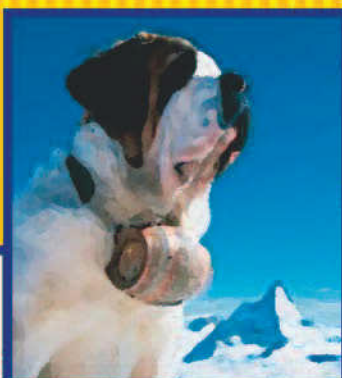


st
scientific tools



Frank Mittelbach
Michel Goossens

Der **L^AT_EX**
Begleiter

2. überarbeitete und erweiterte Auflage

Unter Mitarbeit von J. Braams, D. Carlisle und C. Rowley

PEARSON
Studium

Der L^AT_EX-Begleiter

Zweite überarbeitete und erweiterte Auflage

**Unser Online-Tipp
für noch mehr Wissen ...**



... aktuelles Fachwissen rund
um die Uhr – zum Probelesen,
Downloaden oder auch auf Papier.

www.InformIT.de

Der L^AT_EX-Begleiter

Zweite überarbeitete und erweiterte
Auflage

Frank Mittelbach

L^AT_EX3-Project, Mainz, Deutschland

Michel Goossens

CERN, Genf, Schweiz

mit Johannes Braams, David Carlisle,
und Chris Rowley

und Beiträgen von
Christine Detig und Joachim Schrod



ein Imprint von Pearson Education

München • Boston • San Francisco • Harlow, England
Don Mills, Ontario • Sydney • Mexico City
Madrid • Amsterdam

Bibliografische Information Der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Die Informationen in diesem Buch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht.

Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt. Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht ausgeschlossen werden. Verlag, Herausgeber, Übersetzer und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen. Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und Autoren dankbar.

Es konnten nicht alle Rechteinhaber von Abbildungen ermittelt werden. Sollte dem Verlag gegenüber der Nachweis der Rechtsinhaberschaft geführt werden, wird das branchenübliche Honorar nachträglich gezahlt.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Produkt gezeigten Modelle und Arbeiten ist nicht zulässig.

Dies gilt nicht für die Beispiele in diesem Buch, die unter der LPPL (L^AT_EX Project Public License) stehen und die im Rahmen dieser Lizenz frei benutzbar sind (siehe www.latex-project.org/lppl/).

Fast alle Produktbezeichnungen und weitere Stichworte und sonstige Angaben, die in diesem Buch verwendet werden, sind als eingetragene Marken geschützt. Da es nicht möglich ist, in allen Fällen zeitnah zu ermitteln, ob ein Markenschutz besteht, wird das ® Symbol in diesem Buch nicht verwendet.

Umwelthinweis:

Dieses Produkt wurde auf chlorfrei gebleichtem Papier gedruckt. Die Einschumpffolie – zum Schutz vor Verschmutzung – ist aus umweltverträglichem und recyclingfähigem PE-Material.

10 9 8 7 6 5 4 3 2

09 08 07

ISBN 978-3-8273-7166-9

© 2005 Pearson Studium
ein Imprint der Pearson Education Deutschland GmbH,
Martin-Kollar-Straße 10-12, D-81829 München/Germany
Alle Rechte vorbehalten

<http://www.pearson-studium.de>

Übersetzung: Claudia Krysztofiak, <http://www.krysztofiak.de>

Rebecca Stiels, r_stiels@t-online.de

Diplom-Übersetzerinnen

Lektorat: Irmgard Wagner, Gräfelfing, irmwagner@t-online.de

Fachlektorat: Frank Mittelbach, frank.mittelbach@latex-project.org

Korrektorat: Claudia Krysztofiak, Rebecca Stiels, Hubert Gäßlein

Einbandgestaltung: adesso 21, Thomas Arlt, München

Herstellung: Philipp Burkart, pburkart@pearson.de

Satz: Frank Mittelbach. Gesetzt aus der Lucida Bright 8.5/11.7 Pkt.

Druck und Verarbeitung: Kösel, Krugzell, <http://www.KoeselBuch.de>

Printed in Germany

Korrigierter Nachdruck, Februar 2007



Dieses Buch ist Michael Downes (1958–2003) gewidmet, der uns ein guter Freund war, und ein wunderbarer Kollege im $\text{\LaTeX}$ -Team. Wir vermissen sein freundliches Wesen und seinen stillen Humor, die unser Leben und unsere Arbeit in vieler Hinsicht bereichert haben. Darüber hinaus hat seine Erfahrung, die er in die Unterstützung aller Aspekte des Setzens mathematischer Formeln einfließen ließ, das Leben zahlloser dankbarer $(\text{\LaTeX})$ -Benutzer erleichtert. Sehr *vielen Meisterwerken der Setzkunst* werden für immer an seine stillen, aber tiefen Einsichten erinnern.

Inhaltsverzeichnis

Abbildungsverzeichnis	xix
Tabellenverzeichnis	xxi
Vorwort zur deutschen Ausgabe	xxv
Vorwort zur englischen Originalausgabe	xxvii
Kapitel 1 Einleitung	1
1.1 Ein kurzer Blick in die Vergangenheit	1
1.2 Das heutige System.	6
1.3 Wie man dieses Buch verwendet	11
1.3.1 Was steht wo?	11
1.3.2 Typographische Konventionen	12
1.3.3 Arbeiten mit den Beispielen	15
Kapitel 2 Die Struktur eines L^AT_EX-Dokumentes	17
2.1 Der Aufbau der Quelldateien	17
2.1.1 Verwenden von Paketen und Optionen	19
2.1.2 Aufteilen von Quelldateien	21
2.1.3 Kombinieren mehrerer Dateien	22
2.1.4 optional – Variationen über ein Dokument	23
2.2 Gliederungsbefehle	24
2.2.1 Nummerieren von Überschriften	26
2.2.2 Formatieren von Überschriften	29
2.2.3 Ändern von vorgegebenen Überschriften	36
2.2.4 fncychap – Fertige Layouts für Kapitelüberschriften	36
2.2.5 quotchap – Mottos für Kapitel	37
2.2.6 titlesec – Ein neuer Ansatz für Überschriften	38

2.3	Der Aufbau von Verzeichnissen.	48
2.3.1	Eintragen von Daten in Verzeichnisdateien	49
2.3.2	Formatieren von Verzeichnissen	52
2.3.3	Kombinieren von Verzeichnissen	56
2.3.4	Erstellen weiterer Verzeichnisse	58
2.3.5	shorttoc – Kompakte Inhaltsverzeichnisse	58
2.3.6	minitoc – Mehrfache Inhaltsverzeichnisse.	59
2.3.7	titletoc – Ein anderer Ansatz für Inhaltsverzeichnisse. . .	62
2.4	Verweise in Dokumenten	69
2.4.1	showkeys – Anzeigen der Querverweisschlüssel	71
2.4.2	varioref – Flexiblere Querverweise	72
2.4.3	prettyref – Ausschmücken von Querverweisen	79
2.4.4	titleref – Nicht numerische Verweise	80
2.4.5	hyperref – Dynamische Querverweise.	82
2.4.6	xr – Verweise auf externe Dokumente	82
Kapitel 3	Formatierungswerkzeuge	85
3.1	Textfragmente und Absätze	86
3.1.1	xspace – Korrekte Leerräume nach Makros	86
3.1.2	ellipsis, lips – Auslassungspunkte	87
3.1.3	amsmath – Geschützte Bindestriche	89
3.1.4	relsize – Relative Skalierung der Schriftgröße.	90
3.1.5	textcase – Intelligente Groß- und Kleinschreibung	92
3.1.6	ulem – Betonen durch Unterstreichen.	93
3.1.7	soul – Sperren oder Schafe stehlen	94
3.1.8	url – URLs, Pfadnamen und Ähnliches	99
3.1.9	euro – Konvertieren und Formatieren von Währungen . .	103
3.1.10	lettrine – Schmücken von Absätzen.	106
3.1.11	Randausgleich in L ^A T _E X	108
3.1.12	ragged2e – Verbessern des Randausgleichs	112
3.1.13	setspace – Ändern des Zeilenvorschubs	114
3.1.14	picinpar – Rechteckige Löcher in Absätzen	115
3.2	Fußnoten, Endnoten und Marginalien.	116
3.2.1	Verwenden von Standardfußnoten	117
3.2.2	Anpassen von Fußnoten	119
3.2.3	ftnright – Rechte Fußnoten im Zweispaltensatz	121
3.2.4	footmisc – Verschiedene Fußnotenstile	121
3.2.5	perpage – Zurücksetzen des Zählers auf Seitenbasis . . .	127
3.2.6	manyfoot – Unabhängige Fußnoten.	129
3.2.7	endnotes – Eine Alternative zu Fußnoten	132
3.2.8	Marginalien.	134
3.3	Listen	135
3.3.1	Ändern der Standardlisten	135
3.3.2	paralist – Erweiterte Listenumgebungen	139
3.3.3	amsthm – Theoremähnliche Strukturen.	146
3.3.4	Erstellen eigener Listen.	152
3.4	Wortwörtlicher Text	159
3.4.1	Einfache Verbatim-Erweiterungen.	160
3.4.2	upquote – Anführungsstriche in Programmlistings	161

3.4.3	fancyvrb – Anpassungsfähige Verbatim-Umgebungen . . .	163
3.4.4	listings – Schön gesetzter Programmcode	177
3.5	Zeilen und Spalten	185
3.5.1	lineno – Nummerieren von Textzeilen	185
3.5.2	parallel – Zwei synchronisierte Textstränge.	190
3.5.3	multicol – Setzen in mehreren Spalten	193
3.5.4	changebar – Hinzufügen von Revisionsbalken	198
Kapitel 4	Das Seitenlayout	201
4.1	Geometrische Dimensionen des Layouts.	202
4.2	Verändern des Seitenlayouts.	205
4.2.1	layouts – Darstellen des Layouts	208
4.2.2	Eine Sammlung von Paketen für Seitenlayouts	210
4.2.3	typearea – Ein traditioneller Ansatz.	211
4.2.4	geometry – Layouts mit Auto-Vervollständigung	214
4.2.5	lscap – Setzen einzelner Seiten im Querformat	220
4.2.6	crop – Erzeugen von Beschnittmarken	220
4.3	Dynamische Seitendaten: Seitenzahlen und Textmarken	223
4.3.1	Seitenzahlen in $\text{\LaTeX}$	223
4.3.2	lastpage – Verweise auf die letzte Seite.	224
4.3.3	chappg – Kapitelweise Nummerierung der Seiten.	225
4.3.4	Textmarkenbefehle.	226
4.3.5	extramarks – Eine neue Art von Marken	228
4.4	Layouts für Kolumnentitel	230
4.4.1	Die Low-Level-Schnittstelle	231
4.4.2	fancyhdr – Anpassen von Kolumnentitel-Layouts	232
4.4.3	truncate – Texte auf eine bestimmte Länge kürzen.	240
4.5	Visuelle Formatierung	242
4.5.1	nextpage – Erweiterungen für <code>\clearpage</code>	243
4.6	Layouts mit Klasse	244
4.6.1	KOMA-Script – Ein Ersatz für article et al.	244
4.6.2	memoir – Setzen komplexer Werke.	245
Kapitel 5	Tabellen	247
5.1	$\text{\LaTeX}$ -Standardumgebungen	248
5.1.1	Die tabbing-Umgebung	249
5.1.2	Die tabular-Umgebung	250
5.2	array – Erweiterung von tabular & Co.	252
5.2.1	Einige Beispiele für Präambelbefehle	252
5.2.2	Definieren neuer Spaltenformatkürzel	256
5.3	Errechnen von Spaltenbreiten	257
5.3.1	Explizites Berechnen der Spaltenbreiten	258
5.3.2	tabularx – Spaltenbreiten automatisch berechnen	259
5.3.3	tabulary – Am Inhalt orientierte Spaltenbreiten	261
5.3.4	tabular*, tabularx und tabulary – Ein Vergleich.	263
5.4	Mehrseitige Tabellen.	263
5.4.1	supertabular – Erstellen mehrseitiger Tabellen	264
5.4.2	longtable – Alternative mehrseitige Tabellen	267

5.5	Farbige Tabellen	272
5.6	Anpassen von Linien und Abständen	273
5.6.1	Farbige Tabellenlinien	273
5.6.2	Linien variabler Stärke	274
5.6.3	hhline – Kombinieren horizontaler und vertikaler Linien	274
5.6.4	arydshln – Gestrichelte Linien	275
5.6.5	tabs – Einstellen von Zeilenabständen	277
5.6.6	booktabs – Formale Linien in Tabellen	277
5.7	Sonstige Erweiterungen	280
5.7.1	multirow – Vertikale Ausrichtung in Tabellen	281
5.7.2	dcolum – Ausrichtung am Dezimalpunkt	282
5.8	Tabellenfußnoten	285
5.8.1	minipage-Fußnoten in Tabellen	285
5.8.2	threeparttable – Tabellen und Fußnoten vereint	285
5.9	Verwendungsmöglichkeiten	287
5.9.1	Tabellen mit breiten Einträgen	287
5.9.2	Tabellen in Tabellen	288

Kapitel 6 Gleitobjekte 291

6.1	Parameter für Gleitobjekte	292
6.2	Positionieren von Gleitobjekten.	295
6.2.1	placeins – Gleitobjekte in Schranken	297
6.2.2	afterpage – Kontrolle am Seitenende	297
6.2.3	endfloat – Gleitobjekte am Ende positionieren	298
6.3	Erweiterungen für L ^A T _E Xs Gleitobjektkonzept	300
6.3.1	float – Erstellen eigener Gleitobjektumgebungen	300
6.3.2	caption – Nicht gleitende Abbildungen und Tabellen	304
6.3.3	rotating – Drehen von Gleitobjekten	305
6.3.4	rotfloat – float und rotating kombinieren	306
6.4	Im Fließtext eingebettete Gleitobjekte	307
6.4.1	wrapfig – Text um ein Bild fließen lassen	308
6.4.2	picins – Bilder im Fließtext positionieren	311
6.5	Gleitobjektlegenden	315
6.5.1	caption – Anpassen von Legenden	317
6.5.2	subfig – Gleitobjekte weiter strukturieren	325
6.5.3	subfloat – Serien von Gleitobjekten	331
6.5.4	sidecap – Legenden seitlich setzen	333
6.5.5	fltpage – Legenden auf einer eigenen Seite	335

Kapitel 7 Zeichensätze und Kodierungen 337

7.1	Einleitung	337
7.1.1	Die Geschichte des L ^A T _E X-Fontauswahlverfahrens (New Font Selection Scheme – NFSS)	338
7.1.2	Eingabe- und Ausgabekodierungen	339
7.2	Fontcharakteristika – Zeichensatzmerkmale	342
7.2.1	Dicktengleiche und Proportionalschriften.	342
7.2.2	Schriften mit und ohne Serifen	343

7.2.3	Schriftfamilien und ihre Attribute	343
7.2.4	Fontkodierungen	347
7.3	Zeichensätze im Text	348
7.3.1	Standardfontbefehle in $\text{\LaTeX}$	349
7.3.2	Kombinieren von Standardfontbefehlen	354
7.3.3	Fontbefehle und Deklarationen im Vergleich	355
7.3.4	Zugriff auf alle Zeichen eines Fonts	356
7.3.5	Ändern der voreingestellten Textfonts	357
7.3.6	$\text{\LaTeX}$ 2.09-Fontbefehle	358
7.4	Fonts in mathematischen Formeln	359
7.4.1	Besondere Alphabetbefehle für Formeln	360
7.4.2	Textfontbefehle in Formeln	363
7.4.3	Formellayouts	363
7.5	Die Standardfontunterstützung in $\text{\LaTeX}$	364
7.5.1	Computer Modern – Die $\text{\LaTeX}$ -Standardfonts	365
7.5.2	inputenc – Auswählen der Eingabekodierung	369
7.5.3	fontenc – Auswählen von Fontkodierungen	372
7.5.4	textcomp – Zusätzliche Textsymbole	373
7.5.5	exscale – Skalieren großer Operatoren	379
7.5.6	tracefnt – Überwachen der Fontauswahl	379
7.5.7	nfssfont.tex – Anzeigen von Glyphentabellen	380
7.6	PSNFSS – PostScript Fonts in $\text{\LaTeX}$	381
7.6.1	Schriftproben von PSNFSS-unterstützten Fonts	385
7.6.2	mathptmx – Times Roman in Formeln und Text	387
7.6.3	mathpazo – Palatino in Formeln und Text	389
7.6.4	pifont – Zugriff auf Pi- und Symbolfonts	390
7.7	Eine Sammlung von Fontpaketen	393
7.7.1	eco – Mediävalziffern in Computer Modern	394
7.7.2	ccfonts, concmath – Die Concrete Fonts	395
7.7.3	cmbright – Die Computer Modern Bright Fonts	396
7.7.4	luximono – Ein vielseitiger Schreibmaschinenfont	398
7.7.5	txfonts – Alternative Unterstützung für Times Roman	399
7.7.6	pxfonts – Alternative Unterstützung für Palatino	401
7.7.7	Die Fourier-GUTenberg Fonts	402
7.7.8	Die URW Antiqua und Grotesk Fonts	404
7.7.9	yfonts – Altdeutsche Schriften	405
7.7.10	euler, eulervm – Zugriff auf die Euler Fonts	407
7.8	Die $\text{\LaTeX}$ -Welt der Symbole	411
7.8.1	dingbat – Eine Auswahl von Händen	411
7.8.2	wasysym – Waldis Symbolfont	411
7.8.3	marvosym – Die Schnittstelle zum MarVoSym Font	412
7.8.4	bbding – METAFONT-Alternative zu Zapf Dingbats	414
7.8.5	ifsym – Uhren, Wolken, Berge und andere Symbole	414
7.8.6	tipa – Symbole für das Internationale Phonetische Alphabet	416
7.8.7	Das Euro-Zeichen (€)	418

7.9	Die Low-Level-Schnittstelle	423
7.9.1	Setzen einzelner Zeichensatzattribute	424
7.9.2	Setzen mehrerer Zeichensatzattribute	428
7.9.3	Automatische Fontersetzung	429
7.9.4	Verwendung von Low-Level-Befehlen im Dokument.	430
7.10	Einbinden neuer Zeichensätze	430
7.10.1	Überblick	430
7.10.2	Wie man Tausende von Fonts benennt	431
7.10.3	Deklarieren neuer Schriftfamilien und Gruppen von Schriftformen.	432
7.10.4	Verändern von Schriftfamilien und Schriftschnitten.	441
7.10.5	Einbindung neuer Kodierschemata	442
7.10.6	Interne Dateistruktur	443
7.10.7	Deklarieren neuer Fonts für Formeln	444
7.10.8	Beispiel: Definieren eigener .fd-Dateien	449
7.10.9	Die Reihenfolge der Deklarationen	451
7.11	Kodierungsmodelle in L ^A T _E X	452
7.11.1	Zeichendaten im L ^A T _E X-System	453
7.11.2	Die L ^A T _E X-interne Zeichendarstellung (LICR)	454
7.11.3	Eingabekodierungen	456
7.11.4	Fontkodierungen	460
7.12	Dokumente aus dem letzten Jahrtausend	475
7.12.1	oldfont, rawfonts, newfont – Kompatibilitätspakete	475
7.12.2	latexsym – Zugriff auf L ^A T _E X 2.09 Lasy Fonts	476
Kapitel 8	Höhere Mathematik	477
8.1	Eine Einführung in A _M S-L ^A T _E X	478
8.2	Umgebungen für abgesetzte Formeln	481
8.2.1	Ein Vergleich mit Standard-L ^A T _E X	482
8.2.2	Eine Formel in einer Zeile.	483
8.2.3	Eine Formel in mehreren Zeilen: ohne Ausrichtung	483
8.2.4	Eine Formel in mehreren Zeilen: mit Ausrichtung.	485
8.2.5	Formelgruppen ohne Ausrichtung	487
8.2.6	Formelgruppen mit einfacher Ausrichtung	487
8.2.7	Mehrfache Ausrichtung: align und flalign	487
8.2.8	Formelumgebungen, die Miniseiten erzeugen	489
8.2.9	Unterbrechen abgesetzter Formeln: \intertext.	491
8.2.10	Vertikale Abstände und Seitenumbrüche in und um abgesetzte Formeln	491
8.2.11	Formelnummern und Tags	494
8.2.12	Feinpositionierung von Tags	495
8.2.13	Untersequenzen in der Nummerierung	497
8.2.14	Zurücksetzen des Formelzählers	497
8.3	Matrixähnliche Umgebungen	498
8.3.1	Die cases-Umgebung	498
8.3.2	Die Matrixumgebungen.	498
8.3.3	Mehrzeilige Tief- und Hochstellungen	500
8.3.4	Kommutative Diagramme.	500

8.3.5	delarray – Begrenzungszeichen für Arrays	502
8.4	Komplexere Gebilde und Beschriftungen	502
8.4.1	Beschriftete Pfeile	502
8.4.2	Kettenbrüche	503
8.4.3	Eingerahmte Formeln	503
8.4.4	Grenzpositionen	503
8.4.5	Mehrfachintegrale	504
8.4.6	Modulo-Operationen	505
8.4.7	Brüche und ähnliche Gebilde	505
8.4.8	Punkte als Akzente	507
8.4.9	amsxtra – Akzente als hochgestellte Zeichen	507
8.4.10	Zusätzliche Beschriftungen	507
8.5	Symbole mit variablen Formen	508
8.5.1	Auslassungspunkte	508
8.5.2	Horizontale Erweiterungen	509
8.5.3	Vertikale Erweiterungen	510
8.6	Text in Formeln	512
8.6.1	Der \text-Befehl	512
8.6.2	Operator- und Funktionsnamen	512
8.7	Feinabstimmung des Formellayouts	514
8.7.1	Automatische Skalierung und Abstände	515
8.7.2	Unterformeln	516
8.7.3	Big-g (große) Begrenzungssymbole	517
8.7.4	Verschieben von Wurzelexponenten	517
8.7.5	Ghostbusters™: Gestauchte Phantome	518
8.7.6	Horizontale Abstände	521
8.8	Fonts in Formeln	522
8.8.1	Zusätzliche Mathematikfont-Befehle	523
8.8.2	bm – Fette Zeichen	524
8.8.3	Verschiedene Einstellungen für Mathematikfonts	528
8.9	Symbole in Formeln	540
8.9.1	Mathematik-Symbolklassen	540
8.9.2	Buchstaben, Ziffern und andere reguläre Zeichen	542
8.9.3	Mathematische Akzente	545
8.9.4	Binäre Operatorzeichen	547
8.9.5	Relationssymbole	547
8.9.6	Interpunktion	552
8.9.7	Operatorzeichen	553
8.9.8	Öffnende und schließende Zeichen	553
Kapitel 9	L^AT_EX in einem mehrsprachigen Umfeld	555
9.1	T _E X und nicht englische Sprachen	555
9.1.1	Sprachspezifische Aspekte der Formatierung	557
9.1.2	Kulturspezifische Aspekte der Formatierung	558
9.1.3	Babel – L ^A T _E X spricht mehrere Sprachen	558
9.2	Die babel-Benutzerschnittstelle	560
9.2.1	Einstellen oder Abfragen der aktuellen Sprache	560

9.2.2	Vom Umgang mit Kurzformen	563
9.2.3	Sprachattribute	565
9.3	Sprachspezifische Benutzerbefehle	566
9.3.1	Übersetzungen	566
9.3.2	Verfügbare Kurzformen	568
9.3.3	Sprachspezifische Befehle	574
9.3.4	Erwägungen zum Layout	580
9.3.5	Sprachoptionen und Fontkodierungen	582
9.4	Unterstützung nicht lateinischer Alphabete.	585
9.4.1	Das kyrillische Alphabet	585
9.4.2	Das griechische Alphabet.	590
9.4.3	Das hebräische Alphabet	591
9.5	Anpassen des babel-Systems	596
9.5.1	Silbentrennung für verschiedene Sprachen	596
9.5.2	Die Paketdatei	598
9.5.3	Der Aufbau der babel-Sprachdefinitionsdatei	598
9.6	Andere Ansätze	607
9.6.1	Komplexere Sprachen	608
9.6.2	Omega	608
Kapitel 10 Graphikgenerierung und -bearbeitung		611
10.1	Portable Graphiken und Zierrahmen	613
10.1.1	boxedminipage – Gerahmte Boxen	613
10.1.2	shadow – Schattierte Boxen.	613
10.1.3	fancybox – Zierrahmen	614
10.1.4	epic – Eine erweiterte picture-Umgebung	619
10.1.5	eepic – Erweiterung des epic-Paketes	625
10.1.6	Sprachen für besondere Zwecke	629
10.2	Geräteabhängige Graphikunterstützung.	631
10.2.1	Optionen für graphics und graphicx	632
10.2.2	Die \includegraphics-Syntax im graphics-Paket	634
10.2.3	Die \includegraphics-Syntax im graphicx-Paket	636
10.2.4	Ändern der Voreinstellungen des graphicx-Paketes.	641
10.2.5	Deklarationen zum Einbinden von Bildern	642
10.2.6	Eine Vorsichtsmaßnahme: Kapselung ist wichtig	645
10.3	Manipulieren von graphischen Objekten.	646
10.3.1	Skalieren einer L ^A T _E X-Box	646
10.3.2	Skalieren auf eine bestimmte Größe	647
10.3.3	Drehen einer L ^A T _E X-Box	648
10.3.4	Wiedersehen mit dem rotating-Paket	651
10.4	Seitenbeschreibungssprachen – PostScript, PDF und SVG	652
10.4.1	Die PostScript-Sprache	653
10.4.2	Der PostScript-Treiber dvips	655
10.4.3	pspicture – Erweiterte picture-Umgebung für dvips	657
10.4.4	Das Portable Document Format.	660
10.4.5	Skalierbare Vektorgraphiken	662

Kapitel 11	Indexerstellung	667
11.1	Syntax der Indexeinträge	669
11.1.1	Einfache Indexeinträge	670
11.1.2	Erstellen von Untereinträgen	671
11.1.3	Seitenbereiche und Querverweise.	671
11.1.4	Steuern der Darstellungsform	672
11.1.5	Ausgeben von Sonderzeichen	673
11.1.6	Erstellen eines Glossars.	673
11.1.7	Definieren eigener Indexbefehle	673
11.1.8	Besondere Erwägungen.	674
11.2	makeindex	675
11.2.1	Erstellen eines formatierten Index	675
11.2.2	Einzelne Optionen des <i>MakeIndex</i> -Programms	678
11.2.3	Fehlermeldungen.	679
11.2.4	Anpassen des Index mit <i>MakeIndex</i>	681
11.2.5	<i>MakeIndex</i> -Fallstricke.	687
11.3	xindy – Eine Alternative zu <i>MakeIndex</i>	688
11.3.1	Erzeugen des formatierten Index mit xindy.	690
11.3.2	Internationale Indexierung mit xindy	691
11.3.3	Module für alltägliche Aufgaben	693
11.3.4	Stildateien für individuelle Lösungen	695
11.4	Beeinflussung des Index mit $\text{\LaTeX}$ -Funktionen	702
11.4.1	Ändern des Layouts	702
11.4.2	Kleine Helfer	703
11.4.3	index – Erzeugen mehrerer Indexe	704
Kapitel 12	Quellenverweise	707
12.1	Einleitung	707
12.1.1	Verweisschemata für Bibliographien	708
12.1.2	Markup für Quellenverweise und Bibliographie	710
12.1.3	Erzeugen bibliographischer Daten mit $\text{\BibTeX}$	711
12.2	Das numerische Schema	715
12.2.1	Standard- $\text{\LaTeX}$ – Numerische Verweise	715
12.2.2	cite – Bessere numerische Verweise	718
12.2.3	notocite – Unsortierte Quellenverweise, aber richtig.	722
12.3	Das Autor-Jahr-Schema	722
12.3.1	Frühe Versuche.	724
12.3.2	natbib – Anpassungsfähige Autor-Jahr-Verweise	725
12.3.3	bibentry – Bibliographieeinträge im Fließtext	736
12.4	Das Autor-Nummer-Schema	737
12.4.1	Wiedersehen mit dem natbib-Paket.	737
12.5	Das Kurztitelschema	741
12.5.1	jurabib – Anpassungsfähige Kurztitelverweise	741
12.5.2	camel – Unterstützung für Gesetzestexte.	769
12.6	Mehrere Bibliographien in einem Dokument	771
12.6.1	chapterbib – Eine Bibliographie je $\text{\include}$ -Datei	773
12.6.2	bibunits – Bibliographien für beliebige Einheiten	776

12.6.3	bibtopic – Nach Themen sortierte Literaturhinweise . . .	779
12.6.4	multibib – Separate globale Bibliographien	782
Kapitel 13 Erzeugen von Literaturverzeichnissen		785
13.1	Das BibTeX-Programm und seine Varianten.	786
13.1.1	bibtex8 – Eine 8-Bit-Implementierung von BibTeX	787
13.1.2	Neuere Entwicklungen	788
13.2	Das BibTeX-Datenbankformat.	789
13.2.1	Publikationstypen und Felder.	790
13.2.2	Der Textteil der Felder	794
13.2.3	Kurzformen in BibTeX	798
13.2.4	Die BibTeX-Präambel.	800
13.2.5	Querverweise zwischen Einträgen	801
13.3	Online-Bibliographien	802
13.4	Werkzeuge für Bibliographiedatenbanken.	803
13.4.1	biblist – BibTeX-Datenbankdateien drucken	803
13.4.2	bibttools – Ein Werkzeugset.	804
13.4.3	bibclean usw. – Noch ein Werkzeugset	806
13.4.4	bibttool – Ein Multifunktionswerkzeug	808
13.4.5	pybliographer – Eine erweiterbare Bibliographieverwaltung	813
13.4.6	JBibtexManager – Datenbankverwaltung in Java.	817
13.4.7	BibTexMng – Ein Datenbankmanager für Windows	819
13.5	Formatieren von Bibliographien mit BibTeX-Stilen	820
13.5.1	Eine Sammlung von BibTeX-Stildateien.	820
13.5.2	custom-bib – BibTeX-Stile leicht gemacht.	828
13.6	Die BibTeX-Programmiersprache	835
13.6.1	BibTeX-Befehle und systemimmanente Funktionen.	836
13.6.2	Der Dokumentationsstil btxbst.doc.	837
13.6.3	Kleine Änderungen an Stildateien.	840
Kapitel 14 Dokumentieren eigener L^AT_EX-Pakete		845
14.1	doc – Dokumentieren von Programmcode.	845
14.1.1	Allgemeine Konventionen für Quelldateien	846
14.1.2	Beschreibung neuer Makros und Umgebungen	847
14.1.3	Indexieren aller verwendeten Makros.	849
14.1.4	Der Dokumentationstreiber.	850
14.1.5	Bedingter Code in der Quelldatei	851
14.2	docstrip.tex – Lauffähigen Code erzeugen	857
14.2.1	Aufrufen des docstrip-Programms	858
14.2.2	docstrip-Skriptbefehle	858
14.2.3	Installationsunterstützung und Konfiguration	863
14.2.4	Verwendung mit anderen Programmiersprachen	866
14.3	ltxdoc – Eine einfache Dokumentationsklasse	867
14.3.1	Von ltxdoc bereitgestellte Erweiterungen.	867
14.3.2	Konfiguration der ltxdoc-Ausgabe	868

14.4	Werkzeuge für die Versionsverwaltung	869
14.4.1	rcs – Zugriff auf einzelne Schlüsselwörter	870
14.4.2	rcsinfo – Parsen des Schlüsselwortes \$Id\$	872
Anhang A	Präambeln, Pakete und Klassen – Ein Überblick	873
A.1	Verknüpfen von Markup und Formatierung	873
A.1.1	Befehls- und Umgebungsnamen	874
A.1.2	Definieren neuer Befehle	875
A.1.3	Definieren neuer Umgebungen	880
A.1.4	Definieren und Ändern von Zählern.	883
A.1.5	Definieren und Ändern von Längenparametern	887
A.2	Seiten-Markup – Boxen und Linien	893
A.2.1	LR-Boxen	893
A.2.2	Absatzboxen	896
A.2.3	Linienboxen	899
A.2.4	Arbeiten mit Boxregistern	901
A.2.5	Boxbefehle und Farben	903
A.3	Erweiterte Steuerfunktionen	904
A.3.1	calc – Arithmetische Funktionen	904
A.3.2	ifthen – Verbesserte Steuerfunktionen	905
A.4	Aufbau von Paket- und Klassendateien	910
A.4.1	Die Kenndaten	910
A.4.2	Der Initialisierungsteil	913
A.4.3	Deklariieren von Optionen	913
A.4.4	Ausführen von Optionen	914
A.4.5	Laden von Paketen	916
A.4.6	Der Hauptteil	916
A.4.7	Sonderbefehle für Paket- und Klassendateien.	917
A.4.8	Sonderbefehle Klassendateien	919
A.4.9	Eine minimalistische Klassendatei	922
Anhang B	Finden und Lösen von Problemen	923
B.1	Fehlermeldungen	924
B.1.1	Tod durch zu wenig Speicher.	956
B.2	Warnungen und informelle Meldungen.	961
B.3	T _E X und L ^A T _E X-Tracingbefehle	977
B.3.1	Befehlsdefinitionen und Registerwerte anzeigen	977
B.3.2	Diagnose bei problematischen Seitenumbrüchen.	980
B.3.3	Diagnose und Lösung von Absatzproblemen	985
B.3.4	Weitere Low-Level-Überwachungsfunktionen	989
B.3.5	trace – Selektives Überwachen von Befehlen	992
Anhang C	L^AT_EX-Software und Usergroups	995
C.1	Wie erhält man Hilfe?.	995
C.2	Wo bekommt man diese T _E X-Dateien?	996
C.3	Wie verwendet man CTAN?.	998
C.3.1	Verwenden des T _E X-Dateikatalogs	998
C.3.2	Suchen und Herunterladen von Archivdateien	998

C.3.3	Dateien über die Kommandozeile herunterladen	1000
C.4	Dokumentation auf dem eigenen System	1002
C.4.1	texdoc – Kommandozeilesuche nach Namen	1002
C.4.2	texdoctk – Themensuche mit graphischer Oberfläche . . .	1003
C.5	T _E X-Usergroups	1004
Anhang D	LB2 T_EX-CD	1007
	Der Ursprung – Das T _E X-Live-System	1007
	L ^A T _E X von der CD-ROM installieren	1008
	L ^A T _E X direkt von der CD-ROM ausführen	1008
	Die Beispiele zum L ^A T _E X-Begleiter	1008
	Lizenzen	1009
	Literaturverzeichnis	1011
	Befehls- und Begriffsindex	1033
	Personenverzeichnis	1126
	Biographien	1129
	Entstehungsgeschichte	1135

Abbildungsverzeichnis

1.1	Datenfluss im $\text{\LaTeX}$ -System.	10
2.1	Layout einer abgesetzten Überschrift	30
2.2	Layout einer eingebetteten Überschrift	31
2.3	Layoutparameter für Inhaltsverzeichnisse.	54
3.1	Schematische Darstellung einer Fußnote.	120
3.2	Platzieren von Text und Fußnoten mit dem Paket <code>ftnright</code>	122
3.3	Parameter der <code>list</code> -Umgebung.	153
4.1	Seitenlayoutparameter und ihre Darstellung	203
4.2	Arbeitsweise des Textmarkenmechanismus von $\text{\LaTeX}$	227
6.1	Layout der Abstände des <code>subfig</code> -Paketes	327
7.1	Wichtige Fontcharakteristika.	342
7.2	Buchstaben mit und ohne Serifen im Vergleich	343
7.3	Aufrechte und kursive Zeichen im Vergleich	344
7.4	Echte und falsche Kapitälchen im Vergleich	345
7.5	Outline- und schattierte Schriftformen	345
7.6	Skalierte und entworfene Schriften (Computer Modern)	347
8.1	In Computer Modern gesetzte Beispielseite	528
8.2	Quelltext für die Beispielseiten	529
8.3	In Concrete Fonts gesetzte Beispielseite	530
8.4	In Concrete und Euler gesetzte Beispielseite	530
8.5	In Fourier gesetzte Beispielseite	531
8.6	In Times und Symbol gesetzte Beispielseite.	532
8.7	In Times und TX Fonts gesetzte Beispielseite.	532
8.8	In Times und TM Math Fonts gesetzte Beispielseite	533

8.9	In Palatino und Math Pazo gesetzte Beispielseite	534
8.10	In Palatino und PX Fonts gesetzte Beispielseite	534
8.11	In Palatino und PA Math gesetzte Beispielseite.	535
8.12	In Baskerville gesetzte Beispielseite	536
8.13	In Charter gesetzte Beispielseite	536
8.14	In Lucida Bright gesetzte Beispielseite	537
8.15	In CM Bright gesetzte Beispielseite.	538
8.16	In Helvetica Math gesetzte Beispielseite	538
8.17	In Informal Math gesetzte Beispielseite.	539
9.1	Ein hebräisch-englisches Dokument	594
10.1	Inhalt der Datei <code>w.eps</code>	635
10.2	Eine $\LaTeX$ -Box und mögliche <code>origin</code> -Bezugspunkte	650
10.3	SVG aus einer <code>dvi</code> -Datei generiert	665
11.1	Flussdiagramm für die Indexerstellung.	668
11.2	Schrittweise Entwicklung der Indexerstellung	669
11.3	Einsatz von <code>\index</code> -Befehlen und des <code>showidx</code> -Paketes	676
11.4	Einsatz von <code>\index</code> -Befehlen und des <code>showidx</code> -Paketes (Ausgabe).	677
11.5	Anpassen des Ausgabeformats	685
11.6	Leitpunkte im Index hinzufügen	685
11.7	<code>xindy</code> -Verarbeitungsmodell.	696
12.1	Flussdiagramm des Zusammenspiels von $\BibTeX$ und $\LaTeX$	712
12.2	$\BibTeX$ -Beispieldatenbank <code>tex.bib</code>	714
12.3	$\BibTeX$ -Beispieldatenbank <code>jura.bib</code>	742
13.1	Ausgabe des Programms <code>printbib</code>	805
13.2	Ausgabe des Programms <code>bib2html</code>	806
13.3	Die <code>pybliographic</code> -Benutzeroberfläche	814
13.4	Native Editing in <code>pybliographic</code>	816
13.5	Die <code>JBibtexManager</code> -Benutzeroberfläche (deutsche Version)	818
13.6	Die <code>BibTexMng</code> -Benutzeroberfläche	819
A.1	Eine Klassendatei, die <code>article</code> erweitert	920
C.1	Die Homepage der $\TeX$ Users Group im Internet.	997
C.2	CTAN-Homepage und $\TeX$ -Katalogeintrag	999
C.3	Verwenden der CTAN-Webschnittstelle.	1000
C.4	Dokumentation mit dem <code>texdoctk</code> -Programm suchen	1003

Tabellenverzeichnis

1.1	Wichtige von $\text{\TeX}$ und $\text{\LaTeX}$ verwendete Dateitypen.	9
2.1	Standardgliederungsbefehle in $\text{\LaTeX}$	25
2.2	Sprachspezifische Texte für Überschriften	37
2.3	Die minitoc-Parameter im Überblick.	60
3.1	ISO-Währungscode des <i>Euros</i> und der 12 <i>Euro-Zonen</i> -Länder . . .	103
3.2	Von <code>ragged2e</code> verwendete Parameter	113
3.3	Werte für <code>\baselinestretch</code> bei verschiedenen Schriftgrößen. .	115
3.4	Von <code>footmisc</code> vordefinierte Listen von Fußnotensymbolen.	124
3.5	Befehle zum Steuern der Listenumgebung <code>itemize</code>	136
3.6	Befehle zum Steuern der Listenumgebung <code>enumerate</code>	137
3.7	Von <code>listings</code> unterstützte Sprachen (Winter 2003).	178
3.8	Von <code>multicols</code> verwendete Längenparameter	194
3.9	Von <code>multicols</code> verwendete Zähler	195
4.1	Standardoptionen für Papierformate in $\text{\LaTeX}$	204
4.2	Voreinstellungen der Seitenlayoutparameter (<code>letterpaper</code>). . . .	205
4.3	Layout-Befehle für Kolumnentitel in $\text{\LaTeX}$	231
5.1	Präambel-Formatkürzel der <code>tabular</code> -Umgebung	251
5.2	Zusätzliche Formatkürzel des <code>array</code> -Paketes	252
5.3	Zusätzliche Formatkürzel des <code>tabulary</code> -Paketes	262
7.1	Standardbefehle für Schriftgrade.	353
7.2	Standard-Fontwechselbefehle und Deklarationen	355
7.3	Voreingestellte Fontattribute	357
7.4	Vordefinierte Alphabetbefehle in $\text{\LaTeX}$	361
7.5	Klassifizierung der Computer Modern Fontfamilien	365
7.6	Durch das Paket <code>textcomp</code> bereitgestellte Befehle	374

7.6	Durch das Paket <code>textcomp</code> bereitgestellte Befehle (Forts.)	375
7.7	Von PSNFSS-Paketen verwendete Fonts	382
7.8	Klassifizierung der Fontfamilien der PSNFSS-Distribution	384
7.9	Glyphen des PostScript-Fonts Zapf Dingbats	391
7.10	Glyphen des PostScript-Fonts Symbol.	392
7.11	Klassifizierung der Concrete Fontfamilien.	394
7.12	Klassifizierung der Computer Modern Bright Fontfamilien.	397
7.13	Klassifizierung der LuxiMono Fontfamilien	398
7.14	Klassifizierung der TX Fontfamilien	400
7.15	Klassifizierung der PX Fontfamilien	401
7.16	Klassifizierung der Fourier-GUTenberg Fontfamilien.	402
7.17	Klassifizierung der URW Antiqua und Grotesk Fonts.	404
7.18	Klassifizierung der Euler Mathematikfont-Familien.	408
7.19	Glyphen des wasy Fonts.	412
7.20	Glyphen des MarVoSym Fonts.	413
7.21	Glyphen des METAFONT-Fonts <code>bbding</code>	415
7.22	TIPA-Kürzel	417
7.23	Klassifizierung der EuroSym Fontfamilie.	420
7.24	Klassifizierung der Adobe Euro Fontfamilien.	421
7.25	Klassifizierung von Schriftstärken und -breiten	425
7.26	Klassifizierung der Schriftformen	426
7.27	Standardfontkodierungen unter $\LaTeX$	428
7.28	Karl Berrys Klassifizierungsschema für Fontdateinamen	431
7.29	Glyphen des <code>msbm10</code> Fonts	446
7.30	Klassifizierung mathematischer Symbole	447
7.31	LICR-Objekte, die durch einzelne Zeichen dargestellt werden.	452
7.32	Glyphentabelle eines T1-kodierten Fonts (<code>ecrm1000</code>)	462
7.33	Standard LICR-Objekte.	468
8.1	Formel-Umgebungen des <code>amsmath</code> -Paketes.	481
8.2	Voreingestellte Linienstärke der verschiedenen Formelstile	507
8.3	Vertikal erweiterbare Symbole.	511
8.4	Vordefinierte Operatoren und Funktionen.	513
8.5	Formelstile in Unterformeln	515
8.6	Abstandsbeefehle in Formeln	522
8.7	Abstände zwischen Symbolen	541
8.8	Symbole der Klasse <code>\mathord</code> (lateinische Buchstaben und arabische Ziffern)	543
8.9	Symbole der Klasse <code>\mathord</code> (griechische)	543
8.10	Symbole der Klasse <code>\mathord</code> (buchstabenförmige)	544
8.11	Symbole der Klasse <code>\mathord</code> (verschiedene)	544
8.12	Akzente, die <code>\mathord</code> -Unterformeln erzeugen.	545
8.13	Symbole der Klasse <code>\mathbin</code> (verschiedene)	546
8.14	Symbole der Klasse <code>\mathbin</code> (Kästen).	546
8.15	Symbole der Klasse <code>\mathbin</code> (Kreise)	547
8.16	Symbole der Klasse <code>\mathrel</code> (Gleichheit und Ordnung).	548
8.17	Symbole der Klasse <code>\mathrel</code> (Gleichheit und Ordnung – negiert).	548
8.18	Symbole der Klasse <code>\mathrel</code> (Mengenoperatoren)	549
8.19	Symbole der Klasse <code>\mathrel</code> (Mengenoperatoren – negiert)	549

8.20	Symbole der Klasse <code>\mathord</code> (Pfeile)	550
8.21	Symbole der Klasse <code>\mathord</code> (Pfeile – negiert)	551
8.22	Symbole der Klasse <code>\mathrel</code> (Negationen und Pfeilerweiterungen)	551
8.23	Symbole der Klasse <code>\mathrel</code> (verschiedene)	551
8.24	Symbole der Klassen <code>\mathpunct</code> , <code>\mathord</code> , <code>\mathinner</code> (Interpunktion)	552
8.25	Symbole der Klasse <code>\mathop</code>	552
8.26	Symbolpaare der Klassen <code>\mathopen</code> und <code>\mathclose</code> (erweiterbar).	553
8.27	Symbolpaare der Klassen <code>\mathopen</code> und <code>\mathclose</code> (nicht erweiterbar).	554
9.1	Vom babel-System unterstützte Sprachoptionen	559
9.2	Sprachspezifische Texte in babel (Englische Voreinstellungen) . . .	561
9.3	Sprachspezifische Texte in babel (Deutsch, Französisch, Polnisch, Russisch)	567
9.4	Darstellen von Nummern durch Buchstaben	577
9.5	Mathematische Operatoren in osteuropäischen Sprachen	580
9.6	Glyphentabelle eines T2A-kodierten Fonts (<code>larm1000</code>)	588
9.7	Glyphentabelle eines LGR-kodierten Fonts (<code>grmn1000</code>)	592
9.8	Griechische Transliteration mit lateinischen Buchstaben (LGR) . . .	593
9.9	LGR-Ligaturen, die Glyphen mit einem Akzent erzeugen	593
9.10	Verfügbare Kombinationen aus Spiritus- und Akzentzeichen . . .	593
9.11	Glyphentabelle eines LHE-kodierten Fonts (<code>shold10</code>)	595
9.12	Fontwechselbefehle für hebräische Texte	595
10.1	Überblick über Farb- und Graphikfunktionen der Gerätetreiber . .	633
10.2	Argumente von <code>\DeclareGraphicsRule</code>	644
10.3	Die wichtigsten Optionen des dvips-Programms.	656
11.1	Stilparameter für <i>MakeIndex</i> (Eingabe)	680
11.2	Stilparameter für <i>MakeIndex</i> (Ausgabe).	682
11.3	Von <i>texindy</i> unterstützte Sprachen	692
11.4	<i>xindy</i> -Standardmodule.	694
12.1	Geschlechtsangaben in <i>jurabib</i> (<i>gender</i> -Feld)	760
12.2	Vergleich der Pakete für mehrfache Bibliographien.	772
13.1	Standardpublikationstypen von <i>BibTeX</i>	791
13.2	Standerdeingabefelder in <i>BibTeX</i>	792
13.2	Standerdeingabefelder in <i>BibTeX</i> (Forts.)	793
13.3	Vordefinierte Zeitschriftenkürzel in <i>BibTeX</i> -Stilen	800
13.4	Eine Auswahl von <i>BibTeX</i> -Stildateien	821
13.5	Anforderungen an die Namensdarstellung	828
13.6	In <i>custom-bib</i> unterstützte Sprachen (Sommer 2003)	830
13.7	Befehle der <i>BibTeX</i> -Programmiersprache	838
13.8	Systemimmanente Funktionen der <i>BibTeX</i> -Programmiersprache . .	839

14.1	Überblick über Befehle des <code>doc</code> -Paketes	853
A.1	$\text{\LaTeX}$ -Längenmaße	888
A.2	Vordefinierte horizontale Abstände	889
A.3	Vordefinierte vertikale Abstände	891
A.4	Voreinstellungen für Linienbasisbefehle in $\text{\TeX}$	901
A.5	Interne <code>\boolean</code> -Schalter in $\text{\LaTeX}$	908
A.6	Befehle für Paket- und Klassendateien	911

Vorwort zur deutschen Ausgabe

Seitdem ich das Vorwort zur ersten deutschen Ausgabe des L^AT_EX-Begleiters geschrieben habe, ist mittlerweile ein Jahrzehnt vergangen. Zehn Jahre sind eine lange Zeit für ein Fachbuch über ein sich weiterentwickelndes System, und obwohl sich das Buch über den ganzen Zeitraum stetiger Beliebtheit erfreute, lässt sich nicht abstreiten, dass seine Informationen und Ratschläge langsam anfangen, in Teilen altbacken zu wirken.

Umso mehr freut es mich, den deutschen Lesern mit der nun vorliegenden zweiten Auflage ein radikal erneuertes Buch vorlegen zu können, in das mehrere Jahre Entwicklungsarbeit und nahezu ein Jahr Übersetzung und Anpassung an die Belange der deutschen Sprache eingeflossen sind. Wie im Vorwort zur englischen Ausgabe geschildert, ist dies keine zweite Auflage im üblichen Sinne, sondern ein neues Buch, in das die in den letzten zehn Jahren gewonnene Erfahrung der Autoren und alle wichtigen Neuerungen aus dieser Zeit Eingang gefunden haben.

Die deutsche Übersetzung basiert auf dem zweiten korrigierten Nachdruck der englischen Originalausgabe, wurde aber inhaltlich an vielen Stellen an die Gegebenheiten des deutschen Sprachraums angepasst.

Mein Dank geht an dieser Stelle an die beiden Übersetzerinnen Claudia Krysztofiak und Rebecca Stiels, die sich mit Akribie und Einsatz (oft bis in die frühen Morgenstunden) in die Tiefen der T_EXnischen Details eingearbeitet haben und unsere englische Prosa in leicht verständliches und gut lesbares Deutsch verwandelten. Dank auch an die Mitarbeiter von Pearson Studium, die unkompliziert halfen, alle Stolpersteine aus dem Weg zu räumen.

Last but not least ein immenses Dankeschön an Hubert Gäßlein für seinen großartigen Einsatz beim Korrekturat. Er war es, der den Löwenanteil aller Fehler im ersten Druck der englischen Ausgabe meldete – umso glücklicher war ich, als er seine direkte Mitarbeit bei der deutschen Ausgabe anbot. Was sich jetzt noch an Fehlern finden lässt, geht vermutlich darauf zurück, dass wir den einen oder anderen seiner Korrekturvorschläge missachtet haben oder nachträglich, beim endgültigen Satz, neue Fehler eingebaut haben.

Irren ist menschlich

Jeder gefundene und gemeldete Fehler ist ein Gewinn für alle Leser dieses Buches. Daher schon im Voraus vielen Dank an alle Leser, die uns auf Fehler aufmerksam machen. Die aktuelle Version der Fehlerdatei für den L^AT_EX-Begleiter befindet sich auf der L^AT_EX-Projektsite unter der Internetadresse <http://www.latex-project.org/guides/lb2.err>, zusammen mit einer Onlineversion des Index und weiteren Auszügen aus diesem Buch.

* * *

Die deutsche Ausgabe wurde aber nicht nur inhaltlich, sondern auch visuell an die Gepflogenheiten im deutschen Sprachraum angepasst; ein paar Einzelheiten zur Entstehungsgeschichte finden sich im Nachwort. Während ich im Vorwort der ersten Auflage noch schrieb, dass der optische Randausgleich mithilfe einer Reihe komplizierter Makros erzeugt wurde, und ich hoffte, dass sie eines Tages vielleicht als stabil funktionierendes Paket zur Verfügung stehen würden, lässt sich ein Jahrzehnt später sagen, dass diese Aufgabe nun elegant durch die Satz-Engine selbst (pdfT_EX) erledigt wird.

Bei der Entstehung dieses Buches mitzuwirken, hat allen Beteiligten viel Spaß bereitet – selbst wenn es uns manchmal an den Rand unserer Leistungsfähigkeit gebracht hat. Bleibt zu hoffen, dass sich dies für den Leser in vergleichsweise hohem Nutzen widerspiegelt.

Frank Mittelbach

Rügen, August 2005

Vorwort zur englischen Originalausgabe

Seit dem Erscheinen der ersten Auflage des *L^AT_EX Companion* ist mittlerweile ein Jahrzehnt verstrichen – ein Jahrzehnt, in dem einige den Niedergang von T_EX und L^AT_EX prophezeiten und ankündigten, dass andere Programme die Welt regieren würden. Es gab tatsächlich viele Veränderungen, doch keine der Prophezeiungen ist eingetreten: T_EX ist nicht verschwunden und das Interesse an L^AT_EX ist ungebrochen; nur die Art und Weise, sich mit beiden zu befassen, hat sich im Laufe der Zeit verändert.

Als wir 1993 den ersten *Companion* [57] verfassten, sollte er alles beschreiben, was an Nützlichem in der L^AT_EX-Welt zur Verfügung stand (obwohl er letztendlich nur das beschrieb, was damals bei CERN vorlag). Das hatte den unbeabsichtigten Nebeneffekt, dass die erste Auflage nach Verständnis der meisten Leser *festlegte*, wie eine zu jener Zeit moderne L^AT_EX-Distribution zusammengestellt sein sollte. Glücklicherweise hatten wir in den meisten Fällen eine sinnvolle Auswahl getroffen und der Großteil (wenn auch nicht alle) der in der ersten Auflage beschriebenen Pakete sind auch heute noch weit verbreitet. Oder, um es mit den kürzlich verfassten Worten eines Rezensenten auszudrücken: Obwohl „das Buch in die Jahre gekommen ist, bleibt es größtenteils ein immer noch zuverlässiges Referenzwerk“.

Nichtsdestotrotz hat sich vieles verändert und im Laufe der letzten zehn Jahre wurde L^AT_EX um viele neue und interessante Funktionen erweitert. Die Überarbeitung des *L^AT_EX Companion* endete deshalb damit, dass 90% der ursprünglichen Seiten umgeschrieben wurden und über 600 zusätzliche Seiten mit beeindruckenden neuen Entwicklungen hinzukamen.

Was Sie nun in Händen halten, ist im Grunde ein ganz neues Buch – ein Buch, von dem wir hoffen, dass es die positiven Aspekte der ersten Auflage aufnimmt und erheblich verbessert und gleichzeitig die damals in Inhalt und Darstellung gemachten Fehler nicht mehr zeigt (obgleich wir hier sicherlich

wieder andere Fehler gemacht haben). Als Grundlage für dieses Buch dienten uns die CTAN-Archive. Darüber hinaus durchforsteten wir die Archive der `comp.text.tex`-Newsgroup, um die dringendsten Fragen und Probleme ausfindig zu machen.

Neben den Erläuterungen zu einer guten Auswahl der in den CTAN-Archiven beigesteuerten Pakete beschreibt das Buch viele Aspekte des grundlegenden $\text{\LaTeX}$ -Systems, die im offiziellen $\text{\LaTeX}$ -Manual, dem Buch *$\text{\LaTeX}$: A Document Preparation System* [106] von Leslie Lamport nicht erschöpfend behandelt werden. Dieses Buch soll das *$\text{\LaTeX}$ -Manual* jedoch nicht ersetzen, sondern eher ergänzen: Wer es liest, sollte zumindest bereits den ersten Teil des *$\text{\LaTeX}$ -Manual* (oder eines vergleichbaren Einführungswerkes, wie etwa *Guide to $\text{\LaTeX}$* [103]) gelesen und einige praktische Erfahrung mit dem Erstellen von $\text{\LaTeX}$ -Dokumenten gesammelt haben.

Mit dieser Ausgabe ging auch eine größere Änderung im Feld der Autoren einher: Frank ist nun der federführende Verfasser (so dass er auch für alle Fehler verantwortlich zeichnet) und mehrere Mitglieder des $\text{\LaTeX}$ 3-Projektes haben sich an diesem Buch beteiligt und es durch ihr Wissen und ihre Erfahrung in bestimmten Themengebieten bereichert.

*Ein Dankeschön an
einen großartigen
Menschen!*

Die Arbeiten an diesem Buch wurden durch den plötzlichen Tod unseres guten Freundes, Kollegen und geplanten Mitautors Michael Downes überschattet, dessen bedeutende Beiträge zu $\text{\LaTeX}$ und besonders zu $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\LaTeX}$ vielen gut bekannt sind. Dieses Buch ist ihm und seinem Gedenken gewidmet.

* * *

Wir möchten uns vor allem bei Peter Gordon bedanken, unserem Lektor bei Addison-Wesley, der dieses Buch nicht nur ermöglichte, sondern uns durch seinen unermüdlichen Ansporn immer auf dem richtigen Weg hielt (nur ein paar Jahre zu spät). Als das Manuskript endlich fertig war, zeigte Elizabeth Ryan unendliche Geduld mit unseren Eigenarten und führte uns sicher zum Abschluss.

Unser besonderer Dank gebührt Barbara Beeton, David Rhead, Lars Hellström und Walter Schmidt für das sorgfältige Korrekturlesen einzelner Teile des Manuskriptes. Ihre zahlreichen Kommentare, Vorschläge, Korrekturen und Hinweise haben den Text erheblich verbessert.

Ein ganz besonderes Dankeschön auch an unsere Mitautoren Christine Detig und Joachim Schrod für ihre wertvolle Unterstützung bei Kapitel 11 über die Indexerstellung.

*Verfolgte
Paketentwickler*

Wer seine Hand nahe am Puls des $\text{\LaTeX}$ -Geschehens hat, dem ist vielleicht nicht entgangen, dass es 2002 und 2003 eine erhöhte Anzahl neuer Versionen bewährter $\text{\LaTeX}$ -Pakete gab. Einige dieser Versionen wurden durch die Fragen und Kommentare ausgelöst, die wir bei der Vorbereitung des Manuskriptes für diese zweite Ausgabe an die Paketentwickler richteten. Nahezu alle Paketentwickler reagierten positiv auf unsere Anfragen nach Aktualisierungen, Änderungen und Klärungen und alle investierten ein beträchtliches Maß an Zeit, um uns bei unserer Aufgabe zu unterstützen. Wir möchten uns besonders bei folgenden Personen bedanken: Jens Berger (`jurabib`), Axel Sommerfeldt (`caption`), Steven Cochran (`subfig`), Melchior Franz (`soul`, `euro`) und Carsten Heinz (`listings`). Sie mussten sich um den größten Teil der etwa 6000 E-Mails kümmern, die wir mit den verschiedenen Paketentwicklern austauschten.

Aus ähnlichen Gründen auch ein herzliches Dankeschön an Alexander Rozhenko (manyfoot), Bernd Schandl (paralist), David Kastrup (perpage), Donald Arseneau (cite, relsize, threeparttable, url), Fabrice Popineau (T_EX Live CD), Frank Bennett, Jr. (camel), Gerd Neugebauer (bibtool), Harald Harders (subfloat), Hideo Umeki (geometry), Hubert Gäßlein (sidecap, pict2e), Javier Bezos (titlesec, titletoc), Jean-Pierre Drucbert (minitoc), Jeffrey Goldberg (endfloat, lastpage), John Lavagnino (endnotes), Markus Kohm (typearea), Martin Schröder (ragged2e), Matthias Eckermann (parallel), Michael Covington (upquote), Michel Bovani (fourier), Patrick Daly (custom-bib, natbib), Peter Heslin (ellipsis), Peter Wilson (layouts), Piet van Oostrum (extramarks, fancyhdr), Rei Fukui (tipa), Robin Fairbairns (footmisc), Rolf Niepraschk (sidecap, pict2e), Stephan Böttcher (lineno), Thomas Esser (t_EX distribution), Thomas Henlich (marvosym), Thorsten Hansen (bibunits, multibib) und Walter Schmidt (fix-cm, PSNFSS). Entschuldigung, falls wir jemanden vergessen haben sollten.

Wir danken unseren zahlreichen Kollegen in der (L^A)T_EX-Welt, welche all diese Pakete entwickelt haben, – nicht nur die hier beschriebenen, sondern auch Hunderte von anderen – damit die Anwender ihre Dokumente so setzen können, wie sie es wünschen oder es von ihnen verlangt wird. Ohne die unermüdlichen Anstrengungen dieser Enthusiasten wäre L^AT_EX nicht das großartige und flexible Werkzeug, das es heute ist.

Wir möchten auch Blenda Horn von Y&Y und Michael Vulis von MicroPress für die in diesem Buch verwendeten Schriften danken.

Das Bild von Chris Rowley, das nach einem guten Mittagessen am Hong Kong International Airport aufgenommen wurde, erscheint mit freundlicher Genehmigung von Wai Wong. Das Photo von Michael Downes von der T_EX 2000-Konferenz in Oxford wurde mit freundlicher Genehmigung von Alan Wetmore abgedruckt.

* * *

Wir möchten auch unseren Familien und Freunden für ihre Unterstützung während der Arbeit an diesem Buch danken. Dies mag für manche vielleicht wie ein Alibisatz klingen, er war jedoch bestimmt kaum jemals ernster gemeint als hier.

Chris möchte außerdem der Open University, UK, für ihre Unterstützung bei seiner Arbeit an L^AT_EX danken, und der School of Computer Science and Engineering, University of New South Wales, dafür, dass er seine Arbeiten an diesem Buch in einer so angenehmen Umgebung abschließen durfte.

Frank Mittelbach
Michel Goossens
Johannes Braams
David Carlisle
Chris Rowley

August 2004

KAPITEL 1

Einleitung

1.1 Ein kurzer Blick in die Vergangenheit	1
1.2 Das heutige System.	6
1.3 Wie man dieses Buch verwendet	11

$\text{\LaTeX}$ ist nicht nur ein Satzsystem für mathematische Formeln. Es wird sowohl für kurze Mitteilungen, geschäftliche oder persönliche Korrespondenzen und Newsletter verwendet, als auch zum Verfassen von Artikeln und Büchern natur- und geisteswissenschaftlicher Art. Selbst umfassende Handbücher und Referenzwerke zu den verschiedensten Themen werden mit $\text{\LaTeX}$ erstellt. Inzwischen gibt es für praktisch jede Art von Computer und Betriebssystem eine eigene $\text{\LaTeX}$ -Version. Dieses Buch enthält einen wahren Schatz an Informationen über die vielfältigen gegenwärtigen Einsatzmöglichkeiten des Systems, doch zunächst sollen hier die Hintergründe seiner Entwicklung beleuchtet werden.

Der erste Abschnitt dieses Kapitels blickt auf die Ursprünge und die Entwicklung von $\text{\LaTeX}$ zurück.¹ Im zweiten Abschnitt erhält man einen Überblick über die typischen Dateiarten, mit denen ein modernes $\text{\LaTeX}$ -System arbeitet, und ihre Funktion. Das Kapitel schließt mit einigen Hinweisen dazu, wie man dieses Buch benutzen kann.

1.1 Ein kurzer Blick in die Vergangenheit

Im Mai 1977 begann Donald Knuth von der Stanford University [98] mit seiner Arbeit an dem Textverarbeitungssystem, das heute unter dem Namen „ $\text{\TeX}$ und METAFONT“ [87–91] bekannt ist. Im Vorwort des Buches *The $\text{\TeX}$ book* [87] schreibt Knuth: „ $\text{\TeX}$ ist ein neues Satzsystem, das zum Gestalten schöner Bücher gedacht ist – insbesondere solcher, die sehr viele mathematische Formeln enthalten. Beim Erstellen eines $\text{\TeX}$ -Manuskriptes gibt man

Am Anfang ...

¹Der Artikel *The $\text{\LaTeX}$ legacy: 2.09 and all that* [150] vermittelt einen etwas persönlicheren Eindruck der Geschichte.

dem Computer genaue Anweisungen, wie er dieses Manuskript in Seiten umzusetzen hat, deren typographische Qualität sich mit denjenigen der weltweit besten Setzer vergleichen lässt.“

1979 schrieb Gordon Bell im Vorwort zu einem früheren Buch, *T_EX and METAFONT, New Directions in Typesetting* [85]: „Das System Tau Epsilon Chi (T_EX) von Don Knuth ist die vielleicht bedeutendste Erfindung dieses Jahrhunderts auf dem Gebiet der Satztechnik. Es führt eine Standardsprache in die Computertypographie ein und kann von seiner Bedeutung her nahe der Einführung der Gutenberg-Pressen angesiedelt werden.“

Anfang der 90er Jahre verkündete Donald Knuth offiziell, dass T_EX zugunsten der Stabilität des Systems nicht weiter entwickelt werde [100]. Es ist wohl nicht weiter überraschend, dass T_EX daraufhin durch viele experimentelle Projekte in verschiedene Richtungen erweitert wurde. Einige dieser Projekte tragen nun, zu Beginn des 21. Jahrhunderts, Früchte, so dass alle, die sich mit automatisierter Typographie befassen, eine spannende Zeit erleben.

Donald Knuth hat T_EX ursprünglich nur entwickelt, um damit seine damals laufenden Arbeiten an dem Buch *The Art of Computer Programming* [93] zügig und in hoher typographischer Qualität abzuschließen. Die Weiterentwicklung des Systems, nach seiner Entstehung als Knuths „persönliche Arbeitshilfe“, wurde stark durch die American Mathematical Society beeinflusst und gefördert.

... und Lamport sah, dass es gut war. Während Knuth T_EX entwickelte, begann Leslie Lamport Anfang der 80er Jahre mit seiner Arbeit an dem System zum Formatieren von Dokumenten, das heute unter dem Namen L^AT_EX bekannt ist. Auf Basis der Satzengine und des Makrosystems von T_EX implementierte er eine deklarative Dokumentenbeschreibungssprache, die auf einem System namens Scribe fußt, das von Brian Reid [144] entwickelt wurde. Der Anreiz eines solchen Systems besteht darin, dass man in L^AT_EX mit wenigen High-Level-Deklarationen oder -Befehlen auf einfache Weise eine große Bandbreite an Dokumenten erzeugen kann, ohne dass man sich um ihr typographisches Erscheinungsbild sorgen muss. Zumindest im Prinzip kann man es dem Dokumentengestalter überlassen, das Layout an anderer Stelle festzulegen.

Die zweite Auflage von *L^AT_EX: A Document Preparation System* [106] beginnt mit den folgenden Worten: „L^AT_EX ist ein System zum Setzen von Dokumenten. Seine erste auf breiter Ebene verfügbare Version erschien 1985, rätselhafterweise unter der Versionsnummer 2.09.“ Dieses stabile und gut dokumentierte L^AT_EX-Release führte unmittelbar zu einer rasanten Ausbreitung der auf T_EX basierenden Textverarbeitung über die Gemeinschaft der nordamerikanischen Mathematiker hinaus.

L^AT_EX war die erste in großem Rahmen eingesetzte Sprache zum Beschreiben der logischen Struktur einer Vielzahl von Dokumenten und verbreitete so die Philosophie der logischen Gestaltung, wie sie in Scribe verwendet wurde. Die „logische Gestaltung“ folgt dem Grundsatz, dass Autoren sich nur mit dem logischen Inhalt ihrer Werke befassen sollten und nicht mit deren Erscheinungsbild. Zu dieser Zeit wurde L^AT_EX abwechselnd mal als „T_EX für die Massen“ und mal als ein „von seiner unflexiblen Formatierungssteuerung befreites Scribe“ bezeichnet. Im Laufe des nächsten Jahrzehnts fand es so schnelle Verbreitung, dass Leslie bereits 1994 schreiben konnte: „L^AT_EX ist nun in wissenschaftlichen und akademischen Kreisen sehr beliebt und wird auch

im gewerblichen Bereich großflächig eingesetzt.“ Dieser Grad der Verbreitung erscheint jedoch immer noch gering, gemessen an der heutigen Situation, in der das System für viele Fachleute überall auf der Welt zu einem so selbstverständlichen und unentbehrlichen Arbeitspferd geworden ist, wie die Workstation, auf der es betrieben wird.

Die weltweite Verfügbarkeit von $\text{\LaTeX}$ ließ schnell das internationale Interesse an $\text{\TeX}$ und seiner Verwendung zum Setzen einer Vielzahl von Sprachen steigen. $\text{\LaTeX}$ 2.09 wurde (absichtlich) nicht globalisiert, aber es eignete sich dazu. Darüber hinaus war die beiliegende Dokumentation, aufgrund ihrer klaren Struktur und Geradlinigkeit, durchaus wert übersetzt zu werden. Auf zwei entscheidenden Konferenzen (Exeter, UK, 1988, und Karlsruhe, Deutschland, 1989) wurde die weite Verbreitung von $\text{\LaTeX}$ in Europa klar herausgestellt. Sie führten unmittelbar zu International $\text{\LaTeX}$ [152] und dazu, dass unter Anleitung von Johannes Braams [25] an einer generelleren Unterstützung für viele verschiedene Sprachen und den Wechsel zwischen ihnen gearbeitet wurde (siehe Kapitel 9).

In die weite Welt

Man beachte, dass sich der Begriff der „Sprache“ im Zusammenhang mit Typographie nicht ausschließlich auf die Vielfalt natürlicher Sprachen und Dialekte im Universum bezieht – er hat dort auch eine weiter gefasste Bedeutung. In der Typographie versteht man unter „Sprache“ weit mehr, als nur eine Auswahl an „Zeichen zum Bilden von Wörtern“, da viele wichtige Unterscheidungen von anderen kulturellen Unterschieden herrühren, die sich auf die Traditionen der geschriebenen Kommunikation auswirken. Daher orientieren sich wichtige typographische Unterschiede nicht notwendigerweise an nationalen Grenzen, sondern ergeben sich eher aus verschiedenen Arten von Dokumenten und unterschiedlichen typographischen Schulen.

Einen weiteren wichtigen Beitrag zur Ausbreitung von $\text{\LaTeX}$ leisteten Frank Mittelbach und Rainer Schöpf mit ihrer Pionierarbeit an einer völlig neuen $\text{\LaTeX}$ -Schnittstelle zum Einbinden von Schriften, dem New Font Selection Scheme (NFSS) (siehe Kapitel 7). Sie waren außerdem stark an der Entwicklung des $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\LaTeX}$ -Systems beteiligt, das $\text{\LaTeX}$ um fortgeschrittene Funktionen zum Setzen mathematischer Formeln erweiterte (siehe Kapitel 8).

Die nächste Generation

Als Lohn für ihre Mühen, die unter anderem aus einem ständigen Strom von Fehlermeldungen (und Lösungen) für Leslie bestanden, wurde Frank und Rainer „gestattet“, die technische Unterstützung und Pflege von $\text{\LaTeX}$ zu übernehmen. Als eine ihrer ersten Aktionen konsolidierten sie International $\text{\LaTeX}$, den in Europa entwickelten Standard, als Teil des Systemkernels¹. Sehr bald wurde Version 2.09 formell eingefroren und obwohl die Logeinträge für Änderungen noch einige Monate bis ins Jahr 1992 weiterliefen, stand bereits weitgehend fest, dass das alte System nicht mehr unterstützt werden sollte, da dringend etwas Neues benötigt wurde.

Der weltweite Siegeszug von $\text{\LaTeX}$ führte in den frühen 1990ern in gewissem Sinne zu einem übertriebenen Wildwuchs: Viele $\text{\TeX}$ niker hatten sich darum bemüht, unter der Haube von Leslies „Volkswagen“ solche Extras wie Supermotoren mit Turbolader und -einspritzer in Mehrventilausführung zusammen mit jeglicher Art von „Aussehen ist alles“-Automatik einzubauen – und die wenigsten passten zusammen. Daher wurde die Existenz des neuen

Zu viel des Guten

¹Kernel bedeutet hier Kern oder Herzstück des Systems.

Standard- $\text{\LaTeX}$, das auf den Namen $\text{\LaTeX} 2_{\epsilon}$ getauft wurde, in seiner Ankündigung 1994 folgendermaßen erklärt:

„Im Laufe der Jahre wurden viele Erweiterungen für $\text{\LaTeX}$ entwickelt. Das zeigt natürlich, wie beliebt das System ist, allerdings mit einem weniger schönen Resultat: An verschiedenen Orten kamen $\text{\LaTeX}$ -Formate zum Einsatz, die nicht mehr zueinander kompatibel waren. Um Dokumente von verschiedenen Stellen verarbeiten zu können, waren Systembetreuer daher gezwungen, neben $\text{\LaTeX}$ (mit und ohne NFSS) auch $\text{\texttt{SL\TeX}}$, $\text{\texttt{\AA MS-\LaTeX}}$ und so weiter bereitzustellen. Darüber hinaus ging auch nicht aus jedem Quelldokument eindeutig hervor, in welchem Format es geschrieben war.

Um diese unbefriedigende Situation zu beenden, wurde ein neues $\text{\LaTeX}$ -Release erstellt. Es führt alle derartigen Erweiterungen wieder unter einem Dach zusammen und verhindert so den weiteren Wildwuchs gegenseitig nicht kompatibler Dialekte von $\text{\LaTeX} 2.09$.“

Standard- $\text{\LaTeX}$

Die Entwicklung von „New Standard $\text{\LaTeX}$ “ und der Umgebung zur Pflege des Systems begann 1993, mit etwas Ermutigung und sanftem Nachdruck von Leslie, durch das $\text{\LaTeX} 3$ -Projektteam [122], das damals aus Frank Mittelbach, Rainer Schöpf, Chris Rowley, Johannes Braams, Michael Downes, David Carlisle, Alan Jeffrey und Denys Duchier bestand. Zwar waren die größten Änderungen am $\text{\LaTeX}$ -Basissystem (dem Kernel) und den Standarddokumentenklassen (Stile in 2.09) bereits 1994 abgeschlossen, die zusätzliche Unterstützung für „Typographie in Farbe“, generische Graphiken und Positionierungskontrolle wurden jedoch erst später, weitgehend von David Carlisle, hinzugefügt. Der Zugriff auf Schriften für das neue System umfasste auch Arbeiten von Mark Purtillo über NFSS-Erweiterungen zur besseren Unterstützung von variablen Fontkodierungen und skalierbaren Fonts [32–34].

Das 21. Jahrhundert

Eigentlich wurde diese neue Version ursprünglich mit dem Ziel erstellt, die vielen verschiedenen $\text{\LaTeX}$ -Modelle zu konsolidieren. Dadurch trat jedoch ein wesentlich leistungsfähigeres System zutage, das sowohl über einen robusten Erweiterungsmechanismus (durch Pakete) verfügte, als auch – was wichtig ist – über ein solides Fundament zur technischen Unterstützung und Pflege. So ergab sich durch die Standardisierung und durch die Pflegbarkeit des Quellcodes und der unterstützenden Systeme ein robustes Programm. Dieses System ist denn auch das aktuelle Standard- $\text{\LaTeX}$, das in diesem Buch beschrieben wird. Es hat die meisten der Ziele erreicht oder übertroffen, die 1989 für „ein neues $\text{\LaTeX}$ für das 21. Jahrhundert“ formuliert worden waren [125, 127].

Die besonderen Ansprüche für das aktuelle System lauten: „... bessere Unterstützung von Schriften, Graphiken und Farben; aktiv durch das $\text{\LaTeX} 3$ -Projektteam in Stand gehalten“. Wie diese Ziele im Einzelnen erreicht wurden und die daraus resultierenden Teilsysteme, welche dazu beigetragen haben, dass man den Ansprüchen weitgehend gerecht werden konnte, ergeben ein aufschlussreiches Beispiel für dezentralen Softwaresupport: Die Hauptarbeit wurde in mindestens fünf Ländern bewältigt, und wie die Fehlerdatenbank [112] zeigt, ist die Gesamtzahl derjenigen, die aktive technische Unterstützung leisten, weiterhin hoch.

Obwohl der $\text{\LaTeX}$ -Kern ein wenig an der schleichenden Inflation neu eingebauter Funktionen Ende der 1990er Jahre zu leiden hatte, sorgen nun das Paketsystem gemeinsam mit den klaren Entwicklungsrichtlinien und im rechtlichen Rahmen der $\text{\LaTeX}$ Project Public License (LPPL) [111] dafür, dass $\text{\LaTeX}$ nahezu immer stabil läuft und dabei eine Vielzahl verschiedener Erweiterungen unterstützt. Diese wurden von vielen verschiedenen Leuten beigesteuert, welche, wie das Projektteam freudig bestätigt und der Online-Katalog [171] bezeugt, die verfügbaren Funktionen auf einer breiten Palette verschiedener Bereiche wesentlich verbessert haben.

Das Paketsystem

Alle wichtigen Weiterentwicklungen des Basissystems wurden in den regelmäßigen Ausgaben der $\text{\LaTeX}$ News [113] aufgeführt. Zur Jahrtausendwende konzentrierten sich die Entwicklungsaktivitäten des $\text{\LaTeX}$ 3-Projektteams auf die folgenden Bereiche: Unterstützung für mehrsprachige Dokumente [131], eine „ $\text{\LaTeX}$ -Schnittstelle für Buchdesigner“ [120], wichtige Verbesserungen der Ausgaberroutine [132], bessere Handhabung der Formatierung von Absatzzwischenräumen und eine Lösung für die komplexen Anforderungen an den Vorspann von Zeitschriftenartikeln. Der Programmcode für die Prototypen wurde bereits zur Verfügung gestellt (siehe [119]).

Entwicklungsarbeiten

Das Projektteam hat sich jedoch standhaft geweigert, den Kernel unnötigerweise durch die Integration zusätzlicher Funktionen zu „verbessern“, um nicht in die gleiche Falle zu tappen wie $\text{\LaTeX}$ 2.09 Anfang der 1990er Jahre: den Zerfall in zueinander nicht kompatible Dialekte, bei dem Dokumente, die an einem Ort geschrieben wurden, an einem anderen nicht mehr verarbeitet werden konnten. Man sollte an dieser Stelle nicht vergessen, dass $\text{\LaTeX}$ nicht nur als Satzsystem für Dokumente hoher Qualität dient, sondern als „Lingua franca“ auch die Zusammenarbeit und den Austausch zwischen unterschiedlichen Forschungsgemeinschaften fördert.

Keine neuen Funktionen ...

Durch den $\text{\LaTeX}$ 2 ϵ -Standard lassen sich Dokumente, die 1996¹ geschrieben wurden, auch mit dem heutigen $\text{\LaTeX}$ noch immer verarbeiten. Neue Dokumente laufen auch auf älteren Kernel-Releases, sofern man die Zusatzpakete aktualisiert. Das ist eine Aufgabe, die, im Gegensatz zur Aktualisierung des Kernels, auch Anwender in einer Mehrbenutzerumgebung (etwa an Universitäten oder in Unternehmen) leicht bewältigen können.

Ein stabiler Kernel bedeutet in der Softwareentwicklung jedoch keinen Stillstand: Für die weitere Bedeutung und Beliebtheit von $\text{\LaTeX}$ ist die facettenreiche Sammlung beigesteuerter Pakete ebenso unerlässlich wie der stabile Kern, auf dem diese aufbauen. Der Erfolg des Paketsystems für Erweiterungen außerhalb des Kernels spiegelt sich in der Begeisterung all derer wider, die Pakete dazu beigetragen haben – vielen Dank dafür!

... aber kein Stillstand

Wie man leicht nachvollziehen kann, wenn man das hochverfügbare und stabile Comprehensive $\text{\TeX}$ Archive Network (siehe Anhang C) besucht oder dieses Buch liest (in dem auf Seite 1126 mehr als 250 dieser „netten Jungs“² genannt werden), wurde hier ein wahrer Schatz an $\text{\LaTeX}$ -Paketen und zugehöriger Software geschaffen.

¹Die Zeit zwischen 1994 und 1996 war eine Phase der Konsolidierung für $\text{\LaTeX}$ 2 ϵ , in der noch größere Fehler behoben und Verbesserungen vorgenommen werden mussten, bis das System wirklich stabil lief.

²Leider ist das fast wörtlich zu nehmen: Man muss schon sehr genau hinschauen, um die neun aufgeführten Damen zu finden.

Das Back Office

Allein schon das Bereitstellen der Serviceleistungen, der Werkzeuge und des Supports auf Systemebene für ein so dezentrales Pflege- und Entwicklungssystem stellte eine große intellektuelle Herausforderung dar. Viele der Standardmethoden und Softwaretools für diese Aufgaben gehen davon aus, dass sich die Kollegen hinter der nächsten Tür und nicht auf dem nächsten Kontinent befinden. Zudem waren am Anfang der Entwicklung E-Mail und FTP die einzig zuverlässigen Kommunikationswege. Sowohl die technische Findigkeit als auch die Persönlichkeit aller Beteiligten waren unabdingbare Voraussetzung für ein solches Beispiel an freundlicher „Open Software“-Pflege. Besonders zu erwähnen sind hier jedoch Alan Jeffrey und Rainer Schöpf, die „immer alles repariert haben“.

Ein zentraler Bestandteil dieses Systems, das für die meisten im Verborgenen bleibt, ist das Regressionstestsystem mit seiner riesigen Sammlung von Testdateien [130]. Es wurde von Frank und Rainer in Zusammenarbeit mit Daniel Flipo entworfen und aufgesetzt und hat in der endlosen Fehde mit den Fehlern schon unzählige Male seinen Wert bewiesen.

Forschung

Einige Mitglieder des Projektteams haben auf Basis der Teamerfahrungen ihre eigenen Forschungsarbeiten auf dem Gebiet der Dokumentationswissenschaften über die Grenzen der aktuellen $\text{\LaTeX}$ -Strukturen und -Paradigmen hinaus erweitert. Die folgenden Quellen enthalten einige Beispiele ihrer bis 2003 verfassten Abhandlungen: [30, 31, 35, 36, 123, 128, 139, 148, 149].

Bis 2020?

Bis auf Weiteres hat das Standard- $\text{\LaTeX}$ -System zwei entscheidende Vorteile gegenüber allen anderen vollautomatischen Textverarbeitungssystemen, die in den nächsten 10 Jahren auf der Bildfläche erscheinen werden: Zum einen kann es eine Vielzahl von Elementen in sehr komplexen Dokumenten beliebiger Größe qualitativ hochwertig formatieren; zum zweiten ist es robust in Pflege und Anwendung, so dass es potentiell für mindestens weitere 15 Jahre großflächig im Einsatz bleiben kann.¹

...und in die Zukunft

Ein wichtiges Nebenprodukt der Forschungsarbeiten waren einige Schnittstellen und Erweiterungen, die sich unmittelbar mit Standard- $\text{\LaTeX}$ verwenden lassen. Angesichts des in dieser Weise wachsenden Funktionsumfangs muss man sich früher oder später der Frage stellen, inwieweit ein bloßes Erweitern von $\text{\LaTeX}$ in dieser Form das System noch leistungsfähiger macht, ohne dass es an Robustheit verliert und sich irgendwann nicht mehr pflegen lässt. Dies ist nicht der richtige Ort, um über die Zukunft von $\text{\LaTeX}$ zu spekulieren. Man kann jedoch sicher sein, dass sich das System weiter entwickeln und seinen Einflussbereich ausdehnen wird, sei es im traditionellen Verlagsbereich oder in elektronischen Systemen des Bildungswesens und der Geschäftswelt.

1.2 Das heutige System

Dieser Abschnitt bietet einen Überblick über die große Vielzahl an Dateien, die ein typisches $\text{\LaTeX}$ -System mit all seinen Komponenten verwendet. Ergänzend wird beschrieben, wie die verschiedenen Programmbestandteile zusammenarbeiten. Die meisten Anwender müssen sich eigentlich niemals näher mit der Softwareumgebung, die ihre Arbeit unterstützt, befassen. Dieser Abschnitt ist

¹Einer der Autoren hat öffentlich eine bescheidene Menge Bier darauf gewettet, dass $\text{\TeX}$ bis mindestens 2010 weiterhin allgemein verwendet wird (zumindest von Mathematikern).

jedoch eine nützliche allgemeine Referenz und hilft, einige eher technische Teile dieses Buches besser zu verstehen.

Auch wenn moderne $\text{\LaTeX}$ -Systeme meistens in eine projektorientierte, menügesteuerte Oberfläche integriert sind, hat sich hinter den Kulissen nur wenig gegenüber der hier gegebenen dateibasierten Beschreibung geändert. Die langfristige Stabilität von $\text{\LaTeX}$ ist auch der Grund, warum ein Artikel von Joachim Schrod mit dem Titel *The Components of $\text{\TeX}$* [154] immer noch die beste Quelle für umfassendere Erläuterungen zu einem auf $\text{\TeX}$ basierenden Satzsystem ist. Die folgende Beschreibung setzt voraus, dass man mit einem Standarddateisystem vertraut ist, in dem der „Dateityp“ durch eine „Dateierweiterung“ angegeben wird.

Beim Verarbeiten eines Dokumentes liest und schreibt $\text{\LaTeX}$ verschiedene Dateien, von denen einige in anderen Anwendungen weiterverarbeitet werden. Die entsprechenden Dateitypen sind in Tabelle 1.1 aufgelistet und die schematische Darstellung in Abbildung 1.1 zeigt den Informationsfluss hinter den Kulissen (auf den Seiten 9 und 10).

Die offensichtlichsten wichtigen Dateien in jedem auf $\text{\LaTeX}$ basierenden Dokumentationsprojekt sind die *Quelldateien für die Eingabe*. Dabei handelt es sich typischerweise um eine Masterdatei, die auf weitere untergeordnete Dateien zugreift (siehe Abschnitt 2.1). Diese Dateien haben meistens die Erweiterung `.tex` (Programmdokumentationen für $\text{\LaTeX}$ verwenden üblicherweise die Erweiterung `.dtx`; siehe Kapitel 14). Sie sind im Allgemeinen als „reine Textdateien“ bekannt, da sie mit jedem einfachen Texteditor erstellt werden können. Häufig werden mithilfe der in Abschnitt 10.2 beschriebenen `graphics`-Schnittstelle externe Graphikdateien in ein gesetztes Dokument eingebunden.

Dokumenteingabe

$\text{\LaTeX}$ benötigt zudem verschiedene Dateien mit Struktur- und Layoutdefinitionen: *Klassendateien* mit der Erweiterung `.cls`; *Optionsdateien* mit der Erweiterung `.clo`; *Paketdateien* mit der Erweiterung `.sty` (siehe Anhang A). Viele von ihnen werden bei der grundlegenden Einrichtung des Systems installiert, andere können jedoch auch von einzelnen Anwendern zur Verfügung gestellt werden. Zu $\text{\LaTeX}$ gehören fünf Standarddokumentenklassen: `article`, `report`, `book`, `slides` und `letter`. Diese Dokumentenklassen lassen sich durch die Inhalte anderer Dateien anpassen, die entweder in Klassenoptionen oder durch das Laden zusätzlicher Pakete festgelegt werden. Das Verfahren wird im Abschnitt 2.1 beschrieben. Außerdem verwenden viele $\text{\LaTeX}$ -Dokumente implizit *Sprachdefinitionsdateien* des `babel`-Systems mit der Erweiterung `.ldf` (für „language definition file“, siehe Kapitel 9) und *Kodierungsdefinitionsdateien* der Pakete `inputenc` und `fontenc` mit der Erweiterung `.def` (siehe Kapitel 7).

*Struktur
und Layout*

Die Informationen, die $\text{\LaTeX}$ über die zu setzenden Glyphen benötigt, befinden sich in *$\text{\TeX}$ -Fontmetrikdateien* (Erweiterung `.tfm`). Sie enthalten jedoch nicht die Formen der Glyphen, nur ihre Maße. Welche Fontdateien von $\text{\LaTeX}$ benötigt werden, wird in *Fontdefinitionsdateien* (Erweiterung `.fd`) gespeichert. Beide Dateiarnten werden im Bedarfsfalle automatisch geladen. Kapitel 7 enthält nähere Informationen zu Fontressourcen.

Fontressourcen

$\text{\TeX}$ benötigt noch einige weitere Dateien, denen man mit noch größerer Wahrscheinlichkeit niemals direkt begegnen wird. Eine solche Datei ist zum Beispiel die $\text{\LaTeX}$ -Formatdatei `latex.fmt`, welche die zentralen $\text{\LaTeX}$ -Anweisungen enthält, die bereits zur Verarbeitung mit dem $\text{\TeX}$ -Formatie-

Das $\text{\LaTeX}$ -Format

rungsprogramm vorkompiliert sind. In einigen Situationen muss dieses Format neu kompiliert werden – etwa wenn die Silbentrennungsregeln für L^AT_EX geändert wurden (die mit der Datei `language.dat` konfiguriert werden; siehe Abschnitt 9.5.1), und natürlich, wenn ein neuer L^AT_EX-Kernel eingesetzt wird. Wie ein solches Format im Einzelnen generiert wird, ist von der jeweiligen T_EX-Installation abhängig, so dass in diesem Buch keine näheren Einzelheiten dazu beschrieben werden.

L^AT_EX selbst erzeugt eine Reihe *interner Dateien* (siehe unten) sowie eine sehr wichtige Datei, die alle von T_EX erzeugten Informationen über das Dokument in seiner gesetzten Form enthält.

Formatierte Ausgabe

Diese letztendlich von T_EX, generierte Datei stellt das formatierte Dokument in einem *geräteunabhängigen Format* („device-independent“, Erweiterung `.dvi`) dar. T_EX positioniert Glyphen und Linien mit einer wesentlich größeren Genauigkeit als 0,01 μm (1/4.000.000 Zoll). Aus diesem Grund kann das von T_EX ausgegebene Material definitiv als unabhängig von den Fähigkeiten jedes physikalischen Ausgabegerätes angesehen werden – daher der Name. Einige T_EX-Varianten, wie etwa pdfT_EX [159, 161] und VT_EX [169], können andere geräteunabhängige Dateiformate erzeugen; unter anderem das Portable Document Format (PDF), der Adobe Acrobat-Familie.

Das Dateiformat `.dvi` gibt nur die Namen von Zeichensätzen und die Positionen ihrer Glyphen an – es enthält keine Daten über die Form der Glyphen. Das Dateiformat `.pdf` kann auch Informationen zu ihrer Gestalt enthalten.

Querverweise

Einige der *internen Dateien* enthalten Code, der dazu dient, Informationen von einem L^AT_EX-Lauf für den nächsten zur Verfügung zu stellen, wie zum Beispiel für Querverweise (*Hilfsdateien* mit der Erweiterung `.aux` für „auxiliary“; siehe Abschnitt 2.3) und zum Setzen bestimmter Elemente eines Dokumentes wie etwa des Inhaltsverzeichnisses (Erweiterung `.toc` für „table of contents“, des Abbildungs- (`.lof`, „list of figures“) und des Tabellenverzeichnisses (`.lot`, „list of tables“). Andere Dateien gehören zu bestimmten Paketen (wie etwa `minitoc`, Abschnitt 2.3.6, oder `endnotes`, Abschnitt 3.2.7) oder zu anderen Teilen des Systems (siehe unten).

Fehler, Warnungen
und informelle
Meldungen

Abschließend erzeugt T_EX eine Protokolldatei mit der Erweiterung `.log`, in der all seine Aktivitäten aufgeführt sind. Diese Datei enthält eine Unmenge an Informationen, wie zum Beispiel die Namen der gelesenen Dateien, die Seitenzahlen der verarbeiteten Seiten, Warn- und Fehlermeldungen sowie weitere relevante Daten, die insbesondere für die Fehlersuche nützlich sind (siehe Anhang B).

Indexieren

Eine Datei mit der Erweiterung `.idx` enthält einzelne, unsortierte Einträge, die zu einem Index verarbeitet werden sollen. Diese Einträge müssen mit einem Programm wie *MakeIndex* oder *xindy* (siehe Kapitel 11) sortiert, zusammengestellt und vereinheitlicht werden. Die sortierte Version wird typischerweise als Datei (mit der Erweiterung `.ind`) gespeichert, die wiederum als Eingabedatei für L^AT_EX dient. Die *Indexstildatei* für das Programm *makeindex* hat die Erweiterung `.ist` und seine Protokolldatei die Erweiterung `.ilg`. Bei *xindy* haben die Stildateien die Erweiterung `.xdy`, für die Protokolldatei gibt es keine impliziten Namenskonventionen.

Literaturverweise und
Bibliographie

Daten für Literaturverweise (siehe Kapitel 12) in einem Dokument werden von L^AT_EX normalerweise in die *Hilfsdatei* (`.aux`) geschrieben. Anhand dieser Daten werden zunächst die erforderlichen Informationen aus einer

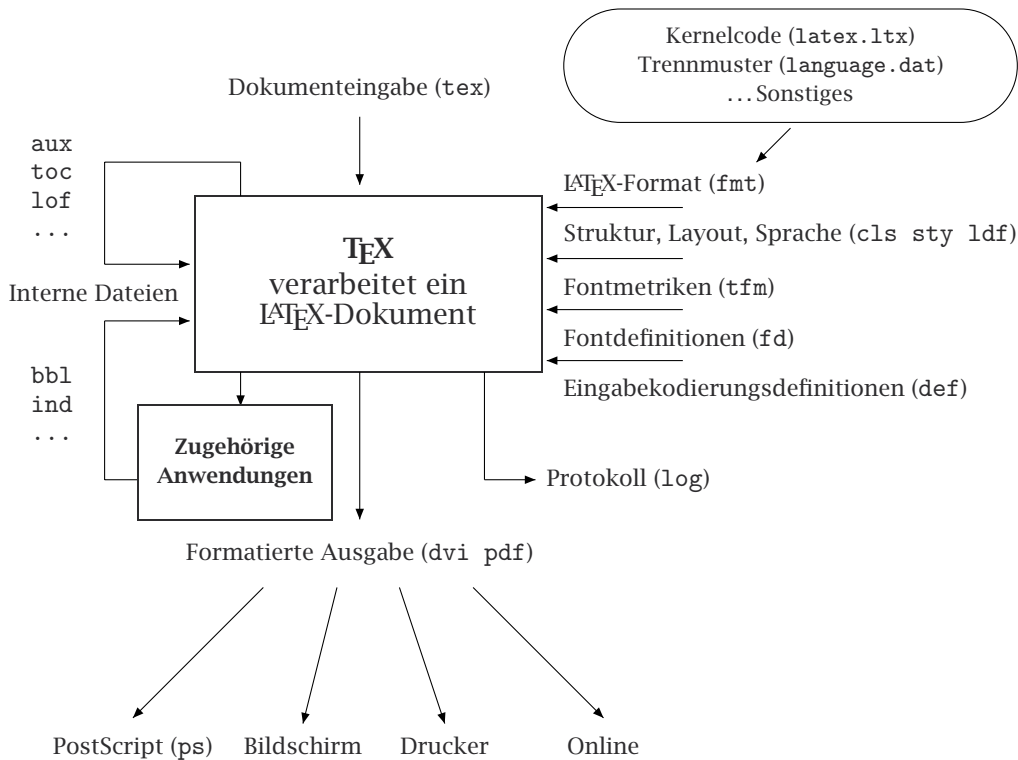
	<i>Dateityp</i>	<i>Übliche Dateierweiterung(en)</i>
<i>Dokumenteingabe</i>	Text	.tex .dtx .ltx
	Bibliographie	.bbl
	Index / Glossar	.ind / .gnd
<i>Graphiken</i>	interne	.tex
	externe	.ps .eps .tif .png .jpg .gif .pdf
<i>Weitere Eingabedaten</i>	Layout und Struktur	.clo .cls .sty
	Kodierungsdefinitionen	.def
	Sprachdefinitionen	.ldf
	Fontzugriffsdefinitionen	.fd
	Konfigurationsdaten	.cfg
<i>Interne Kommunikation (Ein- und Ausgabe)</i>	Hilfsdaten	.aux
	Inhaltsverzeichnis	.toc
	Abbildungs- / Tabellenverzeichnis	.lof / .lot
<i>Low-Level T_EX-Eingabe</i>	Format	.fmt
	Fontmetriken	.tfm
<i>Ausgabe</i>	formatiertes Ergebnis	.dvi .pdf
	Protokoll	.log
<i>Bibliographie (B_BT_EX)</i>	Eingabe / Ausgabe	.aux / .bbl
	Datenbank / Stil / Protokoll	.bib / .bst / .blg
<i>Index (MakeIndex)</i>	Eingabe / Ausgabe	.idx / .ind
	Stil / Protokoll	.ist / .ilg

Tabelle 1.1: Übersicht über die von T_EX und L^AT_EX verwendeten Dateitypen

Literaturdatenbank extrahiert und sortiert. Die sortierte Version wird dann als *Bibliographiedatei* mit der Erweiterung .bbl gespeichert und wieder an L^AT_EX übergeben. Arbeitet das System zu diesem Zweck mit B_BT_EX (siehe Kapitel 13), dann haben die *Literaturdatenbankdateien* die Erweiterung .bib und die Verarbeitungsinformationen sind in einer *Bibliographiestildatei* (Erweiterung .bst) gespeichert. Die entsprechende Protokolldatei hat die Erweiterung .blg.

Da T_EX gewissen Einschränkungen unterliegt und insbesondere nicht mit Graphiken umgehen kann, ist es häufig notwendig, die Formatierung einiger Elemente des gesetzten Dokumentes erst dann abzuschließen, wenn T_EX bereits alles positioniert und diese Informationen in die .dvi-Datei geschrieben hat. Dies geschieht normalerweise, indem man zusätzliche Informationen und Anweisungen für externe Programme an der richtigen „geometrischen Position im gesetzten Dokument“ platziert, und zwar mithilfe des T_EX-Basisbefehls \special, der einfach nur dafür sorgt, dass die jeweilige Information an der richtigen Stelle in der .dvi-Datei vorliegt (siehe Kapitel 10). Bei dieser Information kann es sich ganz schlicht um den Namen einer Graphikdatei

*\special-Befehle
verwenden*

Abbildung 1.1: Datenfluss im $\text{\LaTeX}$ -System

handeln oder es können Anweisungen in einer Graphikbeschreibungssprache sein. Das zurzeit gängigste sekundäre Formatierungsprogramm ist ein *PostScript*-Interpreter. Um diese Methode verwenden zu können, müssen alle von $\text{\TeX}$ in die `.dvi`-Datei geschriebenen Daten, einschließlich derjenigen in `\special`-Befehlen, in PostScript umgewandelt werden. Alle $\text{\LaTeX}$ -Systeme verfügen über Anwendungen, die dies können.

Sehen heißt glauben Sobald das Dokument von $\text{\TeX}$ erfolgreich verarbeitet (und möglicherweise in PostScript umgewandelt) wurde, möchte man sich den formatierten Text wahrscheinlich ansehen. Das geschieht normalerweise auf dem Bildschirm. Eine genaue Überprüfung der gedruckten Ausgabe sollte jedoch immer anhand eines Papierausdrucks in höchstmöglicher Auflösung erfolgen. Die Anwendungen zur Darstellung auf dem Bildschirm sind immer noch (Stand Ende 2003) von System zu System recht unterschiedlich. Einige benötigen eine `.dvi`-Datei, andere verwenden `.ps`-Dateien. Zurzeit ist es besonders beliebt, mit `.pdf`-Dateien zu arbeiten, vor allem, wenn das formatierte Dokument elektronisch verbreitet werden soll. Hin und wieder lässt sich feststellen, dass einige Anwendungen wesentlich bessere Bildschirmausgaben erzeugen als andere. Das liegt oft an den Qualitätsgrenzen der jeweils eingesetzten Technologie und an der Verfügbarkeit geeigneter Schriften.

1.3 Wie man dieses Buch verwendet

Der letzte Abschnitt dieses Kapitels zeigt, wie dieses Buch aufgebaut ist, welchen typographischen Konventionen es folgt und wie man am besten mit den Beispielen in diesem Buch arbeiten kann.

1.3.1 Was steht wo?

Es folgt eine Zusammenfassung der Themengebiete, die in den einzelnen Kapiteln und Anhängen behandelt werden. Im Prinzip kann man alle Kapitel unabhängig voneinander lesen, da gegebenenfalls überall Verweise auf zusätzliche Informationen an anderen Stellen des Buches erfolgen.

- Kapitel 1** gibt eine kurze Einführung in das $\text{\LaTeX}$ -System und seine Geschichte und erläutert auf den noch folgenden Seiten die in diesem Buch verwendeten typographischen Konventionen.
- Kapitel 2** erklärt durch welche Befehle (Markup) der Aufbau eines Dokumentes festgelegt wird, einschließlich der Gliederungsbefehle und Querverweise.
- Kapitel 3** beschreibt die grundlegenden Textsatzbefehle in $\text{\LaTeX}$.
- Kapitel 4** erklärt, wie man das visuelle Layout der Seiten auf verschiedene Weise beeinflussen kann.
- Kapitel 5** zeigt, wie man Material auf einzelnen und mehreren Seiten in Spalten und Zeilen anordnet – kurz gesagt, es behandelt Tabellen aller Art.
- Kapitel 6** befasst sich mit Gleitobjekten und der Formatierung ihrer Legenden.
- Kapitel 7** erläutert das $\text{\LaTeX}$ -Fontauswahlschema in allen Einzelheiten und zeigt, wie man neue Schriften einbindet.
- Kapitel 8** beschäftigt sich mit dem Setzen von Formeln, besonders mit den von der American Mathematical Society unterstützten Paketen.
- Kapitel 9** beschreibt die $\text{\LaTeX}$ -Unterstützung für mehrsprachige Texte und besonders das babel-System.
- Kapitel 10** behandelt die einfacheren Graphikerweiterungen für $\text{\LaTeX}$ und die Verwendung von PostScript.
- Kapitel 11** erklärt, wie man einen Index vorbereitet und setzt. Dabei werden die Programme makeindex und xindy beschrieben.
- Kapitel 12** beschreibt, wie $\text{\LaTeX}$ die verschiedenen gebräuchlichen Verweisschemata für Bibliographien unterstützt.
- Kapitel 13** erläutert, wie man Literaturdatenbanken zusammen mit $\text{\LaTeX}$ verwendet und wie man den Wünschen der Verleger entsprechende Bibliographien erzeugt.

- Kapitel 14** legt dar, wie man $\LaTeX$ -Dateien dokumentiert und wie man solche von anderen bereitgestellte Dateien benutzt.
- Anhang A** befasst sich damit, wie man mit den grundlegenden Programmierstrukturen von $\LaTeX$ umgeht und diese manipuliert, und wie man Klassen- und Paketdateien erzeugt.
- Anhang B** erklärt, wie man Probleme aufspürt und löst.
- Anhang C** zeigt, wie man die in diesem Buch beschriebenen Pakete und Systeme erhält und welche Supportsysteme verfügbar sind.
- Anhang D** stellt kurz die LB2 $\TeX$ -CD-ROM (am Ende des Buches) vor.

An einigen Stellen befasst sich dieses Buch auch mit „Low-Level“- $\TeX$, das eigentlich nichts in einem Buch über $\LaTeX$ zu suchen hat. Nach Kenntnis der Autoren wurden viele dieser Informationen jedoch noch nie im „ $\LaTeX$ -Kontext“ behandelt, obwohl sie sehr wichtig sind. Außerdem wird hier davon ausgegangen, dass es nicht besonders sinnvoll ist, die Leser einfach auf andere Bücher wie *The $\TeX$ book* zu verweisen, da die meisten dort gegebenen Ratschläge zu Plain- $\TeX$ sich entweder nicht direkt auf $\LaTeX$ übertragen lassen oder sogar zu unterschwelligen Fehlern führen. In einigen Abschnitten wurde daher versucht, die Themen durch zusätzliche, im Zusammenhang mit $\LaTeX$ relevante und nützliche Informationen über die zugrunde liegende $\TeX$ -Engine zu in sich geschlossenen Einheiten abzurunden.

1.3.2 Typographische Konventionen

Es ist wichtig, dass Informationen bereits durch ihre Darstellungsform ihre Funktion im Rahmen des Textes vermitteln. Daher werden hier die in diesem Buch verwendeten typographischen Konventionen erläutert.

*Befehle, Umgebungen,
Pakete ...*

Im gesamten Text werden $\LaTeX$ -Befehle und -Umgebungsnamen in einer dicktengleichen (nicht proportionalen) Schrift gesetzt (z.B. `\caption`, `enumerate`, `\begin{tabular}`), während Namen von Paket- und Klassendateien in serifenloser Schrift erscheinen (z.B. `article`). Befehle, die an einem Computer vom Anwender eingegeben werden sollen, werden in einer dicken-gleichen Schrift gezeigt und sind unterstrichen (z.B. Bitte eingeben).

Syntaxbeschreibungen

Die Syntax komplexerer $\LaTeX$ -Befehle wird in einem rechteckigen Kasten wiedergegeben. Die Befehlsargumente sind kleingeschrieben und werden kursiv gesetzt:

```
\titlespacing*{befehl}{links}{oberhalb}{unterhalb}[rechts]
```

Optionale Argumente in $\LaTeX$ werden durch eckige Klammern angezeigt und der Stern deutet auf eine Variante hin (ist also auch optional). Entsprechend besagt der vorige Kasten, dass der Befehl `\titlespacing` in vier verschiedenen Formen auftreten kann:

```
\titlespacing{befehl}{links}{oberhalb}{unterhalb}
\titlespacing{befehl}{links}{oberhalb}{unterhalb}[rechts]
\titlespacing*{befehl}{links}{oberhalb}{unterhalb}
\titlespacing*{befehl}{links}{oberhalb}{unterhalb}[rechts]
```

Bei manchen Befehlen sind nicht alle Kombinationen optionaler Argumente und/oder Sternformen gültig. In diesem Falle werden alle gültigen Alternativen explizit zusammen aufgeführt, wie z.B. die $\text{\LaTeX}$ -Gliederungsbefehle:

$\backslash\text{section}*\{titel\}$	$\backslash\text{section}[toc-eintrag]\{titel\}$

Hier darf das optionale Argument *toc-eintrag* nicht in der Sternform verwendet werden; dadurch ergeben sich die folgenden gültigen Formen:

```
\section*{titel}
\section{titel}
\section[toc-eintrag]{titel}
```

Zeilen, die Beispiele mit $\text{\LaTeX}$ -Befehlen enthalten, werden leicht eingerückt und in einer dicktengleichen Schrift gesetzt, die etwas kleiner ist, als der übrige Text: *Codebeispiele ...*

```
\addtocontents{lof}{\protect\addvspace{10pt}}
\addtocontents{lot}{\protect\addvspace{10pt}}
```

Meistens erfolgen jedoch vollständige Beispiele, die Seite an Seite mit der von *... mit Ausgabe ...* ihnen erzeugten Ausgabe erscheinen:

Die rechte Spalte enthält den von $\text{\LaTeX}$ zu verarbeitenden Quelltext, wobei Präambelmaterial blau dargestellt wird. In der linken Spalte sieht man das gesetzte Resultat.

$\backslash\text{usepackage}\{ragged2e\}$

Die rechte Spalte enthält den von $\backslash\text{LaTeX}\{\}$ zu verarbeitenden Quelltext, wobei Präambelmaterial blau dargestellt wird. In der linken Spalte sieht man das gesetzte Resultat.

Bsp.
1-3-1

Man beachte, dass alle Präambelbefehle im Quelltext der Beispiele immer blau abgedruckt sind.

Für den Fall, dass zum Verdeutlichen eines Sachverhaltes mehrere Seiten erforderlich sind, werden (Teile von) „Doppelseiten“ angezeigt, die normalerweise eingerahmt sind, um zu zeigen, dass es sich um mehrere Seiten handelt. *... für mehrere Seiten ...*

1 EIN TEST
1 Ein Test
Text für unsere Seite, der immer und immer wieder verwendet
Seite 6 von 8

1 EIN TEST
wird.
Text für unsere Seite, der immer und immer wieder
Seite 7 von 8

```
\usepackage{fancyhdr,lastpage}
\pagestyle{fancy}
\fancyhf{} % -- alle Bereiche leeren
\fancyhead[RO,LE]{\leftmark}
\fancyfoot[C]{Seite \thepage\
               von \pageref{LastPage}}

% \sample definiert wie zuvor
\section{Ein Test}
\sample \par \sample
```

Bsp.
1-3-2

Hier sollten einige Dinge beachtet werden:

- Die Beispiele sind normalerweise so angeordnet, dass sie die Seiten 6 und 7 zeigen, also eine Doppelseite.

- Um das Beispiel kurz zu halten, wird normalerweise der Befehl `\sample` mit einem kurzen Text verwendet. Die Definition dieses Befehls ist entweder Teil des Beispiels oder wird, wie hier angedeutet, aus einem früheren Beispiel wiederholt (was an dieser Stelle einfach gelogen ist, da `\sample` noch nicht definiert wurde).
- Die Ausgabe kann Kolumnentitel am Kopf und am Fuß der „Seiten“ enthalten. Im vorigen Beispiel sind beide vorhanden.

... mit umfangreicher
Ausgabe ...

In langen Beispielen, bei denen Quelltext und Ausgabe nicht in geeigneter Weise nebeneinander stehen können, wird das folgende Layout verwendet:

```
\usepackage{ragged2e}
```

Diese lange Zeile wird unansehnlich, wenn man Ein- und Ausgabe zweispaltig nebeneinander setzt.

Je nach Inhalt des Beispiels können zwischen Quelltext und Ausgabe noch Erläuterungen stehen (wie hier).

Diese lange Zeile wird unansehnlich, wenn man Ein- und Ausgabe zweispaltig nebeneinander setzt.

Bsp.
1-3-3

... oder mit Linien
zum Andeuten der
Ränder

In Kapitel 8 taucht noch ein weiteres Beispielformat auf, in dem die Ränder des Beispiels explizit durch zwei dünne blaue vertikale Linien angezeigt werden. Auf diese Weise soll die genaue Positionierung abgesetzter Formeln und ihrer Tags im Verhältnis zum Textrand besser verdeutlicht werden.

$$\begin{aligned}
 (a+b)^3 &= (a+b)(a+b)^2 \\
 &= (a+b)(a^2 + 2ab + b^2) \\
 &= a^3 + 3a^2b + 3ab^2 + b^3 \quad (1)
 \end{aligned}$$

```

\usepackage[tbtags]{amsmath}
\begin{equation}
\begin{split}
(a+b)^3 &= (a+b)(a+b)^2 \\
&= (a+b)(a^2 + 2ab + b^2) \\
&= a^3 + 3a^2b + 3ab^2 + b^3
\end{split}
\end{equation}

```

Bsp.
1-3-4

Alle diese Beispiele sind „vollständig“, wenn man in Gedanken eine `\documentclass`-Zeile (mit der Klasse `article`¹ als Argument) hinzufügt und den Hauptteil des Beispiels in eine `document`-Umgebung setzt. Im Buch wurden die einzelnen Beispiele normalerweise zusätzlich mit folgendem Präambelcode gesetzt:

```

\usepackage[T1]{fontenc}
\usepackage{times}           \usepackage[scaled]{helvet}
\usepackage[ngerman]{babel} \usepackage[ansinew]{inputenc}
\addto\captionsngerman
{\renewcommand\figurename{Abb.\@}%
\renewcommand\listfigurename{Abbildungen}}

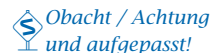
```

¹Außer bei Beispielen zum `\chapter`-Befehl, welche die Klassen `report` oder `book` erfordern.

Die Fontpakete erzwingen Times Roman und Helvetica, das `babel`-Paket wird für korrekte Silbentrennung benötigt und das `inputenc`-Paket, um Umlaute in der Eingabe zu verarbeiten. Die letzten Zeilen sorgen dafür, dass die Beispiele in Kapitel 6 weniger Text erzeugen und so in den schmalen Satzspiegel der Beispiele passen, selbst wenn zwei „Seiten“ nebeneinander gezeigt werden.

Auf diese Weise wurden tatsächlich alle (der fast 1000) Beispiele in diesem Buch hergestellt. Beim Verarbeiten des Buches lesen besondere $\text{\LaTeX}$ -Befehle den Quellcode jedes Beispiels und schreiben ihn jeweils in eine externe Datei, wobei automatisch die Zeilen mit der `\documentclass` und der `document`-Umgebung hinzugefügt werden. Dadurch wird jedes Beispiel zu einem kleinen aber vollständigen $\text{\LaTeX}$ -Dokument. Diese Dokumente werden dann extern verarbeitet (mithilfe eines besonderen Mechanismus, der jedes Beispiel so oft wie nötig ausführt, einschließlich des Erzeugens einer Bibliographie durch `\BibTeX`). Die Ergebnisse werden in kleine EPS-Graphiken konvertiert, die dann beim nächsten Mal, wenn $\text{\LaTeX}$ das ganze Buch verarbeitet, an den entsprechenden Stellen geladen werden. In Abschnitt 3.4.3 auf Seite 171 ist näher beschrieben, wie dieses Verfahren im Einzelnen umgesetzt wurde.

Überall im Buch sind **blaue Marginalien** im Rand verteilt, damit man bestimmte Informationen, die sonst schwieriger auszumachen wären, leichter finden kann. In einigen Fällen sind die Marginalien mit einem Warnzeichen versehen, das darauf hinweist, dass man diese Informationen besser lesen sollte, auch wenn man den entsprechenden Abschnitt ansonsten nur überfliegt.



1.3.3 Arbeiten mit den Beispielen

Dieses Buch wurde mit der Absicht geschrieben, es für die Leser so nützlich wie möglich zu machen. Aus diesem Grunde enthält es fast 1000 vollständige, in sich abgeschlossene Beispiele zu allen Bereichen des Setzens, die in diesem Buch behandelt werden.

Diese Beispiele sind als Quellcode auf CTAN unter der Adresse `info/examples/lb2` erhältlich und zudem im Verzeichnis `Books/lb2/examples` auf der Begleit-CD-ROM zu diesem Buch enthalten. Die Beispiele sind nach Abschnitten durchnummeriert und die jeweilige Nummer wird in einem kleinen Kasten im inneren Rand angezeigt (wie etwa 1-3-4 für das Beispiel auf der vorherigen Seite). Die Namen der externen Dateien setzen sich aus eben diesen Nummern und der Erweiterung `.ltx` für einseitige bzw. `.ltx2` für doppelseitige Beispiele zusammen.

Um die Beispiele weiterzuverwenden, reicht es normalerweise, den Präambelcode (**blau**) in die Präambel des eigenen Dokumentes zu kopieren, und falls nötig, den Text des Dokumentes wie gezeigt anzupassen. In manchen Fällen ist es vielleicht auch besser, den Präambelcode in sein eigenes Paket (oder die eigene Klassendatei) aufzunehmen, so dass man dieses Paket mittels `\usepackage` in mehreren Dokumenten laden kann. Will man Letzteres tun, so muss man zwei Dinge beachten:

- Jeder `\usepackage`-Befehl im Präambelcode muss durch den für Paket- und Klassendateien geeigneten Befehl `\RequirePackage` ersetzt werden (siehe Abschnitt A.4.5).

- Alle `\makeatletter`- und `\makeatother`-Befehle *müssen* aus dem Präambelcode entfernt werden. Das ist sehr wichtig, da eine solche Datei ansonsten nach `\makeatother` nicht mehr richtig gelesen werden kann.

Angenommen man möchte den Quellcode des folgenden Beispiels weiterverwenden (was besonders für mathematisch interessierte Leser nützlich sein könnte):

1 Gleichungen...

$$(a+b)^2 = a^2 + 2ab + b^2 \quad (1.1)$$

$$(a-b)^2 = a^2 - 2ab + b^2 \quad (1.2)$$

```
\makeatletter % '@' jetzt normaler "Buchstabe"
\@addtoreset{equation}{section}
\makeatother % '@' wieder als Sonderzeichen
\renewcommand\theequation{\oldstylenums{\thesection}%
    .\oldstylenums{\arabic{equation}}}\section{Gleichungen\ldots}
\begin{equation} (a+b)^2 = a^2 + 2ab + b^2\end{equation}
\begin{equation} (a-b)^2 = a^2 - 2ab + b^2\end{equation}
```

2 ...abschnittsweise

$$(a+b)(a-b) = a^2 - b^2 \quad (2.1)$$

```
\section{\ldots abschnittsweise}
\begin{equation} (a+b)(a-b) = a^2 - b^2 \end{equation}
```

Bsp. 1-3-5

In diesem Fall gibt es zwei Möglichkeiten: Man kann den Präambelcode (also den blauen Quellcode) in die eigene Dokumentenpräambel kopieren, oder man kann diesen Code - ohne `\makeatletter` und `\makeatother` - in eine Paketdatei (z.B. `reseteqn.sty`) aufnehmen und dieses „Paket“ dann mit `\usepackage{reseteqn}` in der Präambel der eigenen Dokumente laden.

KAPITEL 2

Die Struktur eines $\text{\LaTeX}$ -Dokumentes

2.1 Der Aufbau der Quelldateien	17
2.2 Gliederungsbefehle	24
2.3 Der Aufbau von Verzeichnissen	48
2.4 Verweise in Dokumenten	69

Hinter $\text{\LaTeX}$ steht unter anderem die Idee, das Layout eines Dokumentes so weit wie möglich von seiner Struktur zu trennen, so dass sich der Autor auf den Inhalt konzentrieren kann und sich keine Gedanken über das spätere Aussehen des Dokumentes machen muss [106]. In diesem Kapitel wird erklärt, wie dieses grundlegende Prinzip in $\text{\LaTeX}$ umgesetzt wird.

Im ersten Abschnitt dieses Kapitels wird gezeigt, wie sich Dokumentenklassen, Pakete, Optionen und Präambelbefehle auf die Struktur und das Layout eines Dokumentes auswirken. Zunächst werden die logischen Elemente eines Dokumentes insgesamt betrachtet. Dann wird genauer dargelegt, wie Gliederungsbefehle und ihre Argumente die hierarchische Struktur festlegen und Gliederungsnummern sowie automatisch generierte Kopf- oder Fußzeilen (lebende Kolumnentitel) erzeugen. Anhand von Beispielen werden verschiedene Wege zum Setzen von Überschriften gezeigt. Ebenso wird erklärt, wie man ein Inhaltsverzeichnis zusammenstellt und die Form von Inhalts-, Tabellen- und Abbildungsverzeichnissen seinen eigenen Vorstellungen anpassen kann. Der letzte Abschnitt schließlich gibt eine Einführung in $\text{\LaTeX}$ -Befehle, mit denen sich Querverweise organisieren lassen, sowie deren Geltungsbereiche.

2.1 Der Aufbau der Quelldateien

Mit $\text{\LaTeX}$ lassen sich sehr unterschiedliche Dokumente erzeugen, wie z.B. Artikel, Briefe oder Overheadfolien. Dabei wird schnell deutlich, dass für verschiedene Dokumente auch unterschiedliche logische Strukturen notwendig

sind. Diese wiederum erfordern verschiedene Befehle und Umgebungen. Deshalb werden Dokumente in *Klassen* gleicher Struktur eingeteilt, die allerdings nicht notwendigerweise das gleiche typographische Erscheinungsbild aufweisen. Die Klasse, zu der ein Dokument gehört, wird am Anfang einer L^AT_EX-Datei durch den Befehl `\documentclass` festgelegt, dessen obligatorisches Argument den Namen der *Dokumentenklasse* angibt. Die Dokumentenklasse legt die verfügbaren logischen Befehle und Umgebungen sowie deren Standardformatierung fest (z.B. `\chapter` in der Klasse `report`). Mithilfe einer Liste von *Klassenoptionen* im optionalen Argument lässt sich diese Formatierung verändern. Die Klassenoption `11pt` wird z.B. von den meisten Dokumentenklassen erkannt und bewirkt, dass L^AT_EX elf Punkt als Standardschriftgröße für ein Dokument wählt.

Viele der L^AT_EX-Befehle, die in diesem Buch beschrieben werden, sind nicht nur für eine bestimmte Klasse vorgesehen, sondern können zusammen mit verschiedenen Klassen verwendet werden. Eine Sammlung solcher Befehle ist ein Paket. Die Verwendung bestimmter Pakete in einem L^AT_EX-Dokument wird durch einen oder mehrere `\usepackage`-Befehle hinter `\documentclass` angegeben.

Genau wie die Deklaration `\documentclass` hat `\usepackage` ein obligatorisches Argument, das den Namen des Paketes angibt, und ein optionales Argument, das eine Liste von *Paketooptionen* enthalten kann. Die Paketooptionen beeinflussen das Verhalten des Paketes.

Die Dokumentenklassen und Pakete sind externe Dateien mit den Erweiterungen `.cls` bzw. `.sty`. Der Code für Optionen wird manchmal in separaten Dateien gespeichert (im Falle von Klassenoptionen mit der Erweiterung `.clo`). Normalerweise wird er jedoch direkt in die Klassen- oder Paketdatei integriert (Anhang A enthält weitere Informationen über das Deklarieren von Optionen in Klassen und Paketen). Datei- und Optionsname müssen bei Optionen nicht identisch sein. So gehören z.B. in der Klasse `article` die Datei `size11.clo` und in der Klasse `book` die Datei `bk11.clo` zu der Option `11pt`.

Die
Dokumentenpräambel

Befehle, die im Bereich zwischen der `\documentclass`-Deklaration und `\begin{document}` stehen, befinden sich in der so genannten *Dokumentenpräambel*. In diesem Bereich müssen alle Stilparameter definiert werden, entweder in Paket- oder Klassendateien oder direkt im Dokument, und zwar *vor* dem Befehl `\begin{document}`, der einige der globalen Parameter initialisiert. Eine typische Dokumentenpräambel könnte folgendermaßen aussehen:

```
\documentclass[twocolumn,a4paper]{article}
\usepackage{multicol}
\usepackage[ngerman,french]{babel}
\addtolength\textheight{3\baselineskip}
\begin{document}
```

Diese Dokumentenpräambel weist dem Dokument die Klasse `article` zu und legt fest, dass das Layout durch den Formatierungsbefehl `twocolumn` (zweispaltig setzen) und die Option `a4paper` (Papierformat A4) beeinflusst wird. Der erste `\usepackage`-Befehl informiert L^AT_EX darüber, dass dieses Dokument Befehle und Strukturen aus dem Paket `multicol` enthält. Außerdem wird das Paket `babel` mit den Optionen `ngerman` (Unterstützung der deutschen

Sprache, neue Rechtschreibung) und `french` (Unterstützung der französischen Sprache) geladen. Und schließlich wurde die Standardtexthöhe des Dokumentes um drei Zeilen vergrößert.

Nicht zum Standard gehörige $\text{\LaTeX}$ -Pakete enthalten in der Regel Veränderungen, Erweiterungen oder Verbesserungen¹ gegenüber Standard- $\text{\LaTeX}$, während Befehle in der Präambel nur Änderungen für das aktuelle Dokument definieren. Dadurch ergeben sich mehrere Möglichkeiten, das Layout eines Dokumentes zu verändern:

- Änderung der voreingestellten Parameterwerte einer Klasse durch Optionen für diese Klasse.
- Verwendung von einem oder mehreren Paketen in einem Dokument.
- Änderung der Parametereinstellungen in einem Paket durch Optionen für dieses Paket.
- Definieren von eigenen, lokalen Paketen mit speziellen Parametereinstellungen und Laden dieser Pakete mit dem Befehl `\usepackage` nach dem Paket oder der Klasse, die sie verändern sollen (wie im nächsten Abschnitt erklärt).
- Änderung von Einstellungen in der Präambel zur Feineinstellung.

Wenn man sich intensiver mit $\text{\LaTeX}$ auseinandersetzen möchte, gibt es darüber hinaus natürlich die Möglichkeit, zu verschiedenen Zwecken eigene Pakete zu definieren, welche wiederum über Optionen gesteuert werden können. Nähere Informationen hierzu befinden sich in Anhang A.

2.1.1 Verwenden von Paketen und Optionen

Das heutige $\text{\LaTeX}$ unterscheidet klar zwischen deklarierten Optionen (für Klassen oder Pakete) und allgemeinen Paketdateien. Letztere müssen mit dem Befehl `\usepackage` angegeben werden. Optionen sind Eigenschaften eines gesamten Dokumentes (wenn sie mit `\documentclass` verwendet werden) oder eines einzelnen Paketes (wenn sie mit `\usepackage` verwendet werden).

Eine Option kann nur dann mit `\usepackage` verwendet werden, wenn diese Option auch von dem jeweiligen Paket deklariert wird. Andernfalls gibt $\text{\LaTeX}$ die Fehlermeldung aus, dass die fragliche Option dem gegebenen Paket unbekannt ist. Optionen für `\documentclass` werden etwas anders gehandhabt. Eine Option, die nicht von der jeweiligen Klasse deklariert wird, gilt als „globale Option“.

Alle Optionen von `\documentclass` (deklarierte und globale) werden automatisch als Klassenoptionen an die `\usepackage`-Befehle übergeben. Wenn

¹Viele dieser Pakete sind mittlerweile de facto zu Standards geworden und werden in diesem Buch beschrieben. Das bedeutet jedoch nicht zwangsläufig, dass Pakete, die hier nicht beschrieben werden, weniger wichtig, nützlich oder von geringerer Qualität sind, oder dass sie nicht verwendet werden sollten. Die Autoren konzentrieren sich lediglich auf einige der gängigeren Pakete und erklären zu den anderen, welche Funktionalität in einem gegebenen Bereich möglich ist.

also ein Paket, das mit `\usepackage` geladen wurde, eine dieser Klassenoptionen kennt (d.h. deklariert), kann es entsprechend reagieren. Nicht deklarierte Klassenoptionen werden während der Abarbeitung des jeweiligen Paketes ignoriert. Da alle Optionen in Klassen- oder Paketdateien definiert sein müssen, sind auch ihre Auswirkungen, vom Einstellen interner Schalter bis zum Lesen externer Dateien, durch diese festgelegt. Aus diesem Grund ist ihre Reihenfolge im optionalen Argument von `\documentclass` oder `\usepackage` (normalerweise) unwichtig.

Mehrere Pakete mit den gleichen Optionen (zum Beispiel auch ganz ohne Optionen) lassen sich mit einem einzigen `\usepackage`-Befehl laden, indem die Pakete nacheinander, durch Komma getrennt, im obligatorischen Argument angegeben werden. So ist zum Beispiel

```
\usepackage[german]{babel}   \usepackage[german]{varioref}
\usepackage{multicol}        \usepackage{epic}
```

identisch mit

```
\usepackage[german]{babel,varioref} \usepackage{multicol,epic}
```

Wenn man `german` als globale Klassenoption angibt, wird der `\usepackage`-Befehl noch einmal verkürzt. In diesem Falle wird die Option `german` an alle verwendeten Pakete übergeben und dadurch von all den Paketen verarbeitet, die über diese Option verfügen.

```
\documentclass[german]{book}
\usepackage{babel,varioref,multicol,epic}
```

Das setzt natürlich voraus, dass weder `multicol` noch `epic` ihr Verhalten ändern, wenn `german` als Klassenoption angegeben wird.

Wenn `\begin{document}` erreicht ist, werden alle globalen Optionen dahingehend überprüft, ob sie von irgendeinem Paket genutzt wurden. Für nicht genutzte Optionen wird eine Warnung ausgegeben. Wenn eine Option an keiner Stelle genutzt wird, liegt es meistens daran, dass ein Optionsname falsch geschrieben oder ein `\usepackage`-Befehl, der ein Paket mit dieser Option lädt, gelöscht wurde.

Um eine Dokumentenklasse oder ein Paket selbst zu verändern (z.B. indem man Parameter ändert oder einige Befehle neu definiert), sollte der relevante Code in einer separaten Datei mit der Erweiterung `.sty` gespeichert werden. Diese Datei wird dann mit `\usepackage` nach dem Paket geladen, dessen Verhalten beeinflusst werden soll (oder, im Falle von Klasseneinstellungen, nach der Dokumentenklasse).

Die Veränderungen können statt dessen auch direkt in die Präambel des jeweiligen Dokumentes eingefügt werden. Wenn der eingefügte Code interne L^AT_EX-Befehle enthält, wie z.B. solche mit einem `@`-Zeichen im Namen, muss er zwischen den Befehlen `\makeatletter` und `\makeatother` stehen. Die Verwendung interner Befehle in der Präambel wird auf Seite 875 ausführlicher besprochen.

2.1.2 Aufteilen von Quelldateien

L^AT_EX-Quelldateien lassen sich mit dem Befehl `\include` bequem in mehrere Dateien aufteilen. Dadurch können Dokumente in Teilen neu formatiert werden, indem als Argument für den Befehl `\includeonly` nur die Namen der Dateien eingesetzt werden, die L^AT_EX neu formatieren soll. Bei allen anderen mit `\include`-Befehlen geladenen Dateien liest L^AT_EX nur die Querverweisinformationen und die aktuellen Werte der Zähler (page, chapter, table, figure, equation...) aus den entsprechenden .aux-Dateien ein, sofern diese bereits in einem früheren Lauf erzeugt wurden. Im folgenden Beispiel will der Anwender nur die Dateien `kap1.tex` und `anh1.tex` neu formatieren:

*Dokumente in Teilen
neu formatieren*

```
\documentclass{book}           % Dokumentenklasse ‘book’

\includeonly{kap1,anh1} % nur kap1 und anh1 einschließen

\begin{document}
\include{kap1}           % kap1.tex laden
\include{kap2}           % kap2.tex laden
\include{kap3}           % kap3.tex laden
\include{anh1}           % anh1.tex laden
\include{anh2}           % anh2.tex laden
\end{document}
```

Dabei ist zu beachten, dass L^AT_EX, wenn es eine mit `\include` eingebundene Datei nicht findet, lediglich eine Warnung wie "No file `anh2.tex`" ausgibt und anschließend mit der Formatierung fortfährt.

Wenn die Informationen in den .aux-Dateien aktuell sind, ist es somit möglich, nur einen Teil eines Dokumentes zu formatieren und trotzdem im neu formatierten Teil die korrekten Zähler, Querverweise und Seitennummern zu erhalten. Wenn sich jedoch ein Zähler (auch die Seitenzahl für einen Querverweis) im neu formatierten Teil ändert, kann es sein, dass das gesamte Dokument erneut formatiert werden muss, um fortlaufende und korrekte Index-, Verzeichnis- und Literaturverweise zu erhalten.

Dokumententeile, die mit `\include` geladen werden, beginnen auf einer neuen Seite und rufen am Ende den Befehl `\clearpage` auf. Dadurch bleiben in ihnen enthaltene Gleitobjekte in dem Seitenbereich, der von dem jeweiligen Teil erzeugt wurde. Geeignete Kandidaten für das `\include`-Verfahren sind damit eher ganze Kapitel eines Buches und nicht so sehr kleine Abschnitte eines Textes.

Es ist zwar einerseits von Vorteil, ein längeres Dokument in überschaubare Dateien aufzuteilen, die dann mit einem Texteditor bearbeitet werden können; das teilweise Formatieren sollte jedoch nur mit größter Sorgfalt angewendet werden, und nur als Probelauf für ein oder zwei Kapitel, wenn diese noch nicht fertiggestellt sind. Es gibt letztlich keine andere sichere Methode, eine völlig fehlerfreie Endversion zu erstellen, als das ganze Dokument neu zu formatieren. Dokumente, die sich wegen ihrer Größe nicht in einem einzigen Lauf formatieren lassen, können in mehrere Teile zerlegt werden, um sie dann getrennt voneinander zu verarbeiten. In diesem Falle müssen die einzelnen Teile *in der richtigen Reihenfolge* und wenn nötig mehrmals formatiert werden, um überall korrekte Querverweise und Seitenzahlen zu erhalten.

Interaktives Einbinden

Wer mit `\include`-Befehlen arbeitet, sollte sich einmal das kleine Paket `askinclude` von Pablo Straub ansehen. Es fragt interaktiv, welche Dateien eingebunden werden sollen. Die entsprechenden Dateien können dann durch Kommas getrennt aufgelistet werden, so wie sie im Argument des Befehls `\includeonly` erscheinen müssten. Wenn nur die Eingabetaste betätigt wird, dann werden erneut automatisch die Dateien des vorhergehenden Formatierungslaufes verwendet; im ersten Lauf werden durch dieses Vorgehen alle Dateien eingebunden. Bei Eingabe von `*` werden alle Dateien, durch Eingabe von `-` wird keine Datei eingebunden. Dadurch muss die zentrale Quelldatei nicht verändert werden, wenn man unterschiedliche Teile eines Dokumentes formatieren möchte; eine Funktionalität, die sich beim Schreiben dieses Buches als ausgesprochen hilfreich erwiesen hat.

Ausschließen statt einbinden

Das Paket `excludeonly` von Dan Luecking und Donald Arseneau stellt eine Erweiterung der `\include`-Funktion dar. Es unterstützt den Befehl `\excludeonly`, der als Argument eine durch Komma getrennte Liste von `\include`-Dateinamen erhält und das Einbinden dieser Dateien verhindert. Wenn sowohl `\includeonly` als auch `\excludeonly` eingesetzt werden, verbleiben nur noch die Dateien in Verwendung, die von beiden zugelassen werden. In dem Beispiel

```
\includeonly{kap1,kap2,kap3,anh1}
\excludeonly{kap2,kap3,anh2}
```

werden nur die Dateien `kap1` und `anh1` eingebunden. Dieses Verhalten widerspricht eigentlich dem Namen des Paketes, der darauf schließen lässt, dass nur („only“) die in der Liste angegebenen Dateien ausgeschlossen werden. Dieser Effekt wird mithilfe der Option `only` erzielt, die bewirkt, dass die Angaben von `\includeonly` ignoriert werden.

Das Paket definiert den internen Befehl `\@include` neu, so dass es nicht zusammen mit anderen Paketen oder Klassen funktioniert, die diesen Befehl ebenfalls modifizieren. So treten z.B. in Kombination mit den Dokumentenklassen `paper` oder `thesis` von Wenzel Matiaske Konflikte auf.

2.1.3 Kombinieren mehrerer Dateien

Beim Versenden von L^AT_EX-Dokumenten an andere Personen, ist es häufig erforderlich, lokale oder weniger verbreitete Paketdateien, (z.B. Pakete mit eigenen Änderungen) zusammen mit der Quelldatei zu verschicken. Dabei ist es oft hilfreich, wenn alle zur Formatierung des Dokumentes benötigten Informationen in einer einzelnen Datei zusammengefasst werden können.

Zu diesem Zweck verfügt L^AT_EX über die Umgebung `filecontents`. Diese Umgebung erhält als einziges Argument den Namen einer Datei¹; ihr Text ist der Inhalt dieser Datei. Diese Umgebung kann nur vor einer `\documentclass`-Deklaration stehen. Die Tags `\begin` und `\end` sollten in der Quelldatei jeweils in einer eigenen Zeile stehen. Es sollte vor allen Dingen nichts direkt dahinter stehen, da ansonsten L^AT_EX-Fehler auftreten.

¹Wenn keine Erweiterung angegeben wird, entspricht der tatsächliche Name der externen Datei demjenigen, den L^AT_EX einlesen würde, wenn man die Eingabe als Argument zu `\input` benutzt, d.h. normalerweise wird die Endung `.tex` angefügt.

Wenn L^AT_EX auf diese Umgebung stößt, überprüft es, ob bereits eine Datei mit diesem Namen existiert. Wenn dies nicht der Fall ist, schreibt L^AT_EX den Inhalt der Umgebung wortgetreu in eine Datei dieses Namens im aktuellen Verzeichnis und gibt eine entsprechende Meldung aus. Wenn die Datei bereits existiert, meldet L^AT_EX, dass die Umgebung `filecontents` in diesem Falle ignoriert wurde, weil die Datei schon vorhanden ist.

Die so erzeugte Datei wird durch einige Kommentarzeilen (mit % als Kommentarzeichen) eingeleitet, in denen erklärt wird, dass sie von einer `filecontents`-Umgebung generiert wurde:

```
%% LaTeX2e file 'foo.txt'
%% generated by the 'filecontents' environment
%% from source 'test' on 2003/04/16.
```

Für den Fall, dass der Dateiinhalt nicht verändert werden darf – zum Beispiel weil es sich nicht um eine LaTeX-Datei handelt – ist die `filecontents*`-Umgebung besser geeignet, da sie keine zusätzlichen Kommentarzeilen erzeugt.

Um eine Liste (fast) aller Dateien zu erhalten, die in einem Dokument verwendet wurden, kann der Befehl `\listfiles` in der Präambel angegeben werden. So lässt sich auf einfache Weise festhalten, welche Dateien zusammengehören.

2.1.4 optional – Variationen über ein Dokument

Manchmal ist es sinnvoll, verschiedene Versionen eines Dokumentes in der gleichen Quelldatei vorzuhalten, besonders dann, wenn der größte Teil des Textes für alle Versionen identisch ist. Das Paket `optional` von Donald Arseneau stellt die hierzu erforderlichen Funktionen bereit.

Die variierenden Textteile werden in der Quelle mit dem Befehl `\opt` gekennzeichnet. Beim Formatieren werden einige von ihnen ausgewählt. Der Befehl besitzt zwei Argumente: Ein Label oder eine durch Kommas getrennte Liste von Labels, die beschreiben, zu welcher Textvariante ein Abschnitt gehört, sowie den entsprechenden Text, der fallweise gedruckt wird. Da der Text als Argument angegeben wird, darf er keine `\verb`-Befehle enthalten und alle öffnenden geschweiften Klammern müssen auch wieder geschlossen werden. Dieses Verfahren eignet sich besonders für kürzere Texte. Längere optionale Textteile werden besser in externen Dateien gespeichert, um sie dann, wie in dem folgenden Beispiel, mit den Befehlen `\opt` und `\input` fallweise einzubinden.

Die zu druckenden Teile können auf verschiedene Weise ausgewählt werden. Das folgende Beispiel zeigt die statische Methode, bei der die gewünschten Varianten als Optionen der `\usepackage`-Deklaration angegeben werden.

```
\usepackage[code]{optional}
\opt{doc}{Bei Option doc ausgeben.}
\opt{code}{Bei Option code ausgeben.}
\opt{doc,code}{Bei Option doc oder code ausgeben.}
Immer ausgeben.      \opt{}{und diesen Teil nie!}
\opt{doc}{\input{examples}}
```

Bei Option code ausgeben. Bei Option doc oder code ausgeben. Immer ausgeben.

Bsp.
2-1-1

Alternativ kann man `\AskOptions` in der Präambel angeben. Dann wird der Anwender jedesmal nach einer Liste von Optionen gefragt. Dieses Verfahren kann jedoch sehr mühsam werden, wenn das Dokument häufig formatiert werden muss. Um die Auswahl der geeigneten Optionen zu erleichtern, kann der Befehl `\ExplainOptions` definiert werden. In diesem Fall wird vor jeder Abfrage der Optionen der dazugehörige Text angezeigt.

Bei L^AT_EX-Installationen, die anstelle des Dateinamens L^AT_EX-Code an das Programm durchreichen können, gibt es noch eine dritte Möglichkeit Varianten auszuwählen. Wenn man L^AT_EX mit der Zeile

```
latex "\newcommand\UseOption{doc,code}\input{file}"
```

aufruft, werden die Varianten mit den Labels `doc` und `code` verwendet, und zwar zusätzlich zu denjenigen, die gegebenenfalls mit `\usepackage` angegeben wurden. Die Befehlszeile in dem oben angeführten Beispiel eignet sich für UNIX-Systeme. Auf anderen Plattformen benötigt man möglicherweise eine andere Syntax.

Das Paket `optional` wählt die Varianten während der Formatierung mit L^AT_EX aus. Je nach Anwendung kann ein anderes Verfahren mit einem Präprozessor, der die jeweiligen Varianten aus der zentralen Quelldatei ausliest, besser geeignet sein. Das Programm `docstrip` eignet sich hier zum Beispiel sehr gut. Im Gegensatz zu anderen Präprozessoren hat es den Vorteil, dass es überall eingesetzt werden kann, wo ein L^AT_EX-System installiert ist (siehe auch Abschnitt 14.2).

2.2 Gliederungsbefehle

Die L^AT_EX-Standardklassen (d.h. `article`, `report` und `book`) enthalten Befehle und Umgebungen, mit denen die verschiedenen Elemente der hierarchischen Struktur eines Dokumentes definiert werden (z.B. Kapitel, Abschnitte, Anhänge). Jeder dieser Befehle definiert eine Gliederungsebene innerhalb einer Hierarchie und jedes Strukturelement gehört zu einer Ebene.

Ein typisches Dokument (wie zum Beispiel ein Artikel) besteht aus einem Titel, einigen Abschnitten, die wahrscheinlich in viele weitere Ebenen untergliedert sind, und einem Literaturverzeichnis. Eine solche Struktur wird mithilfe verschiedener Befehle beschrieben: `\maketitle` (erzeugt Dokumententitel), Gliederungsbefehlen wie `\section` und `\subsection`, sowie der Umgebung `thebibliography`. Dabei sollten die Gliederungsebenen beachtet werden. Einen `\subsection`-Befehl sollte man z.B. immer nur innerhalb einer `\section` verwenden.

Umfangreichere Werke (wie Berichte, Anleitungen und Bücher) beginnen mit einer Titelei, sind in Kapitel (und Teile) gegliedert, enthalten Querverweise (Inhalts-, Abbildungs-, Tabellenverzeichnis und Index) und haben wahrscheinlich Anhänge. In solchen Dokumenten lassen sich *Vorspann*, *Hauptteil* und *Nachspann* leicht voneinander unterscheiden. In der L^AT_EX-Klasse `book` können diese drei Bereiche mit den Befehlen `\frontmatter`, `\mainmatter` und `\backmatter` explizit gekennzeichnet werden. In anderen Klassen findet man häufig nur den Befehl `\appendix`, der den Hauptteil vom Nachspann absetzt.

<code>\part</code> (in book und report)	Ebene -1	<code>\part</code> (in article)	Ebene 0
<code>\chapter</code> (nur in book und report)	Ebene 0	<code>\section</code>	Ebene 1
<code>\subsection</code>	Ebene 2	<code>\subsubsection</code>	Ebene 3
<code>\paragraph</code>	Ebene 4	<code>\subparagraph</code>	Ebene 5

Tabelle 2.1: Standardgliederungsbefehle in $\text{\LaTeX}$

Im Vorspann wird normalerweise die so genannte *Sternform* der Gliederungsbefehle `\section` oder `\chapter` verwendet. Diese Form unterdrückt die Nummerierung von Überschriften. Abschnitte mit vorgegebenen Namen, wie „Einleitung“, „Stichwortverzeichnis“ und „Vorwort“ werden gewöhnlich nicht nummeriert. In den Standardklassen rufen die Befehle `\tableofcontents`, `\listoftables`, `\listoffigures` sowie die Umgebungen `theindex` und `thebibliography` intern die Sternform der Befehle `\section` oder `\chapter` auf.

Standard- $\text{\LaTeX}$ verwendet die in Tabelle 2.1 aufgeführten Gliederungsbefehle. Der Befehl `\chapter` definiert die Gliederungsebene null in der hierarchischen Struktur eines Dokumentes, der Befehl `\section` Ebene eins und so fort. Der optionale Befehl `\part` definiert die Ebene minus eins (oder null in Klassen, die den Befehl `\chapter` nicht verwenden). Nicht alle diese Befehle werden in allen Dokumentenklassen definiert. Die Klasse `article` kennt keinen `\chapter`-Befehl und die Klasse `letter` unterstützt überhaupt keine Gliederungsbefehle. Darüber hinaus können Pakete weitere Gliederungsbefehle für zusätzliche Ebenen oder Varianten der vorhandenen Ebenen definieren.

Im Allgemeinen führen die Gliederungsbefehle automatisch eine oder mehrere der folgenden Aktionen aus:

- Erzeugen der Gliederungsnummer entsprechend der aktuellen hierarchischen Ebene.
- Speichern der Überschrift als Eintrag für das Inhaltsverzeichnis (in einer `.toc`-Datei).
- Speichern der Überschrift für die (mögliche) Verwendung in lebenden Kolummentiteln.
- Formatieren der Überschrift.

Alle Gliederungsbefehle, die Überschriften erzeugen, haben eine einheitliche Syntax, die hier anhand des Befehls `\section` dargestellt wird:

<code>\section*{titel}</code> <code>\section[toc-eintrag]{titel}</code>

Die Sternform (z.B. `\section*{...}`) unterdrückt die Nummerierung von Überschriften und erzeugt keinen Eintrag im Inhaltsverzeichnis oder im Kolummentitel. In der zweiten Form wird das optionale Argument `toc-eintrag` verwendet, wenn der Text für Inhaltsverzeichnis und Kolummentitel von der gedruckten Überschrift abweicht. Bei dieser Variante richtet sich die Nummerierung nach dem aktuellen Wert des Zählers `secnumdepth`, der im nächsten Abschnitt ausführlicher besprochen wird.

Probleme beim
manuellen
Formatieren

Wenn man versucht T_EX mit dem „~“-Symbol oder dem Befehl `\` anzuweisen, wie eine bestimmte Überschrift über mehrere Zeilen umbrochen werden soll, können dabei Nebenwirkungen auftreten, wenn das Inhaltsverzeichnis formatiert oder der lebende Kolumnentitel erzeugt wird. In diesem Falle ist die einfachste Lösung, die Überschrift ohne Formatierungsbefehle im optionalen Argument des Gliederungsbefehls zu wiederholen.

Der verbleibende Teil dieses Abschnitts befasst sich dann mit der Frage, in welcher Weise das Erscheinungsbild von Überschriften verändert werden kann. Es wird z.B. gezeigt, wie man einen Befehl, etwa `\section`, mit der oben beschriebenen Syntax so definiert, dass er, falls gewünscht, einen Eintrag im Inhaltsverzeichnis erzeugt, gleichzeitig eine breite Linie über die Überschrift setzt oder *kursiv* gesetzte Zeichen normaler Größe anstatt einer größeren **fetten** Schrift verwendet.

Zunächst zeigen einige Beispiele, wie die Nummerierung von Überschriften verändert werden kann. Weitere Beispiele demonstrieren dann, wie Informationen zu Überschriften ins Inhaltsverzeichnis eingetragen werden. Und schließlich werden Änderungen des allgemeinen Überschriftenlayouts diskutiert, sowie die Möglichkeiten, diese mit L^AT_EX zu definieren.

2.2.1 Nummerieren von Überschriften

Beim Nummerieren verwendet L^AT_EX einen eigenen Zähler für jede Gliederungsebene und stellt daraus die Nummer für die jeweilige Überschrift zusammen.

Eine der am häufigsten gewünschten Änderungen bei der Nummerierung von Überschriften bezieht sich wohl auf die Gliederungsebene, bis zu welcher eine Nummerierung erfolgen soll. Die entsprechende Ebene wird durch einen Zähler namens `secnumdepth` festgelegt. Je höher der Wert des Zählers ist, desto mehr Gliederungsebenen werden nummeriert. In manchen Dokumenten

Überschriften nicht
nummerieren

werden z.B. gar keine Überschriften nummeriert. Anstatt in diesem Falle immer die Sternform der Gliederungsbefehle zu verwenden, ist es viel einfacher, den Zähler `secnumdepth` in der Präambel auf `-2` zu setzen. Dieses Verfahren hat den Vorteil, dass weiterhin ein Eintrag im Inhaltsverzeichnis erfolgen kann und dass die Argumente der Gliederungsbefehle in lebenden Kolumnentiteln erscheinen können. Wie bereits zuvor erwähnt, werden diese Funktionen bei der Sternform unterdrückt.

Alle Überschriften
nummerieren

Um alle Überschriften bis herunter zu `\subparagraph`, oder wie auch immer die tiefste Gliederungsebene einer Klasse heißen mag, zu nummerieren, sollte es normalerweise reichen, den Zähler auf einen genügend hohen Wert zu setzen (zum Beispiel sollte im Allgemeinen eine Deklaration wie `\setcounter{secnumdepth}{10}` ausreichend sein).

Der Befehl `\addtocounter` schließlich, eröffnet eine einfache Methode, einige Ebenen mehr oder weniger zu nummerieren, ungeachtet der Ebenennummer der entsprechenden Gliederungsbefehle. Wenn man zum Beispiel eine weitere nummerierte Ebene benötigt, dann kann man einfach `\addtocounter{secnumdepth}{1}` in die Dokumentenpräambel schreiben, ohne dass man den richtigen Wert nachschlagen muss.

Zu jedem Gliederungsbefehl gehört ein eigener, (normalerweise) gleichnamiger Zähler (so hat z.B. der Befehl `\subsection` den Zähler `subsection`). Dieser Zähler enthält die aktuelle (formatierte) Nummer des entsprechenden Gliederungsbefehls. So repräsentieren zum Beispiel in der Klasse `report` die

Befehle `\chapter`, `\section`, `\subsection`, usw. die hierarchische Struktur des Dokumentes, während ein Zähler wie `subsection` festhält, wie häufig `\subsection` im aktuellen Abschnitt (`\section`) verwendet wurde. Normalerweise gilt: Wird ein Zähler einer gegebenen hierarchischen Ebene um eins erhöht, so wird der Zähler der jeweils nachgeordneten Ebene (d.h. mit der nächsthöheren Ebenennummer) wieder auf null gesetzt. In der Klassendatei `report` befinden sich z.B. folgende Deklarationen:

```
\newcounter{part}           % (-1) Teile
\newcounter{chapter}        % (0) Kapitel
\newcounter{section}[chapter] % (1) Abschnitte
\newcounter{subsection}[section] % (2) Unterabschnitte
\newcounter{subsubsection}[subsection] % (3) ...
\newcounter{paragraph}[subsubsection] % (4) weitere Ebenen
\newcounter{subparagraph}[paragraph] % (5) ...
```

Diese Befehle legen das Verhältnis der verschiedenen Zähler zueinander fest. Der Zähler der Ebene eins (`section`, Abschnitt) wird zurückgesetzt, wenn der Zähler der Ebene null (`chapter`, Kapitel) hochgesetzt wird. Entsprechend wird der Zähler der Ebene zwei (`subsection`, Unterabschnitt) zurückgesetzt, wenn der Zähler von Ebene eins (`section`, Abschnitt) hochgesetzt wird.

Das gleiche Verfahren wird bis hinunter zu dem Befehl `\subparagraph` angewendet. Dabei ist zu beachten, dass der Zähler `part` bei Standardklassen völlig von den anderen Zählern getrennt ist und keine Auswirkungen auf Gliederungsbefehle niedrigerer Ebenen hat. Dadurch werden Kapitel (`\chapter`) in den Klassen `book` oder `report` oder Abschnitte (`\section`) in der Klasse `article` fortlaufend nummeriert, selbst wenn zwischen ihnen ein `\part`-Befehl vorkommt. Dies lässt sich leicht ändern, indem man die Deklaration für den entsprechenden Zähler durch folgenden Ausdruck ersetzt:

```
\newcounter{chapter}[part]
```

Das Verhalten eines bereits existierenden Zählers kann analog mit dem Befehl `\@addtoreset` verändert werden (siehe Anhang A.1.4), z.B.:

```
\@addtoreset{chapter}{part}
```

Da diese Anweisung ein `@`-Zeichen enthält, kann sie nur innerhalb einer Paketdatei oder in der Dokumentenpräambel zwischen den Befehlen `\makeatletter` und `\makeatother` gegeben werden (vgl. die Erklärung auf Seite 875).

Zu jedem Zähler in $\text{\LaTeX}$, einschließlich der Abschnittszähler, gibt es einen Befehl aus dem Namen des Zählers und dem Präfix `\the`, der den Wert des jeweiligen Zählers in formatierter Fassung ausgibt. Im Falle der Gliederungsbefehle wird er wie in der nachfolgenden Definition dazu verwendet, die vollständige Nummerierung zu erzeugen:

```
\renewcommand\thechapter{\arabic{chapter}}
\renewcommand\thesection{\thechapter.\arabic{section}}
\renewcommand\thesubsection{\thesection.\arabic{subsection}}
```

In diesem Beispiel gibt `\thesubsection` den Befehl `\thesection`, gefolgt von einem Punkt und dem Zähler `subsection` in arabischen Ziffern aus. Diese Art der aufeinander aufbauenden Definition vereinfacht die Änderung der Zählerdarstellung, da Änderungen lediglich an einem Ort vorgenommen werden müssen. Wenn zum Beispiel Abschnitte mit Großbuchstaben nummeriert werden sollen, muss nur der Befehl `\thesection` neu definiert werden:

A Anders als sonst

A.1 Auch hier

Aufgrund der Standardeinstellungen verändert sich nicht nur die Abschnittsnummerierung; auch Gliederungsbefehle niedrigerer Ebenen übernehmen dieses Format der Abschnittsnummer.

```
\renewcommand\thesection{\Alph{section}}
```

```
\section{Anders als sonst}
```

```
\subsection{Auch hier}
```

Aufgrund der Standardeinstellungen verändert sich nicht nur die Abschnittsnummerierung; auch Gliederungsbefehle niedrigerer Ebenen übernehmen dieses Format der Abschnittsnummer.

Bsp.
2-2-1

Indem man also die Befehle ändert, die festlegen, wie ein Zähler dargestellt wird, verändert man die Nummer, die von dem entsprechenden Gliederungsbefehl generiert wird. Das Format lässt sich auf diese Weise jedoch nicht beliebig ändern. Angenommen, man möchte die Nummerierung einer Abschnittsüberschrift eingerahmt darstellen, dann könnte man als direkten Lösungsansatz, ausgehend von dem oben gezeigten Beispiel, `\thesubsection` neu definieren, z.B.:

```
\renewcommand\thesubsection
{\fbox{\thesection.\arabic{subsection}}}
```

Dieser Weg ist jedoch nicht korrekt, wie man sieht, wenn man auf einen solchen Abschnitt verweisen will.

```
\renewcommand\thesubsection
{\fbox{\thesection.\arabic{subsection}}}
```

```
\setcounter{section}{3}
```

```
\subsection{Ein Fehler}\label{falsch}
```

Ein Querverweis in diesem Format erzeugt ein lustiges Ergebnis, wie man mit einem Blick auf Unterabschnitt~\ref{falsch} sieht. Man erhält einen eingerahmten Verweis.

Bsp.
2-2-2

3.1 Ein Fehler

Ein Querverweis in diesem Format erzeugt ein lustiges Ergebnis, wie man mit einem Blick auf Unterabschnitt **3.1** sieht. Man erhält einen eingerahmten Verweis.

Mit anderen Worten: Die Befehle, die das Erscheinungsbild der Zähler bestimmen, werden von L^AT_EX auch für Querverweise herangezogen (zu den Befehlen `\label`, `\ref` siehe Abschnitt 2.4). Sie dürfen also nur leicht verändert werden, damit sie auch im Zusammenhang mit dem Befehl `\ref` ein sinnvolles Ergebnis erzeugen. Um Gliederungsnummern zu umrahmen ohne dabei die Ausgabe von `\ref` zu beeinträchtigen, muss der interne L^AT_EX-Befehl `@secntformat` neu definiert werden. Dieser bestimmt das Format des Zählers in einer Abschnittsüberschrift. Die Standarddefinition von `@secntformat` setzt die `\the`-Entsprechung eines Abschnittszählers (d.h. im oben genannten Beispiel verwendet sie den Befehl `\thesection`), gefolgt

von einem festen horizontalen Abstand von 1 em. Um das Problem zu lösen, sollte also das oben gezeigte Beispiel folgendermaßen umformuliert werden:

1 So ist es richtig

Bsp.
2-2-3

Ein Querverweis auf einen Abschnitt erzeugt mithilfe dieser Definition einen korrekten Verweis auf Abschnitt 1.

```
\makeatletter
\renewcommand\@secntformat[1]{\fbox
{\csname the#1\endcsname}\hspace{0.5em}}
\makeatother
\section{So ist es richtig}\label{sec:OK}
Ein Querverweis auf einen Abschnitt
erzeugt mithilfe dieser Definition einen
korrekten Verweis auf Abschnitt~\ref{sec:OK}.
```

Durch den Befehl `\@secntformat` wird nun ausschließlich die Nummer in der Abschnittsüberschrift mit einem Rahmen versehen, so dass die Querverweise richtig dargestellt werden.¹ Gleichzeitig wurde der Abstand zwischen Rahmen und Text auf 0.5 em reduziert (anstatt der Standardeinstellung von 1 em). Die Definition von `\@secntformat` wirkt sich auf alle Überschriften aus, die mit dem Befehl `\@startsection` definiert werden. Dieser wird im nächsten Abschnitt erklärt. Wenn man also mit unterschiedlichen Definitionen von `\@secntformat` für unterschiedliche Überschriften arbeiten will, muss man jede Überschriftendefinition entsprechend anpassen.

2.2.2 Formatieren von Überschriften

L^AT_EX verfügt über den generischen Befehl `\@startsection`, mit dem sich eine Vielzahl von Überschriftenlayouts definieren lassen. Um einen Gliederungsbefehl zu erstellen oder zu verändern, sollte man zunächst prüfen, ob dies mithilfe von `\@startsection` möglich ist. Wenn das gewünschte Layout nicht auf diesem Wege erzielt werden kann, lassen sich mit `\secdef` Abschnittsformate mit beliebigem Layout erzeugen.

Man kann Überschriften grob in zwei große Gruppen unterteilen: abgesetzte und eingebettete Überschriften. Abgesetzte Überschriften sind durch vertikale Zwischenräume vom vorhergehenden und vom nachfolgenden Text getrennt – die meisten Überschriften in diesem Buch haben diese Form.

Eingebettete Überschriften sind ebenfalls durch einen vertikalen Zwischenraum vom vorhergehenden Text getrennt, der nachfolgende Text beginnt jedoch in der gleichen Zeile wie die Überschrift selbst und ist nur durch einen horizontalen Zwischenraum abgetrennt.

Eingebettete Überschriften. Dieses Beispiel zeigt, wie eine eingebettete Überschrift aussieht. Der Absatztext, der auf die Überschrift folgt, setzt die gleiche Zeile fort.

Bsp.
2-2-4

```
\paragraph{Eingebettete Überschriften.}
Dieses Beispiel zeigt, wie eine
eingebettete Überschrift aussieht.
Der Absatztext, der auf die Überschrift
folgt, setzt die gleiche Zeile fort.
```

¹Der Befehl `\@secntformat` verwendet als Argument die Kennung der Gliederungsebene. Diese wird an das Präfix `the` angefügt, und daraus erzeugen `\csname` und `\endcsname`, die zum Generieren von Befehlen dienen, den benötigten Darstellungsbefehl. In dem gezeigten Beispiel wird der Befehl `\@secntformat` mit dem Argument `section` aufgerufen und so der Ersetzungstext `\fbox{\csname thesection\endcsname}\hspace{0.5em}` generiert. Weitere Informationen über den Befehl `\csname` bietet das *T_EXbook* [87].

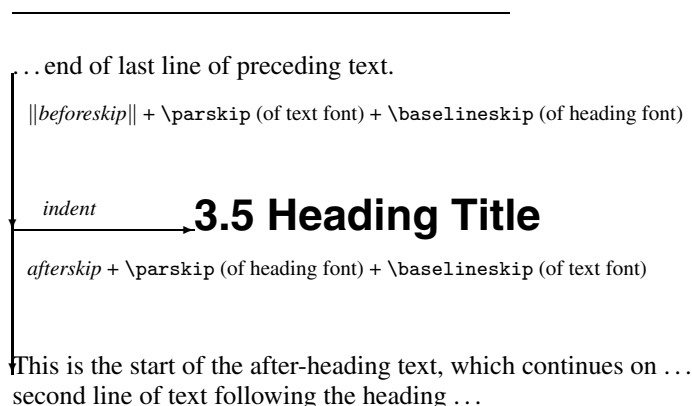


Abbildung 2.1: Layout einer abgesetzten Überschrift (mit layouts erstellt)

Mit dem generischen Befehl `\@startsection` können beide Überschriftenarten definiert werden. Er hat folgende Syntax:

```
\@startsection{name}{ebene}{einzug}{vor-abst}{nach-abst}{layout}
```

name Der Name verweist auf den Überschriftenzähler¹ und ist Bestandteil des Befehlsnamens zur Generierung von lebenden Kolumnentiteln (siehe Seite 227). Wenn der Name des Zählers zum Beispiel *name* ist, dann wäre `\thename` der Befehl, der die aktuelle Nummer der Überschrift generiert und `\namemark` wäre der Befehl für den lebenden Kolumnentitel. Meistens ist *name* der Name des entsprechenden Gliederungsbefehls ohne den vorangestellten Backslash – das ist jedoch nicht zwingend erforderlich.

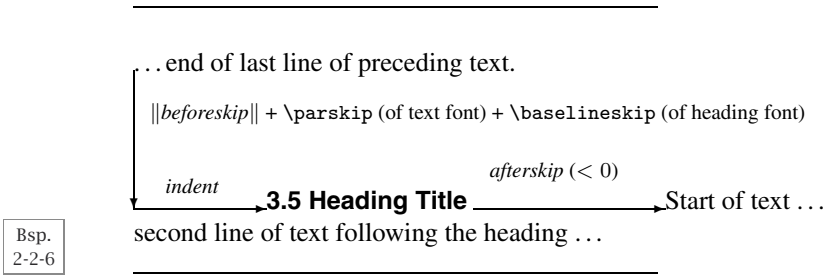
ebene Dieses Argument enthält die Ebenennummer des Gliederungsbefehls. Anhand dieser Zahl wird entschieden, ob die Überschrift nummeriert wird (wenn die Zahl kleiner oder gleich `secnumdepth` ist, siehe Abschnitt 2.2.1 auf Seite 26) und ob sie im Inhaltsverzeichnis erscheint (wenn der Wert kleiner oder gleich `tocdepth` ist, siehe Abschnitt 2.3.2 auf Seite 52). Sie sollte daher die Position in der Hierarchie der Gliederungsbefehle widerspiegeln, wobei der höchste Gliederungsbefehl der Ebene null entspricht.²

einzug Der Einzug der Überschrift gemessen vom linken Rand; bei einem negativen Wert beginnt die Überschrift im äußeren Rand. Ein positiver Wert rückt alle Zeilen der Überschrift um diesen Abstand ein.

vor-abst Der absolute Wert dieses Argumentes legt den vertikalen Abstand vor der Überschrift fest. Bei einem negativen Wert wird der Einzug des nachfolgenden Absatzes unterdrückt. Diese Größe ist elastisch, d.h. sie kann eine Dehn- und Schrumpfkomponekte enthalten. Es ist zu beachten,

¹Dieser Zähler muss existieren; er wird nicht automatisch definiert.

²Tatsächlich hat der Befehl `\part` in den Klassen `book` und `report` die Ebenennummer -1 (siehe Tabelle 2.1).



Bsp.
2-2-6

Abbildung 2.2: Layout einer eingebetteten Überschrift (mit layouts erstellt)

dass $\text{\LaTeX}$ vor der Überschrift einen neuen Absatz beginnt, so dass der Wert von $\backslash parskip$ zu diesem Abstand hinzuaddiert wird.

nach-abst Dieses Argument gibt den Abstand nach einer Überschrift an. Bei einer abgesetzten Überschrift ist es der vertikale Abstand, bei einer eingebetteten Überschrift der horizontale Abstand. Das Vorzeichen von *nach-abst* legt fest, ob eine abgesetzte (*nach-abst* > 0) oder eine eingebettete Überschrift (*nach-abst* ≤ 0) erzeugt wird. Im ersten Fall beginnt ein neuer Absatz, so dass der Wert von $\backslash parskip$ zu diesem Abstand hinzuaddiert wird. Ein unangenehmer Nebeneffekt dieser mehrfachen Verwendung der Werte ist, dass man mithilfe des Befehls $\backslash @startsection$ keine abgesetzte Überschrift erzeugen kann, die zum nachfolgenden Text einen kleineren Abstand als $\backslash parskip$ hat. Wenn man versucht, einen positiven $\backslash parskip$ -Wert durch einen negativen *nach-abst* zu kompensieren, wird aus der abgesetzten eine eingebettete Überschrift.

layout Dieses Argument bestimmt das Textlayout der Überschrift. Es kann beliebige Anweisungen zur Textformatierung enthalten, z.B. $\backslash bfseries$, $\backslash Large$ oder $\backslash raggedright$ (vgl. die nachfolgenden Beispiele).

Die Abbildungen 2.1 und 2.2 zeigen durch das Paket *layouts* generierte graphische Darstellungen dieser Argumente (Texte auf Englisch) für abgesetzte bzw. für eingebettete Überschriften.

Im Folgenden wird anhand praktischer Beispiele gezeigt, wie man mithilfe des Befehls $\backslash @startsection$ und seiner Argumente neue Gliederungsbefehle definieren kann. Angenommen, man möchte den Befehl $\backslash subsection$ der Klasse *article* so verändern, dass man etwa folgendes Ergebnis erhält:

*Eine Hand voll
Beispiele*

... etwas Text oberhalb.

*% neue Definition von \subsection siehe unten
\setcounter{section}{4}% vorhergehende Abschnitte
% simulieren*

4.1 Eine Abschnittsüberschrift

Die Überschrift ist in normaler Größe kursiv gesetzt und der Abstand zum vorangehenden Text beträgt genau eine Zeile. Der Abstand zum nachfolgenden Text beträgt eine halbe Zeile und der Text ist nicht eingerückt.

Bsp.
2-2-7

$\backslash dots\backslash$ etwas Text oberhalb.
 $\backslash subsection\{Eine\ Abschnittsüberschrift\}$
Die Überschrift ist in normaler Größe kursiv gesetzt und der Abstand zum vorangehenden Text beträgt genau eine Zeile. Der Abstand zum nachfolgenden Text beträgt eine halbe Zeile und der Text ist nicht eingerückt.

Dazu muss `\subsection` wie folgt definiert werden:

```
\makeatletter
\renewcommand\subsection{\@startsection
{subsection}{2}{0mm}%           % name, ebene, einzug
{-\baselineskip}%             % vor-abst
{0.5\baselineskip}%           % nach-abst
{\normalfont\normalsize\itshape}}% % layout
\makeatother
```

Das erste Argument ist die Zeichenkette `subsection`. Es gibt an, dass der entsprechende Zähler für Gliederungsnummern verwendet wird. In der Gliederungshierarchie entspricht dies Ebene zwei. Das dritte Argument ist `0mm`, da die Überschrift am linken Rand beginnen soll. Der absolute Wert des vierten Argumentes (*vor-abst*) gibt an, dass ein Abstand von einer Zeile vor der Überschrift gelassen werden muss. Da der Wert negativ ist, wird der Einzug des nachfolgenden Absatzes unterdrückt. Der absolute Wert des fünften Argumentes (*nach-abst*) gibt an, dass nach der Überschrift ein Abstand von einer halben Zeile gelassen werden muss. Da der Wert positiv ist, wird eine abgesetzte Überschrift erzeugt. Schließlich wird, entsprechend dem sechsten Argument, die Überschrift kursiv gesetzt und in der gleichen Größe wie der übrige Text des Dokumentes.

Diese neue Definition ist eigentlich zu stark vereinfacht: Wie bereits erwähnt, wird der aktuelle Wert von `\parskip` jeweils zu dem absoluten Betrag von *vor-abst* und *nach-abst* hinzuaddiert. In Layouts, in denen dieser Wert nicht null ist, muss er also abgezogen werden, um den gewünschten Abstand zu erhalten. Ein weiteres Layout, das manchmal in Romanen verwendet wird, hat folgende Definition:

```
\makeatletter
\renewcommand\section{\@startsection
{section}{1}{1em}%           % name, ebene, einzug
{\baselineskip}%           % vor-abst
{-\fontdimen2\font          % nach-abst
  plus -\fontdimen3\font minus -\fontdimen4\font }%
{\normalfont\normalsize\scshape}}% % layout
\makeatother
```

Diese Definition erzeugt eine eingebettete Überschrift in Kapitälchen. Die Definition des horizontalen Abstandes *nach-abst* nach der Überschrift verdient eine Erklärung: Es handelt sich dabei um den Wert des dehnbaren Wortzwischenraums, der vom aktuellen Zeichensatz abhängig ist. Der Wert ist negativ gewählt, um eine eingebettete Überschrift zu erzeugen. Der Befehl `\fontdimen` wird in Abschnitt 7.10.3 auf Seite 439 näher erläutert. Das Ergebnis ist im nächsten Beispiel dargestellt.

... etwas Text oberhalb.

DER MANN rannte von dem Truck weg.
Er sah, dass er verfolgt wurde, nämlich von

% neue Definition von \section siehe oben

```
\setcounter{secnumdepth}{-2}
```

```
\ldots\ etwas Text oberhalb.
```

```
\section{Der Mann} rannte von dem Truck weg.
```

```
Er sah, dass er verfolgt wurde, nämlich von
```

Bsp. 2-2-8

Für diese Art von Überschriften sollte natürlich die Nummerierung ausgeschaltet werden, indem man den Zähler `secnumdepth` auf den Wert `-2` setzt.

Welche Anweisungen kann man im `layout`-Argument des Befehls `\@startsection` verwenden, um das Überschriftenlayout zu ändern? Abgesehen von den Fontwechselanweisungen (siehe Kapitel 7), gibt es hier nur wenig Auswahl. Der Befehl `\centering` erzeugt eine zentrierte, abgesetzte Überschrift und die Deklaration `\raggedright` richtet den Text linksbündig aus. Ebenso lässt sich `\raggedleft` verwenden, kann aber zu seltsamen Ergebnissen führen. Man kann auch mit `\hrule`, `\medskip`, `\newpage` oder ähnlichen Befehlen arbeiten, um lokale Änderungen vorzunehmen. Das nächste Beispiel zeigt einige mögliche Varianten.

*Einfache
Layoutänderungen für
Überschriften*

1 Ein sehr langer Titel, der sich im Blocksatz über mehrere Zeilen erstreckt (Voreinstellung)

1.1 Unterabschnitt

Die Überschrift ist zentriert und kursiv.

1.2 Unterabschnitt

Die Überschrift ist linksbündig und in serifenloser Schrift gesetzt.

1.3 UNTERABSCHNITT

Die Überschrift ist rechtsbündig und in Großbuchstaben gesetzt.

1.4 Unterabschnitt

Eine Überschrift mit horizontaler Linie oberhalb.

Bsp.
2-2-9

```
\makeatletter
\newcommand\Csub{\@startsection{subsection}%
  {2}{0pt}{-\baselineskip}{.2\baselineskip}%
  {\centering\itshape}}
\newcommand\Lsub{\@startsection{subsection}%
  {2}{0pt}{-\baselineskip}{.2\baselineskip}%
  {\raggedright\sffamily}}
\newcommand\Rsub{\@startsection{subsection}%
  {2}{0pt}{-\baselineskip}{.2\baselineskip}%
  {\raggedleft\MakeUppercase}}
\newcommand\Hsub{\@startsection{subsection}%
  {2}{0pt}{-\baselineskip}{.2\baselineskip}%
  {\hrule\medskip\itshape}}
\makeatother

\section{Ein sehr langer Titel, der sich
        im Blocksatz über mehrere Zeilen
        erstreckt (Voreinstellung)}
\Csub{Unterabschnitt}
Die Überschrift ist zentriert und kursiv.
\Lsub{Unterabschnitt}
Die Überschrift ist linksbündig und in
serifenloser Schrift gesetzt.
\Rsub{Unterabschnitt}
Die Überschrift ist rechtsbündig und in
Großbuchstaben gesetzt.
\Hsub{Unterabschnitt}
Eine Überschrift mit horizontaler Linie
oberhalb.
```

In den $\text{\LaTeX}$ -Standardklassen werden Wörter in langen Überschriften im Blocksatz ausgerichtet und können, falls erforderlich, getrennt werden (siehe voriges Beispiel). Wenn das nicht erwünscht ist, kann der Randausgleich mit der Anweisung `\raggedright` im `layout`-Argument des `\@startsection`-Befehls ausgeschaltet werden. Wenn mit `\` manuelle Zeilenumbrüche erfolgen, muss der Text der Überschrift im optionalen Argument noch einmal ohne Formatierungsanweisungen wiederholt werden. Andernfalls werden die Zeilenumbrüche auch im Inhaltsverzeichnis gesetzt.

*Silbentrennung
und Zeilenumbrüche
in Überschriften*

1 Ein sehr langer Titel, der sich im Flattersatz über mehrere Zeilen erstreckt

```
\makeatletter
\renewcommand\section{\@startsection{section}%
{1}{0pt}{-\baselineskip}{.2\baselineskip}%
{\normalfont\Large\bfseries\raggedright}}
\makeatother
\section{Ein sehr langer Titel,
        der sich im Flattersatz über
        mehrere Zeilen erstreckt}
```

Bsp.
2-2-10

*Einzug nach
einer Überschrift*

Abschließend einige Worte über den Einzug des ersten Absatzes nach einer abgesetzten Überschrift. Die Standardklassen von L^AT_EX folgen der angloamerikanischen Tradition und unterdrücken diese Einzüge. Alle Absätze, die unmittelbar auf eine abgesetzte Überschrift folgen, können mithilfe des Paketes `indentfirst` (David Carlisle) mit einem Einzug versehen werden.

*Komplexe
Überschriftenlayouts*

In den L^AT_EX-Standardklassen erzeugen die Gliederungsbefehle der höchsten Ebenen `\part` und `\chapter` ihre Überschriften ohne `\@startsection`, da deren Layout mit diesem Befehl nicht erzeugt werden kann. Ebenso möchte man vielleicht selbst Gliederungsbefehle ohne Einschränkungen konstruieren. Dabei sind jedoch einige Konventionen zu beachten, damit L^AT_EX beim Ausführen der Befehle alle zum Setzen erforderlichen Schritte vornehmen kann.

Der Befehl `\secdef` bietet beim Definieren solcher Befehle eine einfach zu handhabende Schnittstelle zu den drei möglichen Formen von Abschnittsüberschriften, wie hier anhand des Befehls `\headcmd` dargestellt wird. Die Definition

```
\newcommand\headcmd{\secdef\cmda\cmdb}
```

bewirkt folgende Schritte:

<code>\headcmd{titel}</code>	aufrufen von <code>\cmda[titel]{titel}</code>
<code>\headcmd[toc-Eintrag]{titel}</code>	aufrufen von <code>\cmda[toc-Eintrag]{titel}</code>
<code>\headcmd*{titel}</code>	aufrufen von <code>\cmdb{titel}</code>

Dazu müssen die Befehle `\headcmd`, `\cmda` oder `\cmdb` definiert bzw. umdefiniert¹ werden. `\cmda` besitzt ein optionales Argument, welches den Text für das Inhaltsverzeichnis (`.toc`-Datei) enthält, während das zweite (obligatorische) Argument, genau wie das einzige Argument von `\cmdb`, den zu setzenden Überschriftentext enthält. Die Definitionen müssen also die folgende Struktur aufweisen:

```
\newcommand\headcmd{ ... \secdef \cmda \cmdb }
\newcommand\cmda[2][default]{ ... }
\newcommand\cmdb[1]{ ... }
```

Ein Beispiel für dieses Verfahren zeigt die nachfolgende vereinfachte Variante des Befehls `\appendix`. Sie definiert den Befehl `\section` neu, um Anhangsüberschriften zu erzeugen (durch Aufrufen der Befehle `\Appendix` oder `\sAppendix` mittels `\secdef`). Zudem verändert sie die Darstellung des

¹Umdefiniert für den Fall, dass ein vorhandener Gliederungsbefehl wie `\part` in der Präambel des Dokumentes geändert wird.

Zählers `section` und setzt diesen auf null zurück. Der veränderte `\section`-Befehl beginnt außerdem eine neue Seite mit einem besonderen Seitenlayout (siehe Kapitel 4) und verhindert, dass Gleitobjekte am Kopf der Seite erscheinen. Außerdem wird der Einzug des ersten Absatzes in einem Abschnitt unterdrückt. Dazu wird der Low-Level-Befehl `\@afterheading` verwendet und die boolesche Variable `@afterindent` auf `false` gesetzt. Weiteren Aufschluss zu diesem Thema gibt die `\chapter`-Implementierung in den Standardklassen (Datei `classes.dtx`).

```
\makeatletter
\renewcommand\appendix{%
  \renewcommand\section{%           % \section wird neu definiert...
    \newpage                       %   neue Seite
    \thispagestyle{plain}%         %   Seitenzahlen unten
    \suppressfloats[t]%            %   keine Gleitobjekte oben
    \@afterindentfalse             %   kein Einzug
    \secdef\Appendix\sAppendix}%   %   rufe \Appendix oder \sAppendix auf
  \setcounter{section}{0}%         %   Abschnittsnummer zurücksetzen
  \renewcommand\thesection{\Alph{section}}%   und alphabetisch nummerieren
}
```

Die folgende Definition zeigt, wie `\Appendix` den Zähler `section` mithilfe des Befehls `\refstepcounter` hochsetzt. Letzterer setzt gleichzeitig alle untergeordneten Zähler zurück und definiert den „aktuellen Bezugswert“ für Querverweise (siehe Abschnitt 2.4). Mithilfe des Befehls `\addcontentsline` wird eine Zeile in die `.toc`-Datei geschrieben. Außerdem wird die Überschrift formatiert und durch Aufrufen von `\sectionmark` für lebende Kolumnentitel gespeichert. Der Befehl `\@afterheading` steuert den Einzug des nachfolgenden Absatzes.

```
\newcommand\Appendix[2][?]{%           % Komplexe Form:
  \refstepcounter{section}%             % Zähler erhöhen/ Label setzen
  \addcontentsline{toc}{appendix}%      % toc-Eintrag erzeugen
    {\protect\numberline
      {\appendixname~\thesection}#1}%
  {\raggedleft\large\bfseries \appendixname\ % Formatieren des Titels
    \thesection\par \centering#2\par}%    %   und der Nummer
  \sectionmark{#1}%                     % in Kolumnentitel einfügen
  \@afterheading                         % Steuerung des Einzugs
  \addvspace{\baselineskip}}            % nach-abst
```

Der Befehl `\sAppendix` (Sternform) führt lediglich die Formatierung aus.

```
\newcommand\sAppendix[1]{%             % Vereinfachte (Stern-) Form
  {\raggedleft\large\bfseries\appendixname\par
    \centering#1\par}%
  \@afterheading
  \addvspace{\baselineskip}}
\makeatother
```

Wenn man diese Definitionen verwendet, erhält man folgendes Ergebnis:

Anhang A Die Liste aller Befehle

Dann folgt der Text des ersten Absatzes im Anhang. Und weiterer Text des Anhangs.
Und weiterer Text des Anhangs.

```
% Beispiel benötigt die soeben eingeführten
% Befehle!
\appendix
\section{Die Liste aller Befehle}
Dann folgt der Text des ersten Absatzes im
Anhang. Und weiterer Text des Anhangs.
Und weiterer Text des Anhangs.
```

Bsp.
2-2-11

Man sollte nicht vergessen, dass das oben gezeigte Beispiel nur die vereinfachte Version des undefinierten Befehls `\section` ist. Unter anderem wurde der Zähler `secnumdepth`, der die Nummerierungstiefe angibt, dabei nicht berücksichtigt. Unter Umständen muss man auch speziellen Code für Mehrspaltenformatierung oder für ein- und zweiseitigen Ausdruck vorsehen.

2.2.3 Ändern von vorgegebenen Überschriften

Einige der Standardbefehle für Überschriften erzeugen vordefinierte Texte. Der Befehl `\chapter` erzeugt z.B. normalerweise die Zeichenfolge „Chapter“ vor dem Überschriftentext, der vom Anwender eingegeben wurde. Ebenso erzeugen einige Umgebungen vorgegebene Überschriften. Die Umgebung `abstract` fügt z.B. über dem Text des Anwenders das Wort „Abstract“ ein. Diese Zeichenketten werden in L^AT_EX über Befehle gesteuert (siehe Tabelle 2.2 auf der nächsten Seite), so dass sie auf einfache Weise durch andere bevorzugte Titel ersetzt werden können. Dieses Anpassen wird im folgenden Beispiel gezeigt, wo die Standardbezeichnung „Abstract“ der Dokumentenklasse `article` gegen das Wort „Zusammenfassung“ ausgetauscht wird.

Zusammenfassung

Dieses Buch beschreibt, wie sich das Erscheinungsbild von Dokumenten unter L^AT_EX ändern lässt.

```
\renewcommand\abstractname{Zusammenfassung}
\begin{abstract}
  Dieses Buch beschreibt, wie sich das
  Erscheinungsbild von Dokumenten unter
  LATEX{} ändern lässt.
\end{abstract}
```

Bsp.
2-2-12

Bei Verwendung des `babel`-Systems muss das Undefinieren nicht wie oben mit `\renewcommand`, sondern mithilfe des `\addto`-Befehls geschehen, siehe Beispiel 9-3-2 auf Seite 567.

Die Standardklassendateien von L^AT_EX erzeugen einige weitere Zeichenfolgen. Eine komplette Liste dieser Befehle sowie Erläuterungen zum `babel`-System, das diese Texte in über zwanzig Sprachen übersetzt, befindet sich in Abschnitt 9.1.3, insbesondere in Tabelle 9.2 auf Seite 561.

2.2.4 fncychap – Fertige Layouts für Kapitelüberschriften

Wer ohne großen Mehraufwand originelle Kapitelüberschriften einsetzen möchte, der kann das Paket `fncychap` von Ulf Lindgren nutzen. Es enthält sechs verschiedene Layouts für den Befehl `\chapter`, die sich jeweils mit einer der folgenden Paketoptionen aktivieren lassen: `Sonny`, `Lenny`, `Glenn`,

<i>Befehl</i>	<i>Standard</i>	<i>babel (Option ngerman)</i>
<code>\abstractname</code>	Abstract	Zusammenfassung
<code>\appendixname</code>	Appendix	Anhang
<code>\bibname</code>	Bibliography	Literaturverzeichnis
<code>\chaptername</code>	Chapter	Kapitel
<code>\contentsname</code>	Contents	Inhaltsverzeichnis
<code>\indexname</code>	Index	Index
<code>\listfigurename</code>	List of Figures	Abbildungsverzeichnis
<code>\listtablename</code>	List of Tables	Tabellenverzeichnis
<code>\partname</code>	Part	Teil
<code>\refname</code>	References	Literatur

Tabelle 2.2: Sprachspezifische Texte für Überschriften

Conny, Rejne oder Bjarne. Da dieses Paket den Befehl `\chapter` verändert, kann es nur in Dokumentenklassen eingesetzt werden, die diesen Befehl unterstützen (z.B. mit `report` und `book`, nicht aber mit der Klasse `article` und ihren Derivaten). Im Beispiel werden hier die Ergebnisse der Option `Lenny` dargestellt.



Bsp.
2-2-13

Ein Paket-Test

```
\usepackage[Lenny]{fncychap}
\chapter{Ein Paket-Test}
```

Das Paket bietet auch verschiedene Befehle, mit deren Hilfe die Layouts unterschiedlich abgewandelt werden können. Außerdem enthält es eine kurze Anleitung zum Erstellen eigener Layouts.

2.2.5 quotchap – Mottos für Kapitel

Das Paket `quotchap` von Karsten Tinnfeld stellt eine weitere Möglichkeit dar, Kapitelüberschriften ansprechend zu gestalten. Mit seiner Hilfe kann der Anwender Zitate angeben, die dann oben links im Bereich des Titels für das Kapitel erscheinen.

Ein oder mehrere Zitate für das nächste Kapitel werden in der Umgebung `savequote` vorgehalten. Die Breite des Zitates kann als optionales Argument angegeben werden, wobei der Standardwert 10cm beträgt. Jedes Zitat sollte mit dem Befehl `\qauthor` enden, das die Quelle angibt. Es ist aber auch möglich, manuell ein eigenes Format festzulegen.

Das Paket erzeugt das folgende Standardlayout: Die Zitate sind linksbündig gesetzt, gefolgt von einem vertikalen Abstand, dessen Wert in dem Befehl `\chapterheadstartvskip` gespeichert ist. Danach folgt eine sehr große Kapitelnummer, die rechtsbündig und in Graustufen (60%) gesetzt

ist, gefolgt von dem Text der Kapitelüberschrift, der ebenfalls rechtsbündig erscheint. Nach einem weiteren vertikalen Abstand, der mit dem Befehl `\chapterheadendvskip` festgelegt wird, beginnt der erste Absatz des Kapitels ohne Einzug.

Durch Angabe der Paketoption `nogrey` kann die Nummer in schwarz gesetzt werden. Andere Optionen erlauben es, anstelle der Standardschrift Adobe Bookman einen der frei erhältlichen PostScript-Fonts einzusetzen, wie z.B. Bitstream Charter BT (Option `charter`) oder Adobe Times (Option `times`). Stattdessen kann man auch den Befehl `\chapnumfont` umdefinieren, der den Font für die Kapitelnummer bestimmt. Und schließlich kann man den Font der Kapitelüberschrift beeinflussen, indem man den Befehl `\sectfont`, wie im Beispiel gezeigt, umdefiniert.

Zusammen mit den Änderungsmöglichkeiten durch Umdefinieren der Befehle `\chapterheadstartvskip` und `\chapterheadendvskip` lassen sich somit eine Reihe interessanter Layouts erzeugen. Das folgende Beispiel platziert das Zitat durch einen negativen vertikalen Abstand auf gleicher Höhe mit der Kapitelnummer (in Avantgarde) und setzt den Titel und das Zitat in Helvetica.

Kekse! Ich will Kekse!

Das Krümelmonster

1

Ein Paket-Test

```
\usepackage[avantgarde]{quotchap}
\renewcommand\chapterheadstartvskip
{\vspace*{-5\baselineskip}}
% Helvetica für Titel und Zitat wählen
\usepackage{helvet}
\renewcommand\sectfont{\sffamily\bfseries}
\begin{savequote}[10pc]
\sffamily
Kekse! Ich will Kekse!
\qauthor{Das Krümelmonster}
\end{savequote}
\chapter{Ein Paket-Test}
```

Dieses Paket verändert die Kapitelüberschrift radikal.

Dieses Paket verändert die Kapitelüberschrift radikal.

Bsp.
2-2-14

Wer seine Kapitelüberschriften mit Zitaten versehen will, aber dabei die Layouts des Paketes `fncychap` bevorzugt, kann versuchen, beide miteinander zu kombinieren. Dazu muss `fncychap` *nach* dem Paket `quotchap` geladen werden. Die zuvor beschriebenen Einstellungsmöglichkeiten sind dann zwar nicht mehr verfügbar, aber `savequote` funktioniert weiterhin, auch wenn die Zitate stets in einer festen Position über der Überschrift erscheinen.

2.2.6 titlesec – Ein neuer Ansatz für Überschriften

Dieses Kapitel konzentrierte sich bisher auf die Werkzeuge und Mechanismen, die der L^AT_EX-Kern bereitstellt, um Überschriften zu definieren und zu verändern. Außerdem behandelte es einige Pakete, welche zusätzlich zu den Standardwerkzeugen weitere Funktionen, wie etwa vordefinierte Layouts, bereitstellen.

Das Paket `titlesec` von Javier Bezos verfolgt einen völlig anderen Ansatz, indem es die Gliederungsbefehle, die Überschriften erzeugen, vollständig neu

aufsetzt. Javier hebt mit seinem Paket einige der Einschränkungen des ursprünglichen Funktionsumfangs auf und bietet eine klare und generische Schnittstelle. Der Nachteil des Paketes liegt darin, dass es möglicherweise mit einigen Erweiterungen der Originalschnittstellen nicht kompatibel ist. Die Relevanz dieses Punktes hängt eindeutig von der zu bewältigenden Aufgabe ab. Sobald diese Schnittstelle weiter verbreitet ist, dürfte sich die Frage ohnehin erübrigen.

Das Paket unterstützt zwei Schnittstellen: Eine einfache für kleinere Veränderungen, die hauptsächlich durch Paketoptionen erzielt werden, und eine erweiterte für umfangreichere Modifikationen.

Die Basisschnittstelle

Mithilfe der Basisschnittstelle können die Fontcharakteristika aller Überschriften abgewandelt werden. Das geschieht durch Angabe einer oder mehrerer Optionen, Festlegen einer Schriftfamilie (`rm`, `sf`, `tt`), einer Schriftserie (`md`, `bf`) oder eines Schriftschnitts (`up`, `it`, `sl`, `sc`). Die Größe des Titels lässt sich durch eine der folgenden Optionen beeinflussen: `big` (gleiche Größe wie bei den L^AT_EX-Standardklassen), `tiny` (alle, außer Kapitelüberschriften, in normaler Textgröße) oder `medium` oder `small`, die Layouts zwischen diesen beiden Extremen erzeugen. Die Ausrichtung wird durch `raggedleft`, `center` oder `raggedright` gesteuert, während die vertikalen Abstände mithilfe der Option `compact` reduziert werden können.

Der Befehl `\titlelabel` dient zur Formatierung der Gliederungsnummer einer Überschrift. Dabei bezieht sich `\thetitle` auf den Darstellungsbefehl der aktuellen Gliederungsnummer, wie z.B. `\thesection` oder `\thesubsection`. Die Deklaration wird, wie im nächsten Beispiel ersichtlich, auf alle Überschriften angewendet.

1. Ein Abschnitt

1.1. Ein Unterabschnitt

1.1.1. Eine Passage

```
\usepackage[sf,bf,tiny,center]{titlesec}
\titlelabel{\thetitle.\enspace}
\section{Ein Abschnitt}
\subsection{Ein Unterabschnitt}
\subsubsection{Eine Passage}
```

Drei direkt aufeinander folgende Überschriften
sieht man nicht so häufig \ldots

Bsp.
2-2-15

Drei direkt aufeinander folgende Überschriften sieht man nicht so häufig ...

```
\titleformat*{befehl}{layout}
```

Die Basisschnittstelle verwendet eine weitere Deklaration, und zwar `\titleformat*`, die zwei Argumente besitzt. Das erste Argument (*befehl*) ist ein Gliederungsbefehl, der hier geändert werden soll. Das zweite Argument (*layout*) enthält die Formatierungsanweisung für die jeweilige Überschrift. Diese Deklaration bearbeitet individuelle Gliederungsbefehle und überschreibt dabei alle über Optionen des Paketes bestimmten Font- oder Ausrichtungsangaben, wie im folgenden Beispiel die Optionen `rm`, `it` und `raggedleft`. Der letzte Befehl im *layout*-Argument kann ein Befehl mit einem Argument sein, welcher dann den Text der Überschrift enthält. Im nächsten Beispiel wird diese Funktion verwendet, um den `\subsubsection`-Titel in Kapitälchen zu

setzen (auch wenn das in Kombination mit Ziffern normaler Größe ziemlich hässlich aussieht).

1 Ein Abschnitt

1.1 Ein Unterabschnitt

1.1.1 EINE PASSAGE

Drei direkt aufeinander folgende Überschriften sieht man nicht so häufig ...

```
\usepackage[rm,it,raggedleft,tiny,compact]{titlesec}
\titleformat*{\subsubsection}{\scshape\MakeLowercase}
\section{Ein Abschnitt}
\subsection{Ein Unterabschnitt}
\subsubsection{Eine Passage}
Drei direkt aufeinander folgende Überschriften sieht
man nicht so häufig \ldots
```

Bsp.
2-2-16

Die von `\part` erzeugten Überschriften werden nicht von den Einstellungen an der Basisschnittstelle beeinflusst. Diese lassen sich nur mithilfe der erweiterten Schnittstelle, die im Folgenden beschrieben wird, verändern.

Die erweiterte Schnittstelle

Die erweiterte Schnittstelle umfasst die zwei Befehle `\titleformat` und `\titlespacing`. Mit ihrer Hilfe lassen sich das „innere“ Format (d.h. Fonts, Label, Ausrichtung, ...) bzw. das „äußere“ Format (d.h. Abstände, Einzüge, usw.) festlegen. Dies entspricht der Arbeitsweise vieler Anwender, die entweder den einen oder den anderen Aspekt des Layouts verändern wollen.

```
\titleformat{befehl}[form]{layout}{label-layout}{abstand}
{davor-code}[danach-code]
```

Das erste Argument (*befehl*) ist der Name des Gliederungsbefehls (zum Beispiel `\section`), dessen Layout geändert werden soll. Anders als beim Befehl `\startsection` erwartet dieses Argument den ganzen Befehlsnamen, also mit führendem Backslash. Die anderen Argumente haben folgende Bedeutung:

form Die grundlegende Gestaltung der Überschrift. Es gibt eine Reihe vordefinierter Formen: `hang`, die Standardeinstellung, erzeugt ein hängendes Label (d.h. bei langen Überschriften werden die folgenden Textzeilen um die Breite der Gliederungsnummer eingerückt; wie `\section` in den Standardklassen), `display` setzt das Label und den Text der Überschrift in getrennte Zeilen (wie der Standardbefehl `\chapter`) und `runin` erzeugt eine eingebettete Überschrift (wie der Standardbefehl `\paragraph`).

Außerdem stehen folgende Formen zur Verfügung, zu denen es in Standard-L^AT_EX keine Entsprechungen gibt: `frame` ähnelt `display`, rahmt jedoch den Titel ein; `leftmargin` setzt den Titel in den linken Rand und `rightmargin` setzt ihn in den rechten Rand. Die letzten beiden Formen können Konflikte mit `\marginpar`-Befehlen erzeugen, d.h. es kann zu Überlappungen kommen.

Bei `block` handelt es sich um eine vielseitig verwendbare Form, welche die Überschrift als einen einzelnen Block setzt. Bei zentrierten Layouts sollte sie `hang` vorgezogen werden.

Bei `drop` und `wrap` umfließt der erste Absatz die Überschrift. Während `drop` die Überschrift in einer Box mit fester Breite setzt (*links*-Argument

von `\titlespacing`) und dabei falls nötig umbricht, reduziert `wrap` die Breite dieser Box automatisch auf die längste resultierende Überschriftszeile nach Umbruch.

Da die Schnittstelle (für Programmierer) erweiterbar ist, können neuere Versionen des Paketes zusätzliche Formen enthalten.

layout Deklarationen, die auf den ganzen Titel, also sowohl Label als auch Text, angewendet werden. Sie können nur im vertikalen Modus erlaubte Befehle enthalten, die ausgeführt werden, nachdem der Leerraum oberhalb der Überschrift erzeugt wurde. Befehle, die nur im horizontalen Modus erlaubt sind, sollten in den Argumenten *label-layout* oder *davor-code* verwendet werden.

label-layout Die Formatierung des Labels, also der Gliederungsnummer. Die Nummer selbst wird mit `\thesection` oder dem jeweils entsprechenden Befehl adressiert. Für die Definition von Kapitelüberschriften bietet das Paket den Befehl `\chaptertitlename`, der, je nach Lage der Überschrift im Dokument, `\chaptername` oder `\appendixname` erzeugt.

abstand Länge, die den Abstand zwischen Label und Überschriftstext festlegt. Je nach Inhalt des *form*-Argumentes kann es sich dabei um einen vertikalen oder horizontalen Abstand handeln. Im Falle der Form *frame* bestimmt es z.B. den Abstand zwischen dem Rahmen und dem Text der Überschrift.

davor-code Code, der direkt vor dem Text der Überschrift ausgeführt wird. Sein letzter Befehl kann ein Argument besitzen, das den Text der Überschrift aufnimmt. Dadurch werden auch komplexere Arten der Gestaltung möglich (siehe Beispiel 2-2-19).

danach-code Optionaler Code, der nach dem Formatieren des Textes der Überschrift ausgeführt wird (noch im Geltungsbereich der *format*-Deklarationen). Bei *hang*, *block* und *display* wird er im vertikalen Modus ausgeführt, bei *runin* im horizontalen Modus. Bei anderen Formen hat er keine Auswirkungen.

In der Sternform eines Gliederungsbefehls werden die Argumente *label-layout* und *abstand* ignoriert, da keine Nummerierung erfolgt.

Das nächste Beispiel zeigt eine eher altmodische, eingebettete Überschrift, für die nur das Format definiert wird und nicht die Abstände um die Überschrift. Letztere werden mit dem Befehl `\titlespacing` bearbeitet.

§ 1. DER TITEL. Die Überschrift ist durch einen Punkt und einen Abstand von einem Geviert vom Text getrennt.

```
\usepackage{titlesec}
\titleformat{\section}[runin]{\normalfont\scshape}
{\S\,\oldstylenums{\thesection}.}{.5em}{.}{. \quad}
\section{Der Titel}
Die Überschrift ist durch einen Punkt und einen
Abstand von einem Geviert vom Text getrennt.
```

Die `\section`-Überschriften in $\text{\LaTeX}$ sind standardmäßig nicht eingerückt. (Sie entsprechen normalerweise der *form* *hang*). Wenn man bei einer solchen Überschrift einen normalen Absatzeinzug bevorzugt, kann man vor dem `\S`-Zeichen `\indent` einfügen oder den Einzug über die `\titlespacing`-Deklaration festlegen, die im Folgenden erklärt wird.

Bsp.
2-2-17

```
\titlespacing*{befehl}{links}{oberhalb}{unterhalb}[rechts]
```

Die Sternform des Befehls unterdrückt den Absatzeinzug für den auf die Überschrift folgenden Absatz. Dies gilt nicht für Formen, bei denen Überschrift und Absatztext kombiniert werden, wie z.B. `runin` und `drop`. Das Argument *befehl* beinhaltet den Namen des Gliederungsbefehls, der geändert werden soll. Die weiteren Argumente haben folgende Bedeutung:

links Diese Länge gibt an, um wie viel der linke Rand einer Überschrift bei den Formen `block`, `display`, `hang` oder `frame` vergrößert wird. Im Zusammenhang mit einer `...margin-` oder `drop-`Form bestimmt sie die Breite der Überschrift, bei `wrap` die maximale Breite des Titels und bei `runin` den Einzug vor dem Titel (bei einem negativen Wert läuft die Überschrift in den linken Rand hinein).

oberhalb Diese Länge gibt den zusätzlichen vertikalen Abstand oberhalb der Überschrift an.

unterhalb Diese Länge gibt den Abstand zwischen der Überschrift und dem nachfolgenden Absatz an. Je nach verwendeter Form kann es sich dabei um einen vertikalen oder einen horizontalen Abstand handeln.

rechts Diese Länge gibt an, um wie viel der rechte Rand einer Überschrift bei den Formen `block`, `display`, `hang` oder `frame` vergrößert wird.

Die Argumente *oberhalb* und *unterhalb* erhalten normalerweise elastische Längen um ein flexibles Layout zu erzielen. Zur Vereinfachung der Deklaration kann man alternativ **f* angeben, wobei *f* ein dezimaler Faktor ist. Diese Angabe entspricht *f* ex mit einer gewissen Dehnbarkeit sowie einer geringen erlaubten Stauchung im Bereich *oberhalb* und mit einer noch geringeren Dehnbarkeit und ohne Stauchung im Bereich *unterhalb*.

...etwas Text oberhalb ...

SECTION 1 Ein Titel-Test

Dieser Text zeigt, dass der Absatz nicht eingezogen ist und der Titel beidseitig einen Rand von 1 pc aufweist.

```
\usepackage{titlesec}
\titleformat{\section}[frame]{\normalfont}
  {\footnotesize \enspace SECTION \thesecount}
  {\enspace}{6pt}{\large\bfseries\filcenter}
\titlespacing*{\section}{1pc}{*4}{*2.3}[1pc]
\ldots etwas Text oberhalb \ldots
\section{Ein Titel-Test}
```

Dieser Text zeigt, dass der Absatz nicht eingezogen ist und der Titel beidseitig einen Rand von 1\,pc aufweist.

Bsp.
2-2-18

Werkzeuge für den
Randausgleich

Das vorige Beispiel stellte den Befehl `\filcenter` vor. Außerdem gibt es die Befehle `\filleft`, `\filright` und `\fillast`, wobei letzterer einen Absatz im Blocksatz erzeugt, dessen letzte Zeile zentriert ist. Im Zusammenhang mit `\titleformat` sollten diese Befehle `\raggedleft` oder `\raggedright` vorgezogen werden, denn letztere heben die Einstellungen des `\titlespacing`-Befehls für *links* oder *rechts* auf. Man kann stattdessen auch `\filinner` oder `\filouter` verwenden, die je nach aktueller Seite zu `\filleft` oder `\filright` aufgelöst werden. Da jedoch T_EXs Algorithmus zum Erstellen von Seiten asynchron angelegt ist, werden sie nur bei Gliederungsbefehlen unterstützt, die eine neue Seite beginnen, wie z.B. `\chapter`.

in den meisten Layouts. Das Beispiel 2-2-21 auf Seite 46 zeigt eine Lösung für dieses Problem bei anderen Überschriften. Der Befehl `\wordsep` ist ebenfalls nützlich, um Abstände zu steuern. Er kann in Abhängigkeit von der Größe der Wortzwischenräume (inklusive dehnen und stauchen) für den aktuellen Font Längen festlegen.

Der Einzug für den ersten Absatz nach einer Überschrift kann über die Paketoptionen `indentafter` oder `noindentafter` global festgelegt werden. In diesem Falle werden die Einstellungen über `\titlespacing` (Sternform oder nicht) umgangen.

Einzug nach einer Überschrift

Der Standardabstand zwischen zwei aufeinander folgenden Überschriften ist als Wert des Argumentes *unterhalb* der ersten Überschrift festgelegt. Diese Einstellung lässt sich mit der Option `largestsep` ändern, welche das Maximum von *unterhalb* der ersten Überschrift und *oberhalb* der zweiten Überschrift als Abstand setzt.

Abstand zwischen mehreren Überschriften

L^AT_EX versucht einer Überschrift immer mindestens zwei Zeilen des nächsten Absatzes auf der gleichen Seite folgen zu lassen. Wenn das nicht möglich ist, wird die Überschrift erst auf der nächsten Seite gesetzt. Wenn zwei Zeilen nicht ausreichen, der kann mithilfe der Optionen `nobottomtitles` oder `nobottomtitles*` festlegen, dass die Überschrift immer dann auf die nächste Seite verschoben wird, wenn der verbleibende Platz auf der Seite kleiner ist als der aktuelle Wert von `\bottomtitlespace`. Der Standardwert ist ein Fünftel der Texthöhe (`.2\textheight`). Diese Variable muss mit `\renewcommand` und nicht mit `\setlength` eingestellt werden. Die Sternform der Option ist hier zu bevorzugen, da sie den verbleibenden Platz genauer berechnet. Dies gilt allerdings nicht für mit den Formen `drop`, `margin` oder `wrap` erstellte Überschriften, da diese in Zusammenhang mit der Sternform ungünstig positioniert werden können.

Überschriften am Fuß einer Seite

Bei den meisten Layouts steht die Gliederungsnummer entweder oberhalb oder links der Überschrift. Diese Positionierung kann nicht mithilfe des *label-layout*-Argumentes von `\titleformat` geändert werden. Stattdessen muss man sich die Tatsache zunutze machen, dass *davor-code* den Text der Überschrift aufnehmen kann. Im nächsten Beispiel hat der Befehl `\secformat` ein Argument, das die Formatierung der Überschrift (Text und Nummer) festlegt. Dieser Befehl wird dann im *davor-code*-Parameter von `\titleformat` aufgerufen. Dabei ist zu beachten, dass sich der Fontwechsel für die Nummer nur lokal auswirkt, wenn man sie in geschweifte Klammern setzt. Ohne die Klammern könnte sich die geänderte Schriftgröße unter gewissen Umständen auf die Abstände im Bereich des Titels auswirken.

Handhabung besonderer Layouts

Ein Titel
auf zwei Zeilen

1

```
\usepackage{titlesec}
\newcommand\secformat[1]{%
  \parbox[b]{.5\textwidth}{\filleft\bfseries #1}%
  \quad\rule[-12pt]{2pt}{70pt}\quad
  {\fontsize{60}{60}\selectfont\thesection}}
\titleformat{\section}[block]
  {\filleft\normalfont\sffamily}{}{0pt}{\secformat}
\titlespacing*{\section}{0pt}{*3}{*2}[1pc]
\section{Ein Titel\\ auf zwei Zeilen}
```

Hier erscheint die Gliederungsnummer rechts vom Text der Überschrift.

Bsp.
2-2-19

Hier erscheint die Gliederungsnummer rechts vom Text der Überschrift.

Mit dem gleichen Verfahren lassen sich auch noch andere Änderungen am Text der Überschrift herbeiführen. Ein Punkt nach der Überschrift könnte z.B. folgendermaßen erzeugt werden:

```
\newcommand\secformat[1]{#1.}
```

Er wird dann, wie im vorherigen Beispiel mit dem Befehl `\secformat` im Argument *davor-code* der `\titleformat`-Deklaration aufgerufen.

*Breite der Überschrift
messen*

Die Form `wrap` verfügt über die Möglichkeit, die Breite der einzelnen Textzeilen eines Titels zu messen. Sie gibt die Breite der längsten Zeile als `\titlewidth` aus. Diese Funktionalität kann auf die Formen `block`, `display` und `hang` ausgedehnt werden, indem man das Paket mit der Option `calwidth` lädt und `\titlewidth` nach Bedarf in den `\titleformat`-Argumenten verwendet.

Linien und Leitpunkte

Für Linien und Leitpunkte stellt das Paket den Befehl `\titlerule` bereit. Ohne Argumente erzeugt er eine Linie der Dicke `.4pt`, die sich über die ganze Spalte erstreckt. Dabei werden geänderte Randbereiche aus der `\titlespacing`-Deklaration berücksichtigt. Mithilfe eines optionalen Argumentes kann man die Dicke der erzeugten Linie festlegen. Mit der Sternform von `\titlerule` können anstelle von Linien Leitpunkte (d.h. wiederholte Elemente) verwendet werden. Dieser Befehl verwendet ein optionales Argument für die Breite (*breite*) und ein obligatorisches für den Text (*text*). Sofern das optionale Argument *breite* keine andere Breite angibt, wird der Inhalt von *text* wiederholt gesetzt, und zwar in Boxen mit seiner natürlichen Breite. Andernfalls behalten nur die erste und letzte Box ihre natürliche Breite, um auf beiden Seiten einen sauberen Blocksatz zu gewährleisten.

Mithilfe des Befehls `\titleline` können den `\titleformat`-Argumenten, die vertikale Inhalte erwarten, horizontale Inhalte hinzugefügt werden. Der Befehl verfügt über ein optionales Argument, das die Ausrichtung festlegt, und ein obligatorisches Argument, das das zu setzende Material enthält. Er erzeugt eine Box mit fester Breite und berücksichtigt durch die `\titlespacing`-Deklaration geänderte Randbereiche. Daher sollte das Material entweder elastische Längen enthalten, oder im optionalen Argument muss eine Ausrichtung angegeben sein (erlaubte Werte sind `l`, `r` und `c`).

Die Variante `\titleline*` setzt zunächst das Material aus seinem obligatorischen Argument in einer Box der Breite `\titlewidth` (daher müssen hier möglicherweise elastische Abstände eingefügt werden) und verwendet diese Box dann als Eingabe für `\titleline`, das heißt, sie richtet sie gemäß der Einstellungen im optionalen Argument aus. Gegebenenfalls muss hier die Option `calwidth` verwendet werden, damit `\titlewidth` einen vernünftigen Wert enthält.

Das nächste Beispiel ist etwas konstruiert und, obwohl für Übungszwecke hilfreich, für einen tatsächlichen Einsatz nicht zu empfehlen. Hier werden alle Werkzeuge gemeinsam eingesetzt:

```
\usepackage[noindentafter,calwidth]{titlesec}
\titleformat{\section}[display]                % form
{\filright\normalfont\bfseries\sffamily}      % layout
{\titleline[r]{Abschnitt \Huge\thesection}}    % label-layout
```



```

{1ex} % Abstand
{\titleline*[1]{\titlerule[1pt]}\vspace{1pt}} % davor-code
\titleline*[1]{\titlerule[2pt]}\vspace{2pt}}
[{\titleline*[1]{\titlerule*{\tiny\LaTeX}}}] % danach-code
\titlespacing{\section}{1pc}{*3}{*2}

```

Mit diesen Einstellungen erhält man folgende Ausgabe:

Linien und Leitpunkte

L^AT_EX L^AT_EX L^AT_EX L^AT_EX L^AT_EX L^AT_EX L^AT_EX L^AT_EX

Abschnitt 1

% Code wie oben
\section{Linien und
Leitpunkte}
Die letzte Instanz von
\verb=\titleline*= ist in
Klammern gesetzt. Ohne
diese Klammern würde das
zugehörige optionale
Argument frühzeitig das
äußere optionale Argument
von \verb=\titleformat=
beenden.

Bsp.
2-2-20

Die letzte Instanz von `\titleline*` ist in Klammern gesetzt. Ohne diese Klammern würde das zugehörige optionale Argument frühzeitig das äußere optionale Argument von `\titleformat` beenden.

Standard- $\text{\LaTeX}$ betrachtet den Freiraum vor einer Überschrift als gute Position für einen Seitenumbruch, sofern diese Überschrift nicht unmittelbar auf eine andere Überschrift folgt. Der so genannte „Penalty-Wert“, d.h. der Wert um den ein Umbruch an dieser Stelle erschwert ist, wird in dem internen Zähler `\@secpenalty` gespeichert. In vielen Klassen beträgt er `-300` (negative Werte geben einen Bonus für den Seitenumbruch an der jeweiligen Stelle). Da für die Überschriften aller Gliederungsebenen der gleiche Penalty-Wert gilt, macht es nur selten Sinn, diese Einstellung zu ändern. Mit `titlesec` hat man jedoch genauere Steuerungsmöglichkeiten: Falls ein Befehl `\namebreak` definiert wurde, also etwa `\sectionbreak` (wobei `\name` der Name eines Gliederungsbefehls ist), wird letzterer ausgeführt, anstatt die Standard-Penalty hinzuzufügen. So würde

Umbruch vor einer Überschrift

```
\newcommand\sectionbreak{\clearpage}
```

dazu führen, dass Abschnitte immer am Kopf der Seite erscheinen und zunächst alle aufgelaufenen Gleitobjekte gesetzt werden.

Manche Layouts erfordern, dass der Abstand oberhalb einer Überschrift entgegen der Standardeinstellungen auch am Kopf einer neuen Seite erhalten bleibt. Dies lässt sich mithilfe einer Definition wie der folgenden erreichen:

Abstände oberhalb einer Überschrift immer beibehalten

```
\newcommand\sectionbreak{\addpenalty{-300}\vspace*{0pt}}
```

Der Befehl `\addpenalty` weist auf einen (guten) Platz für einen Seitenumbruch hin. Danach folgt ein Element ohne vertikale Ausdehnung, das nicht verschwinden kann. Dadurch bleibt der Leerraum oberhalb der Überschrift erhalten, selbst wenn der Seitenumbruch an der Penalty erfolgt und die Überschrift damit am Kopf der Seite erscheint.

Bedingte Überschriftenlayouts

Bisher wurde besprochen, wie man mit `\titleformat` und `\titlespacing` unveränderliche Layouts für Gliederungsbefehle definieren kann. Mithilfe des Paketes `titlesec` lässt sich das Layout für gerade und ungerade Seiten auch fallweise ändern. Außerdem können spezielle Layouts für nicht nummerierte Überschriften, wie sie die Sternformen der Gliederungsbefehle erzeugen, definiert werden.

Dies wird durch eine Kombination von Schlüsselwörtern und entsprechenden Werten für das erste Argument von `\titleformat` und `\titlespacing` erreicht. Als Schlüsselwörter sind `name`, `page` (mit den Werten `odd` oder `even`) und `numberless` (mit den Werten `true` oder `false`) verfügbar. Die Syntax, die bisher verwendet wurde,

```
\titleformat{\section}{..}...
```

ist eigentlich nur eine Abkürzung der allgemeinen Form

```
\titleformat{name=\section}{..}...
```

Im Gegensatz zu den Abstandsbefehlen `\filinner` und `\filouter`, die nur bei Überschriften am Anfang einer neuen Seite verwendet werden können, lassen sich mit dem Schlüsselwort `page` Layouts definieren, die sich ohne Einschränkung auf die aktuelle Seite beziehen. Layouts für linke Seiten erfordern den Wert `even`, solche für rechte Seiten den Wert `odd`. Diese Einstellungen wirken sich jedoch nur auf Dokumente aus, die mit der Option `twoside` zweiseitig gesetzt werden. Andernfalls betrachtet L^AT_EX alle Seiten als rechte Seiten. Im folgenden Beispiel wird die Form `block` eingesetzt und die Überschrift je nach aktueller Seite nach links oder rechts verschoben. In ganz ähnlicher Weise kann man Überschriften einrichten, die mithilfe der Formen `leftmargin` und `rightmargin` im Seitenrand platziert werden.

1. Verso

Fülltext auf der aktuellen Seite. Mehr Text auf der Seite.

Fülltext auf dieser Seite.

2. Recto

Mehr Fülltext auf der aktuellen Seite.

```
\usepackage{titlesec}
\titleformat{name=\section,page=odd}[block]
    {\normalfont}{\thesection.}{6pt}
    {\bfseries\filleft}
\titleformat{name=\section,page=even}[block]
    {\normalfont}{\thesection.}{6pt}
    {\bfseries\filright}

\section{Verso} Fülltext auf der
aktuellen Seite. Mehr Text auf der Seite.
\newpage
Fülltext auf dieser Seite.
\section{Recto}
Mehr Fülltext auf der aktuellen Seite.
```

Bsp.
2-2-21

Auf die gleiche Weise kann man mit dem Schlüsselwort `numberless` angeben, dass eine bestimmte `\titleformat`- oder `\titlespacing`-Deklaration nur für Überschriften mit (bzw. ohne) Nummerierung gilt. Ohne das Schlüsselwort gilt eine Deklaration immer für beide Fälle. Daher überschreibt im nachfolgenden Beispiel die zweite Deklaration einen Teil der ersten. Die recht

unterschiedliche Gestaltung der beiden Fälle soll einen Eindruck von der Bandbreite der Möglichkeiten geben. Sie ist nicht als Orientierung für gutes Design gedacht. Es ist wichtig zu wissen, dass weder das *label-layout*- noch das *abstand*-Argument ignoriert wird, wenn man *numberless*, wie im Beispiel zu sehen, auf *true* setzt. Normalerweise würde man wahrscheinlich `{0pt}` als Werte verwenden.

1. Ein Titel

Fülltext auf der Seite. Mehr
Fülltext auf der Seite.

*** *Noch einer*

Fülltext auf der Seite.

```
\usepackage{titlesec}
\titleformat{name=\section}[block]
  {\normalfont}{\thesection.}{6pt}{\bfseries\filright}
\titleformat{name=\section,numberless=true}[block]
  {\normalfont}{***}{12pt}{\itshape\filcenter}
\section{Ein Titel}
Fülltext auf der Seite. Mehr Fülltext auf der Seite.
\section*{Noch einer}
Fülltext auf der Seite.
```

Bsp.
2-2-22

Verändern der Gliederungshierarchie

Mit den bisher behandelten Befehlen lassen sich das Format und die Abstände für existierende Gliederungsbefehle einstellen. Mithilfe der `\titleclass`-Deklaration kann man ganz neue Gliederungsbefehle definieren.

```
\titleclass{befehl}{klasse}
\titleclass{befehl}{klasse}[ebene-oberhalb]
\titleclass{befehl}[start-ebene]{klasse} (mit der Option loadonly)
```

Es gibt drei Klassen von Überschriften: Die Klasse `page` enthält Überschriften, die eine ganze Seite füllen (wie etwa `\part` in den $\text{\LaTeX}$ -Dokumentenklassen `report` und `book`). Die Klasse `top` enthält Überschriften, die eine neue Seite beginnen und dadurch am Kopf der Seite erscheinen; alle anderen Überschriften werden der Klasse `straight` zugeordnet.

Ohne optionales Argument verändert die Deklaration `\titleclass` einfach die Klasse für einen existierenden Gliederungs-*befehl*. So würde z.B.

```
\titleclass\section{top}
```

dazu führen, dass jeder Abschnitt auf einer neuen Seite beginnt.

Wird diese Deklaration jedoch mit dem optionalen Argument *ebene-oberhalb* verwendet, so führt man damit eine neue Gliederungsebene unterhalb von *ebene-oberhalb* ein. Alle Gliederungsbefehle, die auf dieser Ebene existieren, werden in der Hierarchie entsprechend eine Ebene heruntergestuft. So führt zum Beispiel

```
\titleclass\subchapter{straight}[\chapter]
```

die neue Gliederungsebene `\subchapter` zwischen `\chapter` und `\section` ein. Die Deklaration definiert weder ein Layout für diese Überschrift noch

initialisiert sie den erforderlichen Zähler. Das Layout muss mit einem zusätzlichen `\titleformat`- und `\titlespacing`-Befehl definiert werden. Die Zählerdarstellung für `\section` soll sicherlich ebenfalls aktualisiert werden:

```
\titleformat{\subchapter}{..}...
\titlespacing{\subchapter}{..}...
\newcounter{subchapter}
\renewcommand\thesubchapter{\thechapter.\arabic{subchapter}}
\renewcommand\thesection{\thesubchapter.\arabic{section}}
```

Die dritte Variante von `\titleclass` wird nur benötigt, wenn man eine vollständig neue Gliederungsstruktur entwerfen möchte, wie z.B. eine Dokumentenklasse, die nicht auf den Standardklassen basiert. In diesem Falle wird das Paket mit der Option `loadonly` geladen, so dass es nicht versucht, bestehende Gliederungsbefehle mit ihrem aktuellen Layout zu interpretieren. Dann kann man, wie im folgenden Beispiel, mit dem Schreiben von Gliederungsbefehlen beginnen:

```
\titleclass\EbeneA[0]{top}
\titleclass\EbeneB{straight}[\EbeneA]
\titleclass\EbeneC{straight}[\EbeneB]
\newcounter{EbeneA} \newcounter{EbeneB} \newcounter{EbeneC}
\renewcommand\theEbeneB{\theEbeneA-\arabic{EbeneB}}
\renewcommand\theEbeneC{\theEbeneB-\arabic{EbeneC}}
\titleformat{name=\EbeneA}{..}... \titlespacing
                                   {name=\EbeneA}{..}...
\titleformat{name=\EbeneB}{..}... \titlespacing
                                   {name=\EbeneB}{..}...
```

Die *start-ebene* ist normalerweise 0 oder -1. Die Bedeutung dieser Werte ist in der Einleitung zu Abschnitt 2.2 erklärt. Es sollte genau eine `\titleclass`-Deklaration geben, die eben dieses optionale Argument verwendet.

Wer auf diese Weise seine eigenen Dokumentenklassen aufbauen möchte, dem sei ein Blick in die Dokumentation des `titlesec`-Paketes empfohlen. Dort sind weitere Beispiele, sowie nützliche Tipps und Tricks zu finden.

2.3 Der Aufbau von Verzeichnissen

Ein *Inhaltsverzeichnis* (engl. „table of contents“, kurz TOC) ist eine besondere Liste, in der die Titel aller Abschnitte zusammen mit den Zahlen der Seiten, auf denen sie beginnen, ausgegeben werden. Diese Liste kann sehr komplex ausfallen, wenn sie Elemente mehrerer Gliederungsebenen enthält und sie sollte sorgfältig formatiert werden, da sie eine wichtige Navigationshilfe für die Leser darstellt.

Ähnliche Listen, jedoch mit Querverweisen zu den Gleitobjekten in einem Dokument sind das *Tabellenverzeichnis* und das *Abbildungsverzeichnis*. Der Aufbau dieser Verzeichnisse ist einfacher, da ihre Elemente, die Beschriftungen der Gleitobjekte, normalerweise alle zu einer Ebene gehören (siehe jedoch auch Abschnitt 6.5.2).

L^AT_EX kann diese drei Standardverzeichnisse automatisch erzeugen. Dazu wird Text, der durch eines der Argumente der Gliederungsbefehle generiert wurde, in die .toc-Datei eingetragen. In ähnlicher Weise pflegt L^AT_EX zwei weitere Dateien, eine für das Abbildungsverzeichnis (.lof) und eine für das Tabellenverzeichnis (.lot), die jeweils den Text aus dem Argument des \caption-Befehls für Abbildungen bzw. Tabellen enthalten.

Die Informationen, die L^AT_EX in einem vorhergehenden Lauf in diese Dateien geschrieben hat, werden in einem weiteren Lauf gelesen und gesetzt (normalerweise an den Anfang eines Dokumentes). Dies geschieht mithilfe folgender Befehle: \tableofcontents, \listoffigures und \listoftables.

Um diese Verzeichnisse zu erzeugen, benötigt L^AT_EX immer mindestens zwei Durchläufe: einen, um die relevanten Informationen zu sammeln und den zweiten um die Informationen wieder einzulesen und an der richtigen Stelle im Dokument zu setzen. Da somit im zweiten Lauf neues Material gesetzt wird, können sich die Querverweise ändern, so dass ein dritter L^AT_EX-Lauf erforderlich wird. Das ist einer der Gründe, warum traditionellerweise unterschiedliche Nummerierungen für den Vorspann und den Haupttext verwendet werden: In den Tagen des manuellen Setzens bedeutete jeder zusätzliche Durchlauf eine erhebliche Verteuerung für das Endprodukt.

Ein TOC wird in zwei oder drei L^AT_EX-Läufen erzeugt

Die folgenden Abschnitte erläutern, wie diese Verzeichnisse gesetzt und generiert werden. Darüber hinaus wird gezeigt, wie man Informationen direkt in die Hilfsdateien eingeben kann, oder wie man sogar eine Zusatzdatei, die völlig unter der Kontrolle des Anwenders steht, öffnet und in diese schreibt.

2.3.1 Eintragen von Daten in Verzeichnisdateien

Normalerweise werden die Verzeichnisdateien automatisch von L^AT_EX generiert. Diese Schnittstelle, die aus den Befehlen \addcontentsline und \addtocontents besteht, kann mit einiger Sorgfalt auch zur direkten Eingabe von Daten genutzt werden.

`\addcontentsline{ext}{typ}{text}`

Der Befehl \addcontentsline schreibt den *text* mit einigen zusätzlichen Informationen, wie z.B. der Nummer der aktuellen Seite, in eine Datei mit der Endung *ext* (normalerweise .toc, .lof oder .lot). Zerbrechliche Befehle im Argument *text* müssen mit \protect geschützt werden. Das *typ*-Argument ist eine Zeichenfolge, die angibt, welche Art von Verzeichniseintrag vorgenommen wird. Beim Inhaltsverzeichnis (.toc) handelt es sich normalerweise um den Namen des Gliederungsbefehls ohne Backslash, bei .lof- oder .lot-Dateien wird im Allgemeinen figure oder table angegeben.

Die Anweisung \addcontentsline wird normalerweise automatisch durch die Gliederungsbefehle eines Dokumentes oder durch die \caption-Befehle der Gleitumgebungen aufgerufen. Leider verfügt die Schnittstelle nur über ein Argument für den variablen Text, so dass es schwierig ist, eine gegebenenfalls vorhandene Objektnummer korrekt zu identifizieren. Da diese Nummern (z.B. die Gliederungsnummern) in einem Inhaltsverzeichnis typischerweise ein besonderes Format benötigen, ist ihre Identifizierung unverzichtbar. Der aktuelle L^AT_EX-Kern erreicht dieses Ziel mit einem Trick: Er

umschließt die Nummer im Argument *text* folgendermaßen mit dem Befehl `\numberline`:

```
\protect\numberline{nummer}überschrifttext
```

Ein `\caption`-Befehl innerhalb einer `figure`-Umgebung speichert den Text der Bildunterschrift einer Abbildung beispielsweise mit der folgenden Befehlszeile:

```
\addcontentsline{lof}{figure}
{\protect\numberline{\thefigure}bildunterschrift}
```

Aufgrund des `\protect`-Befehls wird `\numberline` unverändert in die externe Datei geschrieben, während `\thefigure` zwischenzeitlich expandiert wird, so dass letztendlich die tatsächliche Zahl für die Nummerierung in die Datei gelangt.

Später, beim Formatieren des Verzeichnisses, kann dann die Nummer mit `\numberline` umformatiert werden, zum Beispiel mit zusätzlichen Abständen oder einer anderen Schrift. Der Nachteil dieser Methode ist, dass sie weniger vielseitig nutzbar ist, als eine Version, die über ein eigenes Argument für die Nummer verfügt. So kann die Nummer z.B. nicht so einfach beliebig manipuliert werden. Außerdem benötigt sie eine geeignete `\numberline`-Definition, die leider auch nicht immer verfügbar ist (siehe die Betrachtung in Abschnitt 2.3.2 auf Seite 52).

Manchmal wird `\addcontentsline` in der Quelldatei verwendet, um die Standardfunktionen von L^AT_EX zu ergänzen. Bei der Sternform eines Gliederungsbefehls erfolgt z.B. kein Eintrag in der `.toc`-Datei. Wenn man also keine nummerierte Überschrift wünscht (Sternform), wohl aber einen Eintrag in der `.toc`-Datei, kann man wie im folgenden Beispiel `\addcontentsline` mit oder ohne `\numberline` verwenden.

Inhaltsverzeichnis	1 Gedanken	<code>\tableofcontents</code>	
Vorwort 1	Wir finden sie alle in [1].	<code>\section*{Vorwort}</code>	
1 Gedanken 2	1.1 Kontaktdaten	<code>\addcontentsline{toc}{section}</code>	
1.1 Kontaktdaten 2	E-Mail an Ben unter [2].	<code>{\protect\numberline{}}Vorwort}</code>	Eine Überschrift mit manuellem Verzeichniseintrag. Vergleiche mit dem Bibliographieeintrag.
Literatur 2	Literatur	<code>\section{Gedanken}</code>	
Vorwort	[1] Ben User, Später wann ist das?, 2010	<code>Wir finden sie alle in \cite{k1}.</code>	
Eine Überschrift mit manuellem Verzeichniseintrag. Vergleiche mit dem Bibliographieeintrag.	[2] BUser@earth.info	<code>\subsection{Kontaktdaten}</code>	
1	2	<code>E-Mail an Ben unter \cite{k2}.</code>	
		<code>\begin{thebibliography}{9}</code>	
		<code>\addcontentsline{toc}</code>	
		<code>{section}{\refname}</code>	
		<code>\bibitem{k1} Ben User, Später wann ist das?, 2010</code>	
		<code>\bibitem{k2} BUser@earth.info</code>	
		<code>\end{thebibliography}</code>	Bsp. 2-3-1

Wenn man `\numberline` wie im „Vorwort“ einsetzt, erhält man einen eingerückten „section“-Eintrag im Inhaltsverzeichnis. Der Platz, an dem normalerweise die Nummer stehen würde, bleibt dabei frei. Wenn man den `\numberline`-Befehl, wie beim Bibliographieeintrag, weglässt, wird die Überschrift stattdessen direkt an den linken Rand gesetzt. Wenn man nach dem Beginn von `theindex` eine ähnliche Zeile hinzufügt, wird dadurch auch der „Index“ im Inhaltsverzeichnis aufgeführt. Leider kann man mit diesem Ansatz nicht die Abbildungs- oder Tabellenverzeichnisse im Inhaltsverzeichnis auflisten, da die Befehle `\listoffigures` bzw. `\listoftables` möglicherweise mehrseitige Verzeichnisse generieren und dadurch die Seitenzahl in `\addcontentsline` falsch sein könnte. Es hilft auch nicht, den Code vor den Befehl zu setzen, da diese Verzeichnisbefehle häufig eine neue Seite beginnen. Eine mögliche Lösung wäre, die Befehlsdefinition aus der Klassendatei zu kopieren und `\addcontentsline` direkt dort einzufügen.

Im Verbund mit Standardklassen oder sehr ähnlichen Derivaten kann man das Paket `tocbibind` von Peter Wilson einsetzen, um die Abschnitte „...-verzeichnis“, „Index“, oder „Bibliographie“ ohne weitere Änderungen der Quelldatei im Inhaltsverzeichnis aufzuführen. Das Paket bietet eine Reihe von Optionen, wie z.B. `notbib`, `notindex`, `nottoc`, `notlof` und `notlot` (den entsprechenden Eintrag nicht ins Inhaltsverzeichnis einfügen) sowie `numbib` und `numindex` (den entsprechenden Abschnitt nummerieren). Standardmäßig wird der Abschnitt „Inhalt“ ebenfalls im Inhaltsverzeichnis aufgeführt, was nur selten wünschenswert ist. Dieses Verhalten lässt sich mithilfe der Option `nottoc` abstellen.

*Bibliographie oder
Index im
Inhaltsverzeichnis*

```
\addtocontents{ext}{text}
```

Der Befehl `\addtocontents` besitzt kein *typ*-Argument und ist dazu gedacht, besondere Formatierungsdaten, die sich nicht direkt auf eine Inhaltszeile beziehen, zu übergeben. Der Standardklassenbefehl `\chapter` setzt zusätzliche Leerräume in die `.lof`- und `.lot`-Dateien, um die Einträge verschiedener Kapitel voneinander abzusetzen:

```
\addtocontents{lof}{\protect\addvspace{10pt}}
\addtocontents{lot}{\protect\addvspace{10pt}}
```

Durch den Einsatz von `\addvspace` werden Einträge aus unterschiedlichen Kapiteln um genau 10pt von einander abgesetzt, ohne dass unschöne Lücken entstehen, wenn ein Kapitel keine Zeichnungen oder Tabellen enthält.

Aus diesem Beispiel ist jedoch auch ein gewisses Risiko der Schnittstelle ersichtlich: Auch wenn die Befehle `\addcontentsline`, `\addtocontents` und `\addvspace` scheinbar normale Anwenderbefehle sind (sie enthalten kein `@`-Zeichen in ihren Namen), so können sie doch sehr leicht seltsame Fehler produzieren.¹ Es ist besonders zu beachten, dass `\addvspace` nur im vertikalen Modus verwendet werden kann. Daher kann eine Zeile wie die oben aufgeführte nur funktionieren, wenn ein vorhergehender `\addcontentsline`-Befehl bereits im vertikalen Modus endet.

*Mögliche Probleme
mit \addvspace*

¹In Anhang A.1.5 auf Seite 891 wird `\addvspace` ausführlicher behandelt.

Man muss also verstehen, wie solche Verzeichniseinträge eigentlich erzeugt werden, bevor man beliebige Formatierungsanweisungen dazwischen setzen kann. Das ist Thema des nächsten Abschnitts.

Mögliche Probleme
mit `\include`

Wenn entweder `\addcontentsline` oder `\addtocontents` in der Quelldatei eines Dokumentes verwendet werden, gilt eine wichtige Einschränkung: Keiner der Befehle kann auf der gleichen Ebene eingesetzt werden, wie eine `\include`-Anweisung. Das heißt, dass beispielsweise die Befehlsfolge

```
\addtocontents{toc}{\protect\setcounter{tocdepth}{1}}
\include{sect1}
```

für den Fall, dass `sect1.tex` einen `\section`-Befehl enthält, überraschenderweise eine `.toc`-Datei mit folgendem Inhalt erzeugen würde:

```
\contentsline {section}{\numberline {1}Section from sect1}{2}
\setcounter {tocdepth}{1}
```

wobei die Zeilen nicht in der richtigen Reihenfolge erscheinen. Dieses Problem lässt sich lösen, indem man die `\addtocontents`- oder `\addcontentsline`-Anweisung in die Datei verlagert, die mit `\include` geladen wird, oder indem man gänzlich auf `\include` verzichtet.

2.3.2 Formatieren von Verzeichnissen

Wie bereits angesprochen werden Inhaltsverzeichnisse durch den impliziten oder expliziten Einsatz der Befehle `\addcontentsline` und `\addtocontents` erzeugt. Genau betrachtet bewirkt

```
\addcontentsline{ext}{typ}{text}
```

dass die Zeile

```
\contentsline{typ}{text}{seite}
```

in eine Hilfsdatei mit der Erweiterung `ext` geschrieben wird, wobei `seite` der aktuellen Seitenzahl im Dokument entspricht. Die Befehlszeile `\addtocontents{ext}{text}` ist einfacher: Sie schreibt lediglich `text` in die Hilfsdatei. Eine typische Verzeichnisdatei besteht also aus einer Reihe von `\contentsline`-Befehlen, in die möglicherweise weitere Formatierungsanweisungen eingestreut sind, die durch Aufrufe von `\addtocontents` hinzugefügt wurden. Mithilfe des Befehls `\contentsline` kann jeder Anwender ein Inhaltsverzeichnis manuell erzeugen.

Weiter unten befindet sich ein typisches Beispiel dazu. Die meisten (wenn auch nicht alle) Gliederungsnummern erscheinen dabei als Argumente des Befehls `\numberline`, damit die Einträge mit einem geeigneten Einzug formatiert werden können. L^AT_EX ist hier leider nicht konsistent. Die Standardklassen verwenden für `\part`-Überschriften nicht `\numberline`, sondern geben stattdessen den Abstand zwischen Nummer und Text explizit an. Seit

Unvereinbarkeit  mit `\part`

dem 2001/06/01-Release von L^AT_EX kann `\numberline` auch an dieser Stelle eingesetzt werden, bei älteren Releases führt dies jedoch möglicherweise zu Fehlern.

		<code>\setcounter{tocdepth}{3}</code>
		<code>\contentsline {part}{I\hspace{1em}Teil}{2}</code>
I Teil	2	<code>\contentsline{chapter}{\numberline{1}A-Ebene}{2}</code>
		<code>\contentsline{section}%</code>
		<code>{\numberline{1.1}B-Ebene}{3}</code>
1 A-Ebene	2	<code>\contentsline{section}%</code>
1.1 B-Ebene	3	<code>{\numberline{1.1.1}C-Ebene}{4}</code>
1.1.1 C-Ebene	4	<code>\contentsline{subsection}%</code>
Leere Nummer .	5	<code>{\numberline{}Leere Nummer}{5}</code>
C-Ebene ohne Nummer .	6	<code>\contentsline{subsection}{C-Ebene ohne Nummer}{6}</code>

Bsp.
2-3-2

Der Befehl `\contentsline` erhält als erstes Argument *typ* und ruft mit dieser Information den entsprechenden Befehl `\l@typ` auf, der die eigentliche Formatierung vornimmt. In der Klassendatei muss deshalb für jeden Typ je ein Befehl definiert sein. Die Klassendatei `report` enthält zum Beispiel die folgenden Definitionen:

```

\newcommand\l@section      {\@dottedtocline{1}{1.5em}{2.3em}}
\newcommand\l@subsection  {\@dottedtocline{2}{3.8em}{3.2em}}
\newcommand\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
\newcommand\l@paragraph   {\@dottedtocline{4}{10em}{5em}}
\newcommand\l@subparagraph{\@dottedtocline{5}{12em}{6em}}
\newcommand\l@figure      {\@dottedtocline{1}{1.5em}{2.3em}}
\newcommand\l@table       {\l@figure}

```

Wenn man in `\l@typ` den Befehl `\@dottedtocline` aufruft, der fünf Argumente besitzt, und die drei Argumente *ebene*, *einzug* und *num-breite* angibt, werden die verbleibenden Argumente *text* und *seite* des `\contentsline`-Befehls als Argumente 4 und 5 von `\@dottedtocline` übernommen.

Einige Gliederungsebenen erstellen ihre Verzeichniseinträge allerdings auf einem komplizierteren Wege, so dass die Standardklassen für `\l@part` und `\l@chapter` (oder `\l@section` in der Klasse `article`) Definitionen enthalten, die nicht den Befehl `\@dottedtocline` verwenden. Im Allgemeinen benutzen sie spezielle Formatierungsbefehle. So lassen sie etwa die Punkte aus und wählen eine größere Schrift.

Um das Layout für ein Inhaltsverzeichnis zu definieren, müssen also die geeigneten `\l@typ`-Befehle deklariert werden. Das lässt sich, wie bereits zuvor gezeigt, auf einfache Weise mithilfe des Befehls `\@dottedtocline` erreichen, einem internen Befehl, der nun im Folgenden genauer betrachtet wird.

`\@dottedtocline{ebene}{einzug}{num-breite}{text}{seite}`

Die letzten beiden Argumente von `\@dottedtocline` stimmen mit den letzten Argumenten des Befehls `\contentsline` überein, der normalerweise

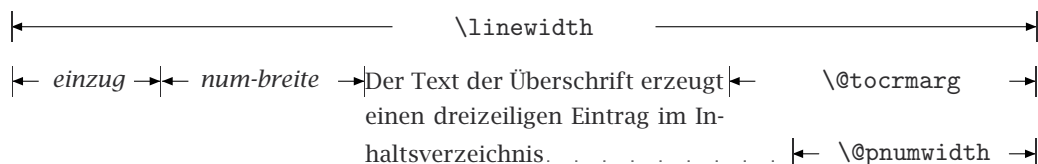


Abbildung 2.3: Layoutparameter für Inhaltsverzeichnisse

selbst einen `\@dottedtocline`-Befehl aufruft. Die anderen Argumente haben folgende Funktionen:

ebene Die Gliderungsebene des Befehls `\contentsline` für einen Eintrag. Mithilfe des Zählers `tocdepth` kann der Anwender steuern, wie viele Gliderungsebenen angezeigt werden. Ebenen mit einer höheren Nummer als dem in `tocdepth` angegebenen Wert erscheinen nicht im Inhaltsverzeichnis.

einzug Der gesamte Einzug vom linken Rand.

num-breite Die Breite der Box, welche die Gliederungsnummer der Überschrift enthält, wenn das Argument *text* mit dem Befehl `\numberline` versehen ist. Dieser Wert dient bei mehrzeiligen Einträgen gleichzeitig als zusätzlicher Einzug für die zweite und alle weiteren Zeilen.

Zusätzlich verwendet der Befehl `\@dottedtocline` drei Formatierungsparameter, um das Erscheinungsbild aller Einträge festzulegen. Obwohl diese Parameter Längenwerte speichern, müssen sie alle mit `\renewcommand` geändert werden!

`\@pnumwidth` Die Breite der Box, welche die Seitenzahl enthält.

`\@tocrmarg` Der Einzug vom rechten Rand für alle Zeilen, außer der letzten (d.h. nur relevant bei mehrzeiligen Einträgen). Der Parameter kann eine elastische Länge enthalten. Dann wird das Verzeichnis im Flattersatz formatiert.

`\@dotsep` Der Punktabstand in μ (math units, mathematischen Einheiten).¹ Angegeben wird hier nur die Zahl (etwa 1.7 oder 2) ohne Einheit. Wenn diese groß genug gewählt wird, lassen sich die Punkte völlig unterdrücken. Wird mit dem Befehl `\renewcommand` geändert!

Eine bildliche Darstellung der beschriebenen Effekte wird in Abbildung 2.3 gezeigt.

Wenn die Box, die durch *num-breite* definiert wird, eine Gliederungsnummer enthält, ist diese linksbündig. Den richtigen Einzug für Einträge niedrige-

¹18 μ bilden ein em, wobei Letzteres sich auf `\fontdimen2` des mathematischen Symbols `symbols` bezieht. Nähere Erläuterungen zu den Dimensionsparametern `\fontdimen` werden in Abschnitt 7.10.3 auf Seite 439 gegeben.

rer Gliederungsebenen erhält man, indem man die Werte für die Argumente *einzug* und *num-breite* variiert.

Wenn man mit den Standardklassen (article, report oder book) arbeitet, ist dieses Vorgehen notwendig, sobald zehn oder mehr Abschnitte auftreten, von denen die letzten mehr als neun Unterabschnitte haben. Wie man am nächsten Beispiel sehen kann, geraten die Nummern und der Text zu nahe aneinander oder überlappen sogar, wenn das Argument *num-breite* in den entsprechenden Aufrufen von `\@dottedtocline` nicht vergrößert wird.

*Problem bei vielen
Titeln auf einer Ebene*

10 A-Ebene	3	<code>\contentsline{section}{\numberline{10}A-Ebene}{3}</code>
Bsp. 2-3-3 10.9 B-Ebene	4	<code>\contentsline{subsection}{\numberline{10.9}B-Ebene}{4}</code>
10.10B-Ebene	4	<code>\contentsline{subsection}{\numberline{10.10}B-Ebene}{4}</code>

Hier erhält man ein besseres Resultat, wenn man `\l@section` so umdefiniert, dass mehr Platz für die Gliederungsnummer (das dritte Argument von `\@dottedtocline`) bleibt. Wahrscheinlich müssen auch die anderen Befehle, wie etwa `\l@subsection`, angepasst werden, damit das Verzeichnis ein ausgewogenes Erscheinungsbild erhält.

		<code>\makeatletter</code>
		<code>\renewcommand\l@section{\@dottedtocline{2}{1.5em}{3em}}</code>
		<code>\makeatother</code>
10 A-Ebene	3	<code>\contentsline{section}{\numberline{10}A-Ebene}{3}</code>
Bsp. 2-3-4 10.9 B-Ebene . . .	4	<code>\contentsline{subsection}{\numberline{10.9}B-Ebene}{4}</code>
10.10 B-Ebene . . .	4	<code>\contentsline{subsection}{\numberline{10.10}B-Ebene}{4}</code>

Ein weiterer Fall, der Anpassungen erforderlich macht, ist eine ungewöhnliche Seitennummerierung. Wenn die Seiten zum Beispiel innerhalb der einzelnen Teile als „A-78“, „B-328“, usw. formatiert werden, ist der Platz für die Seitenzahl wahrscheinlich zu schmal. Dies führt zumindest zu einer Menge lästiger „Overfull hbox“-Warnungen, sehr wahrscheinlich aber auch zu einer schlechten Verteilung der Abstände um sie herum. In diesem Fall hilft es, `\@pnumwidth` einen für den breitesten Eintrag ausreichenden Wert zuzuweisen, z.B. folgendermaßen:

```
\makeatletter \renewcommand\@pnumwidth{2cm} \makeatother
```

Wenn man `\@pnumwidth` auf diese Weise anpasst, muss sehr wahrscheinlich auch der Wert von `\@tocmarg` geändert werden, damit das Layout des Inhaltsverzeichnisses stimmig bleibt.

Die Gliederungsebene, bis zu welcher Überschriften noch im Inhaltsverzeichnis erscheinen, wird durch den Zähler `tocdepth` festgelegt. Er kann z.B. mit folgender Deklaration geändert werden:

```
\setcounter{tocdepth}{1}
```

In diesem Fall werden die Abschnittsüberschriften bis hinunter zur ersten Ebene angezeigt, wie z.B. Teile, Kapitel und Abschnitte in der Klasse report.

2.3.3 Kombinieren von Verzeichnissen

Standardmäßig erzeugt L^AT_EX getrennte Listen für das Inhalts-, das Abbildungs- und das Tabellenverzeichnis, die entsprechend über die Befehle `\tableofcontents`, `\listoffigures` und `\listoftables` verfügbar sind. Keine der Standardklassen unterstützt eine Kombination dieser Listen, so dass man, wie manchmal gewünscht, alle Tabellen und Abbildungen, oder sogar alle drei Listen in einem einzigen Verzeichnis abbilden könnte.

Wie könnte man eine solche Aufgabe lösen? Zunächst muss L^AT_EX dazu gebracht werden, in die entsprechende Hilfsdatei zu schreiben, wenn es intern `\addcontentsline` verwendet. So müssen zum Beispiel alle `\caption`-Befehle in eine einzige Datei geschrieben werden, wenn man Abbildungen und Tabellen zu einer gemeinsamen Liste kombinieren will. Ein Blick auf die L^AT_EX-Quelldateien zeigt, dass dieses Ziel leicht zu erreichen ist: Bildunterschriften werden in eine Datei geschrieben, deren Erweiterung in `\ext@figure` angegeben wird, während Tabellenunterschriften zu diesem Zweck `\ext@table` verwenden.

Mit einer geeigneten neuen Definition, etwa von `\ext@table`, kann man L^AT_EX zwingen, alle Verweise auf Abbildungen und Tabellen in die `.lof`-Datei zu schreiben. Aber reicht das? Das Beispiel zeigt, dass es wahrscheinlich nicht reicht, die Einträge einfach nur in eine Datei zu zwingen. Das Verzeichnis lässt nicht erkennen, welche Einträge sich auf Abbildungen beziehen und welche auf Tabellen. Der einzige Hinweis, dass etwas nicht stimmt, ergibt sich aus den identischen Nummern auf der linken Seite.

Eine Abbildung

Abb. 1: Bildunterschrift

Abbildungen und Tabellen

1	Bildunterschrift	1
1	Tabellenunterschrift	1

1 Ein Abschnitt

Etwas Text ... Verweistext zu Abbildung 1 ...

Eine Tabelle

Tabelle 1: Tabellenunterschrift

```
\makeatletter
\renewcommand\ext@table{lof}
\makeatother
\addto\captionsngerman{%
  \renewcommand\listfigurename
    {Abbildungen und Tabellen}}
\listoffigures
\section{Ein Abschnitt}
Etwas Text \ldots
\begin{table}[b]
  \centering
  \fbox{\scriptsize Eine Tabelle}
  \caption{Tabellenunterschrift}
\end{table}
Verweistext zu Abbildung-\ref{fig} \ldots
\begin{figure}
  \centering
  \fbox{\scriptsize Eine Abbildung}
  \caption{Bildunterschrift}\label{fig}
\end{figure}
```

Bsp.
2-3-5

Da in den Buchbeispielen das babel-System benutzt wird, wurde die Textänderung von `\listfigurename` mittels `\addto` in `\captionsngerman` vorgenommen. Das genaue Verfahren wird in Abschnitt 9.5 erklärt.

Die Situation wäre ein wenig besser, wenn alle Abbildungen und Tabellen den gleichen Zähler verwenden würden, so dass die Nummern in der linken Spalte nicht identisch wären. Ein solches Ergebnis lässt sich jedoch leider nur schwer erzielen, da man dazu die Low-Level-Definitionen für die Gleitobjekte verändern muss.

Eine weitere mögliche Lösung besteht darin, `\l@figure` und `\l@table` so zu definieren, dass die entsprechenden Informationen vorliegen. Das Beispiel below zeigt eine mögliche Lösung: Jedem Eintrag wird die Zeichenfolge „(Abbildung)“ oder „(Tabelle)“ angefügt. Theoretisch wäre es auch möglich, den Typ des Gleitobjektes an die Nummer anzufügen. Dazu müssen jedoch viele interne L^AT_EX-Befehle wie etwa `\numberline` undefiniert werden.

Was geschieht, wenn alle Einträge in eine einzige Liste, also das Inhaltsverzeichnis, gezwungen werden? In diesem Falle erhalten wir eine Liste, die entsprechend dem endgültigen Erscheinungsbild der Objekte im Dokument sortiert ist, und die wohl anders aussieht, als erwartet. Im nächsten Beispiel erscheint die Abbildung, die in der Quelldatei an letzter Stelle stand, vor dem Abschnitt, in dem auf sie verwiesen wird, da der Algorithmus zum Positionieren von Gleitobjekten sie an den Kopf der Seite setzt. Dieses Ergebnis mag für Bücher oder Berichte akzeptabel sein, in denen die Hauptüberschriften eine neue Seite beginnen und Gleitobjekte am Kopf dieser Seiten vermieden werden. In anderen Fällen ist es jedoch wahrscheinlich nicht wünschenswert.

Eine Abbildung

Abb. 1: Bildunterschrift

Inhaltsverzeichnis

1	Bildunterschrift (Abbildung) . .	1
1	Ein Abschnitt	1
1	Tabellenunterschrift (Tabelle) . .	1

1 Ein Abschnitt

Etwas Text ... Verweistext zu Abbildung 1 ...

Eine Tabelle

Tabelle 1: Tabellenunterschrift

Bsp.
2-3-6

```

\makeatletter
\renewcommand\ext@figure{toc}
\renewcommand\ext@table{toc}
\renewcommand\l@figure[2]{\@dottedtocline
  {1}{1.5em}{2.3em}{#1~(Abbildung)}{#2}}
\renewcommand\l@table [2]{\@dottedtocline
  {1}{1.5em}{2.3em}{#1~(Tabelle)}{#2}}
\makeatother
\tableofcontents
\section{Ein Abschnitt}
Etwas Text \ldots
\begin{table}[b]
\centering
\fbbox{\scriptsize Eine Tabelle}
\caption{Tabellenunterschrift}
\end{table}
Verweistext zu Abbildung~\ref{fig} \ldots
\begin{figure}
\centering
\fbbox{\scriptsize Eine Abbildung}
\caption{Bildunterschrift}\label{fig}
\end{figure}

```

Zusammenfassend lässt sich sagen, dass es zwar möglich ist, mehrere Verzeichnistypen zu kombinieren, das Ergebnis dabei jedoch wahrscheinlich nicht wie erwartet ausfällt. Auf jeden Fall erfordert ein solcher Ansatz, dass alle `\l@typ`-Befehle sorgfältig umgeschrieben werden, damit das Endergebnis für den Leser von Nutzen ist.

2.3.4 Erstellen weiterer Verzeichnisse

Wenn man ein Verzeichnis aller Beispiele in einem Buch erstellen möchte, muss man zunächst eine neue Verzeichnisdatei erstellen und dann die oben beschriebenen Funktionen verwenden. Zuerst müssen zwei neue Befehle definiert werden. Der erste Befehl, `\ecaption` im nächsten Beispiel, verknüpft eine Beschriftung mit der aktuellen Position im Dokument, indem er sein Argument und die aktuelle Seitenzahl in die Verzeichnisdatei schreibt. Der zweite Befehl, `\listofexamples`, liest die Daten, die im vorigen Lauf in die Verzeichnisdatei geschrieben wurden und fügt sie an der Stelle im Dokument ein, an welcher der Befehl aufgerufen wird.

Der Befehl `\listofexamples` ruft `\@starttoc{ext}` auf, der die externe Datei (mit der Erweiterung *ext*) liest, und sie dann erneut zum Schreiben öffnet. Letzterer wird ebenfalls von den Befehlen `\tableofcontents`, `\listoffigures` und `\listoftables` verwendet. Die Zusatzdatei kann z.B. die Erweiterung *xmp* erhalten. Für die Überschrift des Verzeichnisses kann ein Befehl wie `\chapter*{Verzeichnis der Beispiele}` direkt vor oder in `\listofexamples` gesetzt werden. Falls gewünscht, kann man dem Leser durch Hinzufügen des Befehls `\addcontentsline` außerdem die Existenz dieses Verzeichnisses im Inhaltsverzeichnis anzeigen.

Die tatsächliche Formatierung der individuellen Einträge in der *.xmp*-Datei wird durch den Befehl `\l@example` gesteuert, der auch noch definiert werden muss. Im folgenden Beispiel werden die Bildunterschriften als Absätze formatiert, gefolgt von einer kursiven Seitenzahl.

1 Ausgewählte Aufnahmen

Ravels Boléro von Jacques Loussier
Trio.

Davis' Blue in Green von Cassan-
dra Wilson.

Kommentare

Loussier: Ein seltsames Erlebnis, *I*
Wilson: Eine wunderbare Version, *I*

```
\newcommand\ecaption[1]
    {\addcontentsline{xmp}{example}{#1}}
\makeatletter \newcommand\listofexamples
    {\section*{Kommentare}\@starttoc{xmp}}
\newcommand\l@example[2]
    {\par\noindent#1,~\textit{#2}\par} \makeatother
\section{Ausgewählte Aufnahmen}
Ravels Bol\`ero von Jacques Loussier
Trio.\ecaption{Loussier: Ein seltsames Erlebnis}

Davis' Blue in Green von Cassandra
Wilson.\ecaption{Wilson: Eine wunderbare Version}
\listofexamples
```

Bsp.
2-3-7

Das in Abschnitt 6.3.1 auf Seite 300 beschriebene Paket `float` implementiert den oben vorgestellten Mechanismus mit dem Befehl `\listof{<typ>}`, der die Liste aller Gleitobjekte des Typs *<typ>* erzeugt.

2.3.5 shorttoc – Kompakte Inhaltsverzeichnisse

Bei sehr umfangreichen Dokumenten ist es manchmal hilfreich, neben dem normalen Inhaltsverzeichnis eine kompaktere Übersicht, die nur die Hauptabschnitte auflistet, zur Verfügung zu stellen. Ein solches kompaktes Inhaltsverzeichnis lässt sich mithilfe des Paketes `shorttoc` von Jean-Pierre Drucbert erzeugen.

`\shorttableofcontents{titel}{tiefe}`

Der Befehl `\shorttableofcontents` (bzw. die Kurzform `\shorttoc`) muss vor dem Befehl `\tableofcontents` stehen. Andernfalls bleibt das kompakte Inhaltsverzeichnis leer. Das erste Argument *titel* enthält die Überschrift für das Verzeichnis und das zweite Argument *tiefe* gibt an, bis zu welcher Gliederungsebene Einträge angezeigt werden sollen. Ein kompaktes Verzeichnis, das nur Kapitel und Abschnitte enthält, während die gesamte Gliederungsstruktur bis zu den Passagen im ausführlichen Inhaltsverzeichnis aufgeführt wird, würde man wie folgt definieren:

```
\shorttableofcontents{Inhaltsübersicht}{1}
\setcounter{tocdepth}{3}
\tableofcontents
```

Das Paket unterstützt zwei Optionen, *loose* (Standard) und *tight*, über die sich die vertikalen Abstände im kompakten Verzeichnis einstellen lassen.

2.3.6 minitoc – Mehrfache Inhaltsverzeichnisse

Das Paket `minitoc` wurde ursprünglich von Nigel Ward und Dan Jurafsky geschrieben und von Jean-Pierre Drucbert vollständig überarbeitet. Mit seiner Hilfe lassen sich Teilverzeichnisse (sogenannte „minitoc“s) für Kapitel, Abschnitte oder Teile erzeugen. Es kann auch Mini-Verzeichnisse für Abbildungen und Tabellen generieren, die in einem Kapitel, einem Abschnitt oder einem Teil vorkommen. Das Paket `titletoc`, das in Abschnitt 2.3.7 behandelt wird, stellt ähnliche Funktionen zur Verfügung. Dabei folgt es jedoch einem völlig anderen Ansatz.

Im Folgenden wird genauer beschrieben, wie man mit dem Paket `minitoc` kapitelweise Verzeichnisse erstellt. Das Vorgehen gilt analog für Verzeichnisse je Abschnitt oder Teil, lediglich mit anderen Befehlsnamen. Am Ende dieses Abschnitts befindet sich eine entsprechende Übersicht.

Das Paket unterstützt nahezu alle Sprachoptionen des `babel`-Systems (siehe Abschnitt 9.1.3), welche die verwendeten Überschriften voreinstellen. Das Layout der generierten Verzeichnisse lässt sich außerdem mit den Optionen *loose* (Standard) oder *tight* sowie *dotted* (Standard) oder *undotted* beeinflussen. Eine Reihe von Parametern, die in der Präambel gesetzt werden können (siehe Tabelle 2.3 auf der nächsten Seite), bieten weitere Gestaltungsmöglichkeiten.

Um das `minitoc`-System zu initialisieren, muss ein `\dominitoc`-Befehl vor `\tableofcontents` ausgeführt werden. Sind nur Teilverzeichnisse erwünscht (ohne ein Gesamtverzeichnis), dann muss letzterer Befehl durch `\faketableofcontents` ersetzt werden. Teilverzeichnisse für Abbildungen und Tabellen werden durch ähnliche Befehle (`\dominilof` bzw. `\dominilot`) initialisiert, gegebenenfalls in Kombination mit `\fakelistoffigures` oder `\fakelistoftables`.

Die `\domini...`-Befehle verwenden ein optionales Argument, um die Position der Verzeichnisüberschrift festzulegen: *l* für links (Standard), *c* für zentriert, *r* für rechts und *n* oder *e* für keinen Titel. Die Deklaration ist global und gilt für alle Verzeichnisse im ganzen Dokument.

<code>\minitocdepth</code>	Dieser L ^A T _E X-Zähler gibt an, wie viele Gliederungsebenen im Teilverzeichnis angezeigt werden. Der Wert ist auf 2 voreingestellt.
<code>\mtcindent</code>	Die Breite des linken/rechten Einzugs des Teilverzeichnisses (voreingestellt auf 24pt).
<code>\mtcfont</code>	Dieser Befehl legt den Standardfont für die Verzeichniseinträge fest (voreingestellt auf einen Roman Font kleiner Schriftgröße).
<code>\mtcSfont</code>	Dieser Befehl legt den Font für die <code>\section</code> -Einträge fest (voreingestellt auf einen kleinen, fett gedruckten Roman Font).
<code>\mtcSSfont</code>	Falls angegeben, der Font für <code>\subsection</code> -Einträge. Diese und alle weiteren Gliederungsebenen sind auf <code>\mtcfont</code> voreingestellt.
<code>\mtcSSSfont</code>	Falls angegeben, der Font für <code>\subsubsection</code> -Einträge.
<code>\mtcPfont</code>	Falls angegeben, der Font für <code>\paragraph</code> -Einträge.
<code>\mtcSPfont</code>	Falls angegeben, der Font für <code>\subparagraph</code> -Einträge.
<code>\mtctitle</code>	Titeltext für Teilverzeichnisse (voreingestellt durch die Sprachoption).
<code>\nomtcrule</code>	Diese Deklaration verhindert Linien ober- und unterhalb der Teilverzeichnisse (<code>\mtcrule</code> ermöglicht sie).
<code>\nomtcpagenumbers</code>	Diese Deklaration blendet Seitenzahlen in den Teilverzeichnissen aus (<code>\mtcpagenumbers</code> blendet sie ein).

Tabelle 2.3: Die minitoc-Parameter im Überblick

Die eigentlichen Teilverzeichnisse werden dann mit dem `\minitoc`-Befehl an geeigneten Stellen im Dokument platziert, normalerweise direkt nach einem `\chapter`-Befehl. Die tatsächliche Platzierung bleibt jedoch dem Anwender überlassen. So kann man z.B. Text davor setzen oder sie einrahmen. Wenn eines der Verzeichnisse leer ist, unterdrückt das Paket die Überschrift und gibt eine Warnung aus, um auf mögliche Formatierungsprobleme durch das Material in der Umgebung des Befehls hinzuweisen.

Zum Generieren von Teilverzeichnissen für Abbildungen oder Tabellen verwendet man die Befehle `\minilof` oder `\minilot`, nachdem das System wie zuvor beschrieben initialisiert wurde.

Für jedes Teilverzeichnis wird eine Hilfsdatei mit der Erweiterung `.mtc<n>` erstellt, wobei `<n>` die Kapitelnummer ist.¹ Bei Teilverzeichnissen für Abbildungen und Tabellen werden entsprechend die Endungen `.mlf<n>` und `.mlt<n>` erzeugt.

Normalerweise enthalten Teilverzeichnisse nur Verweise zu Abschnitten und Unterabschnitten. Dies lässt sich mithilfe des Zählers `minitocdepth`, ähnlich wie bei `tocdepth`, ändern. Außerdem kann man andere Fonts für die einzelnen Einträge einstellen, indem man den Befehl `\mtcfont` bzw. die verwandten Befehle aus der Tabelle 2.3 entsprechend bearbeitet. Die um ein

¹Für Betriebssysteme, die nur Dateierweiterungen mit drei Zeichen unterstützen, wie z.B. MS-DOS, wird automatisch eine andere Bezeichnung gewählt. Diese kann über die Option `shorttext` des Befehls `\usepackage` auch erzwungen werden.

Teilverzeichnis erzeugten Rahmenlinien werden durch Angabe von `\mtcrule` (Standard) oder `\nomtcrule` in der Präambel oder vor den einzelnen Verzeichnissen eingestellt. In ähnlicher Weise lassen sich Seitenzahlen in den Teilverzeichnissen mit `\mtcpagenumbers` einblenden (Standard) oder mit `\nomtcpagenumbers` ausblenden.

Da die Teilverzeichnisse zusätzlichen Platz im Dokument einnehmen, wirken sie sich auf die Seitenzahlen aus. Daher sind normalerweise mindestens drei Läufe erforderlich, damit die Verzeichnisse wirklich korrekte Informationen enthalten.

Für Teilverzeichnisse für die Gliederungsebene `\part` existieren ähnliche Befehle, wie die in Tabelle 2.3 auf der gegenüberliegenden Seite dargestellten. Der einzige Unterschied besteht darin, dass sie statt `mini` die Zeichenfolge `part` im Namen führen, und statt `mtc` die Folge `ptc`. Das System wird hier also mit `\doparttoc` initialisiert, `\parttoc` druckt das Teilverzeichnis, mit `\noptcrules` werden die Rahmenlinien ausgeblendet und so fort. Hinzu kommt lediglich die Deklaration `\ptcCfont`, welche den Font für die Kapiteleinträge festlegt und damit natürlich keine Entsprechung hat.

Bei Teilverzeichnissen auf der `\section`-Ebene stellt sich die Situation wiederum ähnlich dar: `mini` wird durch `sect` ersetzt und `mtc` durch `stc` – also z.B. `\dosecttoc`, `\secttoc` und `\stcfont`. Im Zusammenhang mit den Befehlen `\sectlof` oder `\sectlot` kann man auch die Option `placeins` testen, die Gleitobjekte mithilfe des Paketes `placeins` innerhalb ihrer jeweiligen Abschnitte belässt. Die zugehörigen Optionen sind `below` und `section` (siehe Abschnitt 6.2.1 auf Seite 297).

Teilverzeichnisse für Teile oder Abschnitte

1	Afghanistan	
1.1	Geographie .	1
1.1.1	Gesamtfläche	1
1.1.2	Landfläche .	1
1.2	Geschichte .	2
1.1	Geographie	
1.1.1	Gesamtfläche	
	647,500 km ²	
1.1.2	Landfläche	
	647,500 km ²	
		1

1.2	Geschichte	
...		
2	Albanien	
2.1	Geographie .	2
2.1.1	Gesamtfläche	2
2.1.2	Landfläche .	3
2.2	Geschichte .	3
2.1	Geographie	
2.1.1	Gesamtfläche	
	28,750 km ²	
		2

```

\usepackage{minitoc}
\setlength{stcindent}{0pt}
\renewcommand{stctitle{}}
\renewcommand{stcfont}
{\footnotesize}
\setcounter{secttocdepth}{3}

\dosecttoc \faketableofcontents
\section{Afghanistan}\secttoc
\subsection{Geographie}
\subsubsection{Gesamtfläche}
647,500 km\textsuperscript{2}
\subsubsection{Landfläche}
647,500 km\textsuperscript{2}
\subsection{Geschichte} \ldots

\section{Albanien} \secttoc
\subsection{Geographie}
\subsubsection{Gesamtfläche}
28,750 km\textsuperscript{2}
\subsubsection{Landfläche}
27,400 km\textsuperscript{2}
\subsection{Geschichte} \ldots

```


Zum Deaktivieren der `\minitoc`-Befehle muss nur das Paket `minitoc` im `\usepackage`-Befehl gegen `mtcoff` ausgetauscht werden. Dadurch werden alle `minitoc`-bezogenen Befehle im Quelltext ignoriert.

2.3.7 titletoc – Ein anderer Ansatz für Inhaltsverzeichnisse

Das Paket `titletoc` von Javier Bezos wurde ursprünglich als Begleitpaket zu `titlesec` entwickelt. Es kann jedoch auch eigenständig eingesetzt werden. Es verwendet eine eigene Schnittstelle zum Abbilden von Gliederungsstrukturen und umgeht dadurch einige der Einschränkungen des L^AT_EX-Systems.

Verhältnis zu
Standard-L^AT_EX

Das tatsächliche Erstellen der externen Verzeichnisdateien und deren Syntax sind unverändert, so dass dieses Paket sehr gut zusammen mit anderen Paketen funktioniert, die derartige Dateien generieren. Es gibt jedoch eine Ausnahme: Die Verzeichnisdateien sollten mit dem Befehl `\contentsfinish` enden. Bei Standarddateien mit den Erweiterungen `.toc`, `.lof` und `.lot` geschieht dies automatisch. Wenn man jedoch eigene Verzeichnisdateien erstellt (siehe Abschnitt 2.3.4), muss man diese wie im folgenden Beispiel dem Paket `titletoc` bekannt machen:

```
\contentsuse{example}{xmp}
```

Wie in Abschnitt 2.3.2 erklärt, besteht eine Verzeichnisdatei aus `\contentsline`-Befehlen, die manchmal durch beliebigen Code getrennt sein können, der mit `\addtocontents` hinzugefügt wurde. Um solche Verzeichniszeilen mit Standard-L^AT_EX formatieren zu können, musste man zunächst Befehle der Form `\l@typ` definieren. Bei dem Paket `titletoc` ist dieser Schritt nicht mehr notwendig. Stattdessen wird die gewünschte Formatierung mithilfe der `\titlecontents`-Deklaration (für vertikal abgesetzte Einträge) oder ihrer Sternform (für eingebettete Einträge) vorgenommen.

```
\titlecontents{typ}[einzug-links]{code-oberhalb}{nummeriert-format}  
{nicht-nummeriert-format}{h-code}{code-unterhalb}
```

Das erste Argument von `\titlecontents` spezifiziert den *typ* der Verzeichniszeile, die formatiert wird. Es entspricht damit dem ersten Argument von `\contentsline`. Anders ausgedrückt benötigt man für jeden *typ* von Gliederungsbefehl, der im Dokument auftreten kann, eine `\titlecontents`-Deklaration.¹ Die anderen Argumente haben folgende Bedeutung:

einzug-links Dieses Argument gibt den Einzug vom linken Rand für alle Einträge an. In diesen Raum kann man Material hineinssetzen, wie etwa Gliederungsnummern. Auch wenn das Argument in eckigen Klammern angegeben wird, ist es in der aktuellen Paketversion *nicht* optional!

code-oberhalb Dieser Code wird ausgeführt, bevor der Eintrag gesetzt wird. Er kann dazu verwendet werden, vertikale Abstände einzufügen, etwa mit `\addvspace`, oder um Formatierungsanweisungen, wie z.B. Fontwechsel,

¹Das Paket berücksichtigt bestehende `\l@typ`-Deklarationen, die z.B. von der Dokumentenklasse stammen. Daher kann es nur zum Ändern des Layouts einiger Typen verwendet werden.

für den ganzen Eintrag vorzugeben. An dieser Stelle kann man außerdem mit `\filleft`, `\filright`, `\filcenter` oder `\fillast` arbeiten, die bereits aus dem Paket `titlesec` bekannt sind.

nummeriert-format Dieser Code formatiert den Eintrag mitsamt seiner Nummer. Er wird im horizontalen Modus ausgeführt, nachdem der Einzug gesetzt wurde. Das letzte Element kann ein Befehl mit einem Argument sein, das den Eintrags-*text* als Wert erhält. Die unformatierte Gliederungsnummer ist über den Befehl `\thecontentslabel` verfügbar. Weiter unten werden noch andere Möglichkeiten aufgezeigt, wie man sie ansprechen und positionieren kann.

nicht-nummeriert-format Dieser Code dient dazu, den Eintrag zu formatieren, wenn er keine Nummer enthält. Auch hier kann das letzte Element ein Befehl mit einem Argument sein.

h-code Dieser Code wird noch im horizontalen Modus ausgeführt, nachdem der Eintrag bereits formatiert wurde. Er wird normalerweise verwendet, um Füllmaterial, wie etwa eine gepunktete Linie, einzusetzen, und um die in `\thecontentspage` gespeicherte Seitenzahl anzufügen. Mithilfe des Befehls `\titlerule`, der auf Seite 44 besprochen wird, kann man Leitpunkte erzeugen.

code-unterhalb Dieses (optionale) Argument dient dazu Code anzugeben, der im vertikalen Modus ausgeführt wird, nachdem der Eintrag gesetzt wurde, z.B. um nach dem Eintrag einen vertikalen Abstand einzufügen.

Das Paket `titletoc` enthält zwei nützliche Werkzeuge zum Positionieren und Formatieren der Überschriften und Seitenzahlen: `\contentslabel` und `\contentspage`.

`\contentslabel [text] {größe}`

Der Befehl `\contentslabel` dient dazu, den Inhalt des Argumentes *text*, das normalerweise `\thecontentslabel` enthält, linksbündig in einer Box der Breite *größe* zu setzen und diese Box dann links der aktuellen Position zu platzieren. Wenn man diesen Befehl im *nummeriert-format*-Argument von `\titlecontents` verwendet, wird die Nummer in den Rand vor den Eintrags-*text* gesetzt, oder in den Einzug, der mit *einzug-links* erzeugt wurde. Wer ein individuelleres Layout wünscht, kann im optionalen Argument seine eigenen Formate festlegen, im Allgemeinen unter Verwendung des Befehls `\thecontentslabel`.

Das Paket bietet drei Optionen, um die Voreinstellungen des Befehls `\contentslabel` in Verbindung mit dem Argument *text* zu beeinflussen. Mit `rightlabels` wird die Gliederungsnummer in dem entsprechenden Bereich rechtsbündig ausgerichtet. Die voreingestellte Ausrichtung `leftlabels` ist linksbündig. Die Option `dotinlabels` setzt einen Punkt nach der Nummer.

Paketoptionen

`\contentspage [text]`

In ähnlicher Weise setzt `\contentspage` den Inhalt des optionalen Argumentes *text* (Voreinstellung `\thecontentspage`) rechtsbündig in eine Box vorgegebener Breite und sorgt dann dafür, dass diese Box rechts der aktuellen

Position platziert wird, ohne jedoch Raum einzunehmen. Wenn sie an das Ende einer Zeile gesetzt wird, läuft die Box damit in den Rand hinein. In diesem Fall legt jedoch kein obligatorisches Argument die Größe der Box fest: Sie ist für alle Einträge identisch. Ihre Breite entspricht dem Freiraum, der rechts von allen Einträgen verbleibt und kann mit dem Befehl `\contentsmargin`, der weiter unten beschrieben wird, eingestellt werden.

Zu den Beispielen
in diesem Abschnitt



Für die Beispiele in diesem Abschnitt wurden einige Teile der Original-`.toc`-Dateien, die L^AT_EX für dieses Buch erzeugt hat (Kapitel 2 und Teile von Kapitel 3) in die Datei `partial.toc` kopiert. Innerhalb der Beispiele wurde diese Datei dann mit `\input` geladen und `\contentsfinish` manuell hinzugefügt. In einem echten Dokument verwendet man stattdessen den Befehl `\tableofcontents`, um die `.toc`-Datei für das *eigene* Dokument zu laden und zu verarbeiten.

Im ersten Beispiel erhalten die Kapiteleinträge eine neue Formatierung, während die Abschnitteinträge weiterhin dem L^AT_EX-Standard entsprechen. Die Kapiteleinträge sind im Flattersatz (mit `\filright`) und fett gesetzt. Sie erhalten einen Abstand von einem Pica oberhalb, gefolgt von einer breiten Linie. Der eigentliche Eintrag ist um sechs Pica eingerückt. In diesem Leerraum wird das Wort „Kapitel“ in Kapitälchen gesetzt, gefolgt von einem Leerzeichen und der Kapitelnummer (`\thecontentslabel`). Dies geschieht mithilfe der Anweisung `\contentslabel` und ihrem optionalen Argument. Überschriften ohne Nummerierung erhalten keine spezielle Formatierung, so dass sie einfach um sechs Pica eingerückt werden. Der verbleibende Leerraum wird mit `\hfill` gefüllt und die Seitenzahlen werden mit dem Befehl `\contentspage` in den Rand gesetzt. Nach dem Eintrag folgt ein weiterer Abstand von zwei Pica-Punkten, so dass er leicht von einem gegebenenfalls folgenden Abschnitteintrag abgesetzt wäre.

KAPITEL 2	Die Struktur eines L^AT_EX-Dokumentes	17
2.1.	Der Aufbau der Quelldateien	17
2.2.	Gliederungsbefehle	24
2.3.	Der Aufbau von Verzeichnissen	48
2.4.	Verweise in Dokumenten	69
KAPITEL 3	Formatierungswerkzeuge	85
3.1.	Textfragmente und Absätze	86
3.2.	Fußnoten, Endnoten und Marginalien . . .	116
3.3.	Listen	135
3.4.	Wortwörtlicher Text	159

```
\usepackage[dotinlabels]{titletoc}
\titlecontents{chapter} [6pc]
{\addvspace{1pc}\bfseries
\titlerule[2pt]\filright}
{\contentslabel
[\textsc{\chaptername}\
\thecontentslabel]{6pc}}
{}{\hfill\contentspage}
[\addvspace{2pt}]
% Nur Level 0 + 1 zeigen:
\setcounter{tocdepth}{1}
\input{partial.toc}
\contentsfinish
```

Bsp.
2-3-9

Anstatt den ganzen Eintrag einzurücken und dann einen Teil mithilfe von `\contentslabel` in den linken Rand zu verschieben, kann man auch mit `\contentspush` einen ganz ähnlichen Effekt erzielen.

`\contentspush{text}`

Dieser Befehl formatiert den Inhalt des Argumentes *text* und addiert dann gegebenenfalls dessen Breite zu dem Wert von *einzug-links* für alle weiteren

Zeilen des Eintrags. Daher verändert sich der Einzug mit der Breite von *text*. Ein solcher Wechsel ist oftmals unerwünscht, aber in manchen Fällen ergeben andere Lösungen noch schlechtere Ergebnisse. Als Beispiel kann man ein Dokument mit vielen Kapiteln betrachten, die jeweils Dutzende von Abschnitten enthalten. Ein fester Abstand für *einzug-links* müsste die breiteste Nummer enthalten können, die aus fünf oder sechs Ziffern bestehen kann. Dadurch erhält eine Gliederungsnummer wie „1.1“ einen unangemessen weiten Abstand von ihrem Eintragstext. Im Folgenden ist eine Lösung aufgeführt, die mit der Länge der Gliederungsnummer wächst.

		<code>\usepackage{titletoc}</code>
		<code>\titlecontents{section}[0pt]{\addvspace{2pt}\filright}</code>
		<code>{\contentspush{\thecontentslabel\ }}{}</code>
		<code>{\titlerule*[8pt]{.}\contentspage}</code>
12.8 Ein Abschnitt, der im		
TOC umbrochen wird	87	<code>\contentsline{section}{\numberline{12.8}Ein Abschnitt,</code>
12.9 Abschnitt	88	<code>der im TOC umbrochen wird}{87}</code>
12.10 Ein weiterer		<code>\contentsline{section}{\numberline{12.9}Abschnitt}{88}</code>
Abschnitt, der		<code>\contentsline{section}{\numberline{12.10}Ein weiterer</code>
umbrochen wird	90	<code>Abschnitt, der umbrochen wird}{90}</code>
12.11 Letzter.	92	<code>\contentsline{section}{\numberline{12.11}Letzter}{92}</code>
		<code>\contentsfinish</code>

Bsp.
2-3-10

```
\contentsmargin[korrektur]{abstand-rechts}
```

Der rechte Rand kann mithilfe der `\contentsmargin`-Deklaration für alle Einträge auf den Wert von *abstand-rechts* gesetzt werden. Die Voreinstellung für diesen Rand entspricht `\@pnumwidth`, und ist damit in den Standardklassen breit genug für drei Ziffern. Das optionale Argument *korrektur* wird an alle Zeilen eines Eintrags, außer der letzten, angefügt. Mit diesem Argument kann man Details für das Verzeichnislayout einstellen. So lassen sich z.B. die Leitpunkte in einem mehrzeiligen Eintrag so ausrichten, dass sie sich am Text der vorhergehenden Zeilen dieses Eintrags orientieren.

In einem Absatz kombinierte Verzeichniseinträge

Standard- $\text{\LaTeX}$ unterstützt nur die zeilenweise Formatierung von Verzeichniseinträgen. In manchen Fällen ist es jedoch ökonomischer, die Einträge für eine niedrigere Gliederungsebene in einem Absatz aneinander zu reihen. Das Paket `titletoc` ermöglicht diese Art der Formatierung.

```
\titlecontents*{typ}[einzug-links]{davor-code}{nummeriert-format}
{nicht-nummeriert-format}{h-code}[mittel-code]
\titlecontents*{typ}...{h-code}[mittel-code][end-code]
\titlecontents*{typ}...{h-code}[anfangs-code][mittel-code][end-code]
```

Die Deklaration `\titlecontents*` wird für Einträge verwendet, die zusammen mit anderen Einträgen der gleichen oder einer niedrigeren Gliederungsebene in einem Absatz formatiert werden sollen. Die ersten sechs Argumente entsprechen denjenigen des Befehls `\titlecontents`, der auf Seite 62 beschrieben ist. Anstelle eines vertikal orientierten *code-unterhalb*-Argumentes

stellt `\titlecontents*` ein bis drei optionale Argumente bereit. Diese behandeln die unterschiedlichen Situationen, die auftreten, wenn Einträge horizontal aneinander gereiht werden. Alle drei optionalen Argumente sind zunächst leer. Der Ablauf ist rekursiv, nach folgendem Verfahren:

- Ist der aktuelle Eintrag der erste, der Teil des Absatzes werden soll, dann wird vor dem Setzen des Eintrags der Inhalt des Argumentes *anfangs-code* ausgeführt.
- Andernfalls gibt es bereits einen früheren Eintrag in diesem Absatz.
 - Wenn beide Einträge zur gleichen Ebene gehören, wird der *mittel-code* eingefügt.
 - Andernfalls, wenn der aktuelle Eintrag zu einer niedrigeren Ebene gehört, wird sein *anfangs-code* eingefügt und das Verfahren beginnt von vorne.
 - Andernfalls gehört der aktuelle Eintrag zu einer höheren Gliederungsebene. Dann wird zunächst für jede beendete Ebene ihr *end-code* (in umgekehrter Reihenfolge) ausgeführt. Wenn der aktuelle Eintrag nicht zum gleichen Absatz gehört, ist das Verfahren damit abgeschlossen. Ansonsten wird der *mittel-code* für den Eintrag ausgeführt, da es bereits einen früheren Eintrag der gleichen Gliederungsebene geben muss (vorausgesetzt es handelt sich um ein hierarchisch strukturiertes Dokument).

Wenn verschiedene Gliederungsebenen in einem Absatz kombiniert werden sollen, muss das gesamte Absatzlayout im Argument *davor-code* der höchsten Ebene angegeben werden, die in einem solchen Absatz vertreten sein kann. Andernfalls haben die Einstellungen für das Ende des Absatzes keine Gültigkeit und werden daher nicht angewandt. Im folgenden Beispiel bezieht sich `\footnotesize` nur auf die Abschnitteinträge – der Wert von `\baselineskip` für den ganzen Absatz entspricht immer noch der Größe `\normalsize`. Dieses künstliche Beispiel zeigt, wie man mithilfe der drei optionalen Argumente zwei verschiedene Ebenen kombinieren kann. Hier sind insbesondere die Leerzeichen am Anfang einiger Argumente zu beachten, die bei der Aneinanderreihung zum richtigen Ergebnis beitragen.

```
\usepackage{titletoc}
\contentsmargin{0pt}
\titlecontents*{chapter}[0pt]{\sffamily}
    {}{}{, \thecontentspage}[ \textbullet\ ][-\P]
\titlecontents*{section}[0pt]{\footnotesize\slshape}
    {}{}{}[ \{}[; ]\}
\contentsline{chapter}{\numberline{1}Erstes}{1}
\contentsline{chapter}{\numberline{2}Zweites}{4}
\contentsline{section}{\numberline{2.1}Abschn-A}{5}
\contentsline{section}{\numberline{2.2}Abschn-B}{6}
\contentsline{chapter}{\numberline{3}Drittes}{8}
\contentsline{section}{\numberline{3.1}Abschn-C}{8}
\contentsfinish
```

Erstes, 1 • Zweites, 4 {Abschn-A;
Abschn-B} • Drittes, 8 {Abschn-C} ¶

Bsp.
2-3-11

Wie lässt sich dieses Verfahren praktisch anwenden? Im nächsten Beispiel wird die Abschnittsebene kombiniert. Dabei werden die Einträge durch Aufzählungszeichen getrennt, die von dehnbaren Abständen (`\xquad`) umgeben sind. Das Verzeichnis endet mit einem Punkt. Die Kapiteleinträge sind ebenfalls interessant, da die Seitennummer links erscheint. In diesem Design gibt es keine Gliederungsnummern. Da auf der rechten Seite keine Seitenzahl steht, wird der rechte Rand auf null gesetzt.

17 Die Struktur eines L^AT_EX-Dokumentes

Der Aufbau der Quelldateien, 17 • Gliederungsbefehle, 24
• Der Aufbau von Verzeichnissen, 48 • Verweise in Dokumenten, 69.

85 Formatierungswerkzeuge

Textfragmente und Absätze, 86 • Fußnoten, Endnoten und Marginalien, 116 • Listen, 135 • Wortwörtlicher Text, 159.

```
\usepackage{titletoc}
\contentsmargin{0pt}
\titlecontents{chapter}[0pt]
  {\addvspace{1.4pc}\bfseries}
  {\Huge\thecontentspage\xquad}{\Huge}
\newcommand\xquad
  {\hspace{1em plus.4em minus.4em}}
\titlecontents*{section}[0pt]
  {\small}{\Huge}
  {\,\sim\thecontentspage}
  [\xquad\textbullet\xquad][.]
\setcounter{tocdepth}{1}
\input{partial.toc}\contentsfinish
```

Das zweite Beispiel zeigt ein Layout, das der Gestaltung in dem Buch *Methods of Book Design* [170] sehr ähnelt. Das Design verwendet Mediävalziffern, die das Paket `eco` bereitstellt. Die Titel der Kapitel (`\chapter`) sind als Kapitälchen formatiert. Das geschieht mithilfe von `\scshape`, wobei alle Buchstaben des Titels mit `\MakeLowercase` in Kleinbuchstaben umgewandelt werden. (Wie bereits erwähnt, kann das letzte Element der Argumente *nummeriert-format* und *nicht-nummeriert-format* ein Befehl mit einem Argument sein, das den Text der Überschrift erhält.) Die Abschnitte werden alle in einem Absatz zusammengefasst, wobei den Abschnittsnummern jeweils ein §-Zeichen vorangestellt wird.

Die Einträge sind durch einen Punkt, gefolgt von einem Leerzeichen, getrennt und der letzte Abschnitt wird ebenfalls mit einem Punkt beendet.

2 DIE STRUKTUR EINES L^AT_EX-DOKUMENTES 17

§2.1 Der Aufbau der Quelldateien, 17. §2.2 Gliederungsbefehle, 24. §2.3 Der Aufbau von Verzeichnissen, 48. §2.4 Verweise in Dokumenten, 69.

3 FORMATIERUNGSWERKZEUGE 85

§3.1 Textfragmente und Absätze, 86. §3.2 Fußnoten, Endnoten und Marginalien, 116. §3.3 Listen, 135. §3.4 Wortwörtlicher Text, 159.

```
\usepackage{eco,titletoc}
\contentsmargin{0pt}
\titlecontents{chapter}[1.5pc]
  {\addvspace{2pc}\large}
  {\contentslabel{2pc}%
   \scshape\MakeLowercase}
  {\scshape\MakeLowercase}
  {\hfill\thecontentspage}
  [\vspace{2pt}]
\titlecontents*{section}[1.5pc]
  {\small}{\S\thecontentslabel\ }
  {\,\sim\thecontentspage}[. ][.]
\setcounter{tocdepth}{1}
\input{partial.toc}
\contentsfinish
```

Bsp.
2-3-12

Bsp.
2-3-13

Erstellen von partiellen Inhaltsverzeichnissen

Mithilfe des Paketes `titletoc` lassen sich partielle Inhaltsverzeichnisse erzeugen. Es stellt für diesen Zweck vier Befehle zur Verfügung.

`\startcontents [name]`

Ein partielles Verzeichnis beginnt mit `\startcontents`. Es ist möglich, parallel Daten für mehrere partielle Verzeichnisse zu sammeln, etwa für den aktuellen Teil (`\part`) und für das aktuelle Kapitel (`\chapter`). Sie lassen sich durch das optionale Argument *name* unterscheiden. Sein Standardwert ist die Zeichenfolge `default`. Gleichzeitig geschriebene partielle Verzeichnisse dürfen sich überlappen, obwohl sie normalerweise verschachtelt sind. Alle Daten für diese Verzeichnisse werden in einer einzigen Datei mit der Erweiterung `.ptc` gespeichert. Diese Datei wird mit dem ersten `\startcontents`-Befehl erzeugt.

`\printcontents [name] {präfix}{start-ebene}{toc-code}`

Dieser Befehl schreibt das aktuelle partielle Verzeichnis, das zuvor mit `\startcontents` erzeugt wurde. Wenn das optionale Argument *name* verwendet wird, muss ein partielles Verzeichnis mit diesem *namen* bereits erzeugt worden sein.¹

Sehr wahrscheinlich soll das partielle Verzeichnis abweichend vom Hauptinhaltsverzeichnis formatiert werden. Um dies zu ermöglichen, wird das Argument *präfix* jedem Eintrag *typ* vorangestellt, wenn nach einer durch `\titlecontents` oder seiner Sternform bereitgestellten Layoutdefinition gesucht wird. Im Beispiel wird `p-` als *präfix* verwendet. Dann wird eine Formatierung für `p-subsection` definiert, um damit `\subsection`-Einträge im partiellen Verzeichnis zu formatieren.

Das Argument *start-ebene* legt fest, ab welcher Gliederungsebene Einträge im partiellen Inhaltsverzeichnis erscheinen. Im Beispiel wurde der Wert auf 2 eingestellt, damit alle Unterabschnitte und die darunter liegenden Ebenen angezeigt werden.

Die Gliederungsebene, bis zu welcher Einträge aufgenommen werden sollen, wird über *toc-code* gesteuert, indem man den Zähler `tocdepth` auf einen geeigneten Wert setzt. An dieser Stelle können noch weitere Einstellungen für die Initialisierung des partiellen Verzeichnisses vorgenommen werden. Im Beispiel wird jeder rechte Rand gelöscht, da das partielle Verzeichnis insgesamt als ein Absatz formatiert wird.

Wenn man die Pakete `titletoc` und `titlesec` gemeinsam verwendet, kann man sehr leicht partielle Verzeichnisse in Definitionen für Überschriften integrieren, ohne dass man dazu das eigentliche Dokument verändern muss. Im Folgenden wird Beispiel 2-2-18 von Seite 42 so erweitert, dass der Befehl `\section` automatisch ein partielles Verzeichnis all seiner Unterabschnitte ausgibt. Dies wird mithilfe des optionalen Argumentes *danach-code* der `\titleformat`-Deklaration erreicht. Zunächst wird ein vertikaler

¹Zurzeit (Stand 2003) ist das Paket ziemlich intolerant, wenn man versucht eine Verzeichnisliste zu schreiben, die man nicht erst erzeugt hat: In diesem Fall tritt ein unspezifischer T_EX-Low-Level-Fehler auf.

Abstand hinzugefügt, der dafür sorgt, dass an dieser Stelle kein Seitenumbruch erfolgen kann. Dann wird das partielle Verzeichnis mit seinen default-Einstellungen von `\startcontents` (erneut) aufgerufen. Danach wird es über den Befehl `\printcontents` sofort formatiert. Die entsprechenden Argumente wurden zuvor bereits erläutert. Schließlich formatiert `\titlecontents*` die Unterabschnitte im partiellen Inhaltsverzeichnis als einen zusammenhängenden Absatz. Dieser ist im Blocksatz gesetzt, mit zentrierter letzter Zeile (`\fillast`). Reiht man diese Vorgänge aneinander, so erhält man das gewünschte Ergebnis, ohne das Quelldokument in irgendeiner Weise verändern zu müssen. Ein echtes Design würde natürlich auch das Erscheinungsbild der Überschriften für die Unterabschnitte mit jenem der Abschnittsüberschriften harmonisieren.

ABSCHNITT 1
Ein Titel-Test

*Titel Eins – Längerer Titel Zwei –
Noch längerer Titel Vier.*

Dieser Text zeigt, dass der Absatz nicht eingezogen ist.

1.1 Titel Eins

Etwas Text ...

```
\usepackage{titlesec,titletoc}
\titleformat{\section}[frame]{\normalfont}
    {\footnotesize \enspace ABSCHNITT \thesection
    \enspace}{6pt}{\large\bfseries\filcenter}
    [\vspace*{5pt}\startcontents
    \printcontents{p-}{2}{\contentsmargin{0pt}}]
\titlespacing*{\section}{1pc}{*4}{*2.3}{1pc}
\titlecontents*{p-subsection}{0pt}
    {\small\itshape\fillast}{-}{-}{-}[.][.]
\section{Ein Titel-Test}
Dieser Text zeigt, dass der Absatz nicht eingezogen ist.
\subsection{Titel Eins} Etwas Text \ldots \newpage
\subsection{Längerer Titel Zwei} Noch etwas Text.
\stopcontents \subsection{Titel Drei} \resumecontents
\subsection{Noch längerer Titel Vier}
```

Falls erforderlich, kann man das Sammeln von Einträgen für ein partielles Verzeichnis zeitweise (oder endgültig) einstellen. Im vorigen Beispiel wurde der dritte Unterabschnitt mithilfe der entsprechenden Funktion unterdrückt.

`\stopcontents [name]` `\resumecontents [name]`

Der Befehl `\stopcontents` unterbindet das Sammeln der Einträge für das partielle Standardverzeichnis, oder für das partielle Verzeichnis, dessen Name gegebenenfalls im optionalen Argument *name* angegeben wurde. Zu einem späteren Zeitpunkt kann man das Sammeln von Einträgen mit dem Befehl `\resumecontents` wieder aufnehmen. Dieser Befehl ist nicht zu verwechseln mit `\startcontents`, welcher ein *neues* partielles Inhaltsverzeichnis anlegt und die alten Einträge damit unzugänglich macht.

2.4 Verweise in Dokumenten

L^AT_EX verfügt über Befehle, die das Verwalten von Verweisen in einem Dokument vereinfachen. Dabei werden insbesondere *Querverweise* (interne Verweise zwischen Elementen innerhalb eines Dokumentes), *Literaturverweise* (Verweise auf externe Dokumente) und *Indexeinträge* von ausgewählten Worten

Bsp.
2-3-14

oder Ausdrücken unterstützt. Die Funktionen zum Erstellen von Stichwortverzeichnissen werden in Kapitel 11 erläutert, die Funktionen für Literaturverweise in Kapiteln 12 und 13.

Um Querverweise zu Elementen innerhalb eines Dokumentes zu ermöglichen, wird dem jeweiligen Strukturelement zunächst ein „Schlüssel“ (aus einer Folge von ASCII-Zeichen, Zahlen und Punkten) zugewiesen. Dieser Schlüssel wird dann an anderer Stelle verwendet, um auf das Element zu verweisen

```
\label{schlüssel} \ref{schlüssel} \pageref{schlüssel}
```

Der Befehl `\label` weist dem aktuellen „aktiven“ Element eines Dokumentes den im Argument *schlüssel* angegebenen Schlüssel zu. (Welches Element an einem gegebenen Punkt gerade aktiv ist, wird im Folgenden erklärt.) Der Befehl `\ref` erzeugt eine Zeichenfolge, die das gegebene Element identifiziert – wie z.B. die Abschnitts-, Gleichungs- oder Abbildungsnummer – je nachdem welches Strukturelement aktiv war, als der Befehl `\label` gegeben wurde. Der Befehl `\pageref` erzeugt die Seitenzahl der Seite, auf welcher der Befehl `\label` gesetzt wurde. Die Zeichenfolge *schlüssel* muss dabei natürlich eindeutig sein. Ein Präfix mit einer Zeichenfolge, welche die Struktur des fraglichen Elementes identifiziert, ist dabei eine einfache aber wirkungsvolle Hilfe. So kann zum Beispiel *sec* für Abschnitte (sections) verwendet werden, *fig* für Abbildungen (figures), und so fort.

4 Abschnitt

Ein Querverweis auf diesen Abschnitt sieht folgendermaßen aus: „siehe Abschnitt 4 auf Seite 6“.

```
\section{Abschnitt} \label{sec:hier}
Ein Querverweis auf diesen Abschnitt
sieht folgendermaßen aus:
"‘siehe Abschnitt~\ref{sec:hier} auf
Seite~\pageref{sec:hier}’".
```

Bsp.
2-4-1

In Schlüsseln
bedingt zulässige
Zeichen

Es kann riskant sein, Satzzeichen wie etwa einen Doppelpunkt zu verwenden. In bestimmten Sprachen-Formatvorlagen des `babel`-Paketes (siehe Kapitel 9) haben einige dieser Zeichen eine besondere Bedeutung und verhalten sich grundlegend wie Befehle. Zwar bemüht sich das Paket, solche Zeichen als Teile von `\label`-Schlüsseln zuzulassen, aber diese Toleranz stößt gelegentlich an ihre Grenzen. Auch Zeichen, die nicht zum ASCII-Bereich gehören, und über Pakete wie `inputenc` verfügbar sind, werden in solchen Schlüsseln offiziell nicht unterstützt. Ihre Verwendung führt wahrscheinlich zu Fehlern.

Das „zurzeit aktive“ Strukturelement, mit dem ein Querverweislabel erstellt wird, ist folgendermaßen festgelegt: Die Gliederungsbefehle (`\chapter`, `\section`, ...), die Umgebungen `equation`, `figure`, `table` und die `theorem`-Familie sowie die verschiedenen Ebenen der Umgebung `enumerate` und `\footnote` setzen den *aktuellen Bezug*, der die Nummer enthält, die L^AT_EX für das gegebene Element generiert. Dieser Bezug wird normalerweise am Anfang eines Elementes gesetzt, und zurückgesetzt, wenn der Geltungsbereich dieses Elementes verlassen wird

Probleme bei
Querverweisen auf
Gleitobjekte

Ausnahmen von dieser Regel sind die Umgebungen `table` und `figure`, bei denen der aktuelle Bezug durch den Befehl `\caption` gesetzt wird. Dadurch werden mehrere Paare von `\caption` und `\label`-Befehlen innerhalb

einer Umgebung ermöglicht.¹ Da `\caption` die Nummerierung des Bezugs setzt, muss der zugehörige `\label`-Befehl auf `\caption folgen`. Andernfalls wird eine falsche Bezugsnummer gesetzt. Wenn der `\label`-Befehl früher im Gleitobjekt auftaucht, nimmt er den *aktuellen Bezug* eines vorhergehenden Elementes auf, normalerweise, den des Gliederungselementes.

Dies wird anhand des folgenden Beispiels deutlich, wo nur die Label „fig:in2“ und „fig:in3“ richtig gesetzt sind, um die erforderlichen Bezugsnummern für die Abbildungen zu erzeugen. Im Falle des Labels „fig:in4“ zeigt sich, dass Umgebungen (hier `center`) den Geltungsbereich von Bezügen beenden, da dieses Label die Nummer des aktuellen Abschnitts ergibt und nicht die Nummer der Abbildung.

3 Ein Abschnitt

3.1 Ein Unterabschnitt

Vorhergehender Text siehe ‘3.1’.

... Abbildung ...

Abb. 1: Bildunterschrift 1

... Abbildung ...

Abb. 2: Bildunterschrift 2

Verwendete Label: ‘vorher’ (3.1),
‘fig:in1’ (3.1), ‘fig:in2’ (1), ‘fig:in3’ (2),
‘fig:in4’ (3.1), ‘danach’ (3.1).

Bsp.
2-4-2

```
\section{Ein Abschnitt}
\subsection{Ein Unterabschnitt}\label{sec:before}
Vorhergehender Text siehe ‘\ref{sec:before}’.
```

```
\begin{figure}[ht]                                \label{fig:in1}
\begin{center}
\fbbox{\ldots}{ Abbildung \ldots}
\caption{Bildunterschrift 1} \label{fig:in2}
\bigskip
\fbbox{\ldots}{ Abbildung \ldots}
\caption{Bildunterschrift 2} \label{fig:in3}
\end{center}                                       \label{fig:in4}
\end{figure}
\label{sec:after}
```

```
\raggedright
Verwendete Label: ‘vorher’ (\ref{sec:before}),
‘fig:in1’ (\ref{fig:in1}), ‘fig:in2’
(\ref{fig:in2}), ‘fig:in3’ (\ref{fig:in3}),
‘fig:in4’ (\ref{fig:in4}), ‘danach’
(\ref{sec:after}).
```

Für jedes Argument *schlüssel*, das mit `\label{schlüssel}` deklariert wird, speichert $\LaTeX$ den aktuellen Bezug und die Seitenzahl. Mehrere `\label`-Befehle (mit verschiedenen Schlüsseln) innerhalb eines Abschnitts erzeugen also die gleiche Bezugszahl für das Strukturelement, möglicherweise jedoch mit unterschiedlichen Seitenzahlen

2.4.1 showkeys – Anzeigen der Querverweisschlüssel

Viele Anwender drucken Zwischenentwürfe aus, wenn sie längere Dokumente schreiben. Es wäre hilfreich, wenn in solchen Zwischenentwürfen die `\label`-Befehle und ihre Schlüssel sichtbar wären. Das Paket `showkeys` von David Carlisle unterstützt diese Funktion.

¹Es gibt jedoch gute Gründe, warum man nicht mehr als einen `\caption`-Befehl in einer Gleitobjektumgebung verwenden sollte. Es erschwert die korrekte räumliche Gestaltung, und, was noch wichtiger ist, zukünftige Versionen von $\LaTeX$ könnten diese Syntax nicht mehr akzeptieren.

Wenn das Paket geladen wird, verändert es die Befehle `\label`, `\ref`, `\pageref`, `\cite` und `\bibitem` so, dass die verwendeten Schlüssel mit ausgegeben werden. Die Befehle `\label` und `\bibitem` setzen Schlüssel normalerweise in den Rand in eine Box. Die Befehle, die auf die Schlüssel verweisen, schreiben sie in kleiner Schriftgröße über den formatierten Verweistext (wobei sie möglicherweise den vorhandenen Text zum Teil überschreiben). Das Paket versucht die Schlüssel so anzuzeigen, dass die Formatierung des übrigen Textes nicht verändert wird. Dafür gibt es jedoch keine Garantie. Daher sollte man das Paket `showkeys` entfernen oder deaktivieren, bevor man die endgültige Formatierung für ein Dokument vornimmt.

1 Ein Beispiel

sec

Abschnitt ^{sec} II zeigt den Einsatz des `showkeys`-Paketes durch Verweis auf Gleichung ^{eq} (II).

$$a = b$$

(1) eq

`\usepackage{showkeys}`

`\section{Ein Beispiel}\label{sec}`

Abschnitt~\ref{sec} zeigt den Einsatz

des `\texttt{showkeys}`-Paketes durch

Verweis auf Gleichung~(\ref{eq}).

`\begin{equation}`

`a = b \label{eq}`

`\end{equation}`

Bsp.
2-4-3

Das Paket unterstützt die Option `fleqn` der Standardklassen und arbeitet mit $\mathcal{A}\mathcal{M}\mathcal{S}$ -L^AT_EX, `varioref`, `natbib` und vielen weiteren Paketen zusammen. Es ist jedoch fast unmöglich ein fehlerloses Zusammenspiel mit allen Paketen, die mit den Querverweismechanismen arbeiten, zu gewährleisten.

Wer nur die Schlüssel des `\label`-Befehls am Rand sehen möchte, kann die anderen Anzeigen mit der Paketoption `notref` unterdrücken. Sie deaktiviert das Undefinieren von `\ref`, `\pageref` und verwandten Befehlen. Die Option `notcite` hat die gleichen Auswirkungen auf `\cite` und die entsprechenden Befehle der Pakete `natbib` und `harvard`. Alternativ dazu kann man die Label mit der Option `color` weniger störend gestalten.

Außerdem unterstützt das Paket noch die Optionen `draft` (Standard) und `final`. Letztere Option ist auf Paketebene nutzlos, da es dort ebensogut ausreicht, das Paket `showkeys` gar nicht erst einzubinden. Als globale Option einer Klasse ist `final` jedoch sehr praktisch.

2.4.2 varioref – Flexiblere Querverweise

In vielen Fällen ist es hilfreich, einen Verweis auf eine Abbildung oder Tabelle sowohl mit `\ref` als auch `\pageref` zu bilden, insbesondere, wenn eine oder mehrere Seiten zwischen Verweis und Objekt liegen. Daher verwenden manche Anwender einen Befehl wie

`\newcommand\fullref[1]{\ref{#1} auf Seite~\pageref{#1}}`

um die Länge der Eingabe, die ein vollständiger Verweis erfordert, zu reduzieren. Da man aber niemals genau weiß, wo das Objekt eines Verweises letztendlich platziert wird, kann diese Methode auch einen Verweis auf die aktuelle Seite erzeugen. Dies wäre eher störend und sollte daher vermieden werden. Das Paket `varioref` von Frank Mittelbach versucht, dieses Problem zu lösen. Für Einzelverweise stellt es die Befehle `\vref` und `\vpageref` zur Verfügung.

Mehrfache Verweise werden mit `\vrefrange` und `\vpagerefrange` behandelt. In Abhängigkeit von dem Zähler, der für einen Verweis verwendet wird, kann man Querverweise außerdem mit der Deklaration `\labelformat` unterschiedlich formatieren.

`\vref*{schlüssel}`

Der Befehl `\vref` entspricht `\ref`, wenn sich Verweis und `\label` auf der gleichen Seite befinden. Liegen sie eine Seite auseinander, erzeugt `\vref` eine dieser Zeichenfolgen: „on the facing page“ (auf der gegenüberliegenden Seite)¹, „on the preceding page“ (auf der vorherigen Seite) oder „on the following page“ (auf der nächsten Seite). Das Wort „facing“ (gegenüberliegend) wird verwendet, wenn Label und Verweis auf eine Doppelseite fallen. Ist der Abstand größer als eine Seite, stellt `\vref` den Verweis aus `\ref` und `\pageref` zusammen. Wenn allerdings anstatt der herkömmlichen Seitennummerierung mit arabischen Ziffern ein anderes Nummerierungssystem verwendet wird (z.B. `\pagenumbering{roman}`), wird nicht zwischen einer oder mehreren Seiten Abstand unterschieden.

Es gibt einen weiteren Unterschied zwischen `\ref` und `\vref`: Letzterer ersetzt alle vorhergehenden Abstände durch seine eigenen Einstellungen. An einigen Stellen, wie z.B. direkt nach einer öffnenden Klammer, ist dies jedoch unerwünscht. In solchen Fällen empfiehlt es sich, `\vref*` zu verwenden. Er verhält sich genau wie `\vref`, fügt jedoch keine Leerräume vor dem erzeugten Text ein.

`\vpageref*[diese-seite] [andere-seite] {schlüssel}`

Manchmal möchte man nur auf eine Seitenzahl verweisen. Dann sollte der Verweis unterdrückt werden, sofern er sich auf die aktuelle Seite bezieht. Zu diesem Zweck wurde der Befehl `\vpageref` definiert. Er erzeugt die gleichen Ausdrücke wie `\vref`, beginnt jedoch nicht mit `\ref`. Wenn sich Bezug und Label auf der gleichen Seite befinden, verwendet er den Ausdruck, der unter `\reftextcurrent` gespeichert ist.

Wenn man für `\reftextcurrent` einen Ausdruck wie „on this page“ (auf dieser Seite) definiert, kann man vermeiden, dass eine Formulierung wie

... see the diagram `\vpageref{ex:foo}` which shows ...

den Text „... see the diagram which shows ...“ (siehe das Diagramm, welches zeigt, wie) erzeugt, was irreführend sein könnte. Gleiches gilt in anderen Sprachen, wenn man `varioref` mit einer Sprachoption lädt oder die erzeugten Texte anpasst. Siehe dazu Seite 77.

Man kann ein Leerzeichen vor `\vpageref` platzieren. Wenn der Befehl keinen Text erzeugt, wird das Leerzeichen ignoriert. Wenn Text eingefügt wird, fügt der Befehl automatisch einen geeigneten Abstand vor dem Text ein. Die

¹Das Paket erzeugt im Standardmodus englische Texte. Diese können jedoch mithilfe von $\TeX$ -Funktionen für die Unterstützung anderer Sprachen geändert werden. Weitere Informationen hierzu befinden sich im Abschnitt Sprachenunterstützung auf Seite 77. In den folgenden Beispielen werden zunächst die englischen Verweistexte verwendet.

Variante `\vpageref*` entfernt Weißraum vor dem erzeugten Text, ohne eigenen einzufügen. Sie lässt sich also sehr gut verwenden, wenn das Einfügen eines Abstandes Probleme bereiten würde.

Mit den zwei optionalen Argumenten der Befehle `\vpageref` und `\vpageref*` kann man Verweise sogar noch genauer steuern. Das erste Argument gibt den Text an, der eingesetzt werden soll, wenn sich Bezug und Label auf der gleichen Seite befinden. Das ist besonders hilfreich, wenn beide so nahe beieinander liegen, dass sie nur eventuell durch einen Seitenumbruch getrennt werden. In einem solchen Fall weiß man normalerweise, ob der Verweis vor oder nach dem Label erscheint, so dass man eine Formulierung, wie die folgende erzeugen kann:

```
... see the diagram \vpageref[above]{ex:foo} which shows ...
```

Der Ergebnistext ist in diesem Fall: „... see the diagram above which shows ...“, wenn beide auf der gleichen Seite erscheinen, oder „... see the diagram on the page before which shows ...“, wenn sie durch einen Seitenumbruch getrennt sind (oder ähnlich, je nachdem welcher Text für die Befehle `\ref{...}` `..before` und `..after` definiert wurde). Wenn man die optionalen Argumente von `\vpageref` für Gleitobjekte, wie Abbildungen oder Tabellen, verwendet, ist dabei jedoch Folgendes zu beachten: Je nach Einstellung der Parameter zum Positionieren von Gleitobjekten kann das entsprechende Objekt am Anfang der aktuellen Seite, also vor dem Verweis, erscheinen, obwohl sich das Label im Quelltext hinter dem Verweis befindet.¹

Vielleicht zieht man sogar die Formulierung „... see the above diagram“, also eine geänderte Wortwahl und Reihenfolge, vor, wenn Diagramm und Verweis auf der gleichen Seite erscheinen. Tatsächlich ist eine solche Umformulierung in einigen Sprachen sogar erforderlich. Für diese Abwandlung kann das zweite Argument *andere-seite* verwendet werden. Es gibt den Text an, der dem erzeugten Verweis vorangeht, wenn Objekt und Verweis nicht auf der gleichen Seite erscheinen. Um den gewünschten Effekt zu erzielen würde man in diesem Falle also schreiben:

```
... see the \vpageref[above diagram][diagram]{ex:foo}
which shows ...
```

Das Paket `amsmath` stellt den Befehl `\eqref` für Verweise auf Gleichungen bereit. Es versieht die Formelnummer automatisch mit Klammern. Mit der folgenden Definition könnte man außerdem automatisch eine Seitenreferenz hinzufügen:

```
\newcommand\eqvref[1]{\eqref{#1}\ \vpageref{#1}}
```

```
\vrefrange[text-hier]{start-schlüssel}{end-schlüssel}
```

Dieser Befehl ähnelt `\vref`. Er hat jedoch zwei obligatorische Argumente, die sich auf einen Objektbereich beziehen (z.B. eine Reihe von Abbildungen oder

¹Um sicherzustellen, dass Gleitobjekte immer erst hinter ihrer Position im Quelltext gesetzt werden, kann man das Paket `flafter` verwenden, das in Abschnitt 6.2 beschrieben wird.

eine Abfolge von Gleichungen). Er verwendet den Verweistext in Abhängigkeit davon, wie die beiden Label zueinander positioniert sind und bezieht seine Funktionalität hauptsächlich über den Befehl `\vpagerefrange` (der im Folgenden noch beschrieben wird). Das optionale Argument des Befehls enthält den Text, der eingesetzt werden soll, wenn beide Label auf der gleichen Seite liegen. Für diesen Zweck ist die in `\reftextcurrent` gespeicherte Zeichenfolge voreingestellt.

1 Test

Man beachte Gleichungen 1.1 bis 1.3 auf Seiten 6–7 und auch Gleichungen 1.2 bis 1.3 auf der nächsten Seite.

$$a = b + c \quad (1.1)$$

6

Eine zweite Gleichung...

$$b = a + c \quad (1.2)$$

...und noch eine Gleichung:

$$c = a + b \quad (1.3)$$

7

```
\usepackage[ngerman]{varioref}
\renewcommand\theequation
{\thesection.\arabic{equation}}

\section{Test}      Man beachte
Gleichungen~\vrefrange{A}{C} und
auch Gleichungen~\vrefrange{B}{C}.
\begin{equation}
a=b+c\label{A} \end{equation}
Eine zweite Gleichung\ldots
\begin{equation}
b=a+c\label{B} \end{equation}
\ldots und noch eine Gleichung:
\begin{equation}
c=a+b\label{C} \end{equation}
```

Bsp.
2-4-4

`\vpagerefrange*[text-hier]{start-schlüssel}{end-schlüssel}`

Dieser Befehl ähnelt `\vpageref`, besitzt jedoch zwei obligatorische Argumente und zwar zwei Label, die einen Bereich festlegen. Wenn beide Label auf der gleichen Seite liegen, verhält sich der Befehl genauso wie `\vpageref` mit nur einem Label. Andernfalls erzeugt er einen Verweis wie etwa „on pages 15–18“ („auf den Seiten 15–18“; vgl. die Anpassungsmöglichkeiten, die im Folgenden beschrieben werden). Genau wie `\vrefrange` verfügt er über ein optionales Argument, das auf den Text von `\reftextcurrent` voreingestellt ist und eingesetzt wird, wenn sich beide Label auf der gleichen Seite befinden.

Auch hier existiert eine Sternform, `\vpagerefrange*`, die dem erzeugten Text vorangehenden Weißraum entfernt, ohne dafür einen eigenen Abstand einzufügen.

Die Standardverweise, die mit `\ref` erzeugt werden, bestehen aus den Daten des entsprechenden `\label`-Befehls (üblicherweise eine Nummer). Ihre weitere Formatierung liegt beim Anwender. Ein Verweis auf Gleichungen, der immer in der Form „Gleichung (nummer)“ erscheinen soll, muss jedes Mal folgendermaßen geschrieben werden: „Gleichung (`\ref{key}`)“.

Zielabel

`\labelformat{zähler}{formatierungs-code}`

Das Paket `varioref` bietet mit dem Befehl `\labelformat` eine Möglichkeit, solches Beiwerk automatisch zu erzeugen.¹ Der Befehl besitzt zwei Argumente: Den Namen eines Zählers und seine Darstellung, wenn auf ihn Bezug genommen wird. Man muss also den Namen des Zählers kennen, um ihn einsetzen

¹Dieser Befehl ist auch separat über das Paket `fnclab` von Robin Fairbairns verfügbar.

zu können. Das sollte in der Praxis kein Problem sein. Der aktuelle Zähler, genauer gesagt seine Darstellung, wird als Argument übernommen, so dass das zweite Argument #1 enthalten sollte.

Als Nebeneffekt von `\labelformat` kann man `\ref`, je nach Formatierung, nicht am Anfang eines Satzes verwenden, wenn der eingesetzte Text klein geschrieben ist. Um diesem Problem entgegenzuwirken, stellt `varioref` die Befehle `\Ref` und `\Vref` (inklusive ihrer Sternformen) zur Verfügung, die sich wie `\ref` und `\vref` verhalten, jedoch das erste Zeichen der erzeugten Zeichenfolge groß schreiben. Im folgenden Beispiel kann man dieses Verhalten beobachten, wenn „in Abschnitt“ als „In Abschnitt“ erscheint.

1 Ein Beispiel

In Abschnitt 1 sieht man die `\labelformat`-Deklaration mit einem Verweis auf Informationen in Gleichung (1).

$$a = b$$

(1) `\begin{equation}a = b\label{eq}\end{equation}`

```
\usepackage{varioref}
\labelformat{section}{in Abschnitt~#1}
\labelformat{equation}{in Gleichung~(#1)}
\section{Ein Beispiel}\label{sec}
\Ref{sec} sieht man die
\verb=\labelformat=-Deklaration mit einem
Verweis auf Informationen \ref{eq}.
```

Bsp.
2-4-5

`\Ref` und `\Vref` funktionieren nur dann korrekt, wenn das erste Zeichen in `\labelformat` ein einfaches ASCII-Zeichen ist. Andernfalls führt der Befehl nicht zur Großschreibung sondern schlimmstenfalls zu einigen Fehlermeldungen. Wenn an dieser Stelle ein komplexes Zeichen (z.B. ein Buchstabe mit Akzent) steht, muss dieses explizit in Klammern gesetzt werden, um den Teil zu kennzeichnen, der in Großschreibung gesetzt werden soll. Für Verweise auf Abbildungen im Ungarischen könnte man z.B. folgenden Ausdruck formulieren: `\labelformat{figure}{\{'a\}bra~\thefigure}`. Man beachte auch, dass die beiden Befehle mit älteren Versionen von `babel` nicht korrekt zusammenarbeiten.

Aktuelle
babel-Version nötig! 

Als zweites Beispiel für den Einsatz von `\labelformat` soll folgende Situation betrachtet werden: In den Dokumentenklassen `report` und `book` werden Fußnoten kapitelweise nummeriert. Dadurch werden Querverweise auf Fußnoten zweideutig, da nicht klar ist, ob sie sich auf einen Eintrag im aktuellen oder in einem anderen Kapitel beziehen. Diese Zweideutigkeit kann man auflösen, indem man im Verweis immer die Kapitelnummer nennt, oder indem man die Kapitelnummer des `\labels` mit der aktuellen vergleicht und erstere nur hinzufügt, wenn beide voneinander abweichen. Dieses Verhalten wird durch folgenden Code erzeugt:

```
\usepackage{ifthen,varioref}
\labelformat{footnote}{#1\protect\iscurrentchapter{\thechapter}}
\newcommand\iscurrentchapter[1]{%
  \ifthenelse{equal{#1}{\thechapter}}{}{ in Kapitel~#1}}
```

Der Trick besteht darin, mit einem `\protect`-Befehl zu verhindern, dass der Befehl `\iscurrentchapter` beim Erzeugen des Labels ausgewertet wird. Wenn dann der Befehl `\ref` ausgeführt wird, vergleicht `\iscurrentchapter` sein Argument (also die Kapitelnummer des Labels) mit der aktuellen Kapitelnummer, und setzt, sofern sie sich unterscheiden, den passenden Text.

Das Paket stellt außerdem den Befehl `\vrefpagemum` bereit, mit dem man eigene kleine Befehle schreiben kann. Damit lassen sich Funktionen wie die beiden zuletzt beschriebenen realisieren. Der Befehl verfügt über zwei Argumente: Das zweite ist ein Label (wie es in `\label` oder `\ref` verwendet wird) und das erste ist ein beliebiger Befehlsname (es muss ein eigener Name sein), der die Seitenzahl für das Label erhält. Wenn man also zwei (oder mehr) Label hat, kann man ihre Seitenzahlen auslesen, vergleichen und dann entscheiden, was gesetzt werden soll.

*Eigene Befehle für
Querverweise
formulieren*

Das nächste Beispiel zeigt ein nicht allzu ernst gemeintes Verfahren, das die Label von zwei Formeln miteinander vergleicht und entsprechend ihrer Position zueinander einen bestimmten Text ausgibt. Dabei sollte man die Ergebnisse auf der ersten Seite mit denen der zweiten Seite vergleichen.

Test: die Gleichungen 1 und 2 auf dieser Seite Test: die Gleichung 1 auf dieser Seite und 3 auf Seite 8 $a = b + c \quad (1)$ $b = a + c \quad (2)$	Test: die Gleichungen 1 und 2 auf der vorherigen Seite Test: die Gleichung 1 auf der vorherigen Seite und 3 auf der nächsten Seite
6	7

Bsp.
2-4-6

```
\usepackage[ngerman]{varioref}
\usepackage{ifthen}
\newcommand\veqns[2]{%
  \vrefpagemum\firstnum{#1}%
  \vrefpagemum\secondnum{#2}%
  die Gleichung%
  \ifthenelse{equal\firstnum\secondnum}%
    {en \ref{#1}}%
    { \ref{#1}\vpageref{#1}}%
  \space und \ref{#2}\vpageref{#2}}
Test: \veqns{A}{B} \par Test: \veqns{A}{C}
\begin{equation} a=b+c \label{A}\end{equation}
\begin{equation} b=a+c \label{B}\end{equation}
\newpage
Test: \veqns{A}{B} \par Test: \veqns{A}{C}
\newpage
\begin{equation} c=a+b \label{C}\end{equation}
```

Das Paket unterstützt die Optionen des babel-Systems (siehe Abschnitt 9.1.3). Die Deklaration `\usepackage[ngerman]{varioref}` erzeugt somit passende Texte für die deutsche Sprache. Außerdem unterstützt das Paket die Optionen `final` (voreingestellt) und `draft`. Letztere ändert bestimmte Fehlermeldungen (wie beschrieben auf Seite 79) in Warnungen. Diese Funktion kann beim Schreiben eines Dokumentes hilfreich sein.

Paketoptionen

Um eine weitere Anpassung zu ermöglichen, sind alle Zeichenfolgen durch Makros festgelegt (welche durch die Sprachenoptionen vordefiniert werden). Rückbezüge verwenden `\reftextbefore`, wenn sich das Label auf der nicht sichtbaren, vorherigen Seite befindet und `\reftextfacebefore`, wenn es auf der vorherigen, gegenüberliegenden Seite erscheint (wenn also die Seitenzahl ungerade ist).

*Individuelle
Anpassungen*

Ebenso wird `\reftextafter` verwendet, wenn das Label auf der nächsten Seite, zu der man umblättern muss, erscheint, und `\reftextfaceafter`, wenn es sich auf der nächsten, gegenüberliegenden Seite befindet. Diese vier Formulierungen können mit `\renewcommand` verändert werden.

Der Befehl `\reftextfaraway` wird verwendet, wenn Label und Verweis mehr als eine Seite auseinanderliegen oder nicht numerisch sind. Dieses Makro unterscheidet sich von den bisherigen darin, dass es über ein Argument,

das symbolische Label, verfügt, so dass man in seinem Ersetzungstext den Befehl `\pageref` verwenden kann.

Wenn man diese Makros z.B. in deutschen Dokumenten einsetzen möchte, würde man eine Formulierung wie die folgende verwenden:

```
\renewcommand\reftextfaraway[1]{auf Seite~\pageref{#1}}
```

Der Befehl `\reftextpagerange` hat zwei Argumente und erzeugt Text, der auf einen Seitenbereich verweist (die Argumente sind Schlüssel für `\pageref`). Weiter unten folgt eine Auflistung der deutschen Standardtexte.

Auch `\reftextlabelrange` verfügt über zwei Argumente. Dieser Befehl beschreibt eine Serie von Abbildungen, Tabellen oder anderen Elementen, auf die sich die Label beziehen. Der englische Standardtext ist hier „`\ref{#1}` to~`\ref{#2}`“ (`\ref{#1}` bis~`\ref{#2}`).

Um zufällige Variationen der generierten Texte zu erhalten, kann man innerhalb der Makros für die Verweistexte den Befehl `\reftextvario` verwenden. Der Befehl hat zwei Argumente, von denen er mal das eine und mal das andere für den Ausdruck einsetzt, je nachdem, wie häufig die Befehle `\vref` oder `\vpageref` bereits in einem Dokument verwendet wurden.

In diesem Buch wurden folgende Einstellungen für die verschiedenen Makros verwendet:

```
\renewcommand\reftextfaceafter {auf der
    \reftextvario{nächsten}{gegenüberliegenden} Seite}
\renewcommand\reftextfacebefore{auf der
    \reftextvario{vorherigen}{gegenüberliegenden} Seite}
\renewcommand\reftextafter{auf der nächsten Seite}
\renewcommand\reftextbefore{auf der vorherigen Seite}
\renewcommand\reftextcurrent{auf dieser Seite}
\renewcommand\reftextfaraway[1]{auf Seite~\pageref{#1}}
\renewcommand\reftextpagerange[2]{auf den
    Seiten~\pageref{#1}--\pageref{#2}}
\renewcommand\reftextlabelrange[2]{\ref{#1} bis~\ref{#2}}
```

Wenn man also das Paket seinen eigenen Vorlieben anpassen möchte, braucht man nur die entsprechenden neuen Definitionen für die oben aufgeführten Befehle in eine Datei mit der Erweiterung `.sty` (z.B. `vrlocal.sty`) zu schreiben. Wenn man außerdem die Anweisung `\RequirePackage{varioref}` (siehe Abschnitt A.4 auf Seite 910) an den Beginn der Datei setzt, lädt dieses lokale Paket automatisch das Paket `varioref`. Bei Verwendung des `babel`-Systems sollten neue Definitionen für einzelne Sprachen mit `\addto` hinzugefügt werden. Dieses Verfahren wird in Abschnitt 9.5 erklärt.

Manch einer mag vielleicht keine textuellen Querverweise, möchte jedoch Seitenverweise unterdrücken, wenn Label und Bezug auf der gleichen Seite liegen. Dies lässt sich folgendermaßen, mithilfe des Befehls `\thevpagerefnum`, erreichen:

```
\renewcommand\reftextfaceafter {auf Seite~\thevpagerefnum}
\renewcommand\reftextfacebefore{auf Seite~\thevpagerefnum}
```

```
\renewcommand\reftextafter      {auf Seite~\thepagerefnum}
\renewcommand\reftextbefore     {auf Seite~\thepagerefnum}
```

Innerhalb eines `\reftext...`-Befehls ergibt `\thepagerefnum` die aktuelle Seitenzahl, sofern diese bekannt ist, oder sonst zwei Fragezeichen.

Eine andere interessante Konfiguration besteht darin, die Verweise dann zu unterdrücken, wenn das Ziel des Verweises auf einer gegenüberliegenden Seite liegt, also sichtbar ist. Dazu muss das von `\vpageref` normalerweise eingefügte Leerzeichen wieder entfernt werden:

*Kein Verweis bei
sichtbarem Ziel*

```
\renewcommand\reftextfaceafter {\unskip}
\renewcommand\reftextfacebefore{\unskip}
```

Wenn man Befehle wie die zuvor beschriebenen definiert, wird man mit einigen interessanten Problemen konfrontiert. Ein generierter Text wie „auf der nächsten Seite“ könnte z.B. über zwei Seiten umbrochen werden. Dieses Problem lässt sich durch einen Algorithmus kaum befriedigend lösen; eine solche Situation kann sogar dazu führen, dass ein Dokument ständig von einem Status in den anderen wechselt (d.h. ein Verweistext wird eingefügt; dieser erweist sich als falsch; im nächsten Lauf wird ein anderer eingefügt, wodurch der erste wieder richtig wird; ein Text wird...). Die aktuelle Version des Paketes `varioref` orientiert sich am Ende des erzeugten Verweistextes. Das Beispiel

 *Einige
Warnungen*

Tabelle 5 auf dieser *(Seitenumbruch)* Seite

wäre also korrekt, wenn sich Tabelle 5 auf der gleichen Seite wie das Wort „Seite“ befindet, nicht wie das Wort „dieser“. Dieses Verhalten ist jedoch nicht immer wirklich zufriedenstellend und kann in manchen Fällen zu einer Schleife führen, so dass $\LaTeX$ immer wieder einen zusätzlichen Lauf benötigt. Daher erzeugen derartige Situationen immer eine $\LaTeX$ -Fehlermeldung, damit man das Problem untersuchen und entscheiden kann, ob man an dieser Stelle nicht besser den Befehl `\ref` verwendet.

Man sollte sich auch der Probleme bewusst sein, die entstehen können, wenn man `\reftextvario` verwendet: Wenn man an mehreren nahe beieinander liegenden Stellen auf das gleiche Objekt verweist, kann die jedesmal wechselnde Wortwahl seltsam wirken.

Eine abschließende Warnung: Jeder Einsatz von `\vref` oder `\vpageref` erzeugt intern zwei Makronamen. Wenn man diesen Befehl sehr häufig verwendet, kann das bei kleineren $\TeX$ -Installationen dazu führen, dass man für neue Namen („string pool“ oder „multiletter control sequences“) keinen Raum mehr hat, oder dass kein Hauptspeicher mehr zur Verfügung steht. Aus diesem Grund gibt es zusätzlich den Befehl `\fullref`. Er kann immer dann verwendet werden, wenn man sicher ist, dass Label und Verweis nicht auf nahe beieinander liegenden Seiten erscheinen.

2.4.3 prettyref – Ausschmücken von Querverweisen

Die Querverweismechanismen von $\LaTeX$ erzeugen zwar die Elementnummer (oder die Seitennummer), aber leider lassen sie den Anwender mit der dazugehörigen Gestaltung völlig allein. Daher ist es schwierig, einheitliche Verweise

zu erzielen. Wenn z.B. ein Verlagshaus fordert, dass auf Abbildungen immer mit „Abb.xx“ verwiesen wird, muss man diese Einträge im ganzen Quelldokument manuell ändern.

Das Paket `prettyref` von Kevin Ruland unterstützt automatische Formatierungen mit zusätzlichen Zeichenfolgen, sofern die *schlüssel* der `\label`-Befehle einer bestimmten Struktur folgen. Diese Zeichenfolgen müssen die Form „*<Präfix>:<name>*“ (wie z.B. `fig:main`) haben, wobei weder *präfix* noch *name* einen Doppelpunkt enthalten dürfen. Diese Form wird bereits von vielen Anwendern benutzt. Die zusätzlichen Texte werden mit dem Befehl `\prettyref` erzeugt, ohne dass das Paket die Standardbefehle `\ref` und `\pageref` beeinflusst. Im Unterschied hierzu ändert die `\labelformat`-Deklaration des Paketes `varioref` überall die Darstellung der Verweislabel.

`\newreformat{präfix}{code}`

Dieser Befehl formatiert Querverweise, in deren Schlüssel das Argument *präfix* als Präfix erscheint. Das Argument *code* verweist mit `#1` auf den verwendeten Schlüssel, so dass er an `\ref`, `\vref`, und so weiter übergeben werden kann. Dieses Format wird verwendet, wenn man den Schlüssel mit dem Befehl `\prettyref` einsetzt.

4 Abschnitt

Ein Verweis auf die Gleichung in diesem Abschnitt sieht aus wie folgt:
„siehe (1) in Abschnitt 4“.

 $a = b$

(1)

```
\usepackage{prettyref}
\newreformat{sec}{Abschnitt~\ref{#1}}
\section{Abschnitt} \label{sec:hier}
Ein Verweis auf die Gleichung in diesem Abschnitt
sieht aus wie folgt:
"‘siehe \prettyref{eq:a} in \prettyref{sec:hier}’".
\begin{equation} a = b \label{eq:a} \end{equation}
```

Bsp.
2-4-7

Das Beispiel zeigt, dass die Formatierung für das *<präfix>* „eq“ bereits in dem Paket `prettyref` enthalten ist. Es kennt einige weitere vordefinierte Formate, aber da die meisten von ihnen zwischen dem erzeugten Text und der Nummer einen Umbruch erlauben, sollte man wohl besser eigene definieren.

Da das Paket nicht zwischen Verweisen am Anfang und in der Mitte eines Satzes unterscheidet, ist es wahrscheinlich nicht in allen Fällen geeignet. Auch kann man den Doppelpunkt zwischen *<präfix>* und *<name>* nicht ersetzen, so dass es nicht mit Sprachen kombiniert werden kann, in denen der Doppelpunkt bereits eine besondere Bedeutung hat. In einem solchen Fall bietet sich gegebenenfalls das Paket `fancyref` von Axel Reichert an, das ähnliche Funktionen bereitstellt aber intern mit komplexeren Einstellungen arbeitet.

2.4.4 titleref – Nicht numerische Verweise

In manchen Dokumenten muss man auf die Titel der Abschnitte verweisen, anstatt auf ihre Nummern. Vielleicht weil sie nicht nummeriert sind, oder auch weil ein bestimmter Stil gefordert ist. Das Paket `titleref` von Donald Arseneau stellt die hierzu erforderlichen Funktionen zur Verfügung. Es enthält den Befehl `\titleref`, mit dem auf die Titel von Abschnitten und Gleitobjekten verwiesen werden kann.

Nummerierte Abschnitte und Gleitobjekte werden im Querverweis genauso dargestellt, wie sie im Inhaltsverzeichnis erscheinen würden (egal ob ein Verzeichnis gedruckt wird oder nicht). Wenn also das optionale Argument eines Gliederungsbefehls oder einer Bildunterschrift einen Kurztitel nennt, dann wird dieser Titel von `\titleref` gesetzt. Der Verweistitel für nicht nummerierte Abschnitte entspricht dem gesetzten Titel. Folglich verwandeln sich die Argumente nicht nummerierter Abschnitte in bewegte Argumente, die sehr seltsame Fehler erzeugen, wenn sie zerbrechliche Befehle enthalten, die nicht mit `\protect` geschützt sind.

§ Nicht nummerierte Abschnitte bekommen bewegte Argumente

Ein `\titleref`-Verweis auf ein Label ohne Titel (z.B. ein Label in einer Fußnote oder eine Nummerierung) übernimmt einfach irgendeinen vorhergehenden Titel, normalerweise den des aktuellen Abschnitts.

Wie im nächsten Beispiel dargestellt, ist der Titel des aktuellen Abschnitts über `\currenttitle` verfügbar, egal ob er mit einem `\label`-Schlüssel versehen ist oder nicht. Das Beispiel zeigt auch, dass `\titleref` und `\ref` nebeneinander existieren können.

1 Textuelle Verweise

In Abschnitt „Textuelle Verweise“ zeigt der Verweis auf Abschnitt „Beispiel“, dass man auf nicht nummerierte Abschnitte verweisen kann.

```
\usepackage{titleref}
\setcounter{secnumdepth}{1}
\section{Textuelle Verweise}\label{num}
In Abschnitt "\currenttitle{" zeigt der
Verweis auf Abschnitt "\titleref{ex}",
dass man auf nicht nummerierte Abschnitte
verweisen kann.
```

Ein Beispiel

Bsp.
2-4-8

Der aktuelle Abschnitt wird in Abschnitt 1 erwähnt.

```
\subsection[Beispiel]{Ein Beispiel}\label{ex}
Der aktuelle Abschnitt wird in
Abschnitt~\ref{num} erwähnt.
```

Das Format des Verweises auf den Titel wird durch Umdefinieren des Befehls `\theTitleReference` gesteuert. Er verfügt über zwei Argumente: Eine *nummer*, wie sie von `\ref` dargestellt würde, und einen *titel*. Enthält ein Dokument Verweise auf nicht nummerierte Titel, dann sollte das Argument *nummer* nicht verwendet werden, da es einfach eine beliebige Nummer enthält. So zeigt der Befehl `\titleref` im nächsten Beispiel die Zahl „1“, obwohl er auf einen nicht nummerierten Abschnitt verweist.

1 Textuelle Verweise

In Abschnitt 1 *Textuelle Verweise* zeigt der Verweis auf Abschnitt 1 *Beispiel*, dass man auf nicht nummerierte Abschnitte verweisen kann.

```
\usepackage{titleref}
\renewcommand\theTitleReference[2]
{\emph{#1\ #2}}
\setcounter{secnumdepth}{1}
\section{Textuelle Verweise}\label{num}
In Abschnitt \currenttitle{ zeigt der Verweis
auf Abschnitt \titleref{ex}, dass man auf
nicht nummerierte Abschnitte verweisen kann.
```

Ein Beispiel

Bsp.
2-4-9

Der aktuelle Abschnitt wird in Abschnitt 1 erwähnt.

```
\subsection[Beispiel]{Ein Beispiel}\label{ex}
Der aktuelle Abschnitt wird in
Abschnitt~\ref{num} erwähnt.
```

Konflikte mit anderen Paketen

Das Paket ist so voreingestellt, dass es zusätzlichen Code in Befehle einfügt, die üblicherweise für Überschriften, Bildunterschriften und entsprechende Elemente verwendet werden. Im Zusammenspiel mit anderen Paketen, die Titel nach einer eigenen Methode setzen, kann es daher zu Konflikten kommen. In diesem Fall kann man das Paket mit der Option `usetoc` anweisen, einen völlig anderen Ansatz zu verwenden. Wie der Name schon verrät, zeichnet das Paket dann die Daten des Inhaltsverzeichnisses auf, indem es `\addcontentsline` neu definiert. Als Konsequenz darf der Befehl `\label` nicht mehr im Argument *titel* erscheinen, sondern muss diesem folgen. Außerdem darf zwischen Überschrift und Label kein `\addcontentsline`-Befehl auftauchen, der zu einer anderen Überschrift gehört.

Da die Sternformen von Gliederungsbefehlen keine Verzeichniseinträge erzeugen, werden diese weiterhin undefiniert. Dies kann durch Hinzufügen der Option `nostar` verhindert werden. Dann kann man allerdings auch nicht mehr auf ihre Titel verweisen.

2.4.5 hyperref – Dynamische Querverweise

Sebastian Rahtz hat (mit Beiträgen von Heiko Oberdiek und David Carlisle) das Paket `hyperref` entwickelt, mit dem man alle Querverweise (Zitate, Inhaltsverzeichnisse, usw.) in Hyperlinks verwandeln kann. Es erweitert existierende Befehle um Funktionen, die `\special`-Befehle erzeugen, mit deren Hilfe entsprechend ausgerüstete Treiber Querverweise in Hyperlinks umwandeln können. Das Paket wird in [59, S. 35–67] ausführlich erläutert. Es verfügt über ein eigenes Handbuch, das selbst Hyperlinks enthält, die mit dem Paket erzeugt wurden.

Das Paket `hyperref` lässt sich relativ einfach bedienen. Wenn man es als *Letztes* in die Liste der geladenen Pakete aufnimmt, werden alle Querverweise im Dokument in Hyperlinks umgesetzt. Es verfügt über einige Optionen, mit denen man einstellen kann, wie die Hyperlinks aussehen oder funktionieren. Die wichtigsten Optionen sind `colorlinks`, um Links einzufärben anstatt einzurahmen, und `backref`, um Links in die Bibliographie einzufügen, die auf die Stellen verweisen, an denen die jeweiligen Einträge zitiert wurden.

Das Paket bietet eine Reihe von Einstellungsmöglichkeiten, um das Verhalten von PDF-Dateien, die aus einem Dokument erzeugt wurden, und von PDF-Betrachtern, wie z.B. Adobe Reader, zu beeinflussen, siehe etwa [59].

2.4.6 xr – Verweise auf externe Dokumente

Auf Basis der Arbeit von Jean-Pierre Drucbert entwickelte David Carlisle das Paket `xr`, ein System für Verweise auf externe Dokumente.

Wenn ein Dokument z.B. auf Abschnitte in einem anderen Dokument namens `other.tex` verweisen muss, dann kann man das Paket `xr` in der Hauptdatei angeben und den Befehl `\externaldocument{other}` in die Präambel schreiben. Danach kann man mit `\ref` und `\pageref` auf alles verweisen, was entweder im Hauptdokument oder im Dokument `other.tex` mit dem Befehl `\label` versehen wurde. Man kann eine beliebige Anzahl solcher externer Dokumente deklarieren.

Wenn eines der externen Dokumente oder das Hauptdokument den gleichen `\label`-Schlüssel verwendet, erzeugt dies einen Konflikt, da der gleiche Schlüssel mehrmals definiert wird. Um dieses Problem zu vermeiden, verfügt `\externaldocument` über ein optionales Argument. Wenn man den Befehl in der Form `\externaldocument[A-]{other}` deklariert, erhalten alle Verweise in die Datei `other.tex` das Präfix `A-`. Wenn also ein Abschnitt in der Datei `other.tex` das Label `\label{intro}` erhält, kann darauf mit `\ref{A-intro}` verwiesen werden. Das Präfix muss nicht unbedingt `A-` sein; es kann eine beliebige Zeichenfolge sein, die sicherstellt, dass alle Label, die aus externen Dateien importiert werden, eindeutig sind.

Wenn im Gebrauch befindliche Pakete bestimmte aktive Zeichen (wie zum Beispiel : im Französischen oder " im Deutschen) deklarieren, sollten diese jedoch nicht in `\label`-Befehlen verwendet werden. Ebenso sollten sie nicht im optionalen Argument von `\externaldocument` erscheinen.

Das Paket kann nicht mit `hyperref` kombiniert werden, da beide den internen Mechanismus für Querverweise verändern. Stattdessen kann man das Paket `xr-hyper` einsetzen, das speziell auf die Zusammenarbeit mit `hyperref` zugeschnitten ist.

Formatierungswerkzeuge

3.1 Textfragmente und Absätze	86
3.2 Fußnoten, Endnoten und Marginalien	116
3.3 Listen	135
3.4 Wortwörtlicher Text.	159
3.5 Zeilen und Spalten	185

Die Art und Weise, in der Informationen visuell präsentiert werden, kann das Verständnis einer Aussage durch den Leser stark beeinflussen. Darum ist es wichtig, die besten verfügbaren Werkzeuge zu verwenden, um die genaue Bedeutung der eigenen Worte zu vermitteln. Gleichzeitig gilt jedoch, dass Formen der visuellen Darstellung dem Benutzer immer nur das Verständnis eines Textes erleichtern dürfen, nicht aber seine Aufmerksamkeit ablenken. Daher sind eine einheitliche Darstellung und einheitliche Konventionen für die visuellen Merkmale ein absolutes Muss. Ebenso sollte die Art und Weise, in der gegebene Strukturelemente hervorgehoben werden, im gesamten Dokument immer gleich sein. Diese Regeln lassen sich leicht festschreiben, indem man für jedes Dokumentenelement, das besonders behandelt werden soll, einen eigenen Befehl oder eine Umgebung definiert und diese Befehle und Umgebungen in einer Paketdatei oder in der Dokumentenpräambel zusammenfasst. Indem man dann ausschließlich diese Befehle verwendet, ist eine einheitliche Präsentationsform gewährleistet.

Das vorliegende Kapitel erklärt verschiedene Wege, Teile eines Dokumentes hervorzuheben. Der erste Abschnitt befasst sich mit der Hervorhebung kurzer Textfragmente oder Absätze und beschreibt Werkzeuge, mit denen diese verändert werden können.

Der zweite Abschnitt wirft einen Blick auf die verschiedenen Arten von Anmerkungen, wie z.B. Fußnoten, Marginalien und Endnoten, und erläutert, wie ihr Layout angepasst werden kann.

Der dritte Abschnitt behandelt dann das Formatieren von Listen. Zunächst werden die verschiedenen Parameter und Befehle untersucht, mit denen die $\text{\LaTeX}$ -Listenumgebungen `enumerate`, `itemize` und `description`

gesteuert werden. Anschließend werden die Erweiterungen, die durch das `paralist`-Paket bereitgestellt werden, sowie das Konzept der „theoremähnlichen Strukturen“ am Beispiel des `amsthm`-Paketes besprochen. Für die Struktur- und Layoutanforderungen der meisten Leser sind diese Erweiterungen ausreichend. Für alle anderen wird im restlichen Teil des Abschnitts die allgemeine Listenumgebung `list` vorgestellt. Es wird gezeigt, wie sich durch Variieren der Parameterwerte, mit denen die Umgebung `list` gesteuert wird, Layouts anpassen lassen.

Der vierte Teil beschreibt, wie die wortwörtliche Ausgabe von Text (englisch: verbatim) simuliert werden kann. In diesem Zusammenhang gilt ein spezielles Augenmerk den leistungsfähigen Paketen `fancyvrb` und `listings`.

Das Kapitel schließt mit der Betrachtung von Paketen, die sich mit der Zeilennummerierung, Handhabung von Spalten (etwa von parallelem Text in zwei Spalten) oder den Problemen beim mehrspaltigen Satz beschäftigen.

3.1 Textfragmente und Absätze

Dieser Abschnitt befasst sich mit Textfragmenten und erläutert, wie sie bearbeitet und hervorgehoben werden können, indem man ihnen ein anderes Erscheinungsbild zuweist als dem übrigen Text.

Zunächst wird besprochen, wie man Befehle definiert, die automatisch korrekten Leerraum nach ihrer Anwendung erzeugen. Anschließend wird gezeigt, wie professionelle Auslassungspunkte erzeugt werden.

Zur Hervorhebung von Text können Parameter wie Schriftform, -stärke und -grad angepasst werden (siehe Abschnitt 7.3.1 auf Seite 349). Genauso kann Text unterstrichen oder die Laufweite einer Schrift verändert werden. Mittel, mit denen sich diese Effekte erzielen lassen, werden durch die Pakete `relsize`, `textcase`, `ulem` und `soul` angeboten.

Der Rest des Abschnitts wendet sich absatzbezogenen Fragen zu, wie etwa der Erzeugung von großen Anfangsbuchstaben zu Beginn eines Absatzes, der Änderung der Absatzausrichtung oder Einstellung des Zeilenvorschubs innerhalb eines Absatzes sowie dem Einfügen von rechteckigen Löchern in einem Absatz, die unter anderem mit kleinen Bildern gefüllt werden können.

3.1.1 `xspace` – Korrekte Leerräume nach Makros

Das kleine Paket `xspace` (von David Carlisle) definiert den Befehl `\xspace`, der am Ende von Makros verwendet werden sollte, die hauptsächlich zur Textproduktion dienen. Er fügt ein Leerzeichen ein, sofern dem Makro nicht bestimmte Satzzeichen folgen.

Durch die Verwendung von `\xspace` müssen Befehle ohne Argument nicht mehr mit `\_` oder `{}` abgeschlossen werden, damit nach ihnen ein Leerzeichen ausgegeben wird. Wenn jedoch eine dieser Konstruktionen auf `\xspace` folgt, fügt `\xspace` kein zusätzliches Leerzeichen ein. Das bedeutet, dass man `\xspace` ohne Risiko an das Ende eines bereits existierenden Makros anfügen kann, ohne dass man Änderungen in seinem Dokument vornehmen müsste. Mögliche Kandidaten für die Verwendung von `\xspace` sind

Befehle für Abkürzungen, wie „z.B.“ oder „d.h.“.

```
\newcommand\zB{\z.B.\xspace}
\newcommand\xdh{\d.h.\xspace}
\newcommand\usw{\usw.\@ \xspace}
```

Wichtig ist die richtige Verwendung des Befehls \@ zur Erzeugung der richtigen Art von Leerraum: Rechts von einem Punkt verhindert er, dass zusätzlicher Abstand eingefügt wird, da der Punkt in diesem Fall nicht als ein Satzendezeichen gewertet wird. Links von einem Punkt bewirkt er, dass L^AT_EX den Punkt immer als Satzendezeichen interpretiert.

Unter Umständen kann es vorkommen, dass \xspace an der falschen Stelle einen Leerraum einfügt. In solchen Fällen kann man an das Makro einfach {} anschließen, wodurch der Leerraum unterdrückt wird.

Die Bundesrepublik Deutschland
hat 16 Länder.
Großbritannien, die Bundesrepublik
Deutschland und Frankreich haben
enge kulturelle Verbindungen.

```
\usepackage{xspace}
\newcommand\BRD{Bundesrepublik Deutschland\xspace}
\newcommand\GB {Großbritannien\xspace}
Die \BRD hat 16 Länder.\@ \GB, die \BRD und
Frankreich haben enge kulturelle Verbindungen.
```

Bsp.
3-1-1

3.1.2 ellipsis, lips – Auslassungspunkte

Auslassungspunkte werden im Allgemeinen durch drei aufeinander folgende Punkte dargestellt (auch als *Ellipse* bekannt). Der Abstand der Punkte untereinander hängt von typographischen und verlagsspezifischen Konventionen ab. Hier sind deutliche Unterschiede zu beobachten. Im Französischen werden nach Hart [65] oder dem *Chicago Manual of Style* [38], „points de suspension“ dicht aneinander gesetzt und sie folgen dem vorangegangenen Wort mit einem Leerraum auf der rechten Seite:

C'est une chose... bien difficile.

Im Deutschen haben Auslassungspunkte gemäß Duden [45] einen Leerraum auf der linken *und* rechten Seite, es sei denn, sie stellen ausgelassene Buchstaben innerhalb eines Wortes dar oder ein Satzzeichen folgt:

Du E... du! Scher dich zum ...!

In der britischen oder amerikanischen Typographie dagegen werden die Punkte mit vollem Wortabstand gesetzt und recht komplizierte Regeln legen fest, wie andere Satzzeichen an beiden Enden behandelt werden.

L^AT_EX bietet die Befehle \dots und \textellipsis, um Auslassungspunkte mit kleinem Abstand zu erzeugen. Leider produziert die Standarddefinition (die aus Plain-T_EX stammt) einen ungleichmäßigen Abstand auf der linken und rechten Seite – dies ist für den korrekten Satz von einigen der obigen Beispiele völlig ungeeignet. Der zusätzliche kleine Abstand rechts der Auslassungspunkte ist in manchen Situationen richtig (z.B. wenn ein Satzzeichen folgt). Wenn den Auslassungspunkten ein Leerzeichen folgt, sieht dieser

Abstand jedoch recht unpassend aus und sollte am besten weggelassen werden, wie das Beispiel unten zeigt (auch wenn durch Auslassen des Abstandes im zweiten Fall das Ausrufezeichen ein wenig zu nah an die Auslassungspunkte rutscht).

Ein Beispiel zum Vergleich:
Du E... du! Scher dich zum ...!
Du E... du! Scher dich zum ...!

```
\newcommand\lips{\dots\unkern}
Ein Beispiel zum Vergleich:\\
Du E\dots\ du! Scher dich zum \dots!\\
Du E\lips\ du! Scher dich zum \lips!
```

Bsp.
3-1-2

Dieses Problem wird mit dem Paket `ellipsis` von Peter Heslin angepackt. Es definiert den Befehl `\dots` neu. Für die Entscheidung, ob ein Nachabstand gesetzt werden soll, wird das auf die Auslassungspunkte folgende Zeichen herangezogen. Ein zusätzlicher Leerraum wird nur eingefügt, wenn das folgende Zeichen im Befehl `\ellipsis punctuation` aufgelistet ist, der standardmäßig den Wert „`,. . : ; ! ?`“ hat. Bei einigen Sprachunterstützungspaketen werden bestimmte Zeichen aktiviert. Werden diese in der Liste verwendet, muss sie neu deklariert werden, damit das Paket die Zeichen weiterhin erkennt.

Der Abstand zwischen den Punkten sowie der hinter den Auslassungspunkten gegebenenfalls eingefügte Leerraum kann über den Befehl `\ellipsis gap` festgelegt werden. Automatische Anpassungen, die vom Schriftgrad abhängen, lassen sich über fontabhängige Einheiten, wie `em`, oder einen Bruchteil eines `\fontdimen`-Befehls steuern (siehe Seite 439).

Ein Beispiel zum Vergleich:
Du E... du! Scher dich zum ...!
Du E... du! Scher dich zum ...!
Du E... du! Scher dich zum ...!

```
\usepackage{ellipsis}
Ein Beispiel zum Vergleich:\\
Du E\dots\ du! Scher dich zum \dots!\\
\renewcommand\ellipsis gap{1.5\fontdimen3\font}
Du E\dots\ du! Scher dich zum \dots!\\
\renewcommand\ellipsis gap{0.3em}
Du E\dots\ du! Scher dich zum \dots!
```

Bsp.
3-1-3

Für den Spezialfall, dass Auslassungspunkte mitten im Wort benötigt werden (oder aus anderen Gründen nur ein kleiner Abstand auf beiden Seiten gewünscht ist), bietet das Paket den Befehl `\midwordellipsis`. Wenn das Paket mit der Option `m1a` (Modern Language Association Style) geladen wurde, werden die Auslassungspunkte automatisch eingeklammert und hinter dem letzten Punkt folgt kein zusätzlicher Abstand.

Gemäß dem *Chicago Manual of Style* [38] werden Auslassungspunkte mit vollem Wortabstand zwischen den Punkten gesetzt. Dieses Erscheinungsbild erreicht man mit dem Paket `lips`¹ von Matt Swift. Es implementiert den Befehl `\lips`, der den Empfehlungen in diesem Referenzbuch folgt. Auslassungspunkte, die beispielsweise das Fehlen des Satzendes markieren, sollten nach [38, §10.48–63] aus vier Punkten bestehen, wobei der *erste* Punkt das Satzendezeichen darstellt.² Der Befehl `\lips` realisiert dies, indem er „`\lips.`“ als „`\lips`“ interpretiert, wie das nächste Beispiel zeigt.

¹`lips` ist, genau genommen, Teil einer größeren Paketsammlung. Wenn nur dieses Paket verwendet werden soll, muss auch das Paket `moredefs` desselben Autors geladen werden.

²Nicht, dass die Autoren dieses Buches irgendeine Logik darin sehen könnten ...

In . . . werden Auslassungspunkte normalerweise mit vollem Wortabstand gesetzt. . . Ein Beispiel hierfür ist dieser Absatz.

Bsp.
3-1-4

```
\usepackage{moredefs,lips}
```

In \lips werden Auslassungspunkte normalerweise mit vollem Wortabstand gesetzt \lips. Ein Beispiel hierfür ist dieser Absatz.

Der Befehl \lips sucht nach Satzzeichen, die auf den Befehl folgen, und sorgt dafür, dass Auslassungspunkte und Satzzeichen im Fall von „,:;?!)’]/“ nicht durch einen Zeilenumbruch getrennt werden. In anderen Fällen (z.B. einer öffnenden Klammer) ist ein Zeilenumbruch zugelassen. Die obige Liste wird in \LPNobreakList gespeichert und kann bei Bedarf angepasst werden. Um einen umbruchgeschützten Leerraum nach \lips zu erzwingen, muss der Befehl durch eine Tilde ergänzt werden (~).

Bei Verwendung der Option mla werden die erzeugten Auslassungspunkte automatisch eingeklammert und ein Punkt nach dem Befehl \lips wird nicht an den Anfang verschoben. Wenn erforderlich, erzeugt \olips die ursprüngliche Version ohne Klammern.

In . . . werden Auslassungspunkte mit vollem Wortabstand zwischen den Punkten gesetzt [. . .]. Ein Beispiel hierfür ist dieser Absatz.

Bsp.
3-1-5

```
\usepackage{moredefs}\usepackage[mla]{lips}
```

In \olips werden Auslassungspunkte mit vollem Wortabstand zwischen den Punkten gesetzt \lips. Ein Beispiel hierfür ist dieser Absatz.

3.1.3 amsmath – Geschützte Bindestriche

Das amsmath-Paket, das ausführlich in Kapitel 8 besprochen wird, bietet ebenfalls einen Befehl zur Verwendung in Absätzen. Der Befehl \nobreakdash unterdrückt jede Möglichkeit, nach dem Bindestrich einen Zeilenumbruch vorzunehmen. Eine sehr verbreitete Verwendung von \nobreakdash ist die Unterdrückung von unerwünschten Zeilenumbrüchen in Fällen wie „ p -adic“, aber es gibt noch ein weiteres Beispiel: Wenn „Seite 3–9“ als Seite 3\nobreakdash--9 kodiert ist, wird zwischen dem Bindestrich und der 9 kein Zeilenumbruch eingefügt.

Dieser Befehl muss *unmittelbar* vor dem Bindestrich (–, -- oder ---) angegeben werden. Das folgende Beispiel zeigt, wie ein Zeilenumbruch nach einem Bindestrich unterdrückt wird, aber die normalen Bindestriche im folgenden Wort zugelassen werden (es reicht, nach dem Bindestrich einen Leerraum ohne horizontale Ausdehnung einzufügen). Bei häufiger Verwendung empfiehlt es sich, Abkürzungen wie \p zu definieren. Danach wird das Wort „dimensionalen“ umbrochen, während ein Umbruch nach „ p –“ verhindert (was im Beispiel zur überlaufenden Zeile führt) und dann „3–9“ auf die nächste Zeile verschoben wird.

Die Verallgemeinerung des n -dimensionalen Falls (mittels Standard p -adic-Topologie) kann man auf den Seiten 3–9 von Band IV finden.

Bsp.
3-1-6

```
\usepackage{amsmath}
```

```
\newcommand\p{\$p\$ \nobreakdash}
```

```
\newcommand\Ndash{\nobreakdash--}
```

```
\newcommand\n[1]{\$n\$ \nobreakdash-\hspace{0pt}}
```

Die Verallgemeinerung des n -dimensionalen Falls (mittels Standard \p-adic{-Topologie) kann man auf den Seiten 3\Ndash 9 von Band IV finden.

3.1.4 relsize – Relative Skalierung der Schriftgröße

Standard- $\LaTeX$ bietet zehn vordefinierte Befehle zur Änderung der Schriftgröße (siehe Tabelle 7.1 auf Seite 353). Die von ihnen erzeugten Schriftgrade werden jeweils von der Dokumentenklasse festgelegt, bleiben aber innerhalb eines Dokumentes vom Wert unverändert. Dies bedeutet, dass z.B. `\small` in einem Dokument immer denselben Wert wählt, ungeachtet der umgebenden Bedingungen.

In vielen Fällen ist es jedoch wünschenswert, die Schriftgröße relativ zur aktuellen Größe zu ändern. Dies lässt sich durch das Paket `relsize` bewerkstelligen, das ursprünglich von Bernie Cosell entwickelt und später für $\LaTeX 2_\epsilon$ von Donald Arseneau und Matt Swift aktualisiert und erweitert wurde.

Das Paket enthält den deklarativen Befehl `\relsize`, der als Argument eine Zahl erwartet. Diese gibt die Anzahl an Schritten an, um welche die Schriftgröße geändert werden soll. Wenn z.B. die aktuelle Schriftgröße `\Large` ist, ändert `\relsize{-2}` die Schriftgröße auf `\normalsize`. Wenn die gewünschte Anzahl an Schritten nicht verfügbar ist, wird die kleinste (d.h. `\tiny`) bzw. größte (d.h. `\Huge`) Schriftgröße gewählt. Eine Umkehr der Skalierung durch Negation des Argumentwertes muss also nicht notwendigerweise wieder zur Ausgangsgröße führen. Besser ist es, nur lokale Veränderungen vorzunehmen (etwa innerhalb einer Gruppe geschweiffter Klammern) und $\LaTeX$ die Rückänderungen selbst zu überlassen.

Das Paket definiert außerdem `\smaller` und `\larger`, welche einfach Abkürzungen für den Befehl `\relsize` mit den Argumenten `-1` bzw. `1` sind. Weitere bequeme Alternativen sind `\textsmaller` und `\textlarger`, die als Argument den zu verkleinernden oder zu vergrößernden Text haben. Diesen vier Befehlen kann als optionales Argument die Anzahl der Skalierungsschritte mitgegeben werden, wenn ein anderer Wert als der voreingestellte Wert 1 benötigt wird.

Großer Text mit ein paar
kleinen Worten integriert.

KAPITÄLCHEN (gefälscht)

KAPITÄLCHEN (echt; vgl. Lauflänge und
Strichstärke zur vorherigen Zeile).

```
\usepackage{relsize}
\Large Großer Text mit ein paar \
  {\relsize{-2}kleinen Worten} integriert.
\par\medskip
\normalsize\noindent
K\textsmaller[2]{APITÄLCHEN} (gefälscht)\
\textsc{Kapitälchen} (echt; vgl. \ Lauf\länge
und \ Strichstärke zur vorherigen Zeile).
```

Bsp.
3-1-7

Die obige Beschreibung für `\relsize` ist nicht ganz korrekt: Bei jedem Schritt versucht der Befehl, die Schriftgröße um 20% zu vergrößern oder zu verkleinern, und wählt dann den Schriftgrößenbefehl von $\LaTeX$, welcher der gewünschten Zielgröße am nächsten kommt. Anschließend werden die gewünschte Schriftgröße und die Zielgröße verglichen. Wenn sie um mehr als den aktuellen Wert von `\RSPercentTolerance` (angegeben in Prozent) differieren, ruft das Paket den Befehl `\fontsize` mit der Zielgröße als einem der Parameter auf. Bei diesem Prozess versucht $\LaTeX$ s Zeichensatz-Auswahlschema, einen Font zu finden, welcher der Anforderung am nächsten kommt. Standardmäßig ist `\RSPercentTolerance` ein leeres Makro, das als 30 (Prozent) interpretiert wird, wenn die aktuelle Schriftformgruppe nur

aus separaten Größen besteht (siehe Abschnitt 7.10.3), und als 5, wenn die Schriftformdefinition einen Bereich von Schriftgrößen umfasst.

In einigen Fällen kann ein fester Faktor von 1.2 für jeden Schritt eine zu große Einschränkung darstellen. Aus diesem Grund bietet das Paket zusätzlich den allgemeineren deklarativen Befehl `\relscale{faktor}` und seine Variante `\textscale{faktor}{text}`, mit denen die Schriftgröße basierend auf dem gegebenen *faktor*, wie etwa 1.3 (Vergrößerung um 30%), ausgewählt werden kann.

Zwei weitere Befehle dienen zur Skalierung im mathematischen Modus: `\mathsmaller` und `\mathlarger`. $\TeX$ erkennt nur vier verschiedene mathematische Schriftgrößen, von denen zwei (`\displaystyle` und `\textstyle`) für die meisten Symbole nahezu identisch sind; dadurch ist das Anwendungsgebiet dieser Befehle etwas begrenzt. Die Situation verbessert sich geringfügig, wenn zusätzlich `exscale` geladen wird: Der Befehl `\mathlarger` schaltet dann, bei Verwendung in `\displaystyle`, intern auf eine größere Textschriftgröße um und wählt anschließend den zu dieser Schriftgröße gehörenden `\displaystyle`.

Bsp.
3-1-8

$$\sum \neq \sum \quad \text{und } \frac{1}{2} \neq \frac{1}{2} \text{ aber } N = N$$

```
\usepackage{exscale,relsize}
\[ \sum \neq \mathlarger{\sum} \]
und $\frac{1}{2} \neq \frac{\mathlarger{1}}{2}$ aber $N = \mathlarger{N}$
```

Diese Befehle versuchen, bei großen Operatoren die hoch- und tiefgestellten Indizes richtig zu positionieren. Zum Beispiel:

Bsp.
3-1-9

$$\sum_{i=1}^n \neq \sum_{i=1}^n \neq \sum_{i=1}^n \quad \int_0^\infty \neq \int_0^\infty \neq \int_0^\infty$$

```
\usepackage{exscale,relsize}
\[
\mathsmaller{\sum_{i=1}^n} \neq
\sum_{i=1}^n \neq \mathlarger{\sum_{i=1}^n}
\quad \mathsmaller{\int_0^\infty} \neq
\int_0^\infty \neq \mathlarger{\int_0^\infty}
\]
```

Zu beachten ist, dass die Verwendung dieser Befehle innerhalb von Formeln die wahre Natur der mathematischen Objekte in ihrem Argument verbergen, so dass die Abstände in der Formel ohne zusätzliche Hilfe missraten können. Wie das folgende Beispiel zeigt, muss man deshalb möglicherweise explizit `\mathrel`, `\mathbin` oder `\mathop` verwenden, um korrekte Abstände zu erhalten.

Bsp.
3-1-10

$$a \times b \neq a \times b \neq a \times b$$

```
\usepackage{exscale,relsize}
\[
a \times b \neq a \mathlarger{\times} b \neq
a \mathbin{\mathlarger{\times}} b
\]
```

Aufgrund dieser Unwägbarkeiten sollten die Befehle `\mathlarger` und `\mathsmaller` nicht blindlings eingesetzt werden, sie sind nicht in jedem Fall empfehlenswert.

3.1.5 textcase – Intelligente Groß- und Kleinschreibung

Die $\text{\LaTeX}$ -Standardbefehle `\MakeUppercase` und `\MakeLowercase` ändern die Schreibweise der Zeichen ihrer Argumente in Groß- bzw. Kleinschreibung, wodurch die Makros nach Bedarf expandiert werden. Beispielsweise ergibt

```
\MakeUppercase{Am \today}
```

„AM 29. MÄRZ 2005“. Manchmal werden dadurch mehr Zeichen als gewünscht geändert. Wenn der Text z.B. eine mathematische Formel enthält, ist die Umsetzung in Großbuchstaben nicht so empfehlenswert, weil sich damit die Bedeutung ändert. Auf ähnliche Weise stellen die Argumente der Befehle `\label`, `\ref` und `\cite` semantische Informationen dar, die bei Änderung der Schreibweise zu falschen oder fehlenden Querverweisen führen, weil $\text{\LaTeX}$ nach den falschen Einträgen sucht.

```
\MakeTextUppercase{text} \MakeTextLowercase{text}
```

Das Paket `textcase` von David Carlisle löst die oben angesprochenen Probleme, indem es `\MakeTextUppercase` und `\MakeTextLowercase` als Alternativbefehle bereitstellt, die mathematische Formeln und Querverweise erkennen und diese unberücksichtigt lassen.

1 Groß-/Kleinschreibung

TEXT IM ABSCHNITT 1, ÜBER $a = b$
UND $\alpha \neq a$

```
\usepackage{textcase}
```

```
\section{Groß-/Kleinschreibung}\label{exa}  
\MakeTextUppercase{Text im Abschnitt~\ref{exa},  
über  $a=b$  und  $\alpha \neq a$  }
```

Bsp.
3-1-11

Manchmal sollten Teile des Textes aus dem einen oder anderen Grund unverändert bleiben. Mit `\NoCaseChange` bietet das Paket eine allgemeine Vorgehensweise, um solche Teile zu markieren. Ein Beispiel:

```
\usepackage{textcase}
```

EIN BISSCHEN TEXT Noch mehr TEXT

```
\MakeTextUppercase{Ein bisschen Text  
\NoCaseChange{Noch mehr} Text}
```

Bsp.
3-1-12

Wenn notwendig, kann dieses Verfahren verwendet werden, um syntaktische Informationen zu verbergen, wie in

```
\NoCaseChange{\begin{tabular}{ll}}...\NoCaseChange{\end{tabular}}
```

Dadurch wird verhindert, dass `tabular` und `ll` fälschlicherweise großgeschrieben werden.

Dies alles funktioniert nur, wenn sich der betreffende Text auf der obersten Ebene befindet. Alles, was in geschweiften Klammern steht (mit Ausnahme der Argumentklammern von `\label`, `\ref`, `\cite` oder `\NoCaseChange`), wird in Groß- bzw. Kleinschreibung umgewandelt, ungeachtet dessen, um was es sich handelt.

```
\usepackage{textcase}
```

DIESE BEIDEN SCHLAGEN IMMER
FEHL $A + B = C$, LEIDER

```
\MakeTextUppercase{Diese beiden schlagen immer  
\textbf{fehl  $a+b=c$ }}  
\emph{\NoCaseChange{, leider}}}
```

Bsp.
3-1-13

Im obigen Fall lässt sich diese Stolperfalle umgehen, indem man die Formel aus dem Argument von `\textbf` herausnimmt und `\emph` in das Argument von `\NoCaseChange` einsetzt. In anderen Fällen ist diese Art der Korrektur möglicherweise nicht machbar. Dann besteht die (etwas mühselige) Lösung darin, den problematischen Teil in einem gesonderten Makro zu verbergen und dieses beim Wechsel der Groß- und Kleinschreibung mit `\protect` zu schützen. Diese Methode funktioniert auch für die L^AT_EX-Standardbefehle, wie das nächste Beispiel zeigt.

Bsp.
3-1-14

DIES FUNKTIONIERT $a + b = c$ **IMMER**

```
\newcommand\mymath{\$a+b=c\$}
\MakeUppercase{Dies
\textbf{funktioniert \protect\mymath} immer}
```

Einige Klassen und Pakete verwenden intern `\MakeUppercase`, zum Beispiel in Kopfzeilen. Wenn stattdessen `\MakeTextUppercase` verwendet werden soll, muss das `textcase`-Paket mit der Option `overload` geladen werden. Diese Option ersetzt die L^AT_EX-Standardbefehle durch die vom Paket definierten Varianten.

3.1.6 ulem – Betonen durch Unterstreichen

L^AT_EX ermutigt dazu, einzelne Elemente nicht durch explizite Befehle für den Fontwechsel, wie `\bfseries` oder `\itshape`, zu betonen, sondern eher mit dem Befehl `\emph` oder der Deklaration `\em` hervorzuheben. Das Paket `ulem` (von Donald Arseneau) definiert den Befehl `\emph` neu, so dass er anstelle von kursivem Text unterstrichenen Text erzeugt. Innerhalb des unterstrichenen Textes können Zeilenumbrüche und sogar eine einfache Silbentrennung vorgenommen werden.

Jedes Wort wird in einer unterstrichenen Box gesetzt, wodurch die automatische Silbentrennung ausgeschaltet wird. Weiche Trennstriche (`\-`) können aber trotzdem explizit eingesetzt werden. Die Unterstreichung setzt sich zwischen den Worten fort und wird wie normale Leerzeichen gedehnt. Da Leerzeichen normalerweise Wörter begrenzen, können bei syntaktischen Leerzeichen (z.B. „2.3 pt“) Schwierigkeiten auftreten. Es wird einiger Aufwand betrieben, um solche Leerzeichen zu handhaben. Wenn Probleme auftreten, kann man versuchen, den störenden Befehl in Klammern zu setzen, da alle Einträge in Klammern in eine `\mbox` gesetzt werden. Dadurch unterdrücken Klammern die Dehnung und den Umbruch des Textes, den sie einschließen. Konstruktionen, die in mehreren Abstufungen hervorgehoben werden sollen, werden von diesem Paket nicht immer korrekt behandelt. (Man beachte den Aufwand, der im folgenden Beispiel betrieben wurde, um korrekte Wortzwischenräume zu erhalten: Jedes einzelne weiter hervorgehobene Wort wurde mit einem eigenen `\emph`-Befehl versehen.)

Bsp.
3-1-15

Nein, ich habe nicht in dem Film Die Verfolgung und Ermordung des Jean-Paul Marat, vorgeführt von den Insassen des Asylum of Charenton unter Leitung des Marquis de Sade, mitgespielt! Aber ich habe ihn gesehen.

```
\usepackage{ulem}
```

Nein, ich habe `\emph{nicht}` in dem Film `\emph{\emph{Die} \emph{Verfolgung} \emph{und} \emph{Ermordung} \emph{des} \emph{Jean-Paul} \emph{Marat}}`, vorgeführt von den Insassen des Asylum of Charenton unter Leitung des Marquis de-Sade, mitgespielt! Aber ich `\emph{habe}` ihn gesehen.

Alternativ kann eine Unterstreichung explizit über den Befehl `\uline` erzeugt werden. Außerdem gibt es einige weitere Varianten, die in der Satztechnik üblich sind. Diese sind im nächsten Beispiel zu sehen.

Doppelt <u>unterstrichen</u> , mit Wellenlinie unterschlängelt, einfach (durchgestrichen) oder vollständig ~~ausgeschaltet~~ .	<pre>\usepackage{ulem}</pre> Doppelt <code>\uline{unterstrichen}</code>, mit Wellenlinie <code>\uwave{unterschlaengelt}</code>, einfach (<code>\sout{durchgestrichen}</code>) oder vollständig <code>\xout{ausge-x-t}</code>.

 Bsp.
3-1-16

Die Neudefinition von `\emph` kann mit `\normalem` und `\ULforem` ein- und ausgeschaltet werden. Alternativ kann das Paket mit der Option `normalem` geladen werden, wodurch diese Neudefinition unterdrückt wird. Eine weitere Paketoption ist `UWforbf`, die, wo immer möglich, `\textbf` und `\bfseries` durch `\uwave` ersetzt.

Die Position der Linie, die durch `\uline` erzeugt wird, kann explizit durch Angabe eines Wertes für die Länge `\ULdepth` gesetzt werden. Der Standardwert ist fontabhängig und wird durch den ansonsten unsinnigen (da viel zu großen) Wert `\maxdimen` ausgedrückt. Auf ähnliche Weise kann die Dicke der Linie über den Befehl `\ULthickness` gesteuert werden; er muss aus historischen Gründen mithilfe von `\renewcommand` geändert werden.

3.1.7 soul – Sperren oder Schafe stehlen

Frederic Goudy soll gesagt haben „Jeder, der Frakturschrift sperrt, stiehlt auch Schafe.“ Egal, ob die Aussage wahr oder erfunden ist – das Thema „Sperren“ ruft heftige Diskussionen unter Typographen hervor und wird in den meisten Fällen als Unart betrachtet, weil es den „Grauwert“ des Textes ändert und damit den Lesefluss stört. Trotzdem gibt es durchaus Gründe für eine Sperrung. Layoutschriften benötigen beispielsweise häufig einen lockeren Satz und in den meisten Schriftarten wird großgeschriebener Text auf diese Weise aufgewertet. Manchmal wird Sperrung auch zur Betonung eingesetzt, dabei tritt das Grauwertproblem jedoch besonders deutlich hervor.

T_EX bietet nur wenige Möglichkeiten zum Sperren von Text. Theoretisch ist es am besten, speziell dafür vorgesehene Zeichensätze zu verwenden, anstatt zu versuchen, das Problem mithilfe eines Makropaketes zu lösen. Da dies jedoch voraussetzt, dass ein solcher Zeichensatz verfügbar ist, müssen sich die meisten Benutzer anders behelfen. In der Praxis ist daher eine makrobaasierte Lösung viel einfacher umzusetzen, auch wenn dies bedeutet, dass man einige Beschränkungen hinnehmen muss. Einige Informationen über den Lösungsansatz mit Fonts sind in der Dokumentation des `fontinst`-Paketes [74,75] aufgeführt.

Das `soul`-Paket von Melchior Franz bietet Möglichkeiten zur Sperrung und zur Unterstreichung, erhält dabei aber die T_EX-Funktion der automatischen Silbentrennung. Diese Funktion ist im `ulem`-Paket nicht verfügbar. Bei `soul` wird der Text Zeichen für Zeichen syntaktisch analysiert, was zu einer Reihe von Eigenheiten und Einschränkungen führt. Benutzer, die nur ein paar Wörter unterstreichen möchten und keine automatische Silbentrennung benötigen, sind daher besser mit `ulem` bedient, das bei der Eingabe weniger pingelig ist.

<code>\caps{text}</code>	<code>\hl{text}</code>	<code>\so{text}</code>	<code>\st{text}</code>	<code>\ul{text}</code>

Die Verwendung der fünf wichtigsten Anwenderbefehle des `soul`-Paketes zeigt das nächste Beispiel. In den Fällen, in denen es der Silbentrennungsalgorithmus von $\text{T}_{\text{E}}\text{X}$ nicht schafft, die richtige Trennstelle zu finden, kann man wie gewohnt die Stelle mit dem Befehl `\-` markieren. Wenn das `color`-Paket geladen ist, funktioniert `\hl` wie ein Textmarker, wobei er standardmäßig den Hintergrund gelb einfärbt; ansonsten verhält er sich wie `\ul` und unterstreicht sein Argument.

Mit dem `soul`-Paket können Wörter und Phrasen gesperrt werden. Großbuchstaben werden mit einem anderen Befehl `GSPERRT`. Auch zum Unterstreichen, ~~Durchstreichen~~ und Hervorheben gibt es eigene Befehle.

`\usepackage{soul,color}`

Mit dem `\texttt{soul}`-Paket können `\so{Wörter und Phrasen gesperrt}` werden. Großbuchstaben werden mit einem anderen Befehl `\caps{GSPERRT}`. Auch zum `\ul{Unterstreichen}`, `\st{Durchstreichen}` und `\hl{Hervorheben}` gibt es eigene Befehle.

Bsp.
3-1-17

Normalerweise interpretiert das `soul`-Paket den Text im Argument von `\so`, `\st` usw. Buchstabe für Buchstabe. Wenn Buchstaben jedoch durch mehr als ein Zeichen dargestellt werden (z.B. akzentuierte Zeichen), kann diese Vorgehensweise mit hässlichen $\text{T}_{\text{E}}\text{X}$ -Fehlermeldungen fehlschlagen. Glücklicherweise kennt das Paket bereits alle üblichen Akzentbefehle, so dass diese richtig behandelt werden. Alle anderen, z.B. diejenigen, die vom Paket `textcomp` bereitgestellt werden, können mithilfe einer `\soulaccent`-Deklaration dem `soul`-Paket bekannt gemacht werden. Eine andere Möglichkeit ist, die Zeichen einzuklammern.

Bsp.
3-1-18

ä ù Õ X Y

`\usepackage{soul} \usepackage{textcomp}`
`\soulaccent{\capitalgrave}`

`\Huge \st{"a \'u ~0 \capitalgrave X {\capitalbreve Y}}`

Das `soul`-Paket weiß bereits, dass Anführungsstriche, Gedankenstriche und Geviertstriche aus mehreren Zeichen bestehen, und behandelt diese richtig. Andere syntaktische Ligaturen, wie das spanische Ausrufezeichen, müssen zur korrekten Bearbeitung in Klammern eingfasst werden.

Bsp.
3-1-19

„Also“, sagte er.
¡HOLA – MEIN **FREUND**!

`\usepackage{soul}`

`\so{"'Also'}, sagte er. \\\`
`\caps{!'}Hola -- mein \textbf{Freund}!}}`

Das `soul`-Paket erkennt auch Formeln, solange diese in `$`-Zeichen eingeschlossen sind (die Form `\(...\)` wird nicht unterstützt), sowie alle Fontbefehle mit Argument, wie `\textbf`. Ein selbstdefinierter Fontbefehl oder solche, die von einem Paket bereitgestellt werden, müssen allerdings mit der Deklaration `\soulregister` im `soul`-Paket registriert werden. Bei dieser Deklaration wird der Fontbefehl als erstes Argument und die Anzahl der Argumente (d.h. 0 oder 1) für diesen Befehl als zweites Argument angegeben. Innerhalb des `soul`-Paketes bietet keiner der Fontbefehle Kursivkorrekturen. Wenn eine solche erforderlich ist, muss sie manuell per `\/` durchgeführt werden.

Hier sehen wir **soul** `\newcommand\textsfbf[1]{\textsf{\bfseries#1}}`
 in Aktion: $x \neq y$ OK? `\usepackage{soul} \soulregister{\textsfbf}{1}`
`\so{Hier sehen wir \textsfbf{soul} in \emph{Aktion}:}`
`$x\neq y$ OK?}`

Bsp.
3-1-20

Bei genauerer Betrachtung sieht man, dass die Fontbefehle eine direkt vorangehende und nachfolgende Sperrung unterdrücken, wie zwischen „Aktion“ und dem Doppelpunkt. Dies lässt sich korrigieren, indem man den Befehl `\>` hinzufügt, der einen größeren Abstand erzeugt.

`\usepackage{soul}`
 leidend vs. leidend `\so{l\textbf{ei}dend vs. l\>\textbf{ei}\>dend}`

Bsp.
3-1-21

Eingeklammerter Text wird während der syntaktischen Analyse als ein Objekt betrachtet und daher nicht gesperrt. Dies ist ein sehr willkommener Effekt, wenn bestimmte Ligaturen innerhalb eines gesperrten Textes zusammengehalten werden sollen. Diese Methode funktioniert jedoch nur, wenn der Text innerhalb der Klammern keine Trennstellen zur Silbentrennung enthält. Wenn Trennstellen vorhanden sind, wird die Paketfehlermeldung „Reconstruction failed“ ausgegeben. Um die Trennstellen zu verbergen, muss der Text in eine `\mbox` eingefügt werden, wie die zweite Textzeile des nächsten Beispiels zeigt. (T_EX würde hier in „Es-cher“ trennen – mitten im „sch“, das stets zusammengehalten werden sollte.) Dieser Effekt lässt sich auch durch `\soulomit` erreichen, aber dann lässt sich der Text nur kompilieren, wenn das `soul`-Paket geladen ist.

⚙ ⚙ Vorrichtung
 Gödel, Escher, Bach
Da lässt sich trefflich streiten!

`\usepackage{soul,yfonts}`
`\usepackage[ansinew]{inputenc}`
`\textfrac{\so{S{ch}u{tz}-vorri{ch}tung}} \par`
`\so{Gödel, E\mbox{sch}er, Bach} \par`
`\ul{Da lässt sich tre\soulomit{ffl}ich streiten!}`

Bsp.
3-1-22

Eine der größten Einschränkungen der obigen Befehle ist, dass sie nicht verschachtelt werden können. Jegliche Versuche, `soul`-Befehle zu verschachteln, führen unweigerlich zu systemnahen T_EX-Fehlern. Wenn wirklich eine Verschachtelung erforderlich ist, muss der innere Teil in einer Box platziert werden. Dies bedeutet aber, dass er nicht mehr am Zeilenende umbrochen werden kann.

Dies ist eine Qual für uns alle!

`\usepackage{soul} \newsavebox\soulbox`
`\sbox\soulbox{\so{ eine Qual }}`
`\ul{Dies ist \mbox{\usebox{\soulbox}} für uns alle!}`

Bsp.
3-1-23

Einige andere Befehle zeigen innerhalb des Argumentes von `\so` und `Co` ein besonderes Verhalten. Wie oben zu sehen war, kann eine Sperrung an bestimmten Punkten mit `\<` unterdrückt oder mit `\>` erzwungen werden. Wie bei L^AT_EX üblich, erzeugt `~` ein geschütztes Leerzeichen. Der Befehl `\>` wird zwar unterstützt, allerdings nur in seiner Grundform – d.h. ohne Stern und ohne optionales Argument. Mit `\linebreak` kann an bestimmten Punkten eine Zeile umbrochen werden, doch auch hier wird das optionale Argument nicht unterstützt. Andere L^AT_EX-Befehle werden das Paket wahrscheinlich eher sprengen

- hier hilft nur auszuprobieren, um herauszufinden, was sicher funktioniert und was zur Katastrophe führt. Das nächste Beispiel zeigt ein paar Anwendungen für diese besonderen Fälle.

Bsp.
3-1-24

„Also“, sagte er. Dann wollen wir mal eine kurze und eine gesperrte Zeile produzieren, OK?

```
\usepackage{soul}
\so{"<"}{<Also<"}, sagte er.
Dann wollen wir mal eine kurze und \\
eine gesperrte Zeile\linebreak
produzieren, OK?}
```

```
\sodef{befehl}{font}{zwischenraum}{wortabstand}{endabstand}
```

Mit der Deklaration `\sodef` kann man eigene Befehle zur Sperrung definieren. Außerdem bietet sie die Möglichkeit, die Standardwerte für den Befehl `\so` zu überschreiben.

Dieser Sperrungs-Algorithmus fügt zwischen Zeichen einen bestimmten *zwischenraum*, zwischen Wörtern einen *wortabstand* und am Anfang und Ende eines Textabschnitts einen *endabstand* ein. Letzterer wird nur eingefügt, wenn es an dieser Stelle passend ist. Die Standardwerte für diese Abstände sind auf den Satz von Texten in Frakturschriften ausgerichtet, aber sie lassen sich recht leicht mithilfe der Deklaration `\sodef` den eigenen Anforderungen anpassen. Im Argument *font* können Fontattribute angegeben werden; in den meisten Fällen bleibt es leer. Es empfiehlt sich, in den anderen Argumenten keine expliziten Maße anzugeben, sondern stattdessen auf *em*-Werte zurückzugreifen, damit die Definition vom aktuellen Font und seiner Größe abhängig wird.

Bsp.
3-1-25

Hier **werden einige Wörter** betont.

```
\usepackage{soul}
\sodef\sobf{\bfseries}{.3em}{1em plus .1em}
{1.3em plus.1em minus.2em}
Hier \sobf{werden einige Wörter} betont.
```

Während bei `\so` oder jedem neuen, über `\sodef` definierten Befehl die gespeicherte Definition ausgelesen und ausgeführt wird, funktioniert der Befehl `\caps` anders. Er untersucht den aktuellen Font und versucht, diesen (oder einen ihm möglichst ähnlichen) in einer internen Datenbank zu finden. Danach wendet er die dort gespeicherten Werte zur Sperrung an. Die Datenbank ist erweiterbar. Mit der Deklaration `\capsdef` können Werte für einzelne Fonts oder Fontgruppen bereitgestellt werden. Auf diese Weise kann die Sperrung feinabgestimmt werden – zum Beispiel im Überschriftentext. Es ist sogar möglich, mehrere Datenbanken anzulegen und sie während der Bearbeitung im Dokument auszutauschen.

```
\capsdef{muster}{font}{zwischenraum}{wortabstand}{endabstand}
```

Bis auf das erste Argument, das sich von den anderen stark unterscheidet, sind die anderen Argumente von `\capsdef` mit jenen von `\sodef` identisch. Das erste Argument *muster* definiert den Font oder die Fonts, auf welche die aktuelle Deklaration angewendet werden soll.

Seine Syntax ist *schriftkodierung*, *-familie*, *-serie*, *-form* und *-grad*, getrennt durch Schrägstriche, wobei die Namenskonvention des NFSS verwendet wird. Wenn ein Wert leer ist, bedeutet dies, dass es für dieses Attribut keine Einschränkung gibt, d.h. bei `////` werden alle Fonts gesperrt, bei `/ptm///10` werden alle Times Fonts im Schriftgrad 10 Punkt gesperrt, und bei `OT1/cmr/m/n/` wird Computer Modern (cmr), Serie Medium (m), normale Form (n), kodiert in OT1 in jeglichem Schriftgrad gesperrt. Darüber hinaus ist es möglich, Größenbereiche anzugeben. `5–14` bedeutet beispielsweise $5\text{pt} \leq \text{grad} < 14\text{pt}$ und `14–` heißt, dass alle Grade von 14pt und größer gesperrt werden. Einzelheiten über die Fontnamenskonventionen von NFSS sind in Kapitel 7 aufgeführt.

Wie bei `\sodef` ist das Argument *font* auch in den meisten Deklarationen leer. In einigen Fällen kann es sinnvoll sein, hier `\scshape` zu verwenden, zum Beispiel wenn die Schriftform vor der Sperrung in Kapitälchen umgewandelt werden soll.

`\caps` verwendet immer den ersten passenden Eintrag in seiner Datenbank, daher ist die Reihenfolge der `\capsdef`-Deklarationen wichtig. Nachfolgende Deklarationen werden zuerst untersucht, so dass bestehende Deklarationen überschrieben oder erweitert werden können.

EINE BEISPIEL-ÜBERSCHRIFT

Hier ist die Deklaration `\capsdef` anzuwenden, weil in der Definition für die Überschrift eine serifenlose Schrift spezifiziert ist und die Beispiele in diesem Buch in Times und Helvetica (phv) gesetzt sind.

```
\usepackage{titlesec,soul}
\newcommand{\allcaps[1]{\MakeUppercase{\caps{#1}}}
\titleformat{\section}[block]{\centering\sffamily}
{\thesection.}{.5em}{\allcaps}
\titlespacing*{\section}{0pt}{8pt}{3pt}
\capsdef{/phv///}{\scshape}{.17em}{.55em}{.4em}
\section*{Eine Beispielüberschrift}
Hier ist die Deklaration \verb=\capsdef= anzuwenden, weil
in der Definition für die Überschrift eine serifenlose
Schrift spezifiziert ist und die Beispiele in diesem Buch
in Times und Helvetica (\texttt{phv}) gesetzt sind.
```

Bsp.
3-1-26

Das vorangegangene Beispiel enthielt eine interessante Kombination von `\caps` und `\MakeUppercase`: Der Befehl `\allcaps` wandelt den Text des Argumentes in Großbuchstaben um und sperrt ihn dann mit dem Befehl `\caps`.

<code>\capssave{name}</code>	<code>\capsselect{name}</code>	<code>\capsreset</code>

Mit `\capsreset` wird die Datenbank wieder in ihren ursprünglichen Zustand zurückgesetzt, der nur generische Standardwerte enthält. Mit `\capsdef` können anschließend neue Einträge hinzugefügt werden. Der aktuelle Status der `\caps`-Datenbank kann mit `\capssave` unter einem *namen* abgespeichert werden. Dieser Status kann über `\capsselect` später wieder hergestellt werden. Wenn beim Laden des Paketes die Option `capsdefault` verwendet wird, werden alle Verwendungen von `\caps`, zu denen es keine entsprechende Deklaration gibt, durch Unterstreichen des Textes markiert.

*Benutzerdefinierte
Sperrung für
verschiedene Fälle*

EINE BEISPIELÜBERSCHRIFT

Man beachte die unterschiedliche Laufweite der Sperrung im Überschriftentext und im LAUFENDEN TEXT. Für Times gibt es hier keine Definition, so dass die STANDARDEINSTELLUNG wirksam ist.

Bsp.
3-1-27

```
\usepackage{titlesec} \usepackage[capsdefault]{soul}
\capsdef{/phv///}{\scshape}{.17em}{.55em}{.4em}
\capssave{display} \capsreset
\capsdef{/phv///}{\scshape}{.04em}{.35em}{.35em}
\titlespacing*{\section}{0pt}{8pt}{3pt}
\titleformat{\section}[block]{\centering\sffamily}
{\thesection.}{.5em}{\capsselect{display}\caps}
\section*{Eine Beispielüberschrift}
Man beachte die unterschiedliche Laufweite der Sperrung im
Überschriftentext und im \textsf{\caps{laufenden Text}}.
Für Times gibt es hier keine Definition, so dass die
\caps{Standardeinstellung} wirksam ist.
```

Mit `\setul` bzw. `\setuldepth` kann man die Position und die Dicke der Linie, die vom Befehl `\ul` erzeugt wird, nach eigenen Anforderungen definieren. Der Befehl `\setul` hat zwei Argumente: Im ersten wird die Position der Linie in Bezug auf die Grundlinie und im zweiten die Dicke der Linie angegeben. Alternativ kann man `\setuldepth` verwenden, um anzugeben, dass die Linie unter dem im Argument enthaltenen Text angeordnet werden soll. Mit `\resetul` können die Standardeinstellungen des Paketes wiederhergestellt werden.

*Benutzerdefinierte
Unterstreichung*

Hier testen wir
~~eine Reihe von~~
verschiedenen Einstellungen.
Zurück zur Normalität!

Bsp.
3-1-28

```
\usepackage{soul}
\setul{0pt}{.4pt} \ul{Hier testen wir} \par
\setul{-.6ex}{.3ex} \ul{eine Reihe von} \par
\setuldepth{g} \ul{verschiedenen Einstellungen.} \par
\resetul \ul{Zurück zur Normalität!}
```

Sowohl `\ul` als auch `\st` verwenden standardmäßig eine schwarze Linie. Wenn zusätzlich das Paket `color` geladen wird, können stattdessen auch farbige Linien eingesetzt werden und, wenn gewünscht, kann die Hervorhebungsfarbe von `\hl` geändert werden, wie das Beispiel unten zeigt:

Bsp.
3-1-29

Linien können schwarz oder blau sein.

```
\usepackage{soul,color}
\sethlcolor{green} \setulcolor{blue} \setstcolor{red}
Linien \hl{können} \st{schwarz} oder \ul{blau} sein.
```

3.1.8 url – URLs, Pfadnamen und Ähnliches

E-Mail-Adressen, URLs, Pfad- oder Verzeichnisnamen und ähnliche Objekte zu formatieren ist in der Regel nicht ganz so einfach. Zum Einen enthalten sie oft Zeichen, die eine besondere Bedeutung für $\text{\LaTeX}$ haben, wie `~`, `#`, `&`, `{` oder `}`. Zum Anderen sollten sie möglichst nicht umbrochen werden oder, wenn nicht vermeidbar, zumindest mit Bedacht umbrochen werden. Es ist beispielsweise nicht empfehlenswert, an einem Bindestrich zu umbrechen, denn damit bleibt unklar, ob der Bindestrich aufgrund des Zeilenumbruchs eingefügt wurde (wie bei normalen Wörtern) oder Teil des Ausdrucks ist. Aus demselben Grund sollte man auch keinen Umbruch an einer Leerstelle vornehmen. Als kleine Hilfe entwickelte Donald Arseneau das Paket `url`, das versucht, die meisten dieser Probleme zu lösen.

<code>\url{text}</code>	<code>\url!text!</code>	<code>\path{text}</code>	<code>\path=text=</code>

Der Grundbefehl dieses Paketes ist `\url`, den es in zwei Syntaxvarianten gibt: Das Argument *text* kann entweder in Klammern eingefasst werden (in diesem Fall muss der *text* immer genauso viele öffnende wie schließende Klammern enthalten) oder kann, wie `\verb`, durch willkürliche Zeichen (die nicht im *text* auftauchen) auf beiden Seiten begrenzt werden. (In der Syntaxbox oben werden `!` und `=` verwendet, aber das sind nur Beispiele.) Im zweiten Fall darf es auch eine ungleiche Anzahl an öffnenden und schließenden Klammern im Argument geben.

Der Befehl `\path` funktioniert genauso, außer dass er stets Typewriter Fonts (`\ttfamily`) verwendet, während `\url` individuell angepasst werden kann. Das Argument beider Befehle wird mehr oder weniger wortwörtlich ausgegeben. `\url{~}` erzeugt z.B. eine Tilde. Leerzeichen werden standardmäßig ignoriert, wie das folgende Beispiel zeigt.

Die Webadresse des L^AT_EX-Projektes lautet `http://www.latex-project.org` und mein Homeverzeichnis ist (manchmal) `~frank`.

`\usepackage{url}`

Die Webadresse des L^AT_EX-Projektes lautet `\url{http://www . latex-project . org}` und mein Homeverzeichnis ist (manchmal) `\path+~frank+`.

Bsp.
3-1-30

Zeilenumbrüche können an bestimmten Sonderzeichen durchgeführt werden (standardmäßig nicht zwischen Buchstaben oder Bindestrichen), aber Bindestriche werden von den Befehlen an der Umbruchstelle nicht hinzugefügt. Wenn der *text* die Zeichen `%` oder `#` enthält oder auf `\` endet, kann er nicht im Argument eines anderen Befehls verwendet werden, ansonsten generiert dieser Befehl einen Fehler (genau wie der Befehl `\verb`). Eine weitere Zeichenfolge, die im Argument eines anderen Befehls nicht richtig funktioniert, sind zwei aufeinander folgende `^`-Zeichen. Dieser Fall ist sogar noch schlimmer, denn hierbei wird möglicherweise nicht mal eine Fehlermeldung ausgegeben, sondern einfach eine falsche Ausgabe erzeugt.¹ Dies zeigt auch das nächste Beispiel.

`\usepackage{url}`

`^frank` und `^frank` (OK)

`\url{^frank}` und `\mbox{\url{^frank}}` (OK) `\par`

`^^frank` aber `&rank` (falsch)

`\url{^^frank}` aber `\mbox{\url{^^frank}}` (falsch)

Bsp.
3-1-31

Selbst wenn der *text* keine kritischen Sonderzeichen enthält, ist es nie zulässig, solch einen Befehl in einem bewegten Argument zu verwenden – z.B. im Argument des `\section`-Befehls. Hier wird die Fehlermeldung

```
! Undefined control sequence.
\Url Error ->\url used in a moving argument.
```

ausgegeben, gefolgt von vielen seltsamen Fehlern. Selbst die Verwendung von `\protect` hilft in diesem Fall nicht. Was kann man also machen, wenn man einen Pfadnamen oder eine URL an einer solchen Stellen zitieren muss? Wenn man sorgfältig arbeitet und ausschließlich „sichere“ Zeichen im *text* verwendet, kann man die Befehle beim Laden des Paketes mithilfe der Option `allowmove` für die Verwendung in bewegten Argumenten freigeben. Dies ist

¹Dies hängt vom nachfolgenden Buchstaben ab. Ein großes F erzeugt z.B. einen Fehler.

jedoch nicht wirklich hilfreich, wenn ein Zeichen wie „#“ verwendet werden soll. In diesem Fall muss die Information zunächst mit `\urldef` gespeichert werden, bevor sie später verwendet werden kann.

```
\urldef{befehl}{url-befehl}{text} \urldef{befehl}{url-befehl}=text=
```

Die Deklaration `\urldef` definiert einen neuen Befehl *befehl*, der den *url-befehl* (dies kann `\url`, `\path` oder ein neu definierter Befehl sein – siehe unten) so auf den *text* anwendet, dass er an jeder Stelle (einschließlich bewegter Argumente) eingesetzt werden kann. Der *url-befehl* wird an dieser Stelle nicht ausgeführt, was bedeutet, dass sich Änderungen des Schriftstils auf die Formatierung auswirken (siehe Beispiel 3-1-33). Technisch gesprochen werden hierbei die `\catcodes` der Zeichen im *text* während der Deklaration eingefroren, damit sie an Stellen wie in Argumenten nicht fehlinterpretiert werden können.

1 ^^frank~#\$\$\ geht?

```
\usepackage{url}
\urldef\test\path{^^frank~#$$}
\section{\test{} geht?}
```

Ja, es geht -- im Gegensatz zum vorherigen Beispiel.

Bsp.
3-1-32

Ja, es geht – im Gegensatz zum vorherigen Beispiel.

```
\urlstyle{stil}
```

Die Möglichkeit der Stiländerung wurde bereits erwähnt. Für diese Aufgabe bietet das `url`-Paket den Befehl `\urlstyle` an, der ein obligatorisches Argument hat: einen bestimmten *stil*. Vordefinierte Stile sind *rm*, *sf*, *tt* und *same*. Die ersten drei Werte selektieren die korrespondierende Schriftfamilie, während *same* die aktuelle Schrift verwendet und nur die Zeilenumbruchsregeln ändert.

Der Befehl `\url` verwendet den aktuellen *stil* (voreingestellt *tt*, d.h. Typewriter Font), während `\path` intern immer auf den Stil *tt* umschaltet. Im folgenden Beispiel wird eine URL, die in `\lproject` gespeichert ist, mehrere Male mit verschiedenen Stilen formatiert. Dieses Beispiel sieht immer noch fürchterlich aus, aber wie hätte es wohl ausgesehen, wenn die URL in diesem schmalen Satzspiegel nicht umbrochen werden dürfte?

```
Zapf Chancery! http://www.
latex-project.org (Standard-
einstellung) http://www.latex-
project.org (CM Roman) http:
//www.latex-project.org (CM Sans
Serif) http://www.latex-
project.org (CM Typewriter)
http://www.latex-project.org (Zapf
Chancery)
```

```
\hyphenation{Stan-dard-ein-stellung}
\usepackage[hyphens]{url}
\urldef\lproject\url{http://www.latex-project.org}
\fontfamily{pzc}\selectfont Zapf Chancery!
\lproject\ (Standardeinstellung) \quad
\urlstyle{rm}\lproject\ (CM Roman) \quad
\urlstyle{sf}\lproject\ (CM Sans Serif) \quad
\urlstyle{tt}\lproject\ (CM Typewriter) \quad
\urlstyle{same}\lproject\ (Zapf Chancery)
```

Bsp.
3-1-33

Bei genauer Betrachtung kann man sehen, dass im obigen Beispiel die Option `hyphens` verwendet wurde. Diese Option ermöglicht einen expliziten

Zeilenumbruch an Bindestrichen, welches bei `\url`-ähnlichen Befehlen normalerweise deaktiviert ist. Ohne diese Option sind Umbrüche nur an den Punkten, nach dem Doppelpunkt oder nach „/“ zugelassen.

*Leerzeichen im
Argument*

Wie bereits erwähnt, werden Leerzeichen im `text` standardmäßig ignoriert. Wenn dies nicht erwünscht ist, kann man die Option `obeyspaces` verwenden. Sie kann jedoch dazu führen, dass falsche Leerzeichen eingefügt werden, wenn der Befehl im Argument eines anderen Befehls verwendet wird und der `text` irgendwelche „\“-Zeichen enthält. In diesem Fall lässt sich das Problem mit `\urldef` lösen. Zeilenumbrüche und Leerzeichen werden nur zugelassen, wenn zusätzlich die Option `spaces` verwendet wird.

Das Paket erkennt automatisch, welche Schriftkodierung gerade verwendet wird. Bei T1-kodierten Schriften verwendet es auch die in dieser Kodierung zusätzlich verfügbaren Zeichen, um das Gesamtergebnis zu verbessern.

*Text links oder rechts
anhängen*

Das Paket stellt die beiden Parameter `\UrlLeft` und `\UrlRight` bereit, die standardmäßig keine Auswirkung haben. Sie können aber so umdefiniert werden, dass sie Material links oder rechts vom `text` setzen. Das Material wird auf dieselbe Weise formatiert wie der `text`. Leerzeichen werden ignoriert, es sei denn, dass `\_` verwendet oder `obeyspaces` als Option angegeben wird. Wenn die Befehle auf der obersten Ebene neu definiert werden, wirken sie sich auf jeden `\url`-ähnlichen Befehl aus. Eine Möglichkeit zur Beschränkung ihres Gültigkeitsbereiches ist in Beispiel 3-1-34 dargestellt.

```
\DeclareUrlCommand{befehl}{stil-information}
```

*Definition
URL-ähnlicher Befehle*

Manchmal ist es hilfreich, eigene Befehle zu definieren, die ähnlich wie `\url` oder `\path` funktionieren, aber ihre eigenen Schriften usw. verwenden. Mit dem Befehl `\DeclareUrlCommand` kann ein neuer `\url`-ähnlicher Befehl erzeugt oder ein bestehender modifiziert werden. Er hat zwei Argumente: den Befehl, der definiert oder geändert werden soll, und die *stil-information* (z.B. `\urlstyle`).

Im nächsten Beispiel wird `\email` so definiert, dass E-Mail-Adressen per `\UrlLeft` im `rm`-Stil gesetzt werden, und zwar beginnend mit der Zeichenfolge „E-Mail: “. Das Beispiel zeigt deutlich, dass der Definitionsbereich dieser Neudefinition auf den Befehl `\email` begrenzt ist. Bei genauerer Betrachtung entdeckt man, dass ein Leerzeichen in `\UrlLeft` (wie in der Definition auf der obersten Ebene) ohne Wirkung bleibt, während `\_` das gewünschte Ergebnis erzeugt.

```

\usepackage{url}
\renewcommand\UrlLeft{<url: }
\renewcommand\UrlRight{>}
\DeclareUrlCommand\email{\urlstyle{rm}}%
  \renewcommand\UrlLeft{E-Mail: \ }%
  \renewcommand\UrlRight{}}

<url:http://www.latex-project.org> \url{http://www.latex-project.org}
E-Mail: frank.mittelbach@latex-project.org \email{frank.mittelbach@latex-project.org}
<url:$HOME/figures> uups, falsch! \\ \path{$HOME/figures} uups, falsch!
```

Bsp.
3-1-34

Das `url`-Paket bietet eine Reihe weiterer Parameter, die den Zeilenumbruch beeinflussen, unter anderem `\UrlBreaks`, `\UrlBigBreaks` und

EUR	Europa	GRD	Griechenland
ATS	Österreich	IEP	Irland
BEF	Belgien	ITL	Italien
DEM	Deutschland	LUF	Luxemburg
ESP	Spanien	NLG	Niederlande
FIM	Finnland	PTE	Portugal
FRF	Frankreich		

Tabelle 3.1: ISO-Währungscodes des Euros und der 12 Euro-Zonen-Länder

`\UrlNoBreaks`. Diese Parameter können im Argument *stil-information* von `\DeclareUrlCommand` so umdefiniert werden, dass neue oder spezielle Konventionen verwendet werden. Einzelheiten hierzu enthält die Paketdokumentation, die am Ende der Datei `url.sty` angefügt ist.

3.1.9 euro – Konvertieren und Formatieren von Währungen

Um die Umrechnung zwischen länderspezifischen Währungen und dem Euro zu vereinfachen, entwickelte Melchior Franz das Paket `euro`. Dieses Paket berechnet beliebige Wechselkurse auf Basis des Euros. Die Berechnungen werden mithilfe des `fp`-Paketes von Michael Mehlich mit hoher Genauigkeit durchgeführt. Die Formatierung kann für jede Währung individuell verändert werden, so dass dieses Paket für alle Arten von Anwendungen eingesetzt werden kann, in denen Währungen vorkommen, ungeachtet dessen, ob Umrechnungen erforderlich sind oder nicht.

`\EURO{ausgangs-währung} [ziel-währung] {betrag}`

Der Hauptbefehl `\EURO` wandelt einen *betrag* in der *ausgangs-währung* in die *ziel-währung* um oder, wenn dieses optionale Argument fehlt, in Euro. Die Argumente *ausgangs-währung* und *ziel-währung* werden in ISO-Währungscodes angegeben, so wie sie in Tabelle 3.1 aufgelistet sind. Bei der Eingabe des *betrag*s werden die Dezimalstellen durch einen Punkt abgetrennt, selbst wenn die formatierte Zahl normalerweise anders dargestellt würde.

Bei der Standardeinstellung werden der *betrag* in der *ausgangs-währung* und dahinter der entsprechende Wert in der *ziel-währung* in Klammern angezeigt.

`\usepackage{euro}`
`\EURO{DEM}[FRF]{7} \ \EURO{FRF}[DEM]{23.48}`
`\ \ \EURO{EUR}[DEM]{1.00} \ \EURO{DEM}{2}`

Bsp.
3-1-35

7 DM (23,48 FRF) 23,48 FRF (7 DM)
 1 Euro (1,96 DM) 2 DM (1,02 Euro)

Das Paket bietet eine Anzahl von Optionen, die das allgemeine Layout der Ausgabe verändern können (sofern sie nicht von spezifischeren Formatierungsdeklarationen überschrieben werden, siehe unten). Mit `eco` wird dem Betrag nur der ISO-Code vorangestellt, individuelle Symbole werden nicht verwendet; bei `dots` wird zwischen Dreiergruppen von Ziffern jeweils ein Punkt eingefügt (standardmäßig wird ein kleiner Leerraum verwendet).

Die Paketoptionen

Standardmäßig werden ganze Beträge unverändert, d.h. ohne Dezimalpunkt und Dezimalnullen, ausgegeben. Wenn die Option `table` definiert ist, ändert sich diese Darstellung global, und es wird entweder ein — (Option `emdash`, Voreinstellung), ein – (Option `endash`) oder die richtige Anzahl von Nullen (Option `zeros`) ausgegeben.

	<code>\usepackage[eco,table,endash]{euro}</code>	
DEM 7,– (FRF 23,48) FRF 23,48 (DEM 7,–)	<code>\EURO{DEM}[FRF]{7} \ \EURO{FRF}[DEM]{23.48}</code>	
EUR 1,– (DEM 1,96) DEM 2,– (EUR 1,02)	<code>\\ \EURO{EUR}[DEM]{1.00} \ \EURO{DEM}{2}</code>	

Bsp.
3-1-36

Die spezifischeren Ausgabedefinitionen, die später noch erörtert werden, können überall im Dokument stehen. Es empfiehlt sich jedoch, sie in der Präambel zusammenzuhalten oder sie sogar in die Datei `euro.cfg` einzutragen, die beim Laden des Paketes gelesen wird.

Die Formatierung der Währungszeichen kann mit der Deklaration `\EUROSYM` eingestellt werden; standardmäßig verwendet das Paket für die meisten Währungen die ISO-Codes. Im Beispiel wird die Darstellung für Lira und Euro geändert, indem die Währungssymbole aus dem Paket `textcomp` verwendet werden. Außerdem wird zur einfacheren Darstellung der riesigen Lira-Beträge `dots` verwendet.

	<code>\usepackage{textcomp}\usepackage[dots]{euro}</code>	
	<code>\EUROSYM{ITL}{\textlira}</code>	
	<code>\EUROSYM{EUR}{\texteuro}</code>	
10.000 £ (5,16 €) 1.000 DM (989.999 £)	<code>\EURO{ITL}{10000}\quad \EURO{DEM}[ITL]{1000}</code>	

Bsp.
3-1-37

Das Paket kann jederzeit Währungen neuer Länder aufnehmen, die der Euro-Zone beitreten. Es beherrscht die Umrechnung aus und in alle Währungen, solange ihr Umrechnungskurs zum Euro bekannt ist. Zum Hinzufügen einer neuer Währung wird die Deklaration `\EUROADD` verwendet, die drei Argumente hat: den ISO-Währungscode, das Symbol oder den Text, der für diese Währung angezeigt wird, sowie den Umrechnungskurs zum Euro. Das nächste Beispiel zeigt die Darstellung des Britischen Pfunds. Die Abkürzung `\GBP` vereinfacht die Eingabe ein wenig.

	<code>\usepackage{eurosans,euro}</code>	
14,90 £ (22,41 €)	<code>\EUROADD{GBP}{\textsterling}{0.6648} % 2005/06/21</code>	
10 £ (98,67 FRF)	<code>\newcommand*{\GBP}{\EURO{GBP}} \EUROSYM{EUR}{\euro}</code>	
10 € (6,65 £)	<code>\noindent \GBP{14.9}\\ \GBP[FRF]{10}\\ \EURO{EUR}[GBP]{10}</code>	

Bsp.
3-1-38

Die Umrechnungskurse für die einzelnen Währungen der Euro-Zonenländer sind fest (und im Paket vordefiniert). Bei anderen Währungen können sich die Kurse stündlich ändern, man sollte sich daher auf häufige Änderungen einstellen.

Das Paket ermöglicht es, die Darstellung mithilfe von `\EUROFORMAT`-Deklarationen individuell zu gestalten. Man kann entweder neue Standardwerte bereitstellen oder die Formatierung einzelner Währungen anpassen. Das erste Argument gibt an, welcher Teil der Formatierung verändert werden soll, während das zweite Argument die Formatierung beschreibt.

Das Format `main` legt fest, wie die Ausgangs- und Zielwährungen angeordnet werden sollen. Dabei wird mit den reservierten Schlüsselwörtern `\in` und `\out` auf die Ausgangs- bzw. Zielwährungen verwiesen. Im Beispiel unten wird in der ersten Zeile ein Format definiert, das den Standardeinstellungen sehr ähnlich ist. Die zweite Zeile zeigt das Ergebnis der Umrechnung und in der dritten Zeile ist überhaupt keine Umrechnung zu sehen (obwohl diese hinter den Kulissen durchaus stattgefunden hat). Letzteres ist nützlich, wenn man die Formatierungsfunktion für die Währungen verwenden möchte, aber nicht an einer tatsächlichen Umrechnung interessiert ist.

Bsp.
3-1-39

```
\usepackage{euro}
1 000 DM (= 3 353,85 FRF) \EUROFORMAT{main}{\in\ (=,\,\out)} \EURO{DEM}[FRF]{1000}\par
3 353,85 FRF \EUROFORMAT{main}{\out} \EURO{DEM}[FRF]{1000}\par
1 000 DM \EUROFORMAT{main}{\in} \EURO{DEM}{1000}
```

Die Formate `in` und `out` geben an, wie die Ausgangs- und Zielwährungen formatiert werden sollen. Dies geschieht unter Verwendung der reservierten Schlüsselwörter `\val` (monetärer Betrag), `\iso` (Währungscode) und `\sym` (Währungssymbol, sofern vorhanden, ansonsten der ISO-Code).

Bsp.
3-1-40

```
\usepackage{euro}
\EUROFORMAT{in}{\sym~\val} \EUROFORMAT{out}{\iso~\val}
DM 1 000 (FRF 3 353,85) \EURO{DEM}[FRF]{1000}
```

Interessanter sind vielleicht die Möglichkeiten, die Formatierung der monetären Beträge zu verändern. Hierfür bietet das Paket fünf Deklarationen, die im zweiten Argument von `\EUROFORMAT` verwendet werden können. Die Deklaration `\round` gibt an, an welcher Stelle der monetäre Betrag gerundet wird: Bei positiven Werten wird der ganzzahlige Anteil gerundet, bei negativen Werten die Nachkommastellen. `\round{-3}` bedeutet beispielsweise, dass auf drei Nachkommastellen gerundet wird und auch nur diese drei angezeigt werden. Die Deklaration `\form` hat drei Argumente: das Trennzeichen für die Zifferngruppen des ganzzahligen Anteils (standardmäßig `\,`), das Dezimaltrennzeichen (standardmäßig ein Komma) und das Trennzeichen für die Zifferngruppen der Nachkommastellen (standardmäßig `\,`).

Das erste Argument kann entweder `all` lauten (dann wird die Standard-Zahlenformatierung definiert) oder ein ISO-Währungscode sein (dann wird die Formatierung einer einzelnen Währung geändert).

Bsp.
3-1-41

```
\usepackage{euro} \EUROFORMAT{main}{\out}
1,022-5838 Euro \EUROFORMAT{all}{\round{-4}\form{,}{\textperiodcentered}}{}
-335-3855 FRF \EUROFORMAT{ITL}{\round{2}}
9,900,000 Lit. \noindent \EURO{DEM}{2000}\ \EURO{DEM}[FRF]{-100}\ \
\EURO{DEM}[ITL]{10000}
```

Die Deklaration `\minus` formatiert negative Werte, indem sie ihr erstes Argument vor der Zahl und das zweite Argument nach der Zahl ausführt (voreingestellt `\minus{\$-\$}{}`). Die Zahl selbst wird ohne Vorzeichen formatiert,

dieses wird von der Deklaration hinzugefügt. Die Deklaration `\plus` funktioniert analog für positive Zahlen (voreingestellt `\plus{}{}`).

```
\usepackage{color,euro} \EUROFORMAT{main}{\out}
\EUROFORMAT{all}{\plus{+}$}{\minus{\color{blue}$-$}{}}
+1 022,58 Euro -335,39 FRF \EURO{DEM}{2000}\quad \EURO{DEM}{FRF}{-100}
```

Bsp.
3-1-42

Die Deklaration `\zero` hat drei Argumente, die den Fall behandeln, wenn alles null ist, der ganzzahlige Anteil null ist oder die Nachkommastellen null sind. Im ersten und dritten Argument muss auch das Dezimaltrennzeichen eingegeben werden; dies sollte also dem Standard oder dem Wert des Befehls `\form` entsprechen.

```
\usepackage{eurosans,euro}
\EUROFORMAT{main}{\out} \EUROSYM{EUR}{\euro}
\EUROFORMAT{all}{\zero{0,00}{0}{,--}}
0,00 € 0,51 € 1,— € \EURO{DEM}{0}\quad \EURO{DEM}{1}\quad \EURO{EUR}{1}
```

Bsp.
3-1-43

3.1.10 lettrine – Schmücken von Absätzen

Bei einigen Arten von Veröffentlichungen wird manchmal der Anfangsbuchstabe einiger Absätze durch einen sehr großen Buchstaben dargestellt, der häufig in den Absatz hineingesetzt wird (der um diesen Buchstaben herumfließt). Der restliche Teil des Satzes oder der Phrase folgt gewöhnlich in einer speziellen Schrift. Diese Art des Satzes wird nicht nur für Kapitelanfänge von Romanen, sondern auch zur Aufnahme neuer Gedankengänge bis hin zur reinen Dekoration und Auflockerung eines Zeitschriftentextes verwendet. Sie lässt sich bis zu den Anfangstagen des Buchdrucks zurückverfolgen, als solche Initiale häufig nach dem Druckvorgang von Hand koloriert wurden. Initiale finden sich sogar in Manuskripten des Mittelalters, also noch vor der Erfindung des Buchdrucks.

```
\lettrine[schlüssel-wert-liste]{initial}{text}
```

Das Paket `lettrine` von Daniel Flipo bietet die Möglichkeit, solche Initiale mit dem Befehl `\lettrine` zu erstellen. In der einfachsten Form hat er zwei Argumente: den Buchstaben, der als Initial gesetzt werden soll, und den Folgetext, der in einer besonderen Schrift gesetzt werden soll, standardmäßig ist dies `\scshape`.

LA MOITIÉ DES PASSAGERS,
affaiblis, expirants de ces angoisses inconcevables que le roulis d'un vaisseau porte dans les nerfs et dans toutes les humeurs du corps agitées en sens contraire, ...

```
\usepackage{lettrine} \usepackage[ansinew]{inputenc}
\usepackage[french]{babel}
\lettrine[L]{a moitié des passagers,} affaiblis,
expirants de ces angoisses inconcevables que le
roulis d'un vaisseau porte dans les nerfs et
dans toutes les humeurs du corps agitées en sens
contraire, \ldots
```

Bsp.
3-1-44

Das Initial unterscheidet sich normalerweise vom übrigen Text nur in der Schriftgröße. Wenn man den Befehl `\LettrineFontHook` umdefiniert und die

Standardbefehle des NFSS verwendet, kann man aber auch eine andere Schriftfamilie für das Initial definieren. Auf dieselbe Art kann auch für den Text im zweiten Argument durch Neudefinition von `\LettrineTextFont` die Schrift geändert werden.

Da der Befehl `\lettrine` die Größe des Initials so berechnet, dass es über mehrere Zeilen passt, sollten nur skalierbare Schriften eingesetzt werden, um optimale Ergebnisse zu erzielen. Die Beispiele in diesem Buch sind standardmäßig mit Adobe Times und Helvetica formatiert, hier ergeben sich also keine Probleme. Später gibt es Beispiele in Palatino, bei der es sich auch um eine skalierbare Type 1-Schrift handelt. Bei Verwendung einer Bitmap-Schrift, wie Computer Modern, müssen jedoch spezielle `.fd`-Dateien verwendet werden (siehe Kapitel 7, Seiten 430ff), um zufriedenstellende Ergebnisse zu erzielen.

LA MOITIÉ DES PASSAGERS,
affaiblis, expirants de ces angoisses inconcevables que le roulis d'un vaisseau porte dans les nerfs et dans toutes les humeurs du corps agitées en sens contraire, ...

Bsp.
3-1-45

```
\usepackage{lettrine} \usepackage[ansinew]{inputenc}
\usepackage[french]{babel}
\renewcommand\LettrineFontHook{\sffamily\bfseries}
\renewcommand\LettrineTextFont{\sffamily\scshape}
\lettrine{L}{a moitié des passagers,} affaiblis,
expirants de ces angoisses inconcevables que le
roulis d'un vaisseau porte dans les nerfs et
dans toutes les humeurs du corps agitées en sens
contraire, \ldots
```

Viele Bücher über Typographie geben Empfehlungen über die Art, wie große Initiale in Bezug auf den umgebenden Text am besten formatiert werden sollten. Für höchste Qualität ist es oft erforderlich, die Positionierung je nach Form des Initials manuell nachzukorrigieren. Bei Buchstaben mit einem nach links auslaufenden Balken wird z.B. häufig empfohlen, sie in den Rand hineinragen zu lassen. Der Befehl `\lettrine` berücksichtigt diese Anforderungen durch Bereitstellung eines optionalen Argumentes, in dem die Anpassungen in Form einer durch Kommas getrennten Liste eines Schlüssel-Wert-Paares angegeben werden können.

Die Größe des Initials wird standardmäßig so berechnet, dass sie zwei Textzeilen umfasst (gespeichert in `\DefaultLines`); mit dem Schlüsselwort `lines` kann die Anzahl an Zeilen geändert werden. Es gibt eine Ausnahme: Bei Angabe von `lines=1` wird das Initial trotzdem zwei Zeilen hoch formatiert, aber nicht in den Absatz eingefügt, sondern auf der Grundlinie der ersten Textzeile platziert.

Wenn das Initial sowohl in den Absatz eingebettet sein und über die erste Textzeile herausragen soll, ist das Schlüsselwort `loversize` von Nutzen. Der Wert `.2` vergrößert das Initial um 20%. Der Standardwert für dieses Schlüsselwort wird in `\DefaultLoversize` gespeichert. Dieses Schlüsselwort ist auch in Verbindung mit `lraise` hilfreich (voreingestellt 0 in `\DefaultLraise`). Bei einem Initial mit einer größeren Unterlänge, wie einem „Q“, muss der gesamte Buchstabe möglicherweise etwas angehoben werden, um das Überdrucken der Folgezeilen zu verhindern. In diesem Fall kann man mithilfe von `loversize` die Höhe reduzieren, um das Initial korrekt auszurichten.

Mit dem Schlüsselwort `lhang` kann man angeben, wie weit das Initial in den Rand hineinragen soll. Der Wert wird als Bruchteil angegeben, das heißt,

er liegt zwischen 0 und 1. Der Standardwert hierfür wird in `\DefaultLhang` gespeichert.

QUAND ILS FURENT revenus un peu à eux, ils marchèrent vers Lisbonne; il leur restait quelque argent, avec lequel ils espéraient se sauver de la faim après avoir échappé à la tempête ...

```
\usepackage{palatino,lettrine}
\usepackage[ansinew]{inputenc}
\usepackage[french]{babel}

\lettrine[lines=3, loversize=-0.1, lraise=0.1,
  lhang=.2]{Q}{uand ils furent} revenus un peu à
eux, ils marchèrent vers Lisbonne ; il leur restait
quelque argent, avec lequel ils espéraient se sauver
de la faim après avoir échappé à la tempête \ldots
```

Bsp.
3-1-46

Der Abstand zwischen dem Initial und dem Folgetext in der ersten Zeile wird über den Befehl `\DefaultFindent` gesteuert (Standardwert `Opt`) und kann mithilfe des Schlüsselwortes `findent` überschrieben werden. Der Einzug der nachfolgenden Zeilen ist standardmäßig 0.5em groß (gespeichert in `\DefaultNindent`), kann aber über das Schlüsselwort `nindent` geändert werden. Ein geneigter Einzug ist mithilfe des Schlüsselwortes `slope` möglich, er beginnt ab der dritten Zeile. Auch hier kann der Standardwert, der im Befehl `\DefaultSlope` gespeichert ist, geändert werden. Es ist jedoch fraglich, ob jemals etwas anderes als `Opt` benötigt wird, da eine Neigung normalerweise nur für Buchstaben wie „A“ oder „V“ benutzt wird.

A PEINE ONT-ILS MIS le pied dans la ville en pleurant la mort de leur bienfaiteur, qu'ils sentent la terre trembler sous leurs pas; ...

```
\usepackage{palatino,lettrine}
\usepackage[ansinew]{inputenc}
\usepackage[french]{babel}

\lettrine[lines=4, slope=0.6em, findent=-1em,
  nindent=0.6em]{Å}{ peine ont-ils mis} le pied
dans la ville en pleurant la mort de leur
bienfaiteur, qu'ils sentent la terre trembler sous
leurs pas; \ldots
```

Bsp.
3-1-47

Das Beispiel oben zeigt deutlich, dass die Größenberechnung für das Initial Akzente unberücksichtigt lässt. So sollte es normalerweise auch sein. Natürlich kann die Größe auch hier manuell mit `loversize` angepasst werden.

Wenn links neben dem Initial noch Text eingefügt werden soll (z.B. ein einleitendes Zitat), ist dies mit dem Schlüsselwort `ante` möglich. Dieses Schlüsselwort ist das einzige, für das es keinen Befehl zum Setzen der Standardeinstellung gibt.

Durch Modifikation der Standardeinstellungen kann das Paket so umdefiniert werden, dass die Initiale nach ganz spezifischen Anforderungen formatiert werden. Die Definitionen können entweder in der Präambel oder in einer Datei namens `lettrine.cfg` durchgeführt werden, die beim Auffinden geladen wird.

3.1.11 Randausgleich in L^AT_EX

Zur Formatierung von Absätzen setzt L^AT_EX den bereits im T_EX-Programm implementierten Algorithmus ein, der standardmäßig bündige Absätze erzeugt. Mit anderen Worten: Die Abstände zwischen Wörtern werden leicht gedehnt

oder gestaucht, um Zeilen gleicher Länge zu erzeugen. T_EX erreicht dies mit einem Algorithmus, der versucht, für einen ganzen Absatz eine optimale Lösung zu finden. Dabei benutzt es die aktuellen Einstellungen von etwa 20 internen Parametern. Hierzu gehören Aspekte wie optisch übereinstimmende Zeilen, d.h. dass eine sehr locker gesetzte Zeile nicht direkt unter einer sehr eng beschriebenen Zeile steht und dass mehrere Trennstriche in aufeinander folgenden Zeilen als Zeichen schlechter Qualität betrachtet werden. Das Zusammenspiel dieser Parameter ist sehr kompliziert und selbst Experten können oft nicht voraussagen, wie das Ergebnis ausfallen wird. Da die Standardeinstellungen für fast alle Anwendungen geeignet sind, werden in diesem Buch nur einige dieser Parameter beschrieben. Anhang B.3.3 erörtert, wie die Arbeitsweise des Algorithmus überwacht werden kann. Tiefere Einblicke in die Thematik des automatischen Zeilenumbruchs in Absätzen gibt *The T_EXbook* [87, Kapitel 14], das den Algorithmus im Detail beschreibt, oder auch der sehr interessante Artikel von Michael Plass und Donald Knuth über dieses Thema, der in *Digital Typography* [83] abgedruckt ist.

Der global optimierende Ansatz von T_EX hat einen Nachteil, dem jeder früher oder später begegnen wird: Kleine Änderungen, wie die Korrektur eines Tippfehlers am Ende eines Absatzes, können drastische und überraschende Effekte haben, da sie den Zeilenumbruch des gesamten Absatzes betreffen können. Möglich und absolut nicht unwahrscheinlich ist beispielsweise, dass beim Löschen eines Wortes der Absatz um eine Zeile *länger* wird. Dieses Verhalten kann sehr ärgerlich sein, wenn man sich am Ende eines wichtigen Projektes befindet (wie der zweiten Ausgabe dieses Buches) und eine Korrektur bei den bereits manuell überarbeiteten Seitenumbrüchen ein heilloses Chaos anrichtet. In solch einer Situation ist es das Beste, an strategisch günstigen Stellen `\linebreak`- oder `\pagebreak`-Befehle einzufügen, um T_EX zu zwingen, eine Lösung zu wählen, die es normalerweise als minderwertig betrachtet. Diese manuellen Umbrüche können später leicht entfernt werden, indem man sich eigene Befehle definiert, z.B.

```
\newcommand\finallinebreak{\linebreak}
```

anstatt direkt Standardbefehle von L^AT_EX zu benutzen. Dadurch lassen sich Layoutänderungen für eine bestimmte Version leicht von anderen Verwendungen der ursprünglichen Befehle unterscheiden – diese Methode hat sich auch bei der Erstellung dieses Buches bewährt.

Die Wortzwischenräume in bündig formatierten Absätzen (die Leerräume zwischen einzelnen Wörtern) werden durch verschiedene T_EX-Parameter gesteuert. Die wichtigsten davon sind `\tolerance` und `\emergencystretch`. Wenn diese Parameter richtig eingestellt sind, kann man alle oder zumindest fast alle „Overfull box“-Warnungen vermeiden, ohne dass Zeilen manuell umbrochen werden müssen. Der Parameter `\tolerance` gibt an, wie stark Wortzwischenräume in einem Absatz von ihrer optimalen Größe abweichen dürfen.¹ Dieser Befehl ist ein T_EX-Zähler (kein L^AT_EX-Zähler) und hat daher eine etwas ungewöhnliche Syntax für die Wertzuweisung, z.B. `\tolerance=500`. Niedrigere Werte führen dazu, dass T_EX nur Lösungen in der Nähe des Optimums akzeptiert, höhere Werte erlauben größere Abweichungen beim

¹Die optimale Größe ist fontabhängig; siehe Abschnitt 7.10.3 auf Seite 439.

Vorsicht bei T_EXs
Vorstellung von
„unendlich schlecht“
(infinitely bad)

Setzen. Voreingestellt ist häufig ein Wert von 200. Wenn T_EX nicht innerhalb der vorgegebenen Toleranz bleiben kann, erhält man in der Ausgabe überlaufende Zeilen (d.h. Zeilen die, wie die vorherige, über den Rand hinaus ragen). Wenn der Wert für `\tolerance` erhöht wird, zieht T_EX auch ungünstigere Zeilenumbrüche in Betracht (wie in diesem Absatz), bevor es das Problem für eine manuelle Lösung an den Benutzer übergibt. Werte zwischen 50 und 9999 sind vernünftige Einstellungen, 10000 oder höhere Werte sollten nicht eingesetzt werden, denn diese erlauben es T_EX, beliebig schlechte Zeilenumbrüche vorzunehmen (so wie in dieser Zeile). Wenn man wirklich vollautomatische Zeilenumbrüche benötigt, z.B. bei Anwendungen, bei denen der Text automatisch aus einer Datenbank zusammengestellt wird, ist es besser, den Längenparameter `\emergencystretch` auf einen positiven Wert zu setzen. Wenn T_EX (aufgrund der Einstellungen für `\tolerance`) einen Absatz nicht umbrechen kann, ohne dabei Zeilenüberläufe zu produzieren, und `\emergencystretch` positiv ist, fügt es diese Länge als dehnbaren Leerraum jeder Zeile hinzu. Dadurch werden Zeilenumbrüche akzeptabel, die zuvor verworfen wurden. Das kann zu einigen „Underfull Box“-Warnungen (nicht ganz ausgefüllten Zeilen) führen, da nun alle Zeilen nach einem lockereren Maßstab gesetzt werden. Dieses Ergebnis sieht aber immer noch besser aus als eine einzelne, hässliche Zeile inmitten eines ansonsten perfekt gesetzten Absatzes.

Die oben beschriebenen Parameter werden in L^AT_EX durch zwei vordefinierte Befehle beeinflusst: durch den voreingestellten Befehl `\fussy` und den Befehl `\sloppy`, der relativ ungünstige Zeilenumbrüche zulässt. Der Befehl `\sloppy` wird von L^AT_EX automatisch immer dann verwendet, wenn perfekte Zeilenumbrüche aufgrund des engen Zeilenmaßes unwahrscheinlich sind (z.B. beim Formatieren von `\marginpar`-Befehlen oder von p-Spalten in der Umgebung `tabular`).

Text ohne Randausgleich

Wie man theoretisch hochwertigen, bündigen Text erzeugt, weiß im Prinzip jeder (auch wenn überraschender Weise nur wenige Satzsysteme außer T_EX Algorithmen verwenden, die tatsächlich beabsichtigt eine hohe Qualität produzieren), aber bei nicht bündigem Text sieht das Ganze etwas anders aus. Auf den ersten Blick erscheint das etwas seltsam. Warum sollte es schwierig sein, einen Absatz in Zeilen verschiedener Länge aufzuteilen? Die Antwort ist, dass es keine quantitativ messbaren Qualitätsmaßstäbe gibt, anhand derer man einfach sagen könnte, ob ein bestimmter Umbruch gut oder schlecht ist. Im Vergleich zu dem Ergebnis bei bündigem Text erzielt T_EX beim Erzeugen nicht bündiger Texte sehr schlechte Ergebnisse. Um höchste Qualität zu erhalten, ist deshalb in vielen Fällen ein Eingreifen des Benutzers erforderlich, der an strategischen Punkten explizite Zeilenumbrüche einfügt. Eine gute Einführung in diese Thematik bietet der Artikel von Paul Stiff [155].

Die wichtigste Variante des nicht bündigen Textes ist die, in der die Zeilen linksbündig formatiert sind und rechts flattern. Für diese Art des Satzes bietet L^AT_EX die Umgebung `flushleft`. Sie setzt den gesamten Text in seinem Gültigkeitsbereich „linksbündig“, indem auf der rechten Seite jeder Zeile sehr

dehnbarer Leerraum eingefügt wird; d.h. der interne Parameter `\rightskip` erhält den Wert `Opt plus 1fil`. Diese Einstellung erzeugt häufig sehr „zerupft“ aussehende Absätze, weil alle Zeilen gleich bewertet werden, ungeachtet dessen, wie viel Text sie enthalten. Außerdem wird die Silbentrennung im Wesentlichen unterdrückt, weil Binde- oder Trennstriche zur schlechten Bewertung („badness“) einer Zeile beitragen. Da es kein Mittel gibt, diesem Verhalten gegenzusteuern, wählt der Algorithmus von $\TeX$ vorzugsweise Zeilenumbrüche, bei denen eine Silbentrennung vermieden wird.

„Das Schriftsatzsystem $\LaTeX$ ist eine spezielle Form des $\TeX$ -Programms von Donald Knuth. $\TeX$ ist ein anspruchsvolles Programm zur Erzeugung hochwertig gesetzten Textes, insbesondere mathematischer Texte.“

```
\begin{flushleft}
"‘Das Schriftsatzsystem \LaTeX{} ist eine spezielle
Form des \TeX{}-Pro\gramms von Donald Knuth.
\TeX{} ist ein anspruchsvolles Programm zur
Erzeugung hochwertig gesetzten Textes, insbesondere
mathematischer Texte."’
\end{flushleft}
```

Bsp.
3-1-48

Alles in allem ist die $\LaTeX$ -Umgebung `flushleft` nicht sonderlich gut für längere nicht bündige Texte geeignet, die an der rechten Grenze lediglich um ein gewisses Maß variieren sollen und bei denen an geeigneter Stelle Trennstriche eingefügt werden sollen (siehe nächster Abschnitt für Alternativen). Nichtsdestotrotz kann sie nützlich sein, um einzelne Objekte, wie eine Graphik, linksbündig an den Rand zu setzen, insbesondere da diese Umgebung, genauso wie Listenumgebungen, einen Abstand über und unter sich einfügt.

Eine weitere, bedeutende Einschränkung ergibt sich aus der Tatsache, dass die von dieser Umgebung gewählten Einstellungen nicht allgemeingültig sind. Einige Umgebungen (wie `minipage` oder `tabular`) und Befehle (wie `\parbox`, `\footnote` und `\caption`) setzen die Ausrichtung für Absätze wieder auf Blocksatz zurück, d.h. sie setzen den Abstand zum rechten Rand (`\rightskip`) auf `Opt`. Damit wird der dehnbare Leerraum am rechten Zeilenrand gelöscht. Eine Möglichkeit, dieses Problem automatisch zu lösen, bietet das Paket `ragged2e` (siehe nächster Abschnitt).

Weitere Formate zum Setzen von Absätzen sind: rechtsbündig mithilfe der Umgebung `flushright` und zentriert mithilfe der Umgebung `center`. In diesen beiden Umgebungen werden Zeilenumbrüche normalerweise durch den Befehl `\\` erzeugt, während man beim Flattersatz (mit der oben erörterten Umgebung `flushleft`) $\LaTeX$ die Zeilenumbrüche überlassen kann (sofern man mit der Qualität des Ergebnisses zufrieden ist).

Die drei Umgebungen, die in diesem Abschnitt erklärt wurden, ändern das Absatzformat, indem sie die Parameter ändern, mit deren Hilfe $\TeX$ Absätze formatiert. Wie man aus der folgenden Zuordnungstabelle ersehen kann, sind sie auch als Deklarationen verfügbar:

<i>Umgebung:</i>	<code>center</code>	<code>flushleft</code>	<code>flushright</code>
<i>Befehl:</i>	<code>\centering</code>	<code>\raggedright</code>	<code>\raggedleft</code>

Anders als die entsprechenden Umgebungen beginnen diese Befehle keinen neuen Absatz und fügen keinen vertikalen Leerraum hinzu. Dadurch können sie innerhalb anderer Umgebungen, insbesondere in einer `\parbox`, verwendet werden, um die Ausrichtung in p-Spalten einer `array`- oder `tabular`-

Umgebung zu steuern. Zu beachten ist jedoch, dass das Ende einer Tabellenzeile nicht mit dem Befehl `\` angegeben werden kann, wenn diese Befehle in der letzten Spalte einer `tabular`- oder `array`-Umgebung verwendet werden. Für diesen Zweck kann stattdessen der Befehl `\tabularnewline` verwendet werden (siehe auch Abschnitt 5.2.1).

3.1.12 ragged2e – Verbessern des Randausgleichs

Im vorangegangenen Abschnitt wurden die Mängel der $\text{\LaTeX}$ -Umgebungen `flushleft` und `flushright` angesprochen. Das Paket `ragged2e` von Martin Schröder sucht Alternativen, die nicht solch extrem flatternde Texte erzeugen. Diese Aufgabe ist nicht so trivial, wie es sich anhört, denn es reicht nicht aus, `\rightskip` einen Wert wie `Opt plus 2em` zuzuweisen. Abgesehen davon, dass diese Einstellung dazu führen würde, dass $\text{\TeX}$ versucht, die Zeilenenden innerhalb der Grenze von `2em` zu halten, bleibt ein kleines Problem: Bei den meisten Zeichensätzen ist auch der Wortzwischenraum dehnbar. Wenn `\rightskip` nur eine endliche Dehnbarkeit hat, wird $\text{\TeX}$ an allen Leerräumen gleichmäßig zusätzlichen Leerraum einfügen. Je nach Dichte des Textes entstehen damit unterschiedlich große Wortzwischenräume. Hier muss der Wortzwischenraum so umdefiniert werden, dass er nicht mehr größer oder kleiner werden kann. Dies geschieht über die Angabe eines geeigneten (fontabhängigen) Wertes für `\spaceskip`. Dieser interne $\text{\TeX}$ -Parameter gibt, sofern er nicht null ist, den aktuellen Wortzwischenraum wieder, wodurch der Standardwert überschrieben wird, der vom aktuellen Zeichensatz definiert wurde.

Standardmäßig ändert das Paket nicht die $\text{\LaTeX}$ -Standardbefehle und -umgebungen, die im vorigen Abschnitt erörtert wurden. Stattdessen definiert es eigene Befehle und Umgebungen mit identischen Namen, aber unterschiedlicher Groß-/Kleinschreibung.¹ Die neuen Umgebungen und Befehle sind in der folgenden Zuordnungstabelle aufgeführt:

<i>Umgebung:</i>	<code>Center</code>	<code>FlushLeft</code>	<code>FlushRight</code>
<i>Befehl:</i>	<code>\Centering</code>	<code>\RaggedRight</code>	<code>\RaggedLeft</code>

Sie unterscheiden sich von ihren Gegenstücken aus dem vorigen Abschnitt nicht nur dadurch, dass sie versuchen, weniger flatternde Absätze zu produzieren, sondern auch dadurch, dass die meisten ihrer Formatierungsparameter geändert werden können. Dies erhöht die Flexibilität.

Da es etwas anstrengend ist, ständig die Befehle und Umgebungen mit gemischter Groß-/Kleinschreibung einzugeben, gibt es die Möglichkeit, die ursprünglichen Befehle und Umgebungen, wie `\raggedright`, mit den neuen Definitionen zu überschreiben. Hierzu gibt man beim Laden des Paketes die Option `newcommands` an.

Das Paket bietet eine Vielzahl an Parametern, die das genaue Verhalten der neuen Befehle und Umgebungen definieren (siehe Tabelle 3.2 auf der gegenüberliegenden Seite). Bei `\RaggedRight` oder `FlushLeft` kann der Leerraum, der an der rechten Seite jeder Zeile hinzugefügt wird, über den Parame-

¹Genau genommen widerspricht dies den Standard-Namenskonventionen. In den meisten Paketen weisen Befehle mit gemischter Groß-/Kleinschreibung auf Schnittstellenbefehle hin, die von Entwicklern in Klassendateien oder in der Präambel verwendet werden, jedoch nicht auf Dokumentbefehle.

<code>\RaggedLeftParindent</code>	Opt	<code>\RaggedLeftLeftskip</code>	Opt plus 2em
<code>\RaggedLeftRightskip</code>	Opt	<code>\RaggedLeftParfillskip</code>	Opt
<code>\CenteringParindent</code>	Opt	<code>\CenteringLeftskip</code>	Opt plus 2em
<code>\CenteringRightskip</code>	Opt plus 2em	<code>\CenteringParfillskip</code>	Opt
<code>\RaggedRightParindent</code>	Opt	<code>\RaggedRightLeftskip</code>	Opt
<code>\RaggedRightRightskip</code>	Opt plus 2em	<code>\RaggedRightParfillskip</code>	Opt plus 1fil
<code>\JustifyingParindent</code>	1 em	<code>\JustifyingParfillskip</code>	Opt plus 1fil

Tabelle 3.2: Von ragged2e verwendete Parameter

ter `\RaggedRightRightskip` eingestellt werden und der Leerraum auf der linken Seite mittels `\RaggedRightLeftskip`. Der zu verwendende Absatzeinzug kann über `\RaggedRightParindent` abgerufen werden und der Leerraum zum Auffüllen der letzten Zeile ist über `\RaggedRightParfillskip` verfügbar. Auf ähnliche Weise können auch die Einstellungen für `\Centering` und `\RaggedLeft` geändert werden; dazu muss lediglich `RaggedRight` in den Parameternamen durch `Centering` oder `RaggedLeft` ersetzt werden.

Um ein komplettes Dokument im Flattersatz zu formatieren, muss das `ragged2e`-Paket mit der Option `document` geladen werden. Für den bündigen Satz einzelner Absätze bietet das Paket den Befehl `\justifying` und die Umgebung `justify`. Beide können über die Längenparameter `\JustifyingParindent` und `\JustifyingParfillskip` individuell angepasst werden.

Nicht bündiger Satz als Standard

Ein Dokument mit mäßig flatternden Absätzen und einem Texteinzug von 12pt ließe sich also wie im folgenden Beispiel erzeugen (vgl. Beispiel 3-1-48 auf Seite 111).

„Das **S**chriftsatzsystem **L**^A**T**_E**X** ist eine spezielle Form des **T**_E**X**-Programms von Donald Knuth. **T**_E**X** ist ein anspruchsvolles Programm zur Erzeugung hochwertig gesetzten Textes, insbesondere mathematischer Texte.“

```
\usepackage[document]{ragged2e}
\setlength\RaggedRightRightskip{0pt plus 1cm}
\setlength\RaggedRightParindent{12pt} % Einzug
"‘Das Schriftsatzsystem \LaTeX{} ist eine spezielle
Form des \TeX{}-Pro\gramms von Donald Knuth.
\TeX{} ist ein anspruchsvolles Programm zur
Erzeugung hochwertig gesetzten Textes, insbesondere
mathematischer Texte.'"
```

Bsp.
3-1-49

Bei geringer Zeilenlänge (z.B. in `\marginpars`, `\parboxen`, `minipage`-Umgebungen oder `p`-Spalten von `tabular`-Umgebungen) führt Blocksatz zu eher dürrtigen Ergebnissen. Mit der Option `raggedrightboxes` werden Absätze an solchen Stellen automatisch mit `\RaggedRight` formatiert. Bei Bedarf kann dann mit `\justifying` bündiger Text erzeugt werden.

Nicht bündige Formatierung in engen Spalten

Wenn in den Voreinstellungen `em`-Werte verwendet werden (siehe Tabelle 3.2), ist besondere Vorsicht geboten, weil `em` beim Laden des Paketes in ein reales Maß umgewandelt wird. Daher sollte das Paket geladen werden, *nachdem* die Grundschrift und die Größe eingerichtet wurden – z.B. nach dem Laden der Fontpakete.

Die Standardwerte

Anstatt die in der Tabelle 3.2 aufgeführten Voreinstellungen zu verwenden, kann man das Paket auch anweisen, zunächst **L**^A**T**_E**X**s Standardwerte zu

verwenden, indem man es mit der Option `originalparameters` lädt und dann nach Bedarf die Parameterwerte wie gewünscht ändert.

3.1.13 `setspace` – Ändern des Zeilenvorschubs

Der Befehl `\baselineskip` ist ein $\TeX$ -Parameter, mit dem der *Zeilenvorschub* festgelegt wird. Dies ist der Abstand zweier aufeinander folgender Grundlinien, manchmal auch als Zeilenabstand oder als Durchschuss bezeichnet, siehe aber die Diskussion der Begriffe auf Seite 426. Standard- $\LaTeX$ definiert den Zeilenvorschub um fast 20% größer als die Entwurfsgröße des verwendeten Zeichensatzes. Da es nicht empfehlenswert ist, die Einstellung von `\baselineskip` direkt zu ändern, verfügt $\LaTeX$ über den Befehl `\linespread`, mit dem sich `\baselineskip` global für alle Größen ändern lässt. Nach einer Anweisung wie etwa `\linespread{1.5}\selectfont` tritt der neue Wert sofort in Kraft.¹

Das Paket `setspace` (von Geoffrey Tobin und anderen) definiert Umgebungen und Befehle zum Einfügen variabler Abstände (hauptsächlich doppelt und anderthalb). Um die Abstände im ganzen Dokument zu definieren, gibt es drei Befehle: `\singlespacing`, `\onehalfspacing` und `\doublespacing`; sie werden in der Präambel festgelegt. Daneben kann mit dem Befehl `\setstretch` ein anderer Abstand in der Präambel angegeben werden. Im (obligatorischen) Argument wird der gewünschte Vergrößerungsfaktor angegeben. Wenn keiner dieser Befehle vorhanden ist, wird die Standardeinstellung „einfacher Zeilenvorschub“ verwendet.

Abstände innerhalb des Dokumentes können mit drei spezifischen Umgebungen geändert werden: `singlespace`, `onehalfspace` und `doublespace`. Sie definieren einen einfachen, anderthalbfachen bzw. doppelten Zeilenvorschub. Diese Umgebungen lassen sich nicht verschachteln.

Am Anfang schuf Gott Himmel und Erde. Und die
Erde war wüst und leer, und es war finster auf der Tie-
fe, und der Geist Gottes schwebte auf dem Wasser.

```
\usepackage{setspace}
\begin{doublespace}
Am Anfang schuf Gott Himmel und Erde.
Und die Erde war wüst und leer, und
es war finster auf der Tiefe, und der
Geist Gottes schwebte auf dem Wasser.
\end{doublespace}
```

Bsp.

3-1-50

Mit der Umgebung `spacing` können beliebige Zeilenvorschübe definiert werden. Sie erwartet als obligatorisches Argument den Wert für `\baselinestretch`, der für den Inhalt der Umgebung verwendet werden soll.

Am Anfang schuf Gott Himmel und Erde. Und die
Erde war wüst und leer, und es war finster auf der Tie-
fe, und der Geist Gottes schwebte auf dem Wasser.

```
\usepackage{setspace}
\begin{spacing}{2.0}
Am Anfang schuf Gott Himmel und Erde.
Und die Erde war wüst und leer, und
es war finster auf der Tiefe, und der
Geist Gottes schwebte auf dem Wasser.
\end{spacing}
```

Bsp.

3-1-51

¹Die veraltete $\TeX$ 2.09 Lösung `\renewcommand\baselinestretch{1.5}` benötigt dagegen einen nachfolgenden Schriftgrößenwechsel (wie `\small` oder `\Large`), damit der neue Wert in Kraft tritt.

<i>Abstand</i>	10pt	11pt	12pt
anderthalb	1.25	1.21	1.24
doppelt	1.67	1.62	1.66

Tabelle 3.3: Werte für `\baselinestretch` bei verschiedenen Schriftgrößen

Im obigen Beispiel erzeugt der Koeffizient „2.0“ einen Zeilenvorschub, der größer ist als der „doppelte Zeilenabstand“ (doublespace-Umgebung), der für einige Publikationen benötigt wird. Mit der Umgebung `spacing` wird der Zeilenvorschub zweimal vergrößert – einmal durch `\baselineskip` (durch den der Zeilenvorschub bereits um etwa 20% gegenüber der Schriftgröße erhöht wird) und zum Zweiten durch Setzen von `\baselinestretch`. „Doppelter Abstand“ bedeutet, dass der vertikale Abstand zwischen den Grundlinien ungefähr doppelt so groß ist wie die Zeichengröße. Da `\baselinestretch` das Verhältnis zwischen dem gewünschten Zeilenvorschub und `\baselineskip` angibt, lässt sich der korrekte Wert für `\baselinestretch` bei verschiedenen Grundschriftgraden leicht errechnen. In Tabelle 3.3 sind die Werte für zwei unterschiedliche Abstände und drei Grundschriftgrade dargestellt.

3.1.14 picinpar – Rechteckige Löcher in Absätzen

Mit dem Paket `picinpar` (das Friedhelm Sowa auf der Basis früherer Arbeiten von Alan Hoenig entwickelte) können „Fenster“ innerhalb von Absätzen gesetzt werden. Die Basisumgebung dafür ist `window`. Sie hat ein obligatorisches Argument in Form einer durch Kommas getrennten Liste aus vier Elementen, das im Gegensatz zu den $\text{\LaTeX}$ -Konventionen in eckigen Klammern gesetzt wird. Diese Elemente geben die Anzahl an Zeilen vor dem Beginn des Fensters sowie die Ausrichtung des Fensters im Absatz (l für links, c für zentriert und r für rechts) an, außerdem den im Fenster dargestellten Text, das Bild o. ä. und erläuternden Text über den Inhalt im Fenster (z.B. die Bildunterschrift).

Im vorliegenden Beispiel wird ein vertikal gedrucktes Wort horizontal und vertikal Absatz eingefügt. Es ist leicht vorstellbar, dass sich mit der Umgebung `tabwindow` auch Tabellen einfach einbinden lassen.

Wenn ein Fenster länger ist als der Absatz, in dem es beginnt, dann reicht es über das Ende des Absatzes in den oder die nächsten Absätze hinein.

```
\usepackage{picinpar}
```

```
\begin{window}[1,c,%
```

```
\fbox{\shortstack{H\\a\\l\\l\\o}},]
```

```
\end{window}
```

Im vorliegenden Beispiel wird ein vertikal gedrucktes Wort horizontal und vertikal zentriert in einen Absatz eingefügt. Es ist leicht vorstellbar, dass sich mit der Umgebung `\texttt{tabwindow}` auch Tabellen einfach einbinden lassen. `\par` Wenn ein Fenster länger ist als der Absatz, in dem es beginnt, dann reicht es über das Ende des Absatzes in den oder die nächsten Absätze hinein.

Bsp.
3-1-52

Bei genauerer Betrachtung des obigen Beispiels kann man feststellen, dass der Einzug des zweiten Absatzes nicht stimmt. Dieser Fehler lässt sich einfach beheben, indem der Einzug bei Bedarf explizit mit `\par\indent` angefordert wird.

Ein Fenster wie im vorigen Beispiel zu zentrieren, funktioniert nur, wenn der umfließende Text auf beiden Seiten noch ausreichend breit ist (wobei „ausreichend“ mehr als ein Zoll ist). Ansonsten wird das Paket den Platz einfach mit Weißfläche auffüllen.

Das Paket stellt zwei Varianten der Umgebung namens `figwindow` und `tabwindow` zur Verfügung. Sie formatieren den erläuternden Text als Bild- oder Tabellenunterschriften und fügen eine entsprechende Nummerierung hinzu. Man sollte solch „nicht gleitende“ Gleitobjekte nur mit größter Sorgfalt mit den Standardumgebungen `figure` und `table` vermischen, da letztere hinter die nicht gleitenden Objekte rutschen können, womit eine falsche Reihenfolge der Abbildungs- bzw. Tabellennummern entstünde.

Das nächste Beispiel zeigt solch eine eingebettete Abbildung – eine Karte von Großbritannien in einem Absatz. Leider ist die Formatierung der Bildunterschrift mehr oder weniger fest im Paket integriert; um sie zu ändern, muss der interne Befehl `\@makewincaption` angepasst werden.

Is this a dagger which I see before me,
The handle toward my hand? Come, let
me clutch thee. I have thee not, and yet



Figure 1: United Kingdom

I see thee still. Art thou not, fatal vision,
Sensible To feeling as to sight? or
art thou but A dagger of the mind, a false
creation, Proceeding from the heat-
oppressed brain? I see thee yet, in form as pal-
pable As this which now

I draw. Thou marshall'st me the way that
I was going; And such an instrument I was
to use. Mine eyes are made the fools o' the
other senses, Or else worth all the rest; I see
thee still, And on thy blade and dudgeon
gouts of blood, Which was not so before.
(*Macbeth*, Act II, Scene 1).

```
\usepackage{pictinpar,graphicx}
```

```
\begin{figwindow}[3,1,%  
  \fbox{\includegraphics[width=20mm]{ukmap}},%  
  {United Kingdom}]
```

Is this a dagger which I see before me, The
handle toward my hand? Come, let me clutch
thee. I have thee not, and yet I see thee
still. Art thou not, fatal vision,
sensible To feeling as to sight? or art
thou but A dagger of the mind, a false
creation, Proceeding from the
heat-oppressed brain? I see thee yet, in
form as palpable As this which now I draw.
Thou marshall'st me the way that I was
going; And such an instrument I was to use.
Mine eyes are made the fools o' the other
senses, Or else worth all the rest; I see
thee still, And on thy blade and dudgeon
gouts of blood, Which was not so before.
(\emph{Macbeth}, Act II, Scene 1).

```
\end{figwindow}
```

Bsp.
3-1-53

3.2 Fußnoten, Endnoten und Marginalien

L^AT_EX verfügt über Möglichkeiten „eingefügten“ Text, wie Marginalien, Fußnoten, Abbildungen und Tabellen zu setzen. Der vorliegende Abschnitt befasst sich eingehend mit den verschiedenen Arten von Anmerkungen, während in Kapitel 6 Gleitobjekte ausführlich beschrieben werden.

Den Anfang macht eine Erörterung der Möglichkeiten, die L^AT_EXs Fußnotenbefehle bieten. Es wird gezeigt, wie und wie weit sie an individuelle Bedürfnisse angepasst werden können. Für zweispaltige Dokumente bietet das Paket `ftnright` ein spezielles Layout für Fußnoten, bei dem alle Fußnoten am unteren Rand der rechten Spalte gesetzt werden. Anschließend wird das Paket `footmisc` vorgestellt, das die meisten Einschränkungen der Standardbefehle beseitigt und eine Fülle zusätzlicher Funktionen bietet. Das Paket `manyfoot`

(das mit `footmisc` kombiniert werden kann) erweitert die Unterstützung von Fußnoten auf Bereiche wie die Linguistik, indem es mehrere, voneinander unabhängige Fußnotenbefehle bereitstellt.

Endnoten werden durch das Paket `endnotes` unterstützt, das eine Mischung von Fußnoten und Endnoten ermöglicht. Es kann auch zum Erstellen von Anmerkungsapparaten verwendet werden, wie es manche Verlage vorschreiben. Der Abschnitt schließt mit einer Beschreibung von Marginalien, die bereits Bestandteil von Standard- $\text{\LaTeX}$ sind.

3.2.1 Verwenden von Standardfußnoten

$\text{\LaTeX}$ unterscheidet streng zwischen Fußnoten im Haupttext und Fußnoten in einer `minipage`-Umgebung. Erstere werden mithilfe des Zählers `footnote` nummeriert, während der Befehl `\footnote` innerhalb der `minipage`-Umgebung so definiert wird, dass er den Zähler `mpfootnote` verwendet. Dadurch wird die Darstellung des Fußnotenzeichens je nach Kontext durch den Befehl `\thefootnote` oder `\thempfootnote` festgelegt. Als Standardzeichen wird im Text eine arabische Zahl und in `minipage`-Umgebungen ein Kleinbuchstabe verwendet. Andere Darstellungen für Fußnotenzeichen erhält man durch Neudefinition dieser Befehle:

Bsp.
3-2-1

Text Text Text* Text Text[†] Text.

*Die erste
†Die zweite

```
\renewcommand\thefootnote
{\fnsymbol{footnote}}
Text Text Text\footnote{Die erste}
Text Text\footnote{Die zweite} Text.
```

Fußnoten, die innerhalb einer `minipage`-Umgebung mit dem Befehl `\footnote` erstellt werden, verwenden den Zähler `mpfootnote` und werden am Fuß der Absatzbox gesetzt, die von `minipage` erzeugt wird. Wenn man allerdings in der `minipage`-Umgebung den Befehl `\footnotemark` verwendet, erzeugt dieser Fußnotenzeichen im gleichen Stil und in derselben Reihenfolge wie die Haupttextfußnoten – d.h. der Zähler `footnote` wird weiter hochgesetzt, und die Darstellung erfolgt mithilfe des Befehls `\thefootnote`. Dadurch kann man innerhalb der `minipage`-Umgebung auch eine Fußnote erzeugen, die am Fuß der Seite in einer Sequenz mit den Fußnoten des Haupttextes gesetzt wird: Man platziert den Befehl `\footnotemark` innerhalb der `minipage`-Umgebung und hinter die Umgebung setzt man den entsprechenden Befehl `\footnotetext`.

*Besonderheiten
innerhalb von
minipage*

... Haupttext ...

Fußnoten werden hier mit Kleinbuchstaben nummeriert.^a

Dieser Text bezieht sich auf eine Fußnote am Fuß der Seite.¹ Und eine weitere^b Fußnote.

^aAuf der Miniseite.

^bWieder im Text

... Haupttext ...

¹Am Fuß der Seite

Bsp.
3-2-2

```
\noindent\ldots{} Haupttext \ldots
\begin{center}
\begin{minipage}{.9\linewidth}
Fußnoten werden hier mit Kleinbuchstaben
nummeriert.\footnote{Auf der Miniseite.}
\par Dieser Text bezieht sich auf eine
Fußnote am Fuß der Seite.\footnotemark{}
Und eine weitere\footnote{Wieder im Text}
Fußnote.
\end{minipage}\footnotetext{Am Fuß der Seite}
\end{center}
\ldots{} Haupttext \ldots
```


Wie das vorangegangene Beispiel zeigt, kann beim Verweis auf eine Fußnote innerhalb einer `minipage`-Umgebung `\footnotemark` nicht verwendet werden, weil dieser Befehl auf Fußnoten am Fuß der Seite verweist. Für diesen Fall kann man das Paket `footmisc` laden und anstelle von `\footnotemark` den Befehl `\mpfootnotemark` verwenden. Wie bei `\footnotemark` wird mit `\mpfootnotemark` zunächst der Zähler hochgesetzt und dann der Wert dargestellt. Um auf den vorherigen Wert zu verweisen, muss der Wert in der Regel zuerst heruntergesetzt werden, wie das nächste Beispiel zeigt.

Haupttext ...

Fußnoten werden hier mit Kleinbuchstaben nummeriert.^a

Dieser Text bezieht sich auf die vorangegangene Fußnote.^a Und eine weitere^b Fußnote.

^aAuf der Miniseite.

^bAuch innerhalb.

... Haupttext ...

```
\usepackage{footmisc}
\noindent Haupttext \ldots \begin{center}
\begin{minipage}{.9\linewidth}
Fußnoten werden hier mit Kleinbuchstaben
nummeriert.\footnote{Auf der Miniseite.}
\par Dieser Text bezieht sich auf die
vorangegangene Fußnote.%
\addtocounter{mpfootnote}{-1}%
\mpfootnotemark{}
Und eine weitere\footnote{Auch innerhalb.}
Fußnote.
\end{minipage}
\end{center} \ldots{} Haupttext \ldots
```

Bsp.
3-2-3

Obwohl es in einigen Disziplinen durchaus üblich ist, können mit $\text{\LaTeX}$ innerhalb von Fußnoten keine weiteren `\footnote`-Befehle gesetzt werden. Man kann jedoch den Befehl `\footnotemark` in der ersten Fußnote verwenden und den dazugehörigen Text als Argument des Befehls `\footnotetext` eingeben. Weitere spezielle Fußnoten Anforderungen werden vom Paket `manyfoot` bedient (siehe unten).

Text¹ und noch mehr Text.

¹Eine Beispiel²-Fußnote.

²Eine Fußnoten-Fußnote.

```
Text\footnote{Eine Beispiel\footnotemark{}-%
Fußnote.}\footnotetext{Eine Fußnoten-Fußnote.}
und noch mehr Text.
```

Bsp.
3-2-4

Wie erstellt man einen Querverweis zu einer Fußnote? Zu diesem Zweck kann man den normalen $\text{\LaTeX}$ -Mechanismus von `\label` und `\ref` verwenden. Alternativ besteht die Möglichkeit, sich einen eigenen Befehl zu definieren, um diese Art von Verweisen besonders darzustellen, wie z.B.:

Dies ist irgendein Text.¹

... wie an Fußnote (1) auf Seite 6 ersichtlich, kann man auf Fußnoten verwei-

```
\newcommand{\fnref[1]{\unskip~(\ref{#1})}
Dies ist irgendein Text.\footnote{Text innerhalb
der Bezugsfußnote\label{fn:myfoot}.}

\ldots wie an Fußnote\fnref{fn:myfoot}
auf Seite~\pageref{fn:myfoot} ersichtlich,
kann man auf Fußnoten verweisen \ldots
```

¹Text innerhalb der Bezugsfußnote.

Bsp.
3-2-5

Standard- $\text{\LaTeX}$ erlaubt keine Fußnoten innerhalb von Tabellen. In Abschnitt 5.8 werden einige Wege zur Lösung des Problems aufgezeigt.

3.2.2 Anpassen von Fußnoten

Fußnoten sind in $\text{\LaTeX}$ in der Regel einfach zu benutzen, und $\text{\LaTeX}$ besitzt einen leistungsfähigen Mechanismus, um Text am Fuß einer Seite zu setzen.¹ Eine Fußnote kann dabei aus mehreren Absätzen bestehen und sowohl Listen als auch im Text eingebettete oder abgesetzte Formeln, Tabellen und Ähnliches enthalten.

$\text{\LaTeX}$ bietet verschiedene Parameter, mit denen Fußnoten angepasst werden können. Sie sind in Abbildung 3.1 auf der nächsten Seite schematisch dargestellt und bedeuten:

`\footnotesize` Die Schriftgröße, die für Fußnoten verwendet wird (siehe auch Tabelle 7.1 auf Seite 353).

`\footnotesep` Die Höhe einer Linie ohne horizontale Ausdehnung (englisch: strut), die am Anfang jeder Fußnote platziert wird. Wenn sie größer ist als der von `\footnotesize` verwendete Zeilenvorschub `\baselineskip`, wird über jeder Fußnote ein zusätzlicher vertikaler Leerraum eingefügt. In Anhang A.2.3 werden „struts“ näher erläutert.

`\skip\footins` Ein Low-Level-Längenparameter von $\text{\TeX}$, der den Abstand zwischen dem Haupttext und dem Beginn der Fußnoten definiert. Dieser Wert kann mit dem Befehl `\setlength` oder `\addtolength` verändert werden, indem man im ersten Argument `\skip\footins` verwendet:

```
\addtolength{\skip\footins}{10mm plus 2mm}
```

`\footnoterule` Ein Makro, das die Linie zeichnet, welche die Fußnote vom Haupttext trennt. Das Makro wird sofort nach dem vertikalen Abstand `\skip\footins` ausgeführt. Die Linie sollte rechnerisch keinen vertikalen Raum belegen, d.h. der Raum, den sie belegt, sollte durch einen negativen Abstand kompensiert werden. Die Standarddefinition sieht in etwa so aus:

```
\renewcommand\footnoterule{\vspace*{-3pt}%
\hrule width 2in height 0.4pt \vspace*{2.6pt}}
```

Zu beachten ist, dass hier der $\text{\TeX}$ -Befehl `\hrule` und nicht der $\text{\LaTeX}$ -Befehl `\rule` verwendet wurde. Letzterer leitet einen neuen Absatz ein. Dies erschwert die Berechnung des Leerraums, der benötigt wird, damit der Effekt „null Höhe“ erzielt wird. Aus diesem Grund lässt sich eine originellere „Linie“ am besten erstellen, indem man eine `picture`-Umgebung ohne Ausdehnung verwendet, in welche die Linie platziert wird, ohne vertikalen Abstand hinzuzufügen.

In den Klassen `report` und `book` werden Fußnoten innerhalb von Kapiteln nummeriert; in `article` erhalten Fußnoten eine im ganzen Dokument durchgängige sequentielle Nummerierung. Letzteres kann mit dem Befehl `\@addtoreset` geändert werden (siehe Anhang A.1.4). Dieser Mechanismus

¹Eine interessante und vollständige Abhandlung dieses Themas erschien in der Zeitschrift *Cahiers GUTenberg* [10, 135] der französischen $\text{\TeX}$ -Usergroup.

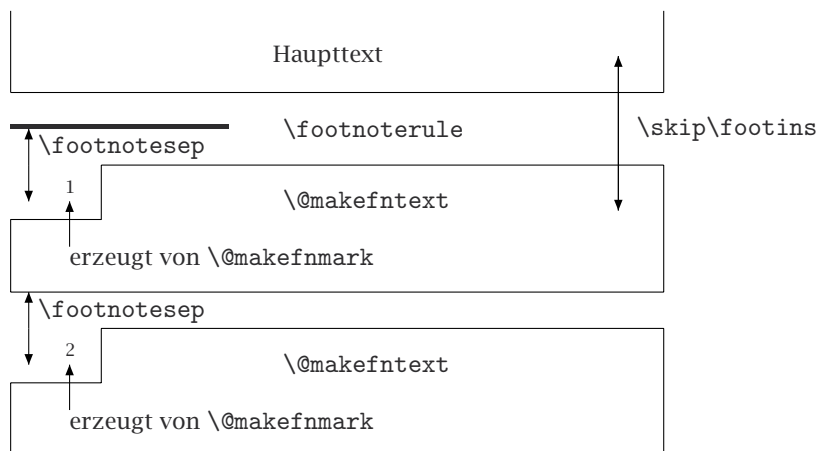


Abbildung 3.1: Schematische Darstellung einer Fußnote

sollte jedoch nicht verwendet werden, um Fußnoten seitenweise zu nummerieren. $\text{\LaTeX}$ arbeitet vorausschauend, während es die endgültigen Seiten erstellt, und würde deshalb viele Fußnoten vollkommen falsch nummerieren. Zur seitenbezogenen Nummerierung von Fußnoten sollte das Paket `footmisc` oder `perpage` verwendet werden (Beschreibung siehe unten).

Der Befehl `\@makefnmark` wird normalerweise verwendet, um ein Fußnotenzeichen zu erstellen. Man würde erwarten, dass dieser Befehl ein Argument hat (die aktuelle Fußnotennummer), aber dies ist nicht der Fall. Stattdessen nutzt er den Befehl `\@thefnmark`, um indirekt auf diese Nummer zu verweisen. Die Erklärung ist einfach: Je nachdem, ob die Nummer innerhalb oder außerhalb einer `minipage`-Umgebung liegt, muss auf einen anderen Zähler zugegriffen werden. Die Definition, die standardmäßig ein hochgestelltes Zeichen erzeugt, sieht in etwa folgendermaßen aus:

```
\renewcommand\@makefnmark
{\mbox{\textsuperscript{\normalfont\@thefnmark}}}
```

Der `\footnote`-Befehl führt `\@makefntext` in einer `\parbox` der Breite `\columnwidth` aus. Die Standardversion sieht in etwa folgendermaßen aus:

```
\newcommand\@makefntext[1]
{\noindent\makebox[1.8em][r]{\@makefnmark}\#1}
```

Hier wird das Fußnotenzeichen rechtsbündig in einer Box der Breite `1.8em` gesetzt, unmittelbar gefolgt vom Fußnotentext. Zu beachten ist die Mehrfachverwendung des Makros `\@makefnmark`; dadurch bewirken alle Änderungen am Makro, dass die Darstellung des Fußnotenzeichens an beiden Stellen geändert wird. Wenn der Text linksbündig gesetzt werden und die Fußnotenmarkierung im Rand stehen soll, sollte die im nächsten Beispiel definierte Variante verwendet werden. In diesem Fall wird das Fußnotenzeichen nicht mithilfe von `\@makefnmark` formatiert, sondern stattdessen per `\@thefnmark`

auf die Nummer zugegriffen. Das Ergebnis ist, dass das Zeichen nicht hochgestellt, sondern auf der Grundlinie platziert wird. Auf diese Weise werden die Markierungen im Text und am Fuß der Seite unterschiedlich formatiert.

Text Text Text¹ Text Text² Text.

Bsp.
3-2-6

1. Die erste
2. Die zweite

```
\makeatletter
\renewcommand\@makefnmark[1]{%
  {\noindent\makebox[0pt][r]{\@thefnmark.\,}\#1}
\makeatother
Text Text Text\footnote{Die erste}
Text Text\footnote{Die zweite} Text.
```

3.2.3 ftnright – Rechte Fußnoten im Zweispaltensatz

Bei zweispaltigem Satz ist es manchmal wünschenswert, alle Fußnoten am Fuß der rechten Spalte zu positionieren. Dies lässt sich mithilfe des Paketes `ftnright` (von Frank Mittelbach) bewerkstelligen. Die Wirkung dieses Paketes ist in Abbildung 3.2 auf der nächsten Seite dargestellt – die erste Seite der Originaldokumentation des Paketes (einschließlich Rechtschreibfehler). In der Abbildung ist deutlich zu sehen, wie die Fußnoten im unteren Teil der rechten Spalte versammelt sind.

Die zentrale Idee bei dem Paket `ftnright` ist, dass alle Fußnoten einer Seite am Fuß der rechten Spalte gesetzt werden. Es wird ausreichend Abstand zwischen Fußnoten und Text gelassen und die Fußnoten werden in einem kleineren Schriftgrad gesetzt.¹ Außerdem werden die Fußnotenzeichen auf der Grundlinie gesetzt und nicht hochgestellt.²

Das Paket kann zusammen mit den meisten Klassendateien von $\text{\LaTeX}$ verwendet werden. Allerdings wird das Paket `ftnright` nur bei Dokumenten wirksam, die mithilfe der Option `twocolumn` für den Befehl `\documentclass` zweispaltig formatiert werden. In den meisten Fällen ist es am sichersten, `ftnright` als letztes Paket zu verwenden, um sicherzugehen, dass seine Einstellungen nicht durch andere Pakete überschrieben werden.

3.2.4 footmisc – Verschiedene Fußnotenstile

Da Standard- $\text{\LaTeX}$ nur eine Art von Fußnoten und nur begrenzte (und irgendwie Low-Level-) Unterstützung zur Anpassung bietet, entwickelten einige Leute kleine Pakete, die Funktionen anbieten, die ansonsten nicht verfügbar sind. Viele dieser früheren Bemühungen wurden von Robin Fairbairns in seinem Paket `footmisc` zusammengefasst, das unter anderem die seitenweise Nummerierung von Fußnoten und das Setzen von mehreren Fußnoten in einem einzigen Absatz am Fuß der Seite unterstützt. In diesem Abschnitt werden die Funktionen dieses Paketes beschrieben und es wird (sofern zutreffend) aufgezeigt, welche älteren Pakete es ersetzt.

¹Manche Zeitschriften benutzen den gleichen Schriftgrad für Fußnoten und Text, so dass man beide nur schwer auseinander halten kann.

²Dies gilt natürlich nur für das Zeichen vor dem Fußnotentext, nicht für die Markierung im Haupttext, wo eine hochgestellte, in kleinerer Schriftgröße gesetzte Zahl oder ein Symbol den Lesefluss nicht unterbrechen.

Footnotes in a multi-column layout

Frank Mittelbach

August 10, 1991

1 Introduction

The placement of footnotes in a multi-column layout always bothered me. The approach taken by $\LaTeX$ (i.e., placing the footnotes separately under each column) might be all right if nearly no footnotes are present. But it looks clumsy when both columns contain footnotes, especially when they occupy different amounts of space.

In the multi-column style option [5], I used page-wide footnotes at the bottom of the page, but again the result doesn't look very pleasant since short footnotes produce undesired gaps of white space. Of course, the main goal of this style option was a balancing algorithm for columns which would allow switching between different numbers of columns on the same page. With this feature, the natural place for footnotes seems to be the bottom of the page¹ but looking at some of the results it seems best to avoid footnotes in such a layout entirely.

Another possibility is to turn footnotes into endnotes, i.e., printing them at the end of every chapter or the end of the entire document. But I assume everyone who has ever read a book using such a layout will agree with me, that it is a pain to search back and forth, so that the reader is tempted to ignore the endnotes entirely.

When I wrote the article about "Future extensions of $\TeX$ " [6] I was again dissatisfied with the outcome of the footnotes, and since this article should show certain aspects of high quality typesetting, I decided to give the footnote problem a try and modified the $\LaTeX$ output routine for this purpose. The layout I used was inspired by the yearbook of the Gutenberg Gesellschaft Mainz [1]. Later on, I found that it is also recommended by Jan White [9]. On the layout of footnotes I also consulted books by Jan Tschichold [8] and Manfred Simoneit [7], books, I would recommend to everyone being able to read German texts.

1.1 Description of the new layout

The result of this effort is presented in this paper and the reader can judge for himself whether it was successful or not.² The main idea for this layout is to assemble the footnotes of all columns on a page and place them all

together at the bottom of the right column. Allowing for enough space between footnotes and text, and in addition, setting the footnotes in smaller type³ I decided that one could omit the footnote separator rule which is used in most publications prepared with $\TeX$.⁴ Furthermore, I decided to place the footnote markers⁵ at the baseline instead of raising them as superscripts.⁶

All in all, I think this generates a neat layout, and surprisingly enough, the necessary changes to the $\LaTeX$ output routine are nevertheless astonishingly simple.

1.2 The use of the style option

This style option might be used together with any other style option for $\LaTeX$ which does not change the three internals changed by `ftnright.sty`.⁷ In most cases, it is best to use this style option as the very last option in the `\documentstyle` command to make sure that its settings are not overwritten by other options.⁸

The $\LaTeX$ style option `ftnright` which is described in this article has the version number v1.0d dated 92/06/19. The documentation was last revised on 92/06/19.

1 You can not use column footnotes at the bottom, since the number of columns can differ on one page.

2 Please note, that this option only changed the placement of footnotes. Since this article also makes use of the `doc` option [4], that assigns tiny numbers to code lines sprinkled throughout the text, the resulting design is not perfect.

3 The standard layout in *TUGboat* uses the same size for footnotes and text, giving the footnotes, in my opinion, much too much prominence.

4 People who prefer the rule can add it by redefining the command `\footnoterule` [2, p. 156]. Please, note, that this command should occupy no space, so that a negative space should be used to compensate for the width of the rule used.

5 The tiny numbers or symbols, e.g., the '5' in front of this footnote.

6 Of course, this is only done for the mark preceding the footnote text and not the one used within the main text where a raised number or symbol set in smaller type will help to keep the flow of thoughts, uninterrupted.

7 These are the macros `\@startcolumn`, `\@makecol` and `\@outputdblcol` as we will see below. Of course, the option will take only effect with a document style using a twocolumn layout (like *ltugboat*) or when the user additionally specifies `twocolumn` as a document style option in the `\documentstyle` command.

8 The *ltugboat* option (which is currently set up as a style option instead of a document style option which it actually is) will overwrite

1

Abbildung 3.2: Platzieren von Text und Fußnoten mit dem Paket `ftnright`

Die Schnittstelle für `footmisc` ist recht einfach: Nahezu alles kann durch Angabe von Optionen angepasst werden, wenn das Paket geladen wird. In einigen Fällen ist allerdings eine weitere Steuerung durch Parameter möglich.

In der Klasse `article` werden Fußnoten im gesamten Dokument fortlaufend nummeriert, in den Klassen `report` und `book` auf Kapitel bezogen. Manchmal ist es jedoch sinnvoller, Fußnoten seitenweise zu nummerieren. Hierzu muss `footmisc` mit der Option `perpage` geladen werden. Das Paket `footnpag` (von Joachim Schrod) bietet dieselbe Funktionalität in einer etwas anderen Implementierung als separates Paket. Eine universelle Variante zur Rücksetzung von Zählern auf Seitenbasis liefert das Paket `perpage` (siehe Abschnitt 3.2.5 auf Seite 127). Da $\TeX$ s Seitenaufbau-Mechanismus asynchron ist, müssen Dokumente immer zweimal bearbeitet werden, um eine korrekte Nummerierung

zu erhalten. Glücklicherweise warnt das Paket mit der Meldung „Rerun to get cross-references right“ vor, wenn die Fußnotennummerierung falsch ist. Das Paket speichert die Information zwischen den einzelnen Läufen in der Datei .aux. Nach vielen Bearbeitungsschritten entspricht diese Information möglicherweise nicht mehr ganz den Tatsachen. In solch einem Fall hilft es, die Datei .aux zu löschen, damit das Paket die richtige Nummerierung schneller findet.¹

Bsp.
3-2-7

Etwas Text* mit einer Fußnote. Mehr[†] *Erstens. [†]Zweitens.

Text. Noch mehr Text.* Und noch[†] *Drittens. [†]Viertens.

```
\usepackage[perpage,symbol]{footmisc}
Etwas Text\footnote{Erstens.} mit einer
Fußnote. Mehr\footnote{Zweitens.} Text.
Noch mehr Text.\footnote{Drittens.} Und
noch\footnote{Viertens.} mehr Text.
```

Für diesen speziellen Fall zeigt das obige Beispiel zwei Seiten nebeneinander, auf denen die Auswirkungen von `perpage` deutlich zu sehen sind. Das Beispiel zeigt auch den Effekt einer weiteren Option, und zwar `symbol`. Diese Option verwendet Fußnotenzeichen anstelle von Nummern. Da es nur eine begrenzte Anzahl solcher Textsymbole gibt, kann diese Option nur verwendet werden, wenn es insgesamt nur wenige Fußnoten gibt oder die Fußnotennummerierung auf jeder Seite neu beginnt. Es stehen sechs verschiedene Textsymbole für Fußnoten zur Verfügung, wobei Standard- $\text{\LaTeX}$ durch Duplikation einiger Symbole bis zu neun Fußnoten bereitstellen kann. Durch Verdreifachung einiger Symbole unterstützt `footmisc` bis zu 16 Fußnoten (insgesamt bzw. pro Seite). Wenn diese Anzahl überschritten wird, wird eine $\text{\LaTeX}$ -Fehlermeldung ausgegeben.

 Fehler:
„Counter too large“

Insbesondere bei der Option `perpage` kann diese Eigenschaft sehr störend sein, weil der Fehler möglicherweise nur dann auftritt, während das Paket noch zu ermitteln versucht, welche Fußnote zu welcher Seite gehört, d.h. er keinen wirklichen Fehler darstellt. Als Ausweichlösung für dieses Problem kann man auch die alternative Option `symbol*` verwenden. Sie erzeugt ebenfalls Fußnotensymbole, aber wenn diese zur Neige gehen, setzt sie die Nummerierung mit arabischen Ziffern fort. In diesem Fall wird am Ende eines Laufs eine Warnung ausgegeben, dass einige Fußnoten nicht mehr im Gültigkeitsbereich liegen, und die Details werden in der Transkriptdatei gespeichert.

<pre>\setfnsymbol{name} \DefineFNSymbols*{name}[typ]{symbol-list}</pre>

Bei Auswahl von `symbol` oder `symbol*` wird die von Leslie Lamport definierte Standardreihenfolge der Fußnotensymbole verwendet. Andere Autoritäten schlagen andere Reihenfolgen vor; aus diesem Grund bietet `footmisc` drei weitere Sequenzen zur Auswahl, die mit der Deklaration `\setfnsymbol` verwendet werden können (siehe Tabelle 3.4 auf der nächsten Seite).

¹In der Tat hat es der Autor dieses Buches geschafft, `footmisc` während der Bearbeitung dieses Kapitels so sehr durcheinander zu bringen (indem er in einem Beispiel `\textheight` änderte), dass das Paket die richtige Nummerierung hinterher nicht mehr fand und ständig einen neuen Bearbeitungslauf verlangte. Durch Löschen der Datei .aux konnte das Problem behoben werden.

lamport	*	†	‡	§	¶		**	††	‡‡	§§	¶¶	***	†††	‡‡‡	§§§	¶¶¶
bringhurst	*	†	‡	§		¶										
chicago	*	†	‡	§		#										
wiley	*	**	†	‡	§	¶										

Tabelle 3.4: Von footmisc vordefinierte Listen von Fußnotensymbolen

Zusätzlich können in der Präambel mit der `\DefineFNSymbols`-Deklaration eigene Sequenzen definiert werden. Die Deklaration hat zwei obligatorische Argumente: den *namen*, mit dem später per `\setfnsymbol` auf die Liste zugegriffen werden kann, und die *symbol-liste*. Aus dieser Liste werden nacheinander Symbole entnommen (wobei Leerzeichen ignoriert werden). Wenn ein Symbol aus mehr als einem Zeichen besteht, muss es in Klammern eingfasst werden. Wenn die Sternform der Deklaration benutzt wird, gibt $\text{\LaTeX}$ eine Fehlermeldung aus, sobald die Symbole zur Neige gehen. Ohne Stern werden arabische Ziffern benutzt und am Ende des $\text{\LaTeX}$ -Laufs wird eine Warnung ausgegeben.

Aufgrund einer unglücklichen Designentscheidung wurden Fußnotensymbole (genau wie einige andere Textsymbole) ursprünglich in den mathematischen Zeichensätzen von $\text{\TeX}$ untergebracht, anstatt in den Textschriften. Das Ergebnis war, dass sie sich nicht änderten, wenn man die Textschrift änderte. In $\text{\LaTeX}$ wurde dieser Mangel teilweise behoben, indem diese Symbole zur Textsymbolkodierung (TS1) hinzugefügt wurden; siehe Abschnitt 7.5.4. Aus Kompatibilitätsgründen werden Fußnotensymbole jedoch standardmäßig immer noch aus den mathematischen Fonts genommen, auch wenn diese Wahl nicht sehr glücklich ist, wenn die Textschrift von Computer Modern in eine andere Schrift geändert wird. Mithilfe des Wertes `text` im optionalen Argument *typ* kann man `footmisc` mitteilen, dass die Liste aus Textsymbolen besteht. Zu beachten ist, dass alle vordefinierten Symbollisten aus mathematischen Symbolen bestehen und gegebenenfalls neu deklariert werden müssen, wenn sie mit anderen Schriften als Computer Modern verwendet werden sollen.

Etwas Text* mit einer Fußnote. Mehr**
Text. Noch mehr Text.*** Und noch****
mehr Text. Ein letzter Text.

```
\usepackage[symbol]{footmisc}
\DefineFNSymbols{stars}[text]{*{**}{***}{****}}
\setfnsymbol{stars}
```

```
Etwas Text\footnote{Erstens.} mit einer Fußnote.
Mehr\footnote{Zweitens.} Text. Noch mehr
Text.\footnote{Drittens.} Und noch\footnote
{Viertens.} mehr Text. Ein letzter Text.
```

Bsp.
3-2-8

Bei vielen kurzen Fußnoten ist die Standardanordnung untereinander am Fuß der Seite möglicherweise keine wirklich befriedigende Lösung. Ein typisches Beispiel sind kritische Editionen, die viele kurze Fußnoten enthalten.¹

¹Die Beschreibung des Paketes `ledmac` [172] zeigt beispielsweise, welche Art von Fuß- und Endnoten üblicherweise in kritischen Editionen verwendet werden. Dieses Paket ist eine einfachere Version des `EDMAC`-Systems [114] für $\text{\LaTeX}$ und wurde kürzlich von Peter Wilson bereitgestellt. Siehe auch das Paket `bigfoot` von David Kastrup.

Die Anordnung der Fußnoten kann über die Option `para` geändert werden, die dann alle Fußnoten in einen einzigen Absatz platziert. Wenn diese Option gesetzt ist, werden Fußnoten nie auf die nächste Seite umbrochen. Der Code für diese Option basiert auf der Arbeit von Chris Rowley und Dominik Wujastyk (verfügbar als Paket `fnpara`), die wiederum von einem Beispiel im *T_EXbook* von Donald Knuth inspiriert wurde.

Bsp.
3-2-9

Etwas Text mit einer Fußnote.¹ Mehr²
Text. Noch mehr Text.³ Ein letzter Text.

¹ Erstens. ² Zweitens. ³ Drittens.

```
\usepackage[para]{footmisc}
```

```
Etwas Text mit einer Fußnote.\footnote{Erstens.}
Mehr\footnote{Zweitens.} Text. Noch mehr
Text.\footnote{Drittens.} Ein letzter Text.
```

Eine weitere Möglichkeit, Fußnoten zu setzen, bietet die Option `side`. Hier werden Fußnoten in den Rand gesetzt, wenn möglich auf dieselbe Zeile, in der auf sie verwiesen wird. Intern werden dabei spezielle `\marginpar`-Befehle verwendet, um den Fußnotentext zu platzieren. Alles, was in Abschnitt 3.2.8 über `\marginpar`-Befehle gesagt wird, gilt also auch hier. Diese Option kann nicht zusammen mit der bereits beschriebenen Option `para` verwendet werden, ist aber mit den meisten anderen kombinierbar.

Bsp.
3-2-10

Etwas Text mit einer Fußnote.¹ Ganz viel zusätzlicher
Text mit einer Fußnote.² Mehr
Text und eine weitere Fußnote.³
Noch mehr Text.⁴ Viele zusätz-
liche Textzeilen, die den Raum
auf der linken Seite füllen sol-

¹Erstens.
²Zweitens.
³Drittens.
⁴Auch noch
viertens.

```
\usepackage[side,flushmargin]{footmisc}
```

```
Etwas Text mit einer Fußnote.\footnote{Erstens.}
Ganz viel zusätzlicher Text mit einer
Fußnote.\footnote{Zweitens.} Mehr Text
und eine weitere Fußnote.\footnote{Drittens.}
Noch mehr Text.\footnote{Auch noch viertens.}
Viele zusätzliche Textzeilen, die den
Raum auf der linken Seite füllen sollen.
```

Bei der Option `flushmargin`, die im vorangegangenen Beispiel verwendet wurde, beginnt der Fußnotentext ohne Einzug, wobei das Fußnotenzeichen in den Rand hineinragt. Der Fußnotentext wird standardmäßig eingezogen. Es ist offensichtlich, dass sich dieses Verhalten nicht mit der Option `para` verträgt. Eine Alternative ist `marginal`. Bei der Verwendung dieser Option reicht das Zeichen noch weiter in den Rand hinein, wie das folgende Beispiel zeigt.

Bsp.
3-2-11

Etwas Text¹ mit einer Fußnote. Mehr
Text.² Noch mehr Text.³ Ein letzter Text.

¹ Erstens.
² Zweitens.
³ Drittens.

```
\usepackage[marginal]{footmisc}
```

```
Etwas Text\footnote{Erstens.} mit einer
Fußnote. Mehr Text.\footnote{Zweitens.}
Noch mehr Text.\footnote{Drittens.}
Ein letzter Text.
```

Anstatt eine der oben beschriebenen Optionen zu verwenden, kann die Position des Fußnotenzeichens alternativ über den `\footnotemargin`-Parameter gesteuert werden. Bei einem negativen Wert wird es in den Rand gesetzt. Der Wert `Opt` entspricht der Verwendung der Option `flushmargin`. Ein positiver Wert bedeutet, dass der Fußnotentext um diesen Betrag eingezo-

gen wird; das Zeichen wird rechtsbündig in den vom Einzug erzeugten Leer-
raum gesetzt.

Etwas Text¹ mit einer Fußnote. Mehr²
Text. Noch mehr Text.³ Ein letzter Text.

¹Erstens.

²Zweitens.

³Drittens.

```
\usepackage{footmisc}
\setlength\footnotemargin{10pt}
```

Etwas Text\footnote{Erstens.} mit einer
Fußnote. Mehr\footnote{Zweitens.} Text.
Noch mehr Text.\footnote{Drittens.}
Ein letzter Text.

Bsp.
3-2-12

Standardmäßig wird der Fußnotentext mit Randausgleich gesetzt, aber dies führt nicht immer zu zufriedenstellenden Ergebnissen, insbesondere nicht mit den Optionen `para` und `side`. Bei `para` lässt sich das nicht ändern, für andere Layouts kann man mit der Option `ragged` auf Flattersatz umschalten. Das nächste Beispiel verwendet nicht `flushmargin`, deshalb wird hier ein Einzug der Breite `\footnotemargin` erzeugt – vgl. hierzu Beispiel 3-2-10 auf der vorherigen Seite.

Etwas Text mit einer
Fußnote.¹ Ganz viel zusätz-
licher Text, um den Leer-
raum im Beispiel hier zu
füllen. Ganz viel zusätzli-
cher Text, um den Leer-

¹Im Rand
sieht Flattersatz
oft besser aus.

```
\usepackage[side,ragged]{footmisc}
```

Etwas Text mit einer Fußnote.\footnote{Im Rand
sieht Flattersatz oft besser aus.}
Ganz viel zusätzlicher Text, um den Leerraum
im Beispiel hier zu füllen. Ganz viel
zusätzlicher Text, um den Leerraum im Beispiel
hier zu füllen.

Bsp.
3-2-13

Die beiden Optionen `norule` und `splitrule` (Dank an Donald Arseneau) ermöglichen eine Modifizierung der Linie, die zwischen Text und Fußnoten steht. Bei Verwendung von `norule` wird die Trennlinie unterdrückt. Als Ausgleich wird der Wert von `\skip\footins` leicht erhöht. Wenn eine Fußnote nicht auf die aktuelle Seite passt, wird sie geteilt und auf der nächsten Seite fortgesetzt, außer wenn die Option `para` verwendet wird (sie unterstützt keine Teilung von Fußnoten). Standardmäßig ist die Linie, die übernommene Fußnoten der vorherigen Seite und Fußnoten der aktuellen Seite vom Text trennt, ein und dieselbe. Mit der Option `splitrule` lässt sich dies jedoch anpassen: Die Linie über geteilten Fußnoten geht über die gesamte Spalte, während die Linie über den normalen Fußnoten die Standarddefinition von `\footnoterule` behält. Genauer gesagt führt diese Option die Befehle `\mpfootnoterule` (zur Verwendung auf Miniseiten), `\pagefootnoterule` (zur Verwendung auf normalen Seiten) und `\splitfootnoterule` (zur Verwendung auf Seiten, die mit einer geteilten Fußnote beginnen) ein. Durch Modifikation ihrer Definitionen kann man das Layout an die eigenen Bedürfnisse anpassen – ähnlich wie es bereits am Beispiel des Befehls `\footnoterule` gezeigt wurde.

Etwas Text mit einer Fußnote.¹ Mehr²
Text. Noch mehr Text.³ Ein letzter Text.

¹ Erste. ² Eine zweite. ³ Eine dritte.

```
\usepackage[norule,para]{footmisc}
```

Etwas Text mit einer Fußnote.\footnote{Erste.}
Mehr\footnote{Eine zweite.} Text. Noch mehr
Text.\footnote{Eine dritte.} Ein letzter Text.

Bsp.
3-2-14

In Klassen wie `article` oder `report`, in denen `\raggedbottom` wirksam ist (was zur Folge hat, dass Spalten unterschiedlich hoch sein können), wer-

den Fußnoten mit einem Abstand von `\skip\footins` an den Spaltentext angehängt. Alternativ können sie auch am Fuß der Spalte platziert werden, wobei hier zwischen Text und Fußnote zusätzlicher Leerraum eingefügt werden muss; dies geht mit der Option `bottom`. In Klassen wie `book`, in denen `\flushbottom` aktiv ist, bleibt diese Option ohne Wirkung.

In einigen Dokumenten, z.B. literarischen Analysen, kann es vorkommen, dass an einer Stelle mehrere Fußnoten gesetzt werden. Leider sind die Standardfußnotenbefehle von $\text{\LaTeX}$ nicht in der Lage, mit dieser Situation umzugehen; die Fußnotenzeichen werden einfach hintereinander aufgereiht, so dass es nicht mehr unterscheidbar ist, ob es sich um die Fußnoten 1 und 2 oder die Fußnote 12 handelt.

Bsp.
3-2-15

Etwas Text¹² mit zwei Fußnoten.
Noch mehr Text.³

¹ Erste. ² Zweite. ³ Dritte.

`\usepackage[para]{footmisc}`

Etwas Text^{\footnote{Erste.}\footnote{Zweite.}} mit
zwei Fußnoten. Noch mehr Text.^{\footnote{Dritte.}}

Dieses Problem lässt sich mithilfe der Option `multiple` lösen. Sie stellt sicher, dass mehrere aufeinander folgende Fußnotenzeichen durch Kommas getrennt werden. Das Trennzeichen kann auch etwas Anderes sein, wie ein kleiner Leerraum. Die Auswahl des Trennzeichens geschieht über den Befehl `\multfootsep`.

Bsp.
3-2-16

Etwas Text^{1,2} mit zwei Fußnoten.
Noch mehr Text.³

¹ Erste. ² Zweite. ³ Dritte.

`\usepackage[multiple,para]{footmisc}`

Etwas Text^{\footnote{Erste.}\footnote{Zweite.}} mit
zwei Fußnoten. Noch mehr Text.^{\footnote{Dritte.}}

Das Paket `footmisc` beschäftigt sich mit einem anderen potentiellen Problem: Wenn Fußnoten in eine Überschrift gesetzt werden, erscheinen sie möglicherweise im Inhaltsverzeichnis oder dem Kolumnentitel, was ungeahnte Folgen haben kann. Natürlich könnte man dieses Problem (manuell) umgehen, indem man das optionale Argument des Gliederungsbefehls verwendet. Oder man gibt die Option `stable` an: Sie verhindert, dass Fußnoten an solchen Stellen erscheinen.

3.2.5 perpage – Zurücksetzen des Zählers auf Seitenbasis

Wie bereits erwähnt, wird die Möglichkeit, Zähler auf Seitenbasis zurückzusetzen, mit dem kleinen Paket `perpage` von David Kastrup eröffnet.

`\MakePerPage[start]{zähler}`

Mit der Deklaration `\MakePerPage` wird der `zähler` auf jeder Seite zurückgesetzt, wobei optional der Startwert `start` (voreingestellt auf 1) definiert werden kann. Zur Verdeutlichung wird hier noch einmal das Beispiel 3-2-7 von Seite 123 aufgenommen, wobei der Beginn einer Fußnotensequenz diesmal

mit dem zweiten Symbol (d.h. „†“ anstatt „*“) gekennzeichnet ist.

Etwas Text [†] mit einer Fußnote. Mehr [‡]
[†] Erstens. [‡] Zweitens.

Text. Noch mehr Text. [†] Und noch [‡]
[†] Drittens. [‡] Viertens.

```
\usepackage[symbol]{footmisc}
\usepackage{perpage}
\MakePerPage[2]{footnote}
```

Etwas Text\footnote{Erstens.} mit einer
Fußnote. Mehr\footnote{Zweitens.} Text.
Noch mehr Text.\footnote{Drittens.} Und
noch\footnote{Viertens.} mehr Text.

Bsp.
3-2-17

Das Paket synchronisiert die Nummerierung über die .aux-Datei des Dokumentes. Dadurch müssen mindestens zwei Kompilierungsläufe durchgeführt werden, bevor die Nummerierung richtig ist. Außerdem kann es vorkommen, dass im ersten Lauf überflüssige Fehlermeldungen „Counter too large“ ausgegeben werden, wenn zur Nummerierung \fnsymbol oder \alph verwendet wird (siehe die Diskussion der Option symbol* für das footmisc-Paket auf Seite 123).

Von den L^AT_EX-Standardzählern lässt sich wahrscheinlich nur der Zähler footnote sinnvoll auf diese Art manipulieren. Man kann sich aber durchaus Anwendungen vorstellen, welche z.B. nummerierte Marginalien verwenden, die etwa wie folgt definiert sind:

```
\newcounter{mnote}
\newcommand\mnote[1]{\{\refstepcounter{mnote}%
\marginpar[\itshape\small\raggedleft\themnote.\ #1]%
{\itshape\small\raggedright\themnote.\ #1}}\}
\usepackage{perpage} \MakePerPage{mnote}
```

Der neue Zähler mnote wird außerhalb von \marginpar hochgesetzt, so dass er nur einmal ausgeführt wird.¹ Außerdem muss der Gültigkeitsbereich für die aktuelle Neudefinition von \label (durch \refstepcounter) begrenzt werden, d.h. die komplette Definition wird eingeklammert. Anmerkungen auf linken Seiten sollten rechtsbündig sein, was mithilfe des optionalen Argumentes von \marginpar realisiert wird.

1. Eins Etwas Text mit einer Fußnote. ¹ Mehr Text. Noch
¹ Zwei.

mehr Text. Und 1. Drei! noch mehr Text. 2. Vier! Ein letzter Text. ²
² Fünf!

% Code wie oben

Etwas Text\mnote{Eins} mit
einer Fußnote.\footnote{Zwei.}
Mehr Text. Noch mehr
Text.\mnote{Drei!} Und noch
mehr\mnote{Vier!} Text. Ein
letzter Text.\footnote{Fünf!}

Bsp.
3-2-18

Eine weitere Anwendung für dieses Paket zeigt das Beispiel 3-2-24 auf Seite 132, in dem mehrere, voneinander unabhängige Fußnotensequenzen jeweils auf Seitenbasis nummeriert werden.

¹Wenn er in beiden Argumenten von \marginpar verwendet wird, wird er zweimal ausgeführt. Es funktioniert auch, wenn man den Zähler nur im optionalen Argument setzt, dies jedoch liegt an der speziellen Implementierung (bei welcher das optionale Argument zuerst ausgewertet wird), die sich gegebenenfalls ändern kann.

3.2.6 manyfoot – Unabhängige Fußnoten

Die meisten Dokumente haben, wenn überhaupt, nur wenige Fußnoten. Für diese reichen L^AT_EXs Standardbefehle und die Erweiterungen aus, die das Paket `footmisc` bietet. Einige Anwendungen, wie kritische Editionen, benötigen allerdings mehrere, unabhängig nummerierte Fußnotensequenzen. Für diese Fälle ist das Paket `manyfoot` von Alexander Rozhenko sehr hilfreich.¹

```
\DeclareNewFootnote[fußnotenstil]{suffix}[nummerierungsstil]
```

Diese Deklaration kann zur Einführung einer neuen Fußnotenebene verwendet werden. In ihrer einfachsten Form benötigt sie lediglich ein *suffix*, wie „B“. Dadurch wird ein neuer Zähler `footnote{suffix}` reserviert, der zur automatischen Nummerierung der Fußnoten auf dieser Ebene dient. Standardmäßig werden hierfür arabische Ziffern verwendet. Mithilfe des optionalen Argumentes *nummerierungsstil* können andere Zählerstile (z.B. `roman` oder `alph`) ausgewählt werden.

Das optionale Argument *fußnotenstil* definiert den Stil, der im Allgemeinen für Fußnoten der neuen Ebene benutzt wird; der voreingestellte Wert ist `plain`. Wenn das Paket mit der Option `para` bzw. `para*` geladen wurde, steht `para` auch als Fußnotenstil zur Verfügung.

Die Deklaration definiert dann automatisch sechs Befehle. Die ersten drei haben folgende Bedeutung:

`\footnote{suffix}[nummer]{text}` Wie `\footnote`, aber für die neue Ebene. Erhöht den Zähler `footnote{suffix}`, falls das optionale Argument *nummer* nicht angegeben ist. Erzeugt Fußnotenzeichen und setzt den *text* an den Fuß der Seite.

`\footnotemark{suffix}[nummer]` Wie `\footnotemark`, aber für die neue Ebene. Erhöht den entsprechenden Zähler (bei fehlendem optionalem Argument) und druckt das seinem Wert entsprechende Fußnotenzeichen.

`\footnotetext{suffix}[nummer]{text}` Wie `\footnotetext`, aber für die neue Ebene. Setzt den *text* an den Fuß der Seite, wobei er zum Erzeugen des Fußnotenzeichens den aktuellen Wert von `footnote{suffix}` oder des optionalen Argumentes verwendet.

In allen drei Fällen hängt der Stil des Zeichens von dem gewählten *nummerierungsstil* ab.

Die übrigen drei Befehle, die durch eine `\DeclareNewFootnote`-Deklaration zur Verwendung im Dokument definiert werden, heißen `\Footnote{suffix}`, `\Footnotemark{suffix}` und `\Footnotetext{suffix}` (d.h. dieselben Namen wie oben, aber mit großem Anfangsbuchstaben F). Der wichtige Unterschied zu den vorherigen Befehlen ist folgender: Anstelle des optionalen Argumentes *nummer* erwarten sie ein obligatorisches *marker*-Argument, das die Verwendung beliebiger Fußnotenzeichen ermöglicht. Einige Beispiele sind unten aufgeführt.

Das Layout der Fußnoten kann verändert werden, indem man zusätzlich zu `manyfoot` das Paket `footmisc` lädt. Die Option `para` von `footmisc` kann jedoch nicht verwendet werden. Im nächsten Beispiel wird das Standardfußno-

¹Ein umfassenderes Paket, `bigfoot`, wird gerade von David Kastrup entwickelt.

tenlayout für Fußnoten der höchsten Ebene verwendet und das eingezogene Layout (Option para) für die zweite Ebene. Wenn alle Fußnotenebenen eingezogen werden sollen, müssen Fußnoten höchster Ebene (z.B. \footnote) vermieden werden. Alle benötigten Ebenen müssen dann über manyfoot definiert werden. Man beachte, dass footmiscs Option multiple alle Fußnoten richtig behandelt.

Etwas Text^{1,a} mit Fußnoten. Noch mehr Text.^b Etwas Text^{2,*} mit Fußnoten. Noch mehr Text.^c

¹Eine erste.

²Noch eine Hauptfußnote.

^aEbene 2. ^bEine zweite. ^{*Ein}manuel-
les Fußnotenzeichen. ^cNoch eine Fußnote zweiter
Ebene.

```
\usepackage[multiple]{footmisc}
\usepackage[para]{manyfoot}
\DeclareNewFootnote[para]{B}{[alph]}

Etwas Text\footnote{Eine erste.}%
\footnoteB{Ebene 2.}
mit Fußnoten. Noch mehr
Text.\footnoteB{Eine zweite.} Etwas
Text\footnote{Noch eine Hauptfußnote.}%
\FootnoteB{*}{Ein manuelles
Fußnotenzeichen.}

mit Fußnoten. Noch mehr
Text.\footnoteB{Noch eine Fußnote
zweiter Ebene.}
```

Bsp.
3-2-19

Im folgenden Beispiel werden die Fußnoten erster Ebene in den Rand verschoben, indem footmisc mit anderen Optionen geladen wird. Dieses Mal wird manyfoot mit der Option para* geladen. Sie unterscheidet sich von der zuvor verwendeten Option para dadurch, dass sie Einzüge für den eingebetteten Fußnotenblock unterdrückt. Außerdem werden Fußnoten der zweiten Ebene jetzt mit römischen Ziffern nummeriert. Zum Vergleich wird im nächsten Beispiel derselbe Text wie im Beispiel 3-2-19 ausgegeben, es wird jedoch ein anderer Satzspiegel verwendet, damit die Marginalien angezeigt werden können.

¹Eine erste.

²Noch eine
Hauptfuß-
note.

Etwas Text^{1,i} mit Fußno-
ten. Noch mehr Text.ⁱⁱ Etwas
Text^{2,*} mit Fußnoten. Noch
mehr Text.ⁱⁱⁱ

ⁱEbene 2. ⁱⁱEine zweite. ^{*Ein}manuel-
les Fußnotenzeichen. ⁱⁱⁱNoch eine Fuß-
note zweiter Ebene.

```
\usepackage[side,flushmargin,ragged,multiple]
{footmisc}
\usepackage[para*]{manyfoot}
\DeclareNewFootnote[para]{B}{[roman]}

Etwas Text\footnote{Eine erste.}%
\footnoteB{Ebene 2.}
mit Fußnoten. Noch mehr
Text.\footnoteB{Eine zweite.} Etwas
Text\footnote{Noch eine Hauptfußnote.}%
\FootnoteB{*}{Ein manuelles
Fußnotenzeichen.}

mit Fußnoten. Noch mehr
Text.\footnoteB{Noch eine Fußnote
zweiter Ebene.}
```

Bsp.
3-2-20

Bei eingebetteten Fußnoten, die mit der Option para bzw. para* erzeugt werden, tritt wahrscheinlich folgender Fehler auf: Sehr lange Fußnoten am Ende der Seite, die aufgrund ihrer Länge umbrochen werden müssten, werden nicht geteilt. Zur Lösung dieses Problems bietet das Paket manyfoot die (halb) manuelle Möglichkeit, an der Stelle, an der die Fußnote umbrochen werden soll, den Befehl \SplitNote einzufügen und die Fußnote zu beenden.

Der restliche Text der Fußnote wird dann in einem späteren Absatz des Dokumentes über ein leeres *marker*-Argument in einen `\Footnotetext{suffix}` platziert.

Bsp.
3-2-21

Etwasⁱ Text mit zwei Fußnoten.ⁱ Mehr Text.ⁱⁱ Noch mehr ⁱEine erste. ⁱEine zweite. ⁱⁱDies ist eine sehr lange Fußnote,	Text. Etwas Text hier und² noch mehr dort. ²Noch eine erste. die hier fortgesetzt wird.

```
\usepackage[para]{manyfoot}
\DeclareNewFootnote[para]{B}[roman]

Etwas\footnote{Eine erste.} Text mit zwei
Fußnoten.\footnoteB{Eine zweite.} Mehr
Text.\footnoteB{Dies ist eine sehr
lange Fußnote,\SplitNote} Noch mehr Text.

Etwas\FootnotetextB{}{die hier
                                fortgesetzt wird.}
Text hier und\footnote{Noch eine erste.}
noch mehr dort.
```

Wenn beide Teile der Fußnote nach einer erneuten Formatierung des Dokumentes auf derselben Seite stehen, werden sie richtig zusammengesetzt, wie das nächste Beispiel zeigt. Dieses verwendet denselben Beispielttext, jedoch einen anderen Satzspiegel. Wenn es jedoch nach einer erneuten Formatierung erforderlich ist, die Fußnote an einer anderen Stelle zu umbrechen, dann ist weiteres manuelles Eingreifen unvermeidlich. Diese Arbeit sollte daher erst zum Schluss durchgeführt werden.

Bsp.
3-2-22

Etwasⁱ Text mit zwei Fußnoten.ⁱ Mehr Text.ⁱⁱ Noch mehr Text. Etwas Text hier und² noch mehr dort. ¹Eine erste. ²Noch eine erste. ⁱEine zweite. ⁱⁱDies ist eine sehr lange Fußnote, die hier fortgesetzt wird.	<pre>\usepackage[para]{manyfoot} \DeclareNewFootnote[para]{B}[roman] Etwas\footnote{Eine erste.} Text mit zwei Fußnoten.\footnoteB{Eine zweite.} Mehr Text.\footnoteB{Dies ist eine sehr lange Fußnote,\SplitNote} Noch mehr Text. Etwas\FootnotetextB{}{die hier fortgesetzt wird.} Text hier und\footnote{Noch eine erste.} noch mehr dort.</pre>

Der vertikale Abstand zwischen zwei untereinander stehenden Fußnotenblöcken wird durch `\skip\footins{suffix}` festgelegt. Standardmäßig entspricht er `\skip\footins` (d.h. dem Abstand zwischen dem Haupttext und den Fußnoten). Ohne nähere Spezifikation werden die Fußnotenblöcke nur durch solche Leerräume voneinander getrennt, aber mit der Option `ruled` wird auch hier `\footnoterule` angewendet. Genau genommen kann beliebiges Material an dieser Stelle platziert werden, indem man den Befehl `\extrafootnoterule` neu definiert – die einzige Beschränkung ist, dass das formatierte Ergebnis dieses Befehls keinen zusätzlichen vertikalen Leerraum beanspruchen darf (Einzelheiten hierzu in der Erörterung von `\footnoterule` auf Seite 119). Es ist sogar möglich, für verschiedene Fußnotenblöcke unterschiedliche Linien zu definieren (die Einzelheiten hierzu findet man in der Paketdokumentation).

Etwas^{1,*} Text mit einer Fußnote.
 Noch^A mehr Text. Etwas Text[†] mit
 einer Fußnote.^B Noch mehr Text für
 das Beispiel.

¹ Eine erste.

* Eine zweite.

[†] Ein Beispiel.

^A Eine dritte.

^B Ein weiteres Beispiel.

```
\usepackage[marginal,multiple]{footmisc}
\usepackage[ruled]{manyfoot}
\DeclareNewFootnote{B}[fnsymbol]
\DeclareNewFootnote{C}[Alph]
\setlength{\skip\footinsB}{5pt minus 1pt}
\setlength{\skip\footinsC}{5pt minus 1pt}

Etwas\footnote{Eine erste.}\footnoteB{Eine zweite.}
Text mit einer Fußnote. Noch\footnoteC{Eine dritte.}
mehr Text. Etwas Text\footnoteB{Ein Beispiel.} mit
einer Fußnote.\footnoteC{Ein weiteres Beispiel.}
Noch mehr Text für das Beispiel.
```

Bsp.
3-2-23

Seitenweise
 Nummerierung der
 Fußnoten

Das vorige Beispiel verwendete zwei zusätzliche *numerierungsstile*, Alph und fnsymbol. Da in beiden Stilen nur wenige Fußnotensymbole verfügbar sind, ist diese Wahl meist nicht empfehlenswert, es sei denn, dass diese Fußnotensequenzen seitenweise nummeriert werden. Die Option `perpage` von `footmisc` hilft in diesem Fall nicht, da sie nur auf Fußnoten der obersten Ebene anwendbar ist. Die gewünschte Wirkung lässt sich erzielen, indem man entweder für die Zähler `footnoteB` und `footnoteC` `\MakePerPage` aus dem Paket `perpage` verwendet (wie unten) oder die Option `perpage` von `manyfoot` einsetzt (welches dazu das Paket `perpage` aufruft. Dadurch werden alle neu definierten Fußnotenebenen seitenweise nummeriert.). Zu beachten ist, dass die Fußnoten der höchsten Ebene, wie im Beispiel, immer noch sequentiell nummeriert werden.

Etwas Text¹ mit
 einer Fußnote. Noch
 mehr^{*,A} Text. Etwas
 Text mit^B einer Fuß-

¹Eine erste.

*Zweitens.

^ADrittens.

^BEin Beispiel.

note hier.^A Mehr Text.
 Und^{2,*} eine letzte Fuß-
 note.

²Wieder.

*Eine letzte.

^AEin weiteres Bei-
 spiel.

```
\usepackage[multiple]{footmisc}
\usepackage{manyfoot,perpage}
\DeclareNewFootnote{B}[fnsymbol]
\DeclareNewFootnote{C}[Alph]
\MakePerPage{footnoteB}\MakePerPage{footnoteC}

Etwas Text\footnote{Eine erste.} mit einer
Fußnote. Noch mehr\footnoteB{Zweitens.}%
\footnoteC{Drittens.} Text. Etwas Text
mit\footnoteC{Ein Beispiel.} einer Fußnote
hier.\footnoteC{Ein weiteres Beispiel.} Mehr
Text. Und\footnote{Wieder.}\footnoteB{Eine
letzte.} eine letzte Fußnote.
```

Bsp.
3-2-24

3.2.7 endnotes – Eine Alternative zu Fußnoten

In geisteswissenschaftlichen Arbeiten werden Anmerkungen üblicherweise am Ende jedes Kapitels oder am Ende des Dokumentes zu sogenannten Endnoten zusammengefasst. Standard- $\text{\LaTeX}$ unterstützt zunächst keine Endnoten. Sie können aber auf verschiedene Weise erzeugt werden.

Das Paket `endnotes` (von John Lavagnino) enthält einen eigenen `\endnote`-Befehl, der den gleichzeitigen Gebrauch von Fußnoten und Endnoten ermöglicht.

Die Syntax auf Dokumentenebene ist den Fußnotenbefehlen nachgebildet, wenn `foot` durch `end` ersetzt wird – so erzeugt `\endnote` beispielsweise

eine Endnote, `\endnotemark` nur das Endnotenzeichen und `\endnotetext` nur den Text. Der Zähler, der die aktuelle Endnotennummer wiedergibt, heißt `endnote`. Er wird hochgezählt, wenn `\endnote` oder `\endnotemark` ohne optionales Argument verwendet wird.

Alle Endnoten werden in einer externen Datei mit der Erweiterung `.ent` gespeichert. Sie werden bereitgestellt, sobald der Befehl `\theendnotes` ausgegeben wird.

Dies ist¹ ein einfacher Text. Dies ist ein einfacher Text.² Noch mehr Text mit einem Endnotenzeichen.¹

Notes

Bsp.
3-2-25

¹Die erste Endnote.

²Die zweite Endnote.

```
\usepackage{endnotes}
```

```
Dies ist\endnote{Die erste Endnote.} ein  
einfacher Text. Dies ist ein einfacher  
Text.\endnote{Die zweite Endnote.} Noch mehr  
Text mit einem Endnotenzeichen.\endnotemark[1]
```

```
\theendnotes % Endnoten hier ausgeben
```

Dieser Prozess unterscheidet sich von der Art und Weise, wie Inhaltsverzeichnisse generiert werden (siehe Abschnitt 2.3). Die Endnoten werden direkt in die Datei geschrieben, so dass nur solche Endnoten sichtbar sind, die früher im Dokument definiert wurden. Der Vorteil dieses Ansatzes ist, dass `\theendnotes` mehrfach aufgerufen werden kann, z.B. am Ende jeden Kapitels. Um zusätzlich die Nummerierung neu zu beginnen, muss der Zähler `endnote` nach dem Aufruf von `\theendnotes` auf null gesetzt werden.

Es gibt verschiedene Möglichkeiten, die von `\theendnotes` erzeugte Überschrift zu steuern. Der Text kann durch Anpassung von `\notesname` geändert werden (voreingestellt ist die Zeichenkette `Notes`). Wenn dies nicht ausreicht, kann auch noch `\noteheading` neu definiert werden, der den Gliederungsbefehl vor der Endnote angibt.

Das Layout für die Endnotennummern wird über `\theendnote` gesteuert, der üblichen Art, die Formatierung von Zählern festzulegen. Das Format des Endnotenzeichens wird von `\makeenmark` mithilfe von `\theenmark` erzeugt, letzteres Makro enthält die formatierte Nummer für das aktuelle Endnotenzeichen.

Dies ist ein einfacher Text.^{a)} Dies ist ein einfacher Text.^{b)} Noch mehr Text mit einem Endnotenzeichen.^{a)}

Kapitelanmerkungen

Bsp.
3-2-26

^{a)}Die erste Endnote.

^{b)}Die zweite Endnote.

```
\usepackage{endnotes}
```

```
\renewcommand\theendnote{\alph{endnote}}  
\renewcommand\makeenmark{\textsuperscript{\theenmark}}  
\renewcommand\notesname {Kapitelanmerkungen}
```

```
Dies ist ein einfacher Text.\endnote{Die erste Endnote.}  
Dies ist ein einfacher Text.\endnote{Die zweite Endnote.}  
Noch mehr Text mit einem Endnotenzeichen.\endnotemark[1]  
\theendnotes
```

Die Schriftgröße für die Liste der Endnoten wird über den Befehl `\notesize` gesteuert, der auf `\footnotesize` voreingestellt ist. Durch Veränderung von `\noteformat` kann man die Darstellung der einzelnen Endnoten in ihrer Liste verändern. Dieser Befehl dient dazu, die Absatzparameter

für die Endnoten einzustellen und die in `\theenmark` gespeicherte Endnotennummer zu setzen. Im nächsten Beispiel erhält der erste Absatz keinen Einzug, und die Nummer wird in den Seitenrand geschrieben.

Dies ist ein einfacher Text. ¹	<code>\usepackage{endnotes}</code>
Dies ist ein einfacher Text. ²	<code>\renewcommand{\enoteformat}{\noindent\raggedright</code>
Noch mehr Text mit einem Endnotenzeichen. ¹	<code>\setlength{\parindent}{12pt}\makebox[0pt][r]{\theenmark.},}}</code>
	<code>\renewcommand{\notesize}{\scriptsize}</code>
	Dies ist ein einfacher Text.\endnote{Die erste Endnote hat sehr viel Text, damit zwei Zeilen erzeugt werden.\par Und sogar noch ein zweiter Absatz.}
	Dies ist ein einfacher Text.\endnote{Die zweite Endnote.}
	Noch mehr Text mit einem Endnotenzeichen.\endnotemark[1]
	<code>\theendnotes</code>

Bsp.
3-2-27

Notes

1. Die erste Endnote hat sehr viel Text, damit zwei Zeilen erzeugt werden.
Und sogar noch ein zweiter Absatz.
2. Die zweite Endnote.

3.2.8 Marginalien

Der $\text{\LaTeX}$ -Befehl `\marginpar` erzeugt eine Marginalie. Er setzt den Text, den er als Argument enthält, in den Seitenrand, wobei die erste Zeile in der gleichen Höhe erscheint wie der `\marginpar`-Befehl im Haupttext. Wenn nur das obligatorische Argument angegeben ist, erscheint der Text bei einseitigem Druck im rechten Rand; bei zweiseitigem Druck erscheint er im äußeren Rand und bei zweispaltigem Druck erscheint er im am nächsten gelegenen Rand. Ist ein optionales Argument angegeben, wird dieses für den linken Rand verwendet, während das obligatorische Argument dann für den rechten Rand gilt.

Diese Voreinstellung lässt sich mit `\reversemarginpar` umkehren (außer bei zweispaltiger Formatierung). Dieser Befehl wirkt sich auf alle nachfolgenden Marginalien aus. Zurück zur Voreinstellung geht es wieder mit `\normalmarginpar`.

Beim Arbeiten mit Marginalien sind einige wichtige Punkte zu berücksichtigen. Zunächst beginnt der Befehl `\marginpar` keinen neuen Absatz. Wenn er also vor dem ersten Wort eines Absatzes erscheint, stimmt die vertikale Ausrichtung nicht mit dem Beginn des Absatzes überein. Außerdem wird das erste Wort im Argument nicht automatisch getrennt. Wenn die Ränder schmal und die Wörter lang sind (wie in der deutschen Sprache), kann es daher erforderlich sein, dem ersten Wort den Befehl `\hspace{0pt}` voranzustellen, damit es getrennt werden kann. Diese beiden möglichen Schwierigkeiten kann man umgehen, indem man einen Befehl wie `\marginlabel` definiert, der mit einer leeren Box `\mbox{}` beginnt, Marginalien linksbündig formatiert und dem Argument den Befehl `\hspace{0pt}` voranstellt.

Etwas Text mit einer Marginalie. Ein weiterer Text mit einer Marginalie. Noch mehr Text. Eine Menge zusätzlicher Text hier, der im Beispiel die Kolumne füllen soll.	EinSuperlangesErstesWort mit Problemen EinSuperlangesErstesWort ohne Probleme	<code>\newcommand{\marginlabel}[1]{\mbox{}}\marginpar{\raggedright\hspace{0pt}\#1}</code> Etwas\marginpar{EinSuperlangesErstesWort mit Problemen} Text mit einer Marginalie. Ein weiterer\marginlabel{EinSuperlangesErstesWort ohne Probleme} Text mit einer Marginalie. Noch mehr Text. Eine Menge zusätzlicher Text hier, der im Beispiel die Kolumne füllen soll.

Bsp.
3-2-28

Natürlich kann die obige Definition keine unterschiedlichen Texte mehr erzeugen, die je nach Randeinstellung variieren. Mit einem kleinen Trick kann dieses Problem gelöst werden, indem man beispielsweise `\ifthenelse`-Konstrukte aus dem Paket `ifthen` verwendet.

Der $\text{\LaTeX}$ -Kernel bemüht sich (ohne bei der Verarbeitung einen zu großen Overhead zu erzeugen) sicherzustellen, dass der Inhalt des `\marginpar`-Befehls stets im richtigen Rand erscheint, und in den meisten Fällen gelingt ihm das auch. Manchmal geht es allerdings schief. Wenn solch ein Missgeschick auftaucht, ist eine Sofortlösung, einen expliziten `\pagebreak` hinzuzufügen, um eine zu weit vorausschauende Seitenerstellung zu verhindern. Natürlich hat dies den Nachteil, dass diese rein optische Korrektur bei späteren Änderungen im Dokument rückgängig gemacht werden muss. Besser ist, das Paket `mparhack` von Tom Sgouros und Stefan Ulrich zu verwenden. Sobald das Paket geladen ist, werden alle `\marginpar`-Positionen beobachtet (wobei intern ein Markierungsverfahren verwendet wird und die Daten in die `.aux`-Datei geschrieben werden). In diesem Fall wird die Warnung „Marginpars may have changed. Rerun to get them right“ ausgegeben. Sie weist darauf hin, dass die Positionen im Vergleich zum vorherigen $\text{\LaTeX}$ -Kompilierungslauf geändert wurden und ein weiterer Lauf erforderlich ist, um das Dokument zu stabilisieren.

*Falsch platzierte
`\marginpars`*

Wie in Tabelle 4.2 auf Seite 205 erläutert, wird der Stil von Marginalien durch drei Parameter festgelegt: `\marginparwidth`, `\marginparsep` und `\marginparpush`.

3.3 Listen

Listen sind ein sehr wichtiges $\text{\LaTeX}$ -Konstrukt. Sie werden zum Bilden vieler der absatzerzeugenden Umgebungen von $\text{\LaTeX}$ verwendet. Die drei Standardlistenumgebungen von $\text{\LaTeX}$ werden im Abschnitt 3.3.1 beschrieben, in dem auch gezeigt wird, wie sie angepasst werden können. Abschnitt 3.3.2, der auf Seite 139 beginnt, stellt das Paket `paralist` im Detail vor, das eine Reihe neuer Listenstrukturen enthält und umfangreiche Methoden zur Anpassung dieser neuen Listen sowie der Standardlisten bietet. Anschließend folgt eine Erörterung von Listen mit Überschriften, wie Theoreme und Übungen. Zum Schluss wird $\text{\LaTeX}$ s generische Listenumgebung in Abschnitt 3.3.4 auf Seite 152 beschrieben.

3.3.1 Ändern der Standardlisten

Es ist relativ einfach, die drei Standardlistenumgebungen von $\text{\LaTeX}$ `itemize`, `enumerate` und `description` eigenen Erfordernissen anzupassen. Die drei folgenden Abschnitte befassen sich nacheinander mit diesen Umgebungen. Änderungen an den Standardeinstellungen lassen sich global in der Präambel vornehmen, indem man dort bestimmte Listenparameter neu definiert. Sie können aber auch lokal durchgeführt werden.

Anpassen der `itemize`-Listenumgebung

Die Label für eine einfache `itemize`-Liste ohne Nummerierung werden durch die Befehle in Tabelle 3.5 auf der nächsten Seite definiert. Eine Liste mit anders

	Befehl	Voreinstellung	Darstellung
Erste Ebene	<code>\labelitemi</code>	<code>\textbullet</code>	•
Zweite Ebene	<code>\labelitemii</code>	<code>\normalfont\bfseries \textendash</code>	–
Dritte Ebene	<code>\labelitemiii</code>	<code>\textasteriskcentered</code>	*
Vierte Ebene	<code>\labelitemiv</code>	<code>\textperiodcentered</code>	.

Tabelle 3.5: Befehle zum Steuern der Listenumgebung `itemize`

aussehenden Labels erhält man, indem man die labelerzeugenden Befehle neu definiert. Die Änderungen können, wie im nächsten Beispiel, lokal für eine Liste vorgenommen oder global bereitgestellt werden. Dafür wird die neue Definition in die Präambel eingefügt. Bei der folgenden einfachen Liste handelt es sich um eine `itemize`-Standardliste mit einem Symbol aus dem PostScript-Zeichensatz Zapf Dingbats (siehe Abschnitt 7.6.4 auf Seite 390) für das Label der obersten Ebene:

	<code>\usepackage{pifont}</code>
	<code>\newenvironment{MYitemize}{\renewcommand\labelitemi</code> <code>{\ding{43}}\begin{itemize}}{\end{itemize}}</code>
☞ Text des ersten Punktes in der Liste.	<code>\begin{MYitemize}</code>
	<code>\item Text des ersten Punktes in der Liste.</code>
☞ Text des ersten Satzes für den zweiten Punkt der Liste. Und der zweite Satz.	<code>\item Text des ersten Satzes für den zweiten</code> <code>Punkt der Liste. Und der zweite Satz.</code>
	<code>\end{MYitemize}</code>

Bsp.
3-3-1

Anpassen der `enumerate`-Listenumgebung

Die $\text{\LaTeX}$ -Umgebung `enumerate` für nummerierte Listen wird durch die Befehle und Darstellungsformen in Tabelle 3.6 auf der nächsten Seite charakterisiert. Die erste Zeile enthält die Namen der Zähler, die zum Nummerieren der vier möglichen Listenebenen verwendet werden. Die zweite und dritte Zeile enthalten die Befehle, welche die Darstellung der Zähler festlegen, und ihre voreingestellten Definitionen in den $\text{\LaTeX}$ -Standardklassen. Die Zeilen vier, fünf und sechs enthalten den Befehl, die Standarddefinition und ein Beispiel für die tatsächliche Nummerierung, wie sie von der Liste ausgegeben wird.

Ein Verweis auf ein nummeriertes Listenelement wird mithilfe von `\theenumi`, `\theenumii` und weiteren, ähnlichen Befehlen gebildet. Ihnen werden entsprechend die Befehle `\p@enumi`, `\p@enumii` usw. als Präfix vorangestellt. Die letzten drei Zeilen der Tabelle 3.6 auf der gegenüberliegenden Seite zeigen den Befehl und die Standarddefinition für Verweise sowie ein Beispiel ihrer Darstellung. Für korrekte Verweise müssen sowohl die Befehlsdefinitionen für die Darstellung als auch diejenigen zum Bilden der Verweise berücksichtigt werden.

Mithilfe dieser Informationen lassen sich nun leicht eigene nummerierte Listen erstellen.

Das erste Beispiel definiert die Zähler der ersten und zweiten Ebene neu, so dass diese große römische Zahlen und lateinische Buchstaben verwenden.

	Erste Ebene	Zweite Ebene	Dritte Ebene	Vierte Ebene
<i>Zähler</i>	<code>enumi</code>	<code>enumii</code>	<code>enumiii</code>	<code>enumiv</code>
<i>Darstellung</i>	<code>\theenumi</code>	<code>\theenumii</code>	<code>\theenumiii</code>	<code>\theenumiv</code>
<i>Voreinstellung</i>	<code>\arabic{enumi}</code>	<code>\alph{enumii}</code>	<code>\roman{enumiii}</code>	<code>\Alph{enumiv}</code>
<i>Labelfeld</i>	<code>\labelenumi</code>	<code>\labelenumii</code>	<code>\labelenumiii</code>	<code>\labelenumiv</code>
<i>Voreinstellung</i>	<code>\theenumi.</code>	<code>(\theenumii)</code>	<code>\theenumiii.</code>	<code>\theenumiv.</code>
<i>Beispiel</i>	1., 2.	(a), (b)	i., ii.	A., B.
<i>Darstellung von Verweisen</i>				
<i>Präfix</i>	<code>\p@enumi</code>	<code>\p@enumii</code>	<code>\p@enumiii</code>	<code>\p@enumiv</code>
<i>Voreinstellung</i>	<code>{}</code>	<code>\theenumi</code>	<code>\theenumi(\theenumii)</code>	<code>\p@enumiii\theenumiii</code>
<i>Verweisbeispiel</i>	1, 2	1a, 2b	1(a)i, 2(b)ii	1(a)IA, 2(b)iiB

Tabelle 3.6: Befehle zum Steuern der Listenumgebung `enumerate`

Die visuelle Darstellung soll aus dem Wert des Zählers gebildet werden, gefolgt von einem Punkt. Damit kann für `\labelenumi` der voreingestellte Wert aus Tabelle 3.6 verwendet werden.

I. Einleitung

A. Anwendungen

Motivation zu Forschung und Anwendung im Zusammenhang mit dem Thema.

B. Aufbau

Erklärung zum Aufbau des Berichtes, was dazugehört und was nicht.

II. Literaturübersicht

q1=I q2=IA q3=IB q4=II

```

\renewcommand\theenumi {\Roman{enumi}}
\renewcommand\theenumii {\Alph{enumii}}
\renewcommand\labelenumii{\theenumii.}

\begin{enumerate}
\item \textbf{Einleitung} \label{q1}
\begin{enumerate}
\item \textbf{Anwendungen} \\\
Motivation zu Forschung und Anwendung
im Zusammenhang mit dem Thema. \label{q2}
\item \textbf{Aufbau} \\\
Erklärung zum Aufbau des Berichtes, was
dazugehört und was nicht. \label{q3}
\end{enumerate}
\item \textbf{Literaturübersicht} \label{q4}
\end{enumerate}
q1=\ref{q1} q2=\ref{q2} q3=\ref{q3} q4=\ref{q4}

```

Bsp.
3-3-2

Nach diesen Neudefinitionen sehen die Verweise etwas seltsam aus. Um dies zu korrigieren, muss die Definition des Präfixbefehls `\p@enumii` geändert werden. Um beispielsweise den Verweis „I-A“ anstelle von „IA“ (wie im vorherigen Beispiel) zu erhalten, benötigt man folgende Definition:

```
\makeatletter \renewcommand\p@enumii{\theenumi--} \makeatother
```

denn der Verweis wird formatiert, indem `\p@enumii`, gefolgt von `\theenumii` ausgeführt wird. Zu beachten ist, dass hier `\makeatletter` und `\makeatother` angegeben werden müssen, weil der Name des neu zu definierenden Befehls ein `@`-Zeichen enthält. Anstelle dieses Low-Level-Verfahrens kann auch der Befehl `\labelformat` aus dem Paket `varioref` verwendet werden, das in Abschnitt 2.4.2 beschrieben wird.

Das `enumerate`-Feld lässt sich ausschmücken, indem man das Labelfeld erweitert. Im nächsten Beispiel wird als Präfix für die Label aller Listenelemente der ersten Ebene das Paragraphenzeichen (§) und als Suffix ein Punkt verwendet (der in den Verweisen weggelassen wird).

```

\renewcommand\labelenumi{\S\theenumi.}
\usepackage{varioref} \labelformat{enumi}{\S#1}

§1. Listentext, weiterer Listentext
§2. Listentext, weiterer Listentext

w1=§1 w2=§2

```

Bsp.
3-3-3

Möglicherweise sollen auch verschiedene Zeichen für aufeinander folgende Label verwendet werden. So nutzt das folgende Beispiel Zeichen des PostScript-Zeichensatzes Zapf Dingbats. In diesem Fall gibt es keinen einfachen Weg, die korrekten Querverweise automatisch über `\ref`-Befehle zu erzeugen. Anstatt `\theenumi` für die Darstellung des `enumi`-Zählers zu verwenden, kann man den Befehl so definieren, dass er aus dem Zählerwert das zu verwendende Symbol berechnet. Die Schwierigkeit besteht hierbei darin, diese Definition so zu gestalten, dass sie den Labelgenerierungsprozess überlebt. Hier hilft ein Trick: Man fügt `\protect`-Befehle hinzu, so dass `\setcounter` und `\ding` nicht ausgeführt werden, wenn das Label in die `.aux`-Datei geschrieben wird. Trotzdem muss sichergestellt sein, dass der aktuelle Wert dort abgelegt wurde. Dies erreicht man, indem man innerhalb von `\setcounter` dem Befehl `\value` den (internen) `TEX`-Befehl `\the` voranstellt (jedoch nicht innerhalb von `\ding`!); ohne dies würden die Verweise alle denselben Wert haben.¹

```

\usepackage{calc,pifont} \newcounter{local}
\renewcommand\theenumi{\protect\setcounter{local}%
{171+\the\value{enumi}}\protect\ding{\value{local}}}%
\renewcommand\labelenumi{\theenumi}

① Listentext, Listentext, Listentext,
Listentext, weiterer Listentext;
② Listentext, Listentext,
Listentext, Listentext, weiterer Listentext;
③ Listentext, Listentext.

l1=① l2=② l3=③

```

Bsp.
3-3-4

Dieselbe Wirkung erzielt man mit der Umgebung `dingautolist`, die im Paket `pifont` definiert ist, einem Teil des PSNFSS-Systems (beschrieben in Abschnitt 7.6.4 auf Seite 390).

¹Für `TEX`nisch Interessierte: `\the` erzeugt, trotz seines Namens, nicht den „Wert“ eines `TEX`s Zählers, sondern nur seinen internen `TEX`-Registernamen. In den meisten Situationen kann dieser als Wert verwendet werden, aber leider nicht innerhalb von `\edef` oder `\write`, bei denen nicht der „Wert“, sondern der interne Name überlebt. Durch Vorstellen des Befehls `\the` vor den internen Registernamen bekommt man auch in diesen Fällen den „Wert“.

Anpassen der Listenumgebung `description`

Zur Anpassung der `description`-Umgebung kann man den Parameter `\descriptionlabel` verändern, der die Label erzeugt. Im folgenden Beispiel wird der Zeichensatz für die Label von fett (Standardeinstellung) in serifenlos geändert.

```

\renewcommand\descriptionlabel[1]{%
    {\hspace{\labelsep}\textsf{#1}}
\begin{description}
A. Listentext, Listentext, Listentext,
    weiterer Listentext;
\item[A.] Listentext, Listentext,
    Listentext, weiterer Listentext;
B. Listentext, Listentext, Listentext,
    weiterer Listentext;
\item[B.] Listentext, Listentext,
    Listentext, weiterer Listentext;
\end{description}

```

Bsp.
3-3-5

Die $\text{\LaTeX}$ -Standardklassen setzen den Anfangspunkt für Labelboxen in einer `description`-Umgebung vor dem linken Rand der umschließenden Umgebung, und zwar mit dem Abstand `\labelsep`. In dem oben aufgeführten Beispiel fügt `\descriptionlabel` daher zunächst den Wert `\labelsep` wieder hinzu, damit das Label linksbündig am Seitenrand beginnt (siehe Seite 156 für eine ausführliche Beschreibung).

3.3.2 `paralist` – Erweiterte Listenumgebungen

Das Paket `paralist` von Bernd Schandl enthält eine Reihe neuer Listenumgebungen und bietet Erweiterungen für $\text{\LaTeX}$ s Standardlistenumgebungen, was deren Anpassung stark vereinfacht. Standardumgebungen und neue Listenumgebungen können ineinander verschachtelt werden und die Umgebungen für nummerierte Listen unterstützen $\text{\LaTeX}$ s Querverweismechanismus (`\label`, `\ref`).

Nummerierte Listen

Alle $\text{\LaTeX}$ -Standardlisten sind abgesetzte Listen, d.h. sie fügen oberhalb und unterhalb der Liste sowie zwischen den einzelnen Listenelementen einen Abstand ein. Manchmal sollen die Punkte in einem Absatz ohne diese optische Unterbrechung nummeriert werden. Zu diesem Zweck wurde die Umgebung `inparaenum` entwickelt. Sie hat ein optionales Argument, das zum Generieren der Labels verwendet werden kann; auf die genaue Syntax wird später noch eingegangen.

```

\usepackage{paralist}
Vielleicht sollen die Punkte in einem
Absatz nummeriert werden, um
\begin{inparaenum}[(a)]
    \item Platz zu sparen
    \item eine nicht so bedeutende Aussage
        zu treffen oder
    \item aus einem anderen Grund.
\end{inparaenum}

```

Vielleicht sollen die Punkte in einem Absatz nummeriert werden, um
(a) Platz zu sparen (b) eine nicht so bedeutende Aussage zu treffen oder
(c) aus einem anderen Grund.

Bsp.
3-3-6

Das ist möglicherweise aber nicht exakt das, was gewünscht war. Viele Leute bevorzugen abgesetzte Listen, wollen aber nicht so viel Weißraum drum herum. In diesem Fall kann vielleicht `compactenum` Abhilfe schaffen. Dieser Befehl formatiert die Liste wie `enumerate`, setzt aber den vertikalen Abstand auf `Opt`.

Manchmal sollen Aufzählungen aber auch so aussehen:

- i) immer noch als abgesetzte Liste;
- ii) Punkte werden wie gewöhnlich, aber mit weniger vertikalem Abstand formatiert, d.h.
- iii) ähnlich einer `enumerate`-Liste.

```
\usepackage{paralist}
```

```
Manchmal sollen Aufzählungen aber auch so aussehen:
\begin{compactenum}[i]
\item immer noch als abgesetzte Liste;
\item Punkte werden wie gewöhnlich, aber mit
      weniger vertikalem Abstand formatiert, d.h.
\item ähnlich einer \texttt{enumerate}-Liste.
\end{compactenum}
```

Bsp.
3-3-7

Genaugenommen stimmt die Aussage von oben nicht – die vertikalen Abstände von `compactenum` können angepasst werden, und zwar mit folgenden Parametern: `\pltopsep` gibt den Abstand über und unter der Umgebung an, `\plpartopsep` gibt den zusätzlichen Abstand an, der hinzugefügt wird, wenn die Umgebung mit einem eigenen Absatz beginnt, `\plitemsep` gibt den Abstand zwischen den Listenelementen an und `\plparsep` gibt den Abstand zwischen Absätzen eines Listenelementes an.

Eine weitere Variante für nummerierte Listen bietet schließlich die Umgebung `asparaenum`; sie formatiert die Listenelemente als separate Absätze, d.h. die erste Zeile wird um `\parindent` eingerückt und die folgenden Zeilen werden bündig zum linken Seitenrand gesetzt.

Manchmal sollen Aufzählungen aber auch so aussehen:

- 1) immer noch als abgesetzte Liste;
- 2) Punkte werden als Absätze formatiert, aber ohne eingerückte Folgezeilen, d.h.
- 3) ähnlich einer `enumerate`-Liste, aber doch sichtbar unterschiedlich.

```
\usepackage{paralist}
```

```
Manchmal sollen Aufzählungen aber auch so aussehen:
\begin{asparaenum}[1]
\item immer noch als abgesetzte Liste;
\item Punkte werden als Absätze formatiert,
      aber ohne eingerückte Folgezeilen, d.h.
\item ähnlich einer \texttt{enumerate}-Liste, aber
      doch sichtbar unterschiedlich.
\end{asparaenum}
```

Bsp.
3-3-8

Wie in den vorangegangenen Beispielen gezeigt, unterstützen alle ein optionales Argument, das beschreibt, wie die Labels für die Listenelemente formatiert werden. Innerhalb des Argumentes haben die Zeichen `A`, `a`, `I`, `i` und `1` eine besondere Bedeutung: Sie werden durch den Nummerierungszähler ersetzt, der im Stil `\Alph`, `\alph`, `\Roman`, `\roman` bzw. `\arabic` angezeigt wird. Alle anderen Zeichen behalten ihre normale Bedeutung. Das Argument `[(a)]` erzeugt also beispielsweise Labels, wie (a), (b), (c) usw., während das Argument `[\S i:]` die Ausgabe `§i`, `§ii`, `§iii` usw. produziert.

Wenn das Label Textfolgen enthält, wie Beispiel 1, Beispiel 2, ... ist Vorsicht geboten. In diesem Fall muss das „i“ eingeklammert werden, was heißt,

dass ein Argument, wie `{\Beispiel} 1`, verwendet wird. Ansonsten gibt es seltsame Ergebnisse, wie das nächste Beispiel zeigt.

Punkt ii zeigt, was schief gehen kann:

Beispiel i: Beim ersten Punkt ist nichts zu sehen, aber

Beispiel ii: der zweite zeigt, was passiert, wenn ein spezielles Zeichen fehlinterpretiert wird.

Bsp.
3-3-9

```
\usepackage{paralist}
```

```
Punkt~\ref{bad} zeigt, was schief gehen kann:
\begin{asparaenum}[Beispiel i:]
\item Beim ersten Punkt ist nichts zu sehen, aber
\item\label{bad} der zweite zeigt, was passiert,
wenn ein spezielles Zeichen fehlinterpretiert wird.
\end{asparaenum}
```

Glücklicherweise entdeckt das Paket für gewöhnlich solch falsche Eingaben und gibt eine Warnung aus. Spezielle Zeichen zu verstecken, indem man sie in geschweifte Klammern einbettet, führt dazu, dass ein Argument wie `[\textbf{a}]` auch nicht funktioniert, weil das „a“ nicht mehr als Sonderzeichen betrachtet wird. Eine provisorische Lösung ist, eine andere Konstruktion zu verwenden, die keine Klammern benötigt, wie `\bfseries`.

Wie oben gezeigt, wird bei der Referenzierung von `\label` nur der Zählerwert in der gewählten Darstellung erzeugt, jedoch kein schmückendes Beiwerk, das im optionalen Argument angegeben ist. Dies gilt für alle Umgebungen für nummerierte Listen.

Mit dieser Syntax ist es nicht möglich anzugeben, dass im Label der äußere und der innere Nummerierungszähler wiedergegeben werden soll, weil die Symbole immer auf den aktuellen Nummerierungszähler verweisen. Es gibt nur eine Ausnahme: Wenn das Paket mit der Option `pointedenum` oder der Option `pointlessenum` geladen wird, werden Labels, wie jene im nächsten Beispiel, erzeugt.

1. Erste Ebene.

1.1. Zweite Ebene.

1.1.1. Dritte Ebene.

1.2. Wieder zweite Ebene.

Bsp.
3-3-10

```
\usepackage{pointedenum}{paralist}
```

```
\begin{compactenum}
\item Erste Ebene.
\begin{compactenum}
\item Zweite Ebene.
\begin{compactenum}\item Dritte Ebene.\end{compactenum}
\item Wieder zweite Ebene.
\end{compactenum}
\end{compactenum}
```

Der Unterschied zwischen den beiden Optionen besteht in der An- bzw. Abwesenheit des abschließenden Punktes. Alternativ zu den Optionen können die Befehle `\pointedenum` und `\pointlessenum` verwendet werden. Sie ermöglichen die Definition eigener Umgebungen, die Labels auf diese Weise formatieren, während andere Listenumgebungen Labels in anderen Formaten darstellen. Kompliziertere Labels, wie jene, die verschiedene Nummerierungszähler aus verschiedenen Ebenen beinhalten, müssen manuell erstellt werden; dazu kann man die in Abschnitt 3.3.1 auf Seite 136 beschriebenen Verfahren verwenden.

Diese Syntax im optionalen Argument, welches die Formatierung der Nummerierungslabels festlegt, wurde zunächst im Paket `enumerate` von

David Carlisle implementiert. Dieses Paket erweitert L^AT_EXs `enumerate`-Umgebung, um solch ein optionales Argument zu unterstützen. Mit `paralist` wird das optionale Argument von allen nummerierten Listen unterstützt, einschließlich der Standardumgebung `enumerate` (für welche es eine aufwärtskompatible Erweiterung darstellt).

Wenn eine der Umgebungen für nummerierte Listen ein optionales Argument verwendet, wird der linke Seitenrand nur so breit gewählt, dass die Labels hineinpassen. Genauer gesagt, wird der Einzug an diejenige Breite des Labels angepasst, die es beim Zählerwert sieben einnimmt. Für den Nummerierungsstil „Roman“ wird dabei eine relativ breite Nummer (vii) erzeugt, ansonsten passiert nichts. Dieses Verhalten wird im nächsten Beispiel gezeigt. Für einige Dokumente mag dieses Verhalten richtig sein, aber wenn ein einheitlicherer Einzug gewünscht ist, sollte die Option `neverdecrease` verwendet werden. Sie stellt sicher, dass der linke Seitenrand immer mindestens so breit ist wie die Standardeinstellung.

`\usepackage{paralist}`

Der linke Seitenrand kann variieren, wenn man nicht aufpasst.

`\begin{enumerate}`

`\item Ein Listenelement in einer normalen`
`\texttt{enumerate}-Umgebung.`

`\end{enumerate}`

`\begin{compactenum}`

`\item Derselbe linke Rand wie`
`\item in diesem Fall.`

`\end{compactenum}`

`\begin{compactenum}[i]`

`\item Aber ein anderer Einzug \item hier.`

`\end{compactenum}`

Bsp.
3-3-12

Der linke Seitenrand kann variieren, wenn man nicht aufpasst.

1. Ein Listenelement in einer normalen `enumerate`-Umgebung.

1. Derselbe linke Rand wie

2. in diesem Fall.

i) Aber ein anderer Einzug

ii) hier.

Auf der anderen Seite kann diese Art von Einstellung erzwungen werden, selbst bei Umgebungen ohne optionales Argument; dafür ist die Option `alwaysadjust` zu verwenden.

`\usepackage[alwaysadjust]{paralist}`

Hier wird stets der kleinstmögliche Einzug erzwungen:

`\begin{enumerate}`

`\item Ein Listenelement in einer normalen`
`\texttt{enumerate}-Umgebung.`

`\end{enumerate}`

`\begin{compactenum}[i]`

`\item Aber ein anderer`
`\item Einzug`
`\item hier.`

`\end{compactenum}`

`\begin{compactenum}[1.]`

`\item Derselbe linke Rand wie`
`\item in der ersten Liste.`

`\end{compactenum}`

Bsp.
3-3-12

Hier wird stets der kleinstmögliche Einzug erzwungen:

1. Ein Listenelement in einer normalen `enumerate`-Umgebung.

i) Aber ein anderer

ii) Einzug

iii) hier.

1. Derselbe linke Rand wie

2. in der ersten Liste.

Schließlich wird mit der Option `neveradjust` in allen Fällen der Standard-einzug verwendet. Dabei ragen alle Labels, die zu breit sind, in den linken Seitenrand hinein.

Mit dieser Option wird das Label in den Seitenrand verschoben.

1. Ein Listenelement in einer normalen `enumerate`-Umgebung.

Aufgabe A) Derselbe linke Rand wie Aufgabe B) in diesem Fall.

- 1) Und derselbe Einzug
- 2) hier.

Bsp.
3-3-13

```
\usepackage[neveradjust]{paralist}
```

Mit dieser Option wird das Label in den Seitenrand verschoben.

```
\begin{enumerate}
  \item Ein Listenelement in einer normalen
    \texttt{enumerate}-Umgebung.
\end{enumerate}
\begin{compactenum}[{Aufgabe} A)]
  \item Derselbe linke Rand wie
  \item in diesem Fall.
\end{compactenum}
\begin{compactenum}[1)]
  \item Und derselbe Einzug \item hier.
\end{compactenum}
```

Aufzählungen

Für Aufzählungen bietet das Paket `paralist` die Umgebungen `compactitem` (eine kompakte Version der Standardumgebung `itemize`), `asparaitem` (die Listenelemente als Absätze formatiert) und `inparaitem` (die im Text eingebettete Aufzählungen erzeugt). Die letzte Umgebung wurde hauptsächlich aus Symmetriegründen hinzugefügt. Alle drei Umgebungen haben ein optionales Argument, welches das Label angibt, das für die Listenelemente verwendet werden soll.

Liste mit besonderen Labels zu erzeugen, ist recht einfach.

- ★ Dieses Beispiel verwendet die Paketoption `neverdecrease`.
- ★ Ohne sie wäre der linke Rand schmaler.

Bsp.
3-3-14

```
\usepackage[neverdecrease]{paralist}
```

Liste mit besonderen Labels zu erzeugen, ist recht einfach.

```
\begin{compactitem}[$\star$]
  \item Dieses Beispiel verwendet die Paketoption
    \texttt{neverdecrease}.
  \item Ohne sie wäre der linke Rand schmaler.
\end{compactitem}
```

Die drei Optionen `neverdecrease`, `alwaysadjust` und `neveradjust`, mit denen man die Labels ausrichten kann, sind auch für Aufzählungen gültig, wie im vorangegangenen Beispiel zu sehen war. Wenn das Paket `paralist` geladen ist, wird $\LaTeX$ s Umgebung `itemize` erweitert, so dass sie auch diese Art des optionalen Argumentes unterstützt.

Beschreibungen

Für Beschreibungen führt das Paket `paralist` drei zusätzliche Umgebungen ein: `compactdesc` (die mit der Standardumgebung `description` von $\LaTeX$ identisch ist, außer dass alle vertikalen Abstände auf null – oder auf jeden anderen, benutzerdefinierten Wert – reduziert werden), `asparadesc` (die jeden Listenelement als Absatz formatiert) und `inparadesc` (die beschreibende Liste innerhalb des laufenden Textes zulässt).

Da `description`-Umgebungen jedes Label mit dem Befehl `\item` definieren, benötigen sie kein optionales Argument.

Wie wär's mit einer beschreibenden Liste im Text?

paralist Ein nützliches Paket, denn es unterstützt die Umgebungen **compact...** die keinen vertikalen Abstand haben, **aspara...**, die als Absätze formatiert werden und **inpara...** als im Text eingebettete Listen.
enumerate Ein jetzt überholtes Paket.

```
\usepackage{paralist}
Wie wär's mit einer beschreibenden Liste im Text?
\begin{compactdesc}
\item[paralist] Ein nützliches Paket, denn es
unterstützt die Umgebungen
\begin{inparadesc} \item[compact\ldots]
die keinen vertikalen Abstand haben,
\item[aspara\ldots,] die als Absätze
formatiert werden und
\item[inpara\ldots] als im Text
eingebettete Listen.
\end{inparadesc}
\item[enumerate] Ein jetzt überholtes Paket.
\end{compactdesc}
```

Bsp.
3-3-15

Anpassen der Voreinstellungen

Neben der Bereitstellung dieser nützlichen neuen Umgebungen ermöglicht das Paket `paralist`, die Voreinstellungen von nummerierten Listen und Aufzählungen zu ändern.

Mithilfe der Deklaration `\setdefaultitem` können die Standardlabels für unterschiedliche Ebenen der Aufzählungen angegeben werden. Sie hat vier Argumente (weil vier Verschachtelungsebenen möglich sind). In jedem Argument wird das gewünschte Label angegeben (so wie im optionalen Argument der Umgebung selbst). Wenn die Voreinstellung für eine Ebene nicht geändert werden soll, wird ein leeres Argument angegeben.

- Die äußere Ebene verwendet das voreingestellte Label.
- Die zweite Ebene verwendet auch einen Punkt.
- Und auf der dritten Ebene ein Stern.

```
\usepackage{paralist}
\setdefaultitem{}{\textbullet}{\$star$}{}
\begin{compactitem}
\item Die äußere Ebene verwendet das voreingestellte Label.
\begin{compactitem}
\item Die zweite Ebene verwendet auch einen Punkt.
\begin{compactitem}
\item Und auf der dritten Ebene ein Stern.
\end{compactitem}
\end{compactitem}
\end{compactitem}
```

Bsp.
3-3-16

Die geänderten Voreinstellungen gelten für alle nachfolgenden Aufzählungsumgebungen. In der Regel wird solch eine Deklaration in die Präambel eingefügt, sie kann aber auch verwendet werden, um Voreinstellungen in der Mitte des Dokumentes zu ändern. Insbesondere können Umgebungen definiert werden, die eine `\setdefaultitem`-Deklaration enthalten, die dann nur für diese bestimmte Umgebung gilt – aber dann auch für Listen, die darin verschachtelt sind.

Es ist wahrscheinlich wenig überraschend, dass es eine ähnliche Deklaration für nummerierte Listen gibt. Mit `\setdefaultenum` kann das voreingestellte Layout solcher Umgebungen gesteuert werden. Auch hier gibt es vier Argumente, die den vier möglichen Ebenen entsprechen. Für jede Ebene kann entweder eine Labeldefinition angegeben werden (über die zuvor bereits beschriebene Syntax) oder das Argument leer gelassen werden, wenn der voreingestellte Wert verwendet werden soll.

- 1) In diesem Beispiel haben alle Ebenen eine schließende Klammer.

- a) Hier Kleinbuchstaben.
i) Hier römische Ziffern.
ii) Wirklich!

Bsp.
3-3-17

```
\usepackage{paralist} \setdefaultenum{1}{a}{i}{A}
\begin{compactenum}
\item In diesem Beispiel haben alle Ebenen eine
      schließende Klammer.
  \begin{compactenum}
    \item Hier Kleinbuchstaben.
  \begin{compactenum}
    \item Hier römische Ziffern. \item Wirklich!
  \end{compactenum}
  \end{compactenum}
\end{compactenum}
```

Es gibt auch die Möglichkeit, den Einzug der einzelnen Listenebenen zu verändern, und zwar mit der Deklaration `\setdefaultleftmargin`. Dieser Befehl hat jedoch sechs Argumente (denn in den Standard- $\text{\LaTeX}$ -Klassen gibt es insgesamt 6 Listenebenen), in denen entweder ein Maß angegeben wird, das die Vergrößerung des Einzugs für die jeweilige Ebene festlegt, oder das leer bleibt, wenn der aktuelle Wert übernommen werden soll, der von der Klasse oder anderenorts definiert wurde. Ein weiterer Unterschied zu den vorangegangenen Deklarationen besteht darin, dass es sich in diesem Fall um absolute Listenebenen handelt, und nicht um relative Ebenen, die sich entweder auf Nummerierungen oder Aufzählungen (die untereinander vermischt werden können) beziehen. Ein Vergleich des nächsten Beispiels mit dem vorhergehenden zeigt die Unterschiede.

- 1) In diesem Beispiel haben alle Ebenen eine schließende Klammer.

- a) Hier Kleinbuchstaben.
i) Hier römische Ziffern.
ii) Wirklich!

Bsp.
3-3-18

```
\usepackage{paralist}
\setdefaultenum{1}{a}{i}{A}
\setdefaultleftmargin{\parindent}{\parindent}
      {\parindent}{\parindent}
\begin{compactenum}
\item In diesem Beispiel haben alle Ebenen eine
      schließende Klammer.
  \begin{compactenum}
    \item Hier Kleinbuchstaben.
  \begin{compactenum}
    \item Hier römische Ziffern. \item Wirklich!
  \end{compactenum}
  \end{compactenum}
\end{compactenum}
```

Standardmäßig werden die Labels in nummerierten Listen und Aufzählungen linksbündig gesetzt. Diese Einstellung lässt sich mit der Option `flushleft` ändern.

Wie bereits beschrieben, kann das Label einer `description`-Liste durch Modifikation von `\descriptionlabel` angepasst werden; dieser Befehl legt auch die Formatierung des Labels in einer `compactdesc`-Umgebung fest. Bei `inparadesc` und `asparadesc` wird jedoch ein anderer Befehl namens `\paradescriptionlabel` für diesen Zweck verwendet. Da diese Umgebungen ihre Labels leicht unterschiedlich positionieren, benötigt man bei ihnen keine Positions Korrektur um den Betrag von `\labelsep` (siehe zum Vergleich Seite 156). Ihre Standarddefinition lautet daher einfach:

```
\newcommand*\paradescriptionlabel[1]{\normalfont\bfseries #1}
```

Schließlich unterstützt das Paket `paralist` die Verwendung einer Konfigurationsdatei namens `paralist.cfg`, die standardmäßig geladen wird, sofern sie existiert. Dies kann durch Angabe der Option `nocfg` verhindert werden.

3.3.3 amsthm – Theoremähnliche Strukturen

Der Begriff „headed lists“ (Listen mit Überschrift) beschreibt typographische Strukturen, die wie andere Listen (z.B. Zitate) einen separaten oder abgesetzten Teil eines Abschnitts oder eines Kapitels bilden und bei denen mindestens Anfang und Ende eindeutig zu identifizieren sein müssen. Dies geschieht in der Regel durch Anpassung des vertikalen Abstandes am Anfang oder durch Anfügen einer Linie – und in diesem Fall auch durch Hinzufügen einer Überschrift, ähnlich einer Gliederungsüberschrift. Auch das Ende kann durch eine Linie oder ein anderes Symbol (z.B. im letzten Absatz) und zusätzlichem vertikalen Abstand abgesetzt werden.

Eine weitere Kennzeichnung dieser Listen ist, dass sie häufig nummeriert sind, entweder mit einer eigenen Nummerierung oder in Verbindung mit der Überschriftennummerierung.

Eine der wohl wichtigsten Vertreter von „Listen mit Überschrift“ sind die sogenannten „theoremähnlichen“ Umgebungen. Sie wurden ursprünglich hauptsächlich in mathematischen Schriften und Büchern verwendet, sind aber genauso auf eine Vielzahl anderer hervorgehobener Textpassagen anwendbar. Tatsächlich können alle Arten von Beispielen und Übungen auf diese Art präsentiert werden, auch wenn sie keine mathematischen Formeln enthalten.

Da ihre historischen Ursprünge in der mathematischen Welt liegen, wird das `amsthm`-Paket [7] von Michael Downes von der amerikanischen Gesellschaft für Mathematik (AMS) stellvertretend für diese Art von Erweiterung beschrieben.¹ Dieses Paket stellt eine verbesserte Version der Standard- $\text{\LaTeX}$ -Deklaration `\newtheorem` bereit, die theoremähnliche Umgebungen definiert.

Wie bei Standard- $\text{\LaTeX}$, haben Umgebungen, die auf diese Art deklariert werden, ein optionales Argument, in dem zusätzlicher Text, sogenannte „Anmerkungen“, zur Überschrift der Umgebung hinzugefügt werden können. Eine Demonstration gibt das Beispiel unten.

¹Wenn die Pakete `amsthm` und `amsmath` zusammen mit einer anderen als einer der AMS-Dokumentenklassen verwendet werden, muss das Paket `amsthm` *nach* `amsmath` geladen werden. Die AMS-Dokumentenklassen umfassen beide Pakete.

```
\newtheorem*{name}{überschrift}
```

Die Deklaration `\newtheorem` hat zwei obligatorische Argumente. Im ersten wird der Name der neuen Umgebung angegeben, den der Autor für dieses Element verwenden möchte. Das zweite enthält den Überschriftentext.

Wird anstelle von `\newtheorem` die Sternform `\newtheorem*` verwendet, dann werden für diese Umgebungen keine automatischen Nummern generiert. Diese Form des Befehls kann nützlich sein, wenn nur ein Lemma oder eine Übung vorhanden ist und diese(s) nicht nummeriert werden soll; außerdem eignet sie sich zur Erzeugung einzelner Theoreme, die unter ihrem Namen bekannt sind.

Lemma 1 (Zentrale Aussage). *Der $\text{\LaTeX}$ -Begleiter vervollständigt jede $\text{\LaTeX}$ -Einführung.*

Mittelbachs Lemma. *Der $\text{\LaTeX}$ -Begleiter enthält Pakete aus allen Anwendungsbereichen.*

```
\usepackage{amsthm}
\newtheorem{lem}{Lemma}
\newtheorem*{ML}{Mittelbachs Lemma}
\begin{lem}[Zentrale Aussage] Der \LaTeX{-}Begleiter
  vervollständigt jede \LaTeX{-}Einführung.
\end{lem}
\begin{ML} Der \LaTeX{-}Be-gleiter enthält
  Pakete aus allen Anwendungsbereichen.
\end{ML}
```

Bsp.
3-3-19

Neben den zwei obligatorischen Argumenten hat `\newtheorem` zwei sich gegenseitig ausschließende optionale Argumente. Sie betreffen die chronologische und hierarchische Abfolge der Nummerierung.

```
\newtheorem{name}[verwende-zähler]{überschrift}
\newtheorem{name}{überschrift}[nummer-innerhalb]
```

Standardmäßig werden die einzelnen theoremähnlichen Umgebungen unabhängig voneinander nummeriert. Wenn z.B. in einem Text Lemmata, Theoreme und Beispiele vorkommen, werden sie in etwa folgendermaßen nummeriert: Beispiel 1, Lemma 1, Lemma 2, Theorem 1, Beispiel 2, Lemma 3, Theorem 2. Wenn Lemmata und Theoreme eine gemeinsame Nummerierung haben sollen, d.h. Beispiel 1, Lemma 1, Lemma 2, Theorem 3, Beispiel 2, Lemma 4, Theorem 5, dann sollte die gewünschte Beziehung in folgender Weise angegeben werden:

```
\newtheorem{thm}{Theorem} \newtheorem{lem}[thm]{Lemma}
```

Das optionale Argument *verwende-zähler* (Wert `thm`) im zweiten Ausdruck bedeutet, dass die `lem`-Umgebung keine eigene Nummerierung einführen soll, sondern stattdessen auch die `thm`-Nummerierung verwenden soll.

Damit eine Theoremumgebung die Abschnittsnummerierung übernimmt, z.B. Übung 2.1, Übung 2.2 usw. in Abschnitt 2, muss an der letzten Stelle der Name des übergeordneten Zählers in eckigen Klammern angegeben werden:

```
\newtheorem{exa}{Übung}[section]
```

Mit dem optionalen Argument `[section]` wird der Zähler `exa` auf 0 zurückgesetzt, sobald der übergeordnete Zähler `section` hochgezählt wird.

Layoutdefinitionen für theoremähnliche Strukturen

Der Spezifikationsteil des `amsthm`-Paketes unterstützt die Idee eines aktuellen Theoremlayouts, das die Formatierung für eine Sammlung von `\newtheorem`-Deklarationen festlegt.¹

```
\theoremstyle{layout}
```

Die drei Theoremlayouts, die durch das Paket bereitgestellt werden, heißen `plain`, `definition` und `remark`. Sie legen verschiedene typographische Stile fest, die den Umgebungen je nach ihrer Bedeutung optisch Nachdruck verleihen. Die Details dieser typographischen Gestaltung können, abhängig von der Dokumentenklasse, variieren. In der Regel setzt `plain` den Haupttext in einem kursiven Schriftschnitt, während die anderen beiden Layouts eine aufrechte Antiqua verwenden.

Neue theoremähnliche Umgebungen in diesen Layouts erstellt man, indem man die `\newtheorem`-Deklarationen in Gruppen aufteilt. Jeder Gruppe wird der passende `\theoremstyle` vorangestellt. Wenn man keinen `\theoremstyle`-Befehl benutzt, wird das Layout `plain` verwendet. Hier einige Beispiele:

Definition 1. Eine typographische Herausforderung ist ein Problem, das sich nicht mithilfe des *LaTeX*-Begleiters lösen lässt.

Theorem 2. *Es gibt keine typographischen Herausforderungen.*

Anmerkung. Der Beweis bleibt dem Anwender überlassen.

```
\usepackage{amsthm}
\theoremstyle{plain}      \newtheorem{thm}{Theorem}
\theoremstyle{definition}\newtheorem{defn}[thm]{Definition}
\theoremstyle{remark}     \newtheorem*{rem}{Anmerkung}

\begin{defn} Eine typographische Herausforderung ist ein
              Problem, das sich nicht mithilfe des
              \emph{\LaTeX{}-Begleiters} lösen lässt.
\end{defn}

\begin{thm}Es gibt keine typographischen Herausforderungen.
\end{thm}

\begin{rem}Der Beweis bleibt dem Anwender überlassen.
\end{rem}
```

Bsp.
3-3-20

Zu beachten ist, dass die nahe liegende Bezeichnung „def“ für eine „Definition“-s-Umgebung nicht funktioniert, weil sie mit dem Low-Level-Befehl von $\TeX$ `\def` kollidiert.

Ein oft verwendetes Layout für Theoremüberschriften ist, die Nummer nicht rechts, sondern links von der Überschrift zu setzen. Da diese Variante normalerweise global angewendet wird, ungeachtet dessen, wie sich einzelne `\theoremstyles` ändern, kann man den Austausch von Nummern durchführen, indem man zu Beginn der Liste von `\newtheorem`-Befehlen, auf die sich dieser Wechsel auswirken soll, die Deklaration `\swapnumbers` hinzufügt.

Erweiterte Anpassung

Umfangreichere Möglichkeiten zur Anpassung bietet das Paket durch die Deklaration `\newtheoremstyle` und eine Reihe von Paketoptionen, über welche die Definitionen für Theoremlayouts geladen werden.

¹Dieses Konzept wurde ursprünglich im inzwischen überholten `theorem`-Paket von Frank Mittelbach eingeführt.

```
\newtheoremstyle{name}{abstand-oben}{abstand-unten}
               {text-layout}{einzug}{kopf-layout}
               {kopf-nach-punkt}{kopf-nach-abstand}{spez-kopf}
```

Ein neues Layout für eine theoremähnliche Struktur wird definiert, indem man diese Deklaration mit den neun obligatorischen Argumenten angibt, die unten beschrieben werden. Bei vielen dieser Argumente erhält man den unten aufgeführten voreingestellten Wert, wenn ein leeres Argument angegeben wird.

name Der Name, der für den Verweis auf das neue Layout verwendet wird.

abstand-oben Der vertikale Abstand über der theoremähnlichen Struktur, eine elastische Länge (voreingestellt `\topsep`).

abstand-unten Der vertikale Abstand unter der theoremähnlichen Struktur, eine elastische Länge (voreingestellt `\topsep`).

text-layout Eine Deklaration des Fonts und anderer Aspekte des Layouts, die für den Haupttext der Liste verwendet werden soll (voreingestellt `\normalfont`).

einzug Der zusätzliche Einzug der ersten Zeile der Liste, eine unelastische Länge (voreingestellt ist kein Einzug).

kopf-layout Eine Deklaration des Fonts und anderer Aspekte des Layouts, die für den Text der Listenüberschrift verwendet werden soll (voreingestellt ist `\normalfont`).

kopf-nach-punkt Der Text (normalerweise Satzzeichen), der nach dem Überschriftentext eingefügt werden soll, einschließlich Anmerkungen.

kopf-nach-abstand Der horizontale Abstand, der nach dem Überschriftentext und der Interpunktion eingefügt werden soll; eine elastische Länge. Dieses Argument darf nicht leer sein. Es gibt zwei Sonderfälle: Das Argument kann ein einzelnes Leerzeichen enthalten, das angibt, dass nur ein normaler Wortzwischenraum eingefügt werden soll, oder – was mehr überrascht – den Befehl `\newline`, der angibt, dass vor dem Beginn des Textkörpers der Liste ein Zeilenumbruch eingefügt werden soll.

spez-kopf Dieses Argument enthält die komplette Spezifikation für die Formatierung der Überschrift. Wenn das Argument leer ist, werden die Einstellungen des Theoremlayouts „plain“ verwendet. Einzelheiten hierzu siehe unten.

Layoutspezifikationen, die für die gesamte Umgebung gelten, sollten am besten in das Argument *text-layout* eingefügt werden. Hierbei ist jedoch darauf zu achten, wie es mit den automatischen Einstellungen interagiert. Einstellungen, die nur für die Überschrift gelten, können im *kopf-layout* angegeben werden.

Im Beispiel unten wird das Theoremlayout `break` definiert, das nach der Überschrift eine neue Zeile beginnt. Der Überschriftentext wird in serifenloser Schrift fett formatiert, gefolgt von einem Doppelpunkt und in den Rand um

12pt ausgerückt. Da die Beispiele im Buch in einem sehr schmalen Satzspiegel gesetzt sind, wurde dem Argument *text-layout* zusätzlich `\raggedright` hinzugefügt.¹

```
\usepackage{amsthm}
\newtheoremstyle{break}%
  {9pt}{9pt}%           Abstand oben und unten
  {\itshape\raggedright}% Textlayout
  {-12pt}%             Überschrift: Einzug
  {\sffamily\bfseries}{:}% Font und Satzzeichen danach
  {\newline}%           Abstand nach Überschrift
  {}%                   Überschriftenspezifikation
  %                     (leer = so wie 'plain' Layout)

\theoremstyle{break}
\newtheorem{exa}{Übung}
\begin{exa}[Aktiver Autor]
  Finde den Autor, von dem die meisten der im
  \LaTeX{}-Begleiter beschriebenen Pakete stammen.
\end{exa}
```

Übung 1 (Aktiver Autor):

Finde den Autor, von dem die meisten der im $\text{\LaTeX}$ -Begleiter beschriebenen Pakete stammen.

Bsp.
3-3-21

Überschriftenformat
spezifizieren

Das Argument *spez-kopf* wird, wenn es nicht leer ist, zum Ersetzungstext eines internen Befehls. Dieser wird zur Formatierung der (maximal) drei Teile verwendet, die in der Überschrift einer theoremähnlichen Umgebung enthalten sind: die Nummer (sofern vorhanden) der Überschrift, ihr Name und gegebenenfalls Anmerkungen des Autors, die dieser beim Gebrauch der Umgebung (im optionalen Argument) angegeben hat. Daher sollte dieser Ersetzungstext Verweise auf drei Argumente enthalten, die dann folgendermaßen ersetzt werden:

- #1 Der unveränderliche Text, der in der Überschrift verwendet wird (z.B. „Übungen“). Er wird aus `\newtheorem`-Deklaration ausgelesen, die zur Definition der Umgebung verwendet wurde.
- #2 Darstellung der Nummer eines Elementes, sofern dieses nummeriert werden soll. Wird normalerweise leer gelassen, wenn die Umgebung nicht nummeriert werden soll.
- #3 Der optionale Anmerkungs-text aus dem optionalen Argument der Umgebung.

Unter der Annahme, dass alle Teile vorhanden sind, könnte der Inhalt des Argumentes *spez-kopf* wie folgt aussehen:

```
#1 #2 \textup{(#3)}
```

Solch eine Deklaration ist natürlich möglich, aber es empfiehlt sich, die drei Argumentreferenzen eher nicht direkt zu verwenden, da dies zu unerwünschtem zusätzlichen Leerraum führen könnte, wenn z.B. die Umgebung nicht nummeriert ist.

Dieser erhöhten Komplexität trägt das Paket mit drei zusätzlichen Befehlen Rechnung, die jeweils ein Argument haben: `\thmname`, `\thmnumber` und `\thmnote`. Diese drei Befehle werden bei jeder Verwendung der Umgebung neu definiert, damit ihre Argumente in der richtigen Weise verarbeitet werden.

¹Das Beispiel funktioniert nicht, wenn `ragged2e` geladen ist (Stand 2005).

Ihre Standardeinstellung ist einfach „das Argument setzen“. Wenn jedoch z.B. die spezielle Instanz unnummeriert ist, wird `\thmnumber` so definiert, dass sein Argument nicht gesetzt wird. Eine bessere Definition des Argumentes *spez-kopf* wäre also

```
\thmname{#1}\thmnumber{ #2}\thmnote{ \textup{(#3)}}
```

Dieses entspricht der Einstellung, die das Standardlayout `plain` verwendet. Eine besondere Beachtung verdienen die Leerzeichen in den beiden letzten Argumenten: Sie geben den Wortzwischenraum an, der für die einzelnen Teile der Überschrift benötigt wird. Da sie jedoch innerhalb der Argumente platziert wurden, haben sie nur eine Auswirkung, wenn der entsprechende Teil der Überschrift gesetzt wird.

Im folgenden Beispiel wird eine „Theorem“-Variante vorgestellt, bei welcher die komplette Theoremüberschrift als optionale Anmerkung angegeben werden muss, wie z.B. beim Zitieren von Theoremen aus anderen Quellen.

Theorem 3.16 in [92].

Durch Fokussierung auf kleine Details kann man die tiefere Bedeutung einer Passage verstehen.

Bsp.
3-3-22

```
\usepackage{amsthm}
\newtheoremstyle{citing}%      Name
    {3pt}{3pt}%                Abstand oben und unten
    {\itshape}%                Textlayout
    {\parindent}{\bfseries}%   Überschrifteinzug und Font
    {.%}%                      Interpunktion nach Überschrift
    { }%                       Abstand nach Überschrift
    {\thmnote{#3}}%            Anmerkung nur, wenn vorhanden
\theoremstyle{citing} \newtheorem*{varthm}{}
\begin{varthm}[Theorem 3.16 in \cite{Knuth90}]
Durch Fokussierung auf kleine Details kann man die
tiefere Bedeutung einer Passage verstehen.
\end{varthm}
```

Beweisführungen und das QED-Zeichen

Von speziell mathematischer Natur ist die von `amsthm` definierte `proof`-Umgebung. Diese Umgebung erzeugt die Überschrift „Proof“ (anpassbar über den Parameter `\proofname`) mit passendem Leerraum und Satzzeichen und fügt am Ende automatisch ein „QED-Zeichen“ ein.¹

Mit einem optionalen Argument der `proof`-Umgebung hat man die Möglichkeit, die Standardbezeichnung „Proof“ durch eine andere zu ersetzen. Wenn die Überschrift z.B. „Beweis des Haupttheorems“ lauten soll, ist sie folgendermaßen zu definieren:

```
\begin{proof}[Beweis des Haupttheorems]
...
\end{proof}
```

Das „QED-Zeichen“ (voreingestellt $\square$) wird automatisch an das Ende der `proof`-Umgebung angehängt. Um das Beweisendezeichen durch ein anderes zu ersetzen, kann man mithilfe von `\renewcommand` den Befehl

¹Die `proof`-Umgebung ist hauptsächlich für kurze Beweise gedacht, die nicht länger als ein oder zwei Seiten sind. Längere Beweise lassen sich im Dokument in der Regel besser als separate `\section` oder `\subsection` realisieren.

`\qedsymbol` neu definieren. Für einen längeren Beweis, der als `\section` oder `\subsection` realisiert wurde, kann man mit dem Befehl `\qed` das Zeichen und den üblichen Vorabstand erzeugen. Dazu muss der Befehl an der Stelle eingefügt werden, an der das Zeichen erscheinen soll.

Die automatische Platzierung des QED-Zeichens kann problematisch sein, wenn der letzte Teil einer `proof`-Umgebung z.B. tabellarisch oder eine abgesetzte Gleichung oder Liste ist. In diesem Fall sollte der Befehl `\qedhere` an einer früheren Stelle eingefügt werden, nämlich dort, wo das QED-Zeichen erscheinen soll. Damit wird sein Erscheinen am logischen Ende der `proof`-Umgebung unterdrückt. Wenn `\qedhere` in einer Gleichung eine Fehlermeldung erzeugt, sollte stattdessen `\mbox{\qedhere}` verwendet werden.

<i>Beweis (Hinlänglichkeit).</i> Dieser Beweis beinhaltet eine Liste: 1. Weil der Beweis zweigeteilt ist, 2. muss <code>\qedhere</code> verwendet werden. □	<pre> \usepackage{amsthm} \begin{proof}[Beweis (Hinlänglichkeit)] Dieser Beweis beinhaltet eine Liste: \begin{enumerate} \item Weil der Beweis zweigeteilt ist, \item muss \verb \qedhere verwendet werden. \qedhere \end{enumerate} \end{proof} </pre>

Bsp.
3-3-23

3.3.4 Erstellen eigener Listen

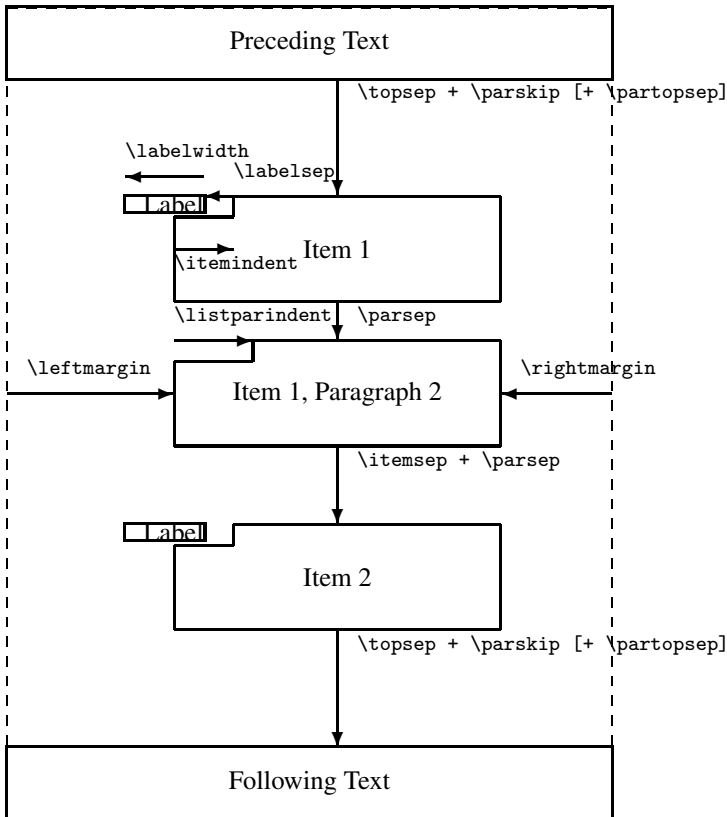
In $\text{\LaTeX}$ werden die meisten Listen, einschließlich derjenigen, die bereits vorgestellt wurden, intern mit der generischen `list`-Umgebung erzeugt. Diese hat folgende Syntax:

```
\begin{list}{standardlabel}{dekl} element-liste \end{list}
```

Das Argument *standardlabel* gibt den Text an, der als Label verwendet wird, wenn ein `\item`-Befehl ohne optionales Argument verwendet wird. Das zweite Argument *dekl* stellt die verschiedenen geometrischen Parameter für die `list`-Umgebung ein, die schematisch in Abbildung 3.3 auf der nächsten Seite dargestellt sind.

Die Standardwerte dieser Parameter hängen in der Regel vom Schriftgrad und der Listenebene ab. Die Parameter mit vertikaler Ausrichtung sind elastische Längen, d.h. sie können gestaucht und gedehnt werden. Sie werden von der `list`-Umgebung wie folgt gesetzt: Wenn die neue Umgebung beginnt, wird der interne Befehl `\@list<level>` ausgeführt. Dabei gibt *<level>* die Verschachtelungstiefe der Liste an, die in römischen Ziffern dargestellt wird (z.B. `\@listi` für die erste Ebene, `\@listii` für die zweite Ebene, `\@listiii` für die dritte Ebene usw.). Jeder dieser Befehle, die von der Dokumentenklasse definiert werden, hält für die jeweilige Ebene die entsprechenden Einstellungen bereit. Typischerweise bietet die Dokumentenklasse separate Definitionen für alle durch Optionen wählbare Hauptschriftgrade des Dokumentes. Bei Wahl der Option `11pt` werden beispielsweise unter anderem die Standardwerte für Listen geändert. In den Standardklassen geschieht dies durch Laden der Datei `size11.clo`, welche die Definitionen für die Schriftgröße 11 pt enthält.

Außerdem ändern die meisten Klassen `\@listi` (d.h. Listenwerte für die erste Ebene) innerhalb der größenändernden Befehle `\normalsize`, `\small`



Bsp.
3-3-24

$\backslash\topsep$ Elastischer Leerraum zwischen erstem Listenpunkt und vorangegangenen Absatz.

$\backslash\partopsep$ Zusätzlicher Leerraum, der zu dem Parameter $\backslash\topsep$ hinzugefügt wird, wenn die Umgebung einen neuen Absatz beginnt.

$\backslash\itemsep$ Elastischer Leerraum zwischen aufeinander folgenden Listenpunkten.

$\backslash\parsep$ Elastischer Leerraum zwischen Absätzen innerhalb eines Listenpunktes.

$\backslash\leftmargin$ Abstand zwischen dem linken Rand der umschließenden Umgebung (oder der Seite bei der obersten Verschachtelungsebene) und dem linken Rand dieser Liste. Muss größer oder gleich null sein. Der Wert hängt von der jeweiligen Listenebene ab.

$\backslash\rightmargin$ Ähnlich wie $\backslash\leftmargin$, aber für den rechten Rand. Voreinstellung ist Opt.

$\backslash\listparindent$ Zusätzlicher Einzug am Anfang jedes Absatzes einer Liste, außer bei denjenigen, die mit $\backslash\item$ beginnen. Kann negativ sein, beträgt aber normalerweise Opt.

$\backslash\itemindent$ Zusätzlicher Einzug, der zum horizontalen Einzug des Textteils der ersten Zeile eines Listenpunktes hinzuaddiert wird. Indem man die Werte von $\backslash\labelsep$ und $\backslash\labelwidth$ davon abzieht, wird mithilfe dieses Bezugspunktes berechnet, wo das Label beginnt. Der Wert dieser Länge beträgt normalerweise Opt.

$\backslash\labelwidth$ Die nominale Weite der Labelbox. Wenn die natürliche Breite des Labels $\leq \backslash\labelwidth$ ist, wird das Label per Voreinstellung rechtsbündig in einer Box der Breite $\backslash\labelwidth$ gesetzt. Ansonsten wird eine Box mit der natürlichen Breite verwendet, welche wiederum einen ebenso großen Einzug für den Text dieser Zeile erzeugt. Die Art, wie das Label formatiert wird, kann mit dem Befehl $\backslashmakelabel$ geändert werden.

$\backslash\labelsep$ Abstand zwischen dem Ende der Labelbox und dem Text des ersten Punktes. Der Standardwert beträgt 0.5em.

Abbildung 3.3: Parameter der list-Umgebung (mit layouts erstellt)

und `\footnotesize` ab. Dies geschieht in der Annahme, dass es Listen innerhalb von Textpassagen gibt, die in solch kleinen Schriftgrößen gesetzt sind. Da aber diese Anpassungen recht unvollständig sind, sind dabei seltsame Effekte möglich, wenn man

- verschachtelte Listen in diesen kleinen Größen verwendet (die inneren Listen erhalten die Standardwerte, die für `\normalsize` bestimmt sind),
- von `\small` oder `\footnotesize` direkt auf eine größere Schrift, wie `\huge` wechselt (eine Liste erster Ebene erbt jetzt die Voreinstellungen von der kleinen Größe, da `\huge` diese nicht anpasst).

Mit einer komplexeren Konfiguration könnten diese Mängel beseitigt werden. Da die einfachere Variante jedoch in den meisten Situationen sehr gut funktioniert, bieten die meisten Klassen nur diese eingeschränkte Unterstützung.

*Globale Änderungen
sind schwierig*

Da die Einstellungen für die Listenparameter vom Schriftgrad und der Verschachtelungstiefe abhängen, können die Parameter nicht global in der Dokumentenpräambel geändert werden. Um Parameter global zu ändern, muss man die verschiedenen, oben angesprochenen `\@list...`-Befehle neu definieren oder eine andere Dokumentenklasse auswählen.

*Seitenumbruch vor, in
und nach Listen*

Seitenumbrüche außerhalb und innerhalb einer Listenstruktur werden über drei $\text{\TeX}$ -Zähler gesteuert: `\@beginparpenalty` (für einen Umbruch vor der Liste), `\@itempenalty` (für einen Umbruch vor einem Listenpunkt innerhalb der Liste) und `\@endparpenalty` (für einen Umbruch nach einer Liste). Standardmäßig werden alle drei auf einen leicht negativen Wert gesetzt; dies bedeutet, dass es zulässig (und sogar empfehlenswert) ist, eine Seite an diesen Stellen zu umbrechen (anstatt an anderen Stellen). Es kann jedoch sein, dass das Ergebnis trotzdem nicht zufriedenstellend ausfällt. Möglicherweise soll von einem Seitenumbruch vor einer Liste eher abgeraten bzw. dieser unterdrückt werden. Dies erreicht man, indem man `\@beginparpenalty` einen hohen Wert zuweist (10000 oder mehr verhindert einen Umbruch unter allen Umständen). Ein Beispiel:

```
\makeatletter \@beginparpenalty=9999 \makeatother
```

$\text{\TeX}$ -Zähler benötigen diese ungewöhnliche Form der Wertzuweisung, und da alle drei ein `@`-Zeichen im Namen haben, müssen sie in `\makeatletter` und `\makeatother` eingebettet werden, wenn die Zuweisung in der Präambel geschieht.

*Viele Umgebungen
sind als Listen
implementiert*

Man muss sich dessen bewusst sein, dass solch eine Einstellung global für alle Umgebungen gilt, die auf der generischen `list`-Umgebung basieren (sofern sie nicht im Argument *dekl* vorgenommen wurde). Außerdem werden verschiedene $\text{\LaTeX}$ -Umgebungen mithilfe von `list` definiert (z.B. `quote`, `quotation`, `center`, `flushleft` und `flushright`). Diese Umgebungen sind „Listen“ mit einem einzigen Listenpunkt, bei denen der Befehl `\item[]` in der Definition der Umgebung angegeben wird. Sie werden hauptsächlich deshalb intern als Listen definiert, damit sie denselben vertikalen Abstand bekommen wie andere abgesetzte Objekte. Dies trägt zu einem einheitlichen Layout bei.

Ein Beispiel hierfür ist die Umgebung `quote`, deren Definition die Ränder links und rechts gleichsetzt. Die einfache Variante `Quote`, die unten gezeigt wird, ist – bis auf die Anführungszeichen um den Text – identisch mit `quote`. Zu beachten sind die besonderen Vorsichtsmaßnahmen, die erforder-

lich sind, um unerwünschten Leerraum vor (`\ignorespaces`) und hinter dem Text (`\unskip`) zu vermeiden. Durch Platzierung der Anführungsstriche in Boxen ohne horizontale Ausdehnung laufen diese in den Seitenrand. Diesen Trick sollte man sich merken: Bei einer Box, der man die horizontale Ausdehnung null zuweist, und rechtsbündiger Ausrichtung des Inhalts ragt der Text nach links heraus.

	<code>\usepackage[ngerman]{babel}</code>
	<code>\newenvironment{Quote}%</code>
	<code>{\begin{list}{}%</code>
	<code>{\setlength\rightmargin{\leftmargin}}%</code>
	<code>\item[]\makebox[Opt][r]{""\ignorespaces}%</code>
<code>{\unskip\makebox[Opt][l]{""}\end{list}}</code>	
... vorangehender Text.	<code>\ldots\ vorangehender Text.</code>
"Zitierter Text, weiterer zitierter Text."	<code>\begin{Quote}</code>
	<code>Zitierter Text, weiterer zitierter Text.</code>
	<code>\end{Quote}</code>
Nachfolgender Text ...	<code>Nachfolgender Text \ldots</code>

Bsp.
3-3-25

Im restlichen Teil dieses Kapitels werden eine Reihe von verschiedenen „beschreibenden“ Listen generiert, um die verschiedenen Möglichkeiten zu erläutern, welche die generische `list`-Umgebung bietet. Den Anfang macht die Standarddefinition der `description`-Umgebung, wie sie in den Standardklassen von $\text{\LaTeX}$, wie `article` oder `report`, vorkommt.¹

```
\newenvironment{description}
  {\begin{list}{\setlength\labelwidth{0pt}%
               \setlength\itemindent{-\leftmargin}%
               \let\makelabel\descriptionlabel}
  {\end{list}}
```

Die Gründe hinter dieser Definition zeigt Abbildung 3.3 auf Seite 153, welche die Beziehung zwischen den verschiedenen Listenparametern erläutert. Die erste Parametereinstellung betrifft `\labelwidth`: Er wird auf null gesetzt, was bedeutet, dass kein Leerraum für das Label reserviert wird. Bei der Formatierung des Labels wird der Text der ersten Zeile damit nach rechts verschoben, damit der erforderliche Leerraum entsteht. Danach wird der Parameter `\itemindent` auf den negierten Wert von `\leftmargin` gesetzt. Im Ergebnis wird das erste Zeichen genau bis an den Rand der umschließenden Umgebung verschoben, alle weiteren Zeilen werden weiterhin um `\leftmargin` eingezogen. Die letzte Deklaration setzt `\makelabel` mit dem Befehl `\descriptionlabel` gleich. Der Befehl `\makelabel` wird von der `list`-Umgebung aufgerufen, sobald das Label eines Listenelementes formatiert werden muss. Er hat ein Argument (das Label) und dient dazu, eine formatierte Version dieses Argumentes zu erzeugen. Um die Definition der `description`-Umgebung abzuschließen, muss als Letztes eine geeignete Definition für `\descriptionlabel` bereit gestellt werden. Dieser Umweg ist nützlich, weil er eine geänderte Labelformatierung ermöglicht, ohne dass der Rest der Definition der Umgebung modifiziert werden muss.

¹Ein Blick in `article.cls` oder `report.cls` zeigt eine leicht optimierte Kodierung, die beispielsweise Low-Level-Zuordnungen nutzt, anstelle von `\setlength`. Vom Konzept her sind beide Definitionen jedoch identisch.

Wie sollte `\descriptionlabel` definiert werden? Der Befehl stellt die Formatierung für das Label bereit. Mit der Standardumgebung `description` wird das Label normalerweise fett formatiert. Zu beachten ist, dass das Label vom folgenden Text durch einen Abstand der Breite `\labelsep` getrennt ist. Aufgrund der obigen Parametereinstellungen beginnt dieser Text im Seitenrand. Ohne Korrektur würde das Label in den Rand gerückt (um die Breite von `\labelsep`). Um dieses Ergebnis zu verhindern, hat die Standarddefinition für den Befehl `\descriptionlabel` folgende seltsame Definition, durch die das Label erst nach rechts gerückt und dann formatiert wird:

```
\newcommand*\descriptionlabel[1]
  {\hspace{\labelsep}\normalfont\bfseries #1}
```

Um diese Abhängigkeit zu beseitigen, müsste man die Einstellung von `\itemindent` ändern, um `\labelsep` zu berücksichtigen, was an sich nicht schwierig wäre. Man könnte dieses Verhalten als historisches Artefakt bezeichnen, aber viele Dokumente verlassen sich auf diese etwas obskure Funktion. Daher ist es schwierig, die Einstellung im $\text{\LaTeX}$ -Kernel zu ändern, ohne diese Dokumente zu zerstören.

Mit den Parametereinstellungen der Standardumgebung `description` beginnt der Text der ersten Zeile bei kurzen Labels vor dem Text der übrigen Zeilen. Wenn der Text stets einen gewissen Betrag eingerückt werden soll, kann man eine Definition ähnlich dem folgenden Beispiel verwenden. Hier wird `\labelwidth` auf 40pt und `\leftmargin` auf `\labelwidth` plus `\labelsep` gesetzt. Dies bedeutet, dass sich `\makelabel` selbst nur um die Formatierung des Labels kümmern muss. Aber da der Befehl jetzt eine positive nominale Labelbreite generiert, muss definiert werden, was passieren soll, wenn das Label klein ist. Mit `\hfil` kann man angeben, wo zusätzlicher Weißraum eingefügt werden soll.

```
\usepackage{calc}
\newenvironment{Description}
  {\begin{list}{}{\let\makelabel\Descriptionlabel
    \setlength\labelwidth{40pt}%
    \setlength\leftmargin{\labelwidth+\labelsep}}}%
  {\end{list}}
\newcommand*\Descriptionlabel[1]{\textsf{#1:}\hfil}
\begin{Description}
\item[Beschreibung]
  Rücksprung aus einer Funktion. Bei Verwendung
  im Hauptprogramm endet das Programm.
\item[Fehler] Keine.
\item[Rückgabewerte]
  \mbox{}\hspace{1cm}
  Alle Parameter werden an die aufrufende
  Komponente zurückgegeben.
\end{Description}
```

Beschreibung: Rücksprung aus einer Funktion. Bei Verwendung im Hauptprogramm endet das Programm.

Fehler: Keine.

Rückgabewerte:

Alle Parameter werden an die aufrufende Komponente zurückgegeben.

Bsp.
3-3-26

Dieses Beispiel zeigt ein typisches Problem beschreibender Listen, wenn der zu beschreibende *Term* breiter ist als der vorgesehene Raum für das Label. Mit der obigen Definition ragt der Text des Terms in den Text der *Be-*

schreibung hinein. Dieser Effekt ist häufig unerwünscht. Um das optische Erscheinungsbild der Liste zu verbessern, wurde einer der beschreibenden Teile in der nächsten Zeile gestartet. Ein Zeilenumbruch wurde erzwungen, indem eine leere Box, gefolgt vom Befehl ‘\\’, auf dieselbe Zeile platziert wurde.

Im übrigen Teil dieses Abschnitts werden verschiedene Möglichkeiten untersucht, die Breite und gegenseitige Positionierung von Begriff und Beschreibung festzulegen. Zunächst lässt sich die Breite des Labels ändern. Dazu wird die Umgebung mit einem Argument deklariert, das die gewünschte Breite des Labelfeldes angibt (normalerweise die Breite des längsten Begriffs). Man beachte auch die neue Definition des Befehls `\makelabel`, die angibt, wie das Label formatiert wird. Da diese neue Definition innerhalb der Definition¹ für die Umgebung `altDescription` steht, muss der Argumentplatzhalter `#` durch `##` dargestellt werden, um L^AT_EX anzuzeigen, dass dieser Platzhalter nicht zu dem Argument der äußeren Umgebung, sondern zu dem Argument von `\makelabel` gehört. Der Parameter `\labelwidth` wird auf die Breite des Argumentes der Umgebung gesetzt, nachdem es durch `\makelabel` formatiert wurde. Auf diese Weise werden Formatierungsbefehle für das Label berücksichtigt, die dessen Breite ändern könnten.

		<pre> \usepackage{calc} \newenvironment{altDescription}[1] {\begin{list}{}% {\renewcommand\makelabel[1]{\textsf{##1:}\hfil}% \settowidth\labelwidth{\makelabel{#1}}% \setlength\leftmargin{\labelwidth+\labelsep}}% {\end{list}} </pre>
Beschreibung:	Rücksprung aus einer Funktion. Bei Verwendung im Hauptprogramm endet das Programm.	<pre> \begin{altDescription}{Rückgabewerte} \item[Beschreibung] </pre>
Fehler:	Keine.	<pre> \item[Rückgabewerte] </pre>
Rückgabewerte:	Alle Parameter werden an die aufrufende Komponente zurückgegeben.	<pre> \item[Fehler] Keine. \item[Rückgabewerte] </pre>
		<pre> Alle Parameter werden an die aufrufende Komponente zurückgegeben. \end{altDescription} </pre>

Bsp.
3-3-27

Eine ähnliche Umgebung (aber unter Verwendung eines optionalen Argumentes) wird in Beispiel A-1-9 auf Seite 883 gezeigt. Verschiedene Listen auf der gleichen Seite, die unterschiedlich breite Labelfelder haben, können jedoch nach typographischen Gesichtspunkten unakzeptabel sein. Daher besteht eine weitere Möglichkeit darin, die Länge des Begriffs auszuwerten. Wenn der Begriff länger ist als `\labelwidth`, wird zusätzlich eine leere Box eingefügt, so dass die Beschreibung erst in der nächsten Zeile beginnt. Dies entspricht dem üblichen Verfahren für die Darstellung von Befehlsoptionen in UNIX-Handbüchern.

Zur Demonstration dieses Verfahrens wird im nächsten Beispiel die in Beispiel 3-3-26 definierte `Description`-Umgebung wiederverwendet, jedoch

¹Dies geschieht aus Demonstrationszwecken. In der Regel ist die Lösung mit externem Namen vorzuziehen, wie bei `\Descriptionlabel` in Beispiel 3-3-26 auf der vorherigen Seite.

mit einer abgewandelten Definition für den Befehl `\Descriptionlabel`.

Beschreibung:

Rücksprung aus einer Funktion. Bei Verwendung im Hauptprogramm endet das Programm.

Fehler: Keine.

Rückgabewerte:

Alle Parameter werden an die aufrufende Komponente zurückgegeben.

```
\usepackage{calc,ifthen} \newlength{\Mylen}
% Definition der Description-Umgebung wie zuvor
\newcommand*\Descriptionlabel[1]{%
  \settowidth\Mylen{\textsf{#1:}}% Breite bestimmen
  \ifthenelse{\lengthtest{\Mylen > \labelwidth}}{%
    {\parbox[b]{\labelwidth}% Term > Labelbreite
      {\makebox[Opt][l]{\textsf{#1:}}\mbox{}}}%
    {\textsf{#1:}}% Term <= Labelbreite
  \hfill}
\begin{Description}
\item[Beschreibung]
  Rücksprung aus einer Funktion. Bei Verwendung
  im Hauptprogramm endet das Programm.
\item[Fehler] Keine.
\item[Rückgabewerte]
  Alle Parameter werden an die aufrufende
  Komponente zurückgegeben.
\end{Description}
```

Bsp.
3-3-28

Der Befehl `\Descriptionlabel` weist der Längenvariable `\Mylen` die Breite des Labels zu. Anschließend wird diese Länge mit `\labelwidth` verglichen. Wenn das Label nicht breiter ist als `\labelwidth`, wird es in derselben Zeile gesetzt wie die Beschreibung; andernfalls wird es in einer Box der Breite 0 gesetzt, so dass der Inhalt so weit wie notwendig über den rechten Rand der Box herausragt. Es wird einer am unteren Rand ausgerichteten `\parbox` platziert, gefolgt von einem erzwungenem Zeilenumbruch, so dass die Beschreibung eine Zeile tiefer beginnt. Dieses etwas komplizierte Verfahren ist notwendig, weil `\makelabel`, und damit auch `\Descriptionlabel`, in einem streng horizontalem Kontext ausgeführt werden, in dem vertikale Abstände oder `\\`-Befehle keine Wirkung haben.

Eine weitere Möglichkeit ist, mehrzeilige Labels zuzulassen.

**Be-
schrei-
bung:**

Rücksprung aus einer Funktion. Bei Verwendung im Hauptprogramm endet das Programm.

Fehler: Keine.

**Rückga-
bewerte:**

Alle Parameter werden an die aufrufende Komponente zurückgegeben.

```
\usepackage{calc}
% Definition der Description-Umgebung wie zuvor
\newcommand*\Descriptionlabel[1]
{\raisebox{Opt}[1ex][Opt]{%
  {\makebox[\labelwidth][l]{%
    {\parbox[t]{\labelwidth}%
      {\hspace{Opt}\textsf{#1:}}}}}}
\begin{Description}
\item[Beschreibung]
  Rücksprung aus einer Funktion. Bei Verwendung
  im Hauptprogramm endet das Programm.
\item[Fehler] Keine.
\item[Rückgabewerte]
  Alle Parameter werden an die aufrufende
  Komponente zurückgegeben.
\end{Description}
```

Bsp.
3-3-29

Im vorangegangenen Beispiel wurde als Grundlage wieder die Umgebung `Description` und dazu eine weitere neue Definition des Befehls `\Descriptionlabel` verwendet. Dieser Befehl soll bewirken, dass längere Labels über mehrere Zeilen umbrochen werden können. Um eine Trennung des ersten Wortes in einem Absatz zu ermöglichen, müssen bestimmte Vorsichtsmaßnahmen beachtet werden. Dazu wurde in der Definition der Befehl `\hspace{0pt}` benutzt. Der Text wird in einer Absatzbox der richtigen Breite `\labelwidth` gesetzt, die dann am oberen und linken Rand in einer weiteren Box ausgerichtet wird. Diese wird wiederum selbst in einer Box platziert, welche die Höhe `1ex` und keine Tiefe hat. Auf diese Weise erkennt $\text{\LaTeX}$ nicht, dass sich der Text unter die erste Zeile erstreckt.

Das letzte Beispiel befasst sich mit der Definition von nummerierten Listen. Wenn man den Befehl `\usecounter` in die Deklaration der `list`-Umgebung einbettet, kann man eine Umgebung mit einem automatisch erhöhten Zähler erstellen. Diese Funktionalität wird anhand der Umgebung `Notes` gezeigt, die eine Reihe von Anmerkungen erzeugt. In diesem Fall stellt das erste Argument der `list`-Umgebung den automatisch generierten Text für den Begriff zur Verfügung.

Nachdem der Zähler `notes` deklariert ist, wird das Standardlabel der Umgebung `Notes` definiert. Es besteht aus dem in Kapitälchen gesetzten Wort `ANMERKUNG`, gefolgt von dem Wert des Zählers `notes`, der als arabische Ziffer mit darauf folgendem Punkt dargestellt wird. Als Nächstes wird `\labelsep` auf einen relativ hohen Wert gesetzt, während die Parameter `\itemindent`, `\leftmargin` und `\labelwidth` so angepasst werden, dass das Label trotzdem am linken Rand beginnt. Schließlich sorgt die bereits erwähnte `\usecounter`-Deklaration dafür, dass der Zähler `notes` bei jedem `\item`-Befehl hochgesetzt wird.

	<pre>\newcounter{notes} \newenvironment{Notes} {\begin{list}{\textsc{Anmerkung} \arabic{notes}.}% {\setlength\labelsep{10pt}% \setlength\itemindent{10pt}% \setlength\leftmargin{0pt}% \setlength\labelwidth{0pt}% \usecounter{notes}}}% {\end{list}}</pre>
ANMERKUNG 1. Dies ist der Text der ersten Anmerkung.	<pre>\begin{Notes} \item Dies ist der Text der ersten Anmerkung.</pre>
ANMERKUNG 2. Dies ist der Text der zweiten Anmerkung.	<pre>\item Dies ist der Text der zweiten Anmerkung. \end{Notes}</pre>

Bsp.
3-3-30

3.4 Wortwörtlicher Text

Oft muss man Informationen unverändert, also „wie über die Tastatur eingegeben“, darstellen. Diese Funktionalität wird über die Standardumgebung `verbatim` von $\text{\LaTeX}$ realisiert. Um den Benutzer zu leiten, kann es jedoch nützlich sein, dabei einige Textteile besonders hervorzuheben, z.B. durch Nummerierung der Zeilen. Nach und nach wurden eine Reihe von Paketen entwickelt,

die sich mit der einen oder anderen Sonderfunktion befassen – leider jedoch jedes mit einer eigenen Syntax.

In diesem Abschnitt werden ein paar solcher Pakete kurz beleuchtet. Da sie in der Vergangenheit intensiv genutzt wurden, findet man sie in älteren Quelldokumenten im Internet oder hat das eine oder andere auch selbst schon verwendet. Der dann folgende Teil konzentriert sich aber auf das Paket `fancyvrb` von Timothy Van Zandt, das all diese Funktionen und viele andere unter dem Dach eines einzigen, leicht anpassbaren Paketes vereint.

Anschließend folgt eine Besprechung des Paketes `listings`, das eine vielseitige Umgebung bereitstellt, in der Computerlistings in vielen Computersprachen auf ansprechende Weise ausgegeben werden können.

3.4.1 Einfache Verbatim-Erweiterungen

Das Paket `alltt` (von Leslie Lamport) definiert die Umgebung `alltt`. Diese verhält sich wie die Umgebung `verbatim`, bis auf die Tatsache, dass der Backslash „\“ und die Klammern „{“ und „}“ ihre übliche Bedeutung behalten. Dadurch können andere Umgebungen und Befehle innerhalb der Umgebung `alltt` verwendet werden. Eine ähnliche Funktionalität bietet das Schlüsselwort `commandchars` der Umgebung `fancyvrb` (siehe Seite 169).

Man kann den *Font* wechseln.

Einige Sonderzeichen: # \$ % ^ & ~ _

```
\usepackage{alltt}
\begin{alltt}
Man kann den \emph{Font} wechseln.
Einige Sonderzeichen: # $ % ^ & ~ _
\end{alltt}
```

Bsp.
3-4-1

In Dokumenten, in denen zahlreiche `\verb`-Befehle benötigt werden, ist das Quelldokument bald schwer zu lesen. Aus diesem Grund führt das Paket `doc`, das in Kapitel 14 beschrieben wird, eine Abkürzung ein, mit der über ein Sonderzeichen Anfang und Ende der wortwörtlichen Ausgabe gekennzeichnet werden können, ohne dass diese ständig wieder mit `\verb` eingeleitet werden muss. Diese Funktion ist auch in einem separaten Paket namens `shortvrb` verfügbar. Dieselbe Funktionalität bietet auch `fancyvrb`, jedoch wird hier eine etwas andere Syntax verwendet (siehe Seite 177).

Mit `\MakeShortVerb` werden Quelldateien leichter lesbar. Und mit der Deklaration `\DeleteShortVerb{\|}` kann man das Zeichen `|` wieder auf die normale Bedeutung zurücksetzen.

```
\usepackage{shortvrb}
\MakeShortVerb{\|}
Mit |\MakeShortVerb| werden Quelldateien
leichter lesbar.
\DeleteShortVerb{\|}\MakeShortVerb{+}
Und mit der Deklaration +\DeleteShortVerb{\|}+
kann man das Zeichen ++ wieder auf die
normale Bedeutung zurücksetzen.
```

Bsp.
3-4-2

Die Variante `\MakeShortVerb*` führt dieselbe Abkürzung für den Befehl `\verb*` ein. Dies wird im nächsten Beispiel gezeigt.

Anstelle von `\verb*` kann man jetzt `\verb++` schreiben.

```
\usepackage{shortvrb} \MakeShortVerb*{++}
Anstelle von \verb*/ / kann man jetzt ++ schreiben.
```

Bsp.
3-4-3

Das Paket `verbatim` (von Rainer Schöpf) reimplementiert die $\LaTeX$ -Umgebungen `verbatim` und `verbatim*`. Einer seiner größten Vorteile ist, dass es unformatierte Texte beliebiger Länge zulässt, was mit den gleichnamigen Standardumgebungen von $\LaTeX$ nicht möglich ist. Außerdem stellt es die Umgebung `comment` zur Verfügung, die jeglichen Text zwischen den Befehlen `\begin{comment}` und `\end{comment}` übergeht. Weiterhin bietet das Paket die Möglichkeit, eigene Erweiterungen zu implementieren, so dass man angepasste, Verbatim-ähnliche Umgebungen definieren kann.

Ein paar dieser Erweiterungen werden im Paket `moreverb` (von Angus Duggan) realisiert. Es bietet einige interessante Verbatim-ähnliche Befehle zum Schreiben und Lesen von Dateien sowie verschiedene Umgebungen zum Erzeugen von Auflistungen und zum Setzen von Tabulatoren. Alle diese Erweiterungen sind auch in konsistenter Art und Weise mit dem `fancyvrb`-Paket verfügbar, deshalb folgt an dieser Stelle nur ein einziges Beispiel, das einen Eindruck der Syntax vermittelt, die das Paket `moreverb` verwendet.

Text vor der Listenumgebung.

Die `listing`-Umgebung nummeriert
4 die in ihr enthaltenen Zeilen.
Sie hat ein optionales Argument,
6 welches den Zähler schritt zwischen
nummerierten Zeilen angibt (die
8 Zeile mit der Nummer 1 wird immer
nummeriert), und ein
10 obligatorisches Argument -- die
Nummer der Startzeile. Die
12 Sternform macht Leerzeichen
sichtbar.

Text zwischen Listenumgebungen.

14 Die `listingcont`-Umgebung fährt da
fort, wo die vorige
16 `listing`-Umgebung aufgehört hat.
Beide Umgebungen setzen Tabstopps
18 mit dem voreingestellten Abstand
von 8 Leerzeichen ein.

Text nach Listenumgebungen.

```
\usepackage{verbatim,moreverb}
```

Text vor der Listenumgebung.

```
\begin{listing*}[2]{3}
```

Die `listing`-Umgebung nummeriert
die in ihr enthaltenen Zeilen.
Sie hat ein optionales Argument,
welches den Zähler schritt zwischen
nummerierten Zeilen angibt (die
Zeile mit der Nummer 1 wird immer
nummeriert), und ein
obligatorisches Argument -- die
Nummer der Startzeile. Die
Sternform macht Leerzeichen
sichtbar.

```
\end{listing*}
```

Text zwischen Listenumgebungen.

```
\begin{listingcont}
```

Die `listingcont`-Umgebung fährt da
fort, wo die vorige
`listing`-Umgebung aufgehört hat.
Beide Umgebungen setzen Tabstopps
mit dem voreingestellten Abstand
von 8 Leerzeichen ein.

```
\end{listingcont}
```

Text nach Listenumgebungen.

Bsp.
3-4-4

3.4.2 upquote – Anführungsstriche in Programm listings

Die Schrift Computer Modern Typewriter, die standardmäßig zum Formatieren von „Verbatim“, d.h. der wortgetreuen Wiedergabe, von Texten verwendet wird, ist eine sehr gut lesbare dicktengleiche Schrift. Durch ihre kleine Lauflänge eignet sie sich gut für Computerprogramme und ähnliche Texte. Ein Vergleich dieser Schrift mit anderen dicktengleichen Schriften ist in Abschnitt 7.7.4 aufgeführt.

Ein potentielles Problem gibt es jedoch, wenn diese Schrift zur Darstellung von Programmlistings und Ähnlichem verwendet wird: Die meisten Leute erwarten, dass rechte Anführungsstriche durch einen senkrechten Anführungsstrich (d.h. ') und linke oder rückwärtige Anführungsstriche durch ein Grave-Akzentzeichen (d.h. `) dargestellt werden. Die Schrift Computer Modern Typewriter zeigt jedoch echte linke und rechte geschweifte Anführungsstriche an (wie man sie in einer normalen Textschrift erwarten würde). Tatsächlich verhalten sich auch die meisten anderen Typewriter-Schriften so, wenn sie mit L^AT_EX verwendet werden. Dies führt zu etwas ungewöhnlichen Ergebnissen, die manche Leute schwer verstehen können. Das folgende Beispiel zeigt das Standardverhalten von drei bekannten Typewriter-Schriften: LuxiMono, Courier und Computer Modern Typewriter.

```

\usepackage[scaled=0.85]{luximono}
\raggedright
\verb+TEST='ls -l |awk '{print $3}'+
\par \renewcommand\ttdefault{pcr}
\verb+TEST='ls -l |awk '{print $3}'+
\par \renewcommand\ttdefault{cmtt}
\verb+TEST='ls -l |awk '{print $3}'+

```

TEST='ls -l |awk '{print \$3}''
TEST='ls -l |awk '{print \$3}''
TEST='ls -l |awk '{print \$3}''

Bsp.
3-4-5

Dieses Verhalten lässt sich ändern, indem man das Paket upquote (von Michael Covington) lädt, das in \verb oder der verbatim-Umgebung anstelle der üblichen geschweiften linken und rechten Anführungsstriche die Zeichen \textasciigrave und \textquotesingle aus dem textcomp-Paket verwendet. In normaler Schreibmaschinenschrift können trotzdem noch geschweifte Anführungsstriche verwendet werden, vgl. hierzu den englischen Text in der letzten Zeile des Beispiels.

```

\usepackage[scaled=0.85]{luximono}
\usepackage{upquote}
\raggedright
\verb+TEST='ls -l |awk '{print $3}'+
\par \renewcommand\ttdefault{pcr}
\verb+TEST='ls -l |awk '{print $3}'+
\par \renewcommand\ttdefault{cmtt}
\verb+TEST='ls -l |awk '{print $3}'+
\par \texttt{but 'text' is unaffected!}

```

TEST='ls -l |awk '{print \$3}''
TEST='ls -l |awk '{print \$3}''
TEST='ls -l |awk '{print \$3}''
but 'text' is unaffected!

Bsp.
3-4-6

Wie dieses Kapitel zeigt, arbeitet das Paket sehr gut mit den „Verbatim“-Erweiterungen zusammen. Einzige Ausnahme bildet das listings-Paket, mit dessen Scan-Mechanismus es kollidiert. Wenn diese Art der Anführungsstriche trotzdem mit listings verwendet werden soll, können sie mit dem Schlüsselwort upquote des Befehls \lstset aktiviert werden.

```

\usepackage{textcomp}
\usepackage{listings} \lstset{upquote}
\begin{lstlisting}[language=ksh]
TEST='ls -l |awk '{ print $3 }''
\end{lstlisting}

```

TEST='ls -l |awk '{ print \$3 }''

Bsp.
3-4-7

3.4.3 fancyvrb – Anpassungsfähige Verbatim-Umgebungen

Das Paket `fancyvrb` von Timothy Van Zandt (das inzwischen von Denis Girou und Sebastian Rahtz gepflegt wird) bietet eine Reihe von sehr anpassbaren Umgebungen und Befehle, mit denen wortwörtliche Ausgaben formatiert und verändert werden können.

Die Arbeitsweise des Paketes basiert auf einer zeilenweisen syntaktischen Analyse der Umgebung oder der Datei (das Konzept stammt ursprünglich vom `verbatim`-Paket), wodurch Zeilen auf unterschiedliche Weise aufbereitet werden können. Durch Integration verschiedener Funktionen aus anderen Paketen bietet es eine echte universelle Produktionsumgebung mit gemeinsamen Syntaxregeln.

Die wichtigste Umgebung, die das Paket bereitstellt, ist die Umgebung `Verbatim`, die, wenn sie ohne Anpassungen benutzt wird, L^AT_EXs Standardumgebung `verbatim` ähnelt. Der Hauptunterschied besteht darin, dass sie ein optionales Argument besitzt, dem in Form einer Schlüssel-Wert-Syntax Informationen zur Anpassung der Umgebung mitgegeben werden können. Es gibt jedoch eine Einschränkung, die man nicht vergessen sollte: Die linke Klammer des optionalen Argumentes muss in derselben Zeile stehen wie `\begin`. Ansonsten wird das optionale Argument nicht erkannt und stattdessen wortwörtlich ausgegeben.

Es sind mehr als 30 Schlüsselwörter verfügbar, deren Verwendung und mögliche Werte im Folgenden ausführlich besprochen werden.

Gegen Ende dieses Abschnitts werden eine Reihe von alternativen Umgebungen und Befehlen erörtert. Auch sie können über das Schlüssel-Wert-Verfahren verändert werden. Zum Schluss wird auf die Möglichkeit eingegangen, eigene Varianten zu definieren.

Schlüsselwörter zur Anpassung der Formatierung

Zur Änderung der Fonts, die von den Verbatim-Umgebungen des Paketes `fancyvrb` verwendet werden, stehen vier Schlüsselwörter zur Verfügung, die den vier Achsen des NFSS entsprechen. Das Schlüsselwort `fontfamily` gibt die zu verwendende Schriftfamilie an. Voreingestellt ist `Computer Modern Typewriter`, so dass sich die Umgebungen ohne Verwendung von Schlüsselwörtern ähnlich der Standardumgebung `verbatim` von L^AT_EX verhalten. Der Wert des Schlüsselwortes kann jedoch jede Schriftfamilie in NFSS-Schreibweise sein, z.B. `pcr` für `Courier` oder `cmss` für `Computer Modern Sans`, auch wenn letztere keine dicktengleiche Schrift ist, wie sie normalerweise für eine wortwörtliche Ausgabe verwendet würde. Das Schlüsselwort erkennt auch die speziellen Werte `tt`, `courier` und `helvetica` und transferiert sie intern in die NFSS-Nomenklatur.

Die Formatierung von wortwörtlichen Ausgaben umfasst auch Sonderzeichen, wie „\“; hier ist darauf zu achten, dass solche Zeichen im gewählten Zeichensatz tatsächlich vorhanden sind. Dies sollte kein Problem darstellen, wenn eine Fontkodierung wie T1 verwendet wird (diese kann man mithilfe des Paketes `fontenc` laden). Bei der Standardfontkodierung OT1 von L^AT_EX ist dies jedoch nicht so einfach, denn in dieser Kodierung enthalten nur ein paar dicktengleiche Schriften, wie die voreingestellte Typewriter-Schrift, alle diese

Sonderzeichen. Welche Form von falscher Ausgabe erzeugt werden könnte, zeigt die zweite Zeile im nächsten Beispiel.

```

\usepackage{fancyvrb}
\usepackage[OT1,T1]{fontenc}
\fontencoding{OT1}\selectfont
\begin{Verbatim}[fontfamily=tt]
'tt' funktioniert in OT1: \sum_{i=1}^n
\end{Verbatim}
\begin{Verbatim}[fontfamily=helvetica]
'Helvetica' geht in OT1 nicht: \sum_{i=1}^n
\end{Verbatim}
\fontencoding{T1}\selectfont
\begin{Verbatim}[fontfamily=helvetica]
... aber in T1 schon: \sum_{i=1}^n
\end{Verbatim}

```

'tt' funktioniert in OT1: $\sum_{i=1}^n$

'Helvetica' geht in OT1 nicht: "sum"-i=1ⁿ

... aber in T1 schon: $\sum_{i=1}^n$

Bsp.
3-4-8

Da alle Beispiele in diesem Buch mit der T1-Kodierung formatiert sind, wird dieses Problem nicht an anderen Stellen im Buch auftreten. Trotzdem sollte man sich dieser Gefahr bewusst sein. Sie ist ein weiterer guter Grund, um anstelle von T_EXs Original-Fontkodierung T1 zu verwenden; eine ausführliche Erörterung gibt Abschnitt 7.2.4 auf Seite 347 wieder.

Die anderen drei Schlüsselwörter, die sich auf die Fonteinstellungen beziehen, sind `fontseries`, `fontshape` und `fontsize`. Sie erben die aktuellen NFSS-Einstellungen aus dem umgebenden Text, wenn nichts anderes angegeben ist. Die ersten beiden erwarten Werte, die an `\fontseries` bzw. `\fontshape` weitergereicht werden können (z.B. `bx` für eine fette Schriftserie mit großer Laufweite oder `it` für einen *kursiven* Schriftschnitt), während `fontsize` sich anders verhält. Es erwartet als Wert einen der höheren NFSS-Befehle für die Angabe der Schriftgröße - z.B. `\small`. Wenn das Paket `relsize` verfügbar ist, kann alternativ eine Schriftgrößenänderung relativ zur aktuellen Schriftgröße des Textes als Wert verwendet werden, wie etwa `\relsize{-2}` im folgenden Beispiel.

```

\usepackage{relsize,fancyvrb}
\begin{Verbatim}[fontsize=\relsize{-2}]
\sum_{i=1}^n
\end{Verbatim}
\begin{Verbatim}[fontshape=sl,fontsize=\Large]
\sum_{i=1}^n
\end{Verbatim}

```

\sum_{i=1}^n

Eine Textzeile für die Darstellung des Textkörpers.

\sum_{i=1}^n

Bsp.
3-4-9

Eine allgemeinere Form, die Formatierung anzupassen, bietet das Schlüsselwort `formatcom`, das jeglichen L^AT_EX-Code akzeptiert und diesen am Anfang der Umgebung ausführt. Um beispielsweise die wortwörtliche Ausgabe einzufärben, könnte man ihm einen Wert wie `\color{blue}` zuweisen. Es ist auch möglich, jede Textzeile separat zu bearbeiten, indem der Befehl `\FancyVerbFormatLine` in geeigneter Weise neu definiert wird. Dieser Befehl wird für jede Zeile ausgeführt, wobei er den Text aus der Zeile als Argument

erhält. Im nächsten Beispiel wird jede zweite Zeile blau eingefärbt. Dieses Ergebnis resultiert aus der Überprüfung des Zählers `FancyVerbLine`, der von der Umgebung automatisch bereitgestellt wird und die aktuelle Zeilennummer enthält.

Diese Zeile sollte blau werden,
während diese schwarz ist. Und
er sieht man, dass gobble nicht
r Leerzeichen löscht.

```
\usepackage{ifthen,color,fancyvrb}
\renewcommand\FancyVerbFormatLine[1]
  {\ifthenelse{\isodd{\value{FancyVerbLine}}}%
    {\textcolor{blue}{#1}}{#1}}
\begin{Verbatim}[gobble=2]
  Diese Zeile sollte blau werden,
    während diese schwarz ist. Und
hier sieht man, dass gobble nicht
nur Leerzeichen löscht.
\end{Verbatim}
```

Bsp.
3-4-10

Wie das vorangegangene Beispiel zeigt, kann das Schlüsselwort `gobble` verwendet werden, um (bis zu neun) Zeichen oder Leerzeichen am Anfang jeder Zeile zu entfernen. Dies ist dann sehr nützlich, wenn alle Zeilen in den Umgebungen eingezogen sind und man die zusätzlichen Leerzeichen löschen möchte, die durch den Einzug entstanden sind. Manchmal ist das Gegenteil erwünscht: Jede Zeile sollte um einen bestimmten Betrag eingezogen werden. Zum Beispiel sind alle `Verbatim`-Umgebungen in diesem Buch um 24pt eingerückt. Dieser Einzug wird von dem Schlüsselwort `xleftmargin` gesteuert. Daneben gibt es das Schlüsselwort `xrightmargin`, das den rechten Einzug festlegt. Es wird allerdings nur selten benötigt, weil die Zeilen von wortwörtlichen Ausgaben nicht umbrochen werden. Ihre visuelle Auswirkung lässt sich vermutlich nur anhand von „overfull box“-Meldungen¹ nachvollziehen (es sei denn, man verwendet Rahmen, wie unten erörtert). Diese zeigen an, dass der Text in den rechten Rand hineinläuft. Möglicherweise nützlicher ist das boolesche Schlüsselwort `resetmargins`, welches festlegt, ob existierende Einzüge von äußeren Umgebungen ignoriert werden sollen.

- Normaler Einzug links:

Wortwörtliche Ausgabe!

- Auf beiden Seiten kein Einzug:

Wortwörtliche Ausgabe!

```
\usepackage{fancyvrb}
\begin{itemize} \item Normaler Einzug links:
\begin{Verbatim}[frame=single,xrightmargin=1pc]
Wortwörtliche Ausgabe!
\end{Verbatim}
\item Auf beiden Seiten kein Einzug:
\begin{Verbatim}[resetmargins=true,
  frame=single]
Wortwörtliche Ausgabe!
\end{Verbatim}
\end{itemize}
```

Bsp.
3-4-11

Das vorangegangene Beispiel demonstriert eine typische Verwendung des Schlüsselwortes `frame`: einen Rahmen um die wortwörtliche Ausgabe ziehen.

¹Ob überlaufende Zeilen innerhalb einer `Verbatim`-Umgebung angezeigt werden, wird vom Schlüsselwort `hfuzz` gesteuert, das den Standardwert 2pt hat. Eine Warnung wird nur ausgegeben, wenn die Boxen um einen größeren Betrag in den Rand hineinragen, als den, der als Wert des Schlüsselwortes angegeben ist.

Mit weiteren Werten lassen sich andere Rahmen erzeugen. Voreingestellt ist `none`, d.h.: kein Rahmen. `topline`, `bottomline` oder `leftline` erzeugen jeweils eine einzelne Linie auf der angegebenen Seite.¹ Der Wert `lines` erzeugt oben und unten eine Linie und `single` zeichnet, wie schon Beispiel 3-4-11 zeigte, einen vollständigen Rahmen. In allen Fällen kann die Dicke der Linien durch Angabe eines Wertes im Schlüsselwort `framerule` geändert werden (voreingestellt ist 0.4pt). Der Abstand zwischen Linien und Text ist über `framesep` steuerbar (voreingestellt ist der aktuelle Wert von `\fboxsep`).

Wenn das `color`-Paket verfügbar ist, können die Linien eingefärbt werden. Dazu wird das Schlüsselwort `rulecolor` benötigt (Standardeinstellung: schwarz). Bei Verwendung eines kompletten Rahmens kann auch der Freiraum zwischen Text und Rahmen eingefärbt werden, und zwar mit `fillcolor`.

Gerahmte Ausgabe!

```
\usepackage{color,fancyvrb}
\begin{Verbatim}[frame=single,rulecolor=\color{blue},
    framerule=3pt,framesep=1pc,fillcolor=\color{yellow}]
Gerahmte Ausgabe!
\end{Verbatim}
```

Bsp.
3-4-12

Leider gibt es keine direkte Methode, um den gesamten Hintergrund einzufärben. Die größte Näherung erreicht man, indem man `\colorbox` innerhalb von `\FancyVerbFormatLine` verwendet. Dieser Ansatz hinterlässt jedoch schmale weiße Linien zwischen den Zeilen und führt auch zu unterschiedlich breiten farbigen Blöcken – es sei denn, dass eine gleiche Zeilenbreite erzwungen wird, z.B. mit `\makebox`.

Wortwörtliche Ausgabe mit
Hintergrundfarbe.

```
\usepackage{color,fancyvrb}
\renewcommand\FancyVerbFormatLine[1]
    {\colorbox{green}{#1}}
\begin{Verbatim}
Wortwörtliche Ausgabe mit
Hintergrundfarbe.
\end{Verbatim}
\renewcommand\FancyVerbFormatLine[1]
    {\colorbox{yellow}{\makebox[\linewidth][1]{#1}}}
\begin{Verbatim}
Wortwörtliche Ausgabe mit
Hintergrundfarbe.
\end{Verbatim}
```

Wortwörtliche Ausgabe mit
Hintergrundfarbe.

Bsp.
3-4-13

Man kann Text als Teil eines Rahmens formatieren, indem man ihn als Wert des Schlüsselwortes `label` angibt. Wenn dieser Text Sonderzeichen enthält, wie etwa Klammern, Gleichheitszeichen oder Kommas, müssen diese maskiert werden, indem sie in ein Klammerpaar eingefasst werden. Ansonsten werden sie fälschlicherweise als Teil der Syntax interpretiert. Der Text erscheint standardmäßig am oberen Rand, wird aber nur ausgedruckt, wenn die Einstellung für den Rahmen an dieser Stelle eine Linie erzeugt. Andere Positionen können mithilfe von `labelposition` festgelegt werden, welches den

¹Es gibt keinen Wert, um eine Linie auf der rechten Seite zu erzeugen.

Wert `none`, `topline`, `bottomline` oder `all` annehmen kann. In letzterem Fall wird der Rahmentext oberhalb und unterhalb ausgegeben. Wenn der Rahmentext ungewöhnlich lang ist, muss der Abstand zwischen Rahmen und wortwörtlicher Ausgabe möglicherweise vergrößert werden, dies geschieht mithilfe des Schlüsselwortes `framesep`. Eine vorangegangene Einstellung für den Text kann mit dem Wert `none` gelöscht werden. Soll der Rahmentext „none“ ausgegeben werden, muss dieser in Klammern eingfasst werden.

Bsp.
3-4-14

Gerahmte Ausgabe
Beispielcode

```
\usepackage{fancyvrb}
\begin{Verbatim}[frame=single,label=\fbox{Beispielcode},
framesep=5mm,labelposition=bottomline]
Gerahmte Ausgabe
\end{Verbatim}
```

Es ist sogar möglich, verschiedene Texte für oben und unten zu definieren. Dazu muss der Text für die obere Position in eckige Klammern eingfasst werden, wie das nächste Beispiel zeigt. Damit diese Anordnung klappt, muss man `frame` entweder auf `single` oder auf `lines` setzen.

Bsp.
3-4-15

—— Anfang des Codes ——
Eine Zeile Programmcode
—— Ende des Codes ——

```
\usepackage{fancyvrb}
\begin{Verbatim}[frame=lines,framesep=5mm,
label={{[Anfang des Codes]Ende des Codes}}]
Eine Zeile Programmcode
\end{Verbatim}
```

Standardmäßig kann die fertig gestellte Ausgabe von `Verbatim`-Umgebungen am Seitenende von $\text{\LaTeX}$ umbrochen werden, wenn sie nicht auf eine einzelne Seite passt. Dies gilt auch in den Fällen, in denen der Text eingerahmt ist. Um einen Seitenumbruch zu verhindern, muss das boolesche Schlüsselwort `samepage` auf `true` gesetzt werden.

Der vertikale Abstand zwischen Zeilen in einer `Verbatim`-Umgebung entspricht dem von normalem Text. Bei Bedarf kann er mithilfe des Schlüsselwortes `baselinestretch` vergrößert werden. Zeilen zu stauchen, so dass sie überlappen, ist nicht möglich. Den Standardzeilenvorschub erhält man wieder, indem man als Wert die Zeichenfolge `auto` angibt.

Bsp.
3-4-16

Dieser Text zeigt etwa doppelten
Zeilenvorschub. Siehe auch die
Besprechung des `setspace`-Paketes an
anderer Stelle.

```
\usepackage{fancyvrb}
\begin{Verbatim}[baselinestretch=1.6]
Dieser Text zeigt etwa doppelten
Zeilenvorschub. Siehe auch die
Besprechung des setspace-Paketes an
anderer Stelle.
\end{Verbatim}
```

Bei der Darstellung von Computerlistings ist es häufig hilfreich, einige oder alle Zeilen zu nummerieren. Hier hilft das Schlüsselwort `numbers`, das den Wert `none`, `left` oder `right` annehmen kann, wodurch die Position der Nummern festgelegt wird. Der Abstand zwischen der Nummer und der wortwörtlichen Ausgabe beträgt standardmäßig 12pt, er kann aber über das

Schlüsselwort `numbersep` geändert werden. In der Regel beginnt die Nummerierung mit jeder neuen Umgebung wieder bei 1. Man kann jedoch beim Schlüsselwort `firstnumber` explizit einen beliebigen ganzzahligen Startwert angeben, der sogar negativ sein darf. Alternativ kann in diesem Schlüsselwort `last` angegeben werden, welches festlegt, dass die Nummerierung dort fortgesetzt werden soll, wo sie in der vorhergehenden `Verbatim`-Umgebung aufgehört hat.

<pre> 1 Zeilen können entweder links 2 oder rechts nummeriert werden. Text dazwischen... 3 Wie hier zu sehen, kann die 4 Nummerierung fortgesetzt werden.</pre>	<pre> \usepackage{fancyvrb} \begin{Verbatim}[numbers=left,numbersep=6pt] Zeilen können entweder links oder rechts nummeriert werden. \end{Verbatim} Text dazwischen\ldots \begin{Verbatim}[numbers=left, firstnumber=last] Wie hier zu sehen, kann die Nummerierung fortgesetzt werden. \end{Verbatim}</pre>

Bsp.
3-4-17

Manche ziehen es vor, nur einige Zeilen zu nummerieren. Auf diese Wünsche geht das Paket mit dem Schlüsselwort `stepnumber` ein. Wenn diesem Schlüsselwort eine positive ganze Zahl zugewiesen wird, werden nur jene Zeilennummern ausgedruckt, die ein Vielfaches dieser Zahl sind. Wie bereits erwähnt, heißt der Zähler, der intern zur Zählung von Zeilen verwendet wird, `FancyVerbLine`. Es überrascht daher wenig, dass das Erscheinungsbild der Nummern durch den Befehl `\theFancyVerbLine` gesteuert wird. Durch Veränderung dieses Befehls können Spezialeffekte erzielt werden. Eine Möglichkeit, bei der die aktuelle Kapitelnummer vorangestellt wird, zeigt das nächste Beispiel. Es demonstriert auch die Verwendung des booleschen Schlüsselwortes `numberblanklines`, das festlegt, ob Leerzeilen nummeriert werden (vorangestellt ist `false`, d.h. sie werden nicht nummeriert).

<pre> 3.2 Normalerweise haben Leerzeilen in einer Verbatim-Umgebung keine Nummern -- hier haben sie welche! 3.4 3.6 Zugegeben, die mit FancyVerbLine hinzugefügten Gliederungsnummern sehen etwas seltsam aus, wenn man 3.8 "'stepnumber'" verwendet.</pre>	<pre> \usepackage{fancyvrb} \renewcommand\theFancyVerbLine {\footnotesize \thechapter.\arabic{FancyVerbLine}} \begin{Verbatim}[numbers=left,stepnumber=2, numberblanklines=true] Normalerweise haben Leerzeilen in einer Verbatim-Umgebung keine Nummern -- hier haben sie welche! Zugegeben, die mit FancyVerbLine hinzugefügten Gliederungsnummern sehen etwas seltsam aus, wenn man "'stepnumber'" verwendet. \end{Verbatim}</pre>

Bsp.
3-4-18

In manchen Situationen ist es hilfreich, Leerräume zu kennzeichnen, indem alle Leerzeichen als `\_` dargestellt werden. Dies lässt sich mit dem boole-

schen Schlüsselwort `showspaces` oder alternativ mit der `Verbatim*`-Variante der Umgebung erreichen.

Ein weiteres Zeichen zur Darstellung von Leerräumen, nämlich der Tabulator, spielt in einigen Programmiersprachen eine wichtige Rolle, so dass es manchmal notwendig sein kann, diesen zu kennzeichnen. Hierzu steht das boolesche Schlüsselwort `showtabs` zur Verfügung. Das dargestellte Tabzeichen wird durch den Befehl `\FancyVerbTab` definiert und kann, wie unten gezeigt, neu definiert werden. Standardmäßig entspricht ein Tabstopp acht Leerzeichen. Dieser Wert kann mit dem Schlüsselwort `tabsize` verändert werden. Weist man dem booleschen Schlüsselwort `obeytabs` den Wert `true` zu, dann erzeugt der Tabstopp jedoch genau die benötigte Anzahl an Leerzeichen, um auf das nächste ganzzahlige Vielfache von `tabsize` zu gelangen. Im nächsten Beispiel enthalten alle Zeilen Tabstopps, die in der Eingabe als Leerzeichen mit einem voreingestellten `tabsize`-Wert von 8 dargestellt werden. Man beachte vor allem in der letzten Zeile den Unterschied zwischen der Ein- und Ausgabe.

```
\usepackage{fancyvrb}
\begin{Verbatim}[showtabs=true]
123456789012345678901234567890
Zwei      Standard      Tabulatoren
\end{Verbatim}
\begin{Verbatim}[obeytabs=true,showtabs=true]
Zwei      echte      Tabulatoren
\end{Verbatim}
\renewcommand\FancyVerbTab{$\triangleright$}
\begin{Verbatim}[obeytabs=true,showtabs=true]
Zwei      neue      Tabulatoren
\end{Verbatim}
\begin{Verbatim}[obeytabs=true,tabsize=3,showtabs=true]
Und       eine       spezielle Tab   einstellung
\end{Verbatim}
```

```
123456789012345678901234567890
Zwei      ↪Standard      ↪Tabulatoren

Zwei      ↪echte      ↪Tabulatoren

Zwei      ▷neue      ▷Tabulatoren

Und      ▷eine ▷spezielle Tab ▷einstellung
```

Bsp.
3-4-19

Wenn innerhalb der wortwörtlichen Ausgabe Befehle ausgeführt werden sollen, müssen diese durch ein spezielles Zeichen markiert werden (um den Beginn eines Befehls zu kennzeichnen). Zwei weitere Zeichen werden als Begrenzungssymbole für Argumente benötigt (d.h. sie übernehmen die Funktion, die in $\text{\LaTeX}$ normalerweise geschweifte Klammern haben). Solche speziellen Zeichen können mit dem Schlüsselwort `commandchars` definiert werden (siehe unten); allerdings dürfen diese Zeichen dann natürlich nicht in der

wortwörtlichen Ausgabe vorkommen. Die Zeichen werden durch Voranstellen eines Backslashes markiert. Damit wird jegliche Bedeutung, die sie in $\text{\LaTeX}$ möglicherweise haben, maskiert. Mit dem Schlüsselwort `commentchar` kann ein Kommentarzeichen definiert werden, was dazu führt, dass alles bis zum nächsten Zeilenumbruch ignoriert wird. Wenn dieses Zeichen beispielsweise in der Mitte einer Zeile eingefügt wird, werden diese und die nächste Zeile zusammengefasst. Die Einstellung für `commandchars` oder `commentchar` kann mithilfe des Schlüsselwortes „none“ aufgehoben werden.

```
\usepackage{fancyvrb}
\begin{Verbatim}[commandchars=\\\[\],commentchar=!]
Text kann |emph[hervorgehoben] werden.
! siehe oben (diese Zeile ist unsichtbar)
Eine Zeile mit Label|label[linea] ! entfernt Zeilenvorschub
ist hier zu sehen.
\end{Verbatim}
In Zeile~\ref{linea} sieht man\ldots
```

Text kann *hervorgehoben* werden.

Eine Zeile mit Label ist hier zu sehen.

In Zeile 2 sieht man...

Bsp.
3-4-20

Wenn man wie im vorangegangenen Beispiel innerhalb der `Verbatim`-Umgebung den Befehl `\label` verwendet, bezieht sich dieser auf die interne Zeilennummer, ungeachtet dessen, ob die Nummer angezeigt wird oder nicht. Hierzu muss das Schlüsselwort `commandchars` angegeben werden, was manchmal ungünstig sein kann, weil dadurch die gewählten Zeichen nicht mehr in der wortwörtlichen Ausgabe verwendet werden können.

Es gibt zwei weitere Schlüsselwörter, mit denen die Art und Weise verändert werden kann, wie der Inhalt einer `Verbatim`-Umgebung analysiert und manipuliert wird: `codes` und `defineactive`. Mit diesen beiden Schlüsselwörtern sind einige nette Tricks möglich, ihr Einsatz ist jedoch nicht so einfach zu erklären. Man braucht ein gutes Verständnis für die Funktionsweise von $\text{\TeX}$. Eine gute Beschreibung liefert die Dokumentation des `fancyvrb`-Paketes.

Teilweise Darstellung der Eingabe

Normalerweise werden innerhalb der `Verbatim`-Umgebung alle Zeilen gesetzt. Man kann jedoch die Ausgabe der Zeilen mit einer Reihe von Schlüsselwörtern begrenzen. `firstline` und `lastline` definieren die erste und, sofern notwendig, letzte darzustellende Zeile. Alternativ können auch bestimmte Zeichenketten innerhalb der Umgebung als Anfang und Ende definiert werden, was dazu führt, dass alle Zeilen zwischen diesen (aber diesmal *exklusive* der angegebenen Zeilen) ausgegeben werden. Die Zeichenketten werden in den Makros `\FancyVerbStartString` und `\FancyVerbStopString` angegeben. Hier muss man jedoch aufpassen: Die Makros müssen mit den Befehlen `\newcommand*` bzw. `\renewcommand*` (re)definiert werden. Die Verwendung

von `\newcommand` funktioniert *nicht*! Solch eine Deklaration aufzuheben, ist sogar noch komplizierter, sie muss mittels `\let` rückgängig gemacht werden:

```
\let\FancyVerbStartString\relax
```

oder aber lokal begrenzt sein, so dass $\text{\LaTeX}$ die Definition am Ende der Klammernguppe automatisch rückgängig macht – alles andere funktioniert nicht.

```
\usepackage{fancyvrb}
\newcommand*\FancyVerbStartString{START}
\newcommand*\FancyVerbStopString{STOP}
\begin{Verbatim}
  Eine nicht dargestellte wortwörtliche Ausgabe.
START
  Nur diese Zeile wird angezeigt.
STOP
  Aber der Rest wird ausgelassen.
\end{Verbatim}
```

Bsp.
3-4-21

Nur diese Zeile wird angezeigt.

Man mag sich fragen, warum es solch eine Funktion überhaupt gibt, denn man könnte die Zeilen, die nicht gesetzt werden, doch einfach weglassen. In einer Umgebung wie `Verbatim` sind die Einsatzmöglichkeiten tatsächlich begrenzt. Bei Verwendung mit anderen Funktionen des Paketes, die Daten in Dateien schreiben und sie wieder auslesen, bietet diese Funktion jedoch eine hervorragende Lösung für ansonsten unlösbare Probleme.

So nutzen etwa alle Beispiele in diesem Buch dieses Verfahren. Das Beispiel wird zusammen mit einer Dokumentenpräambel und weiterem Material in eine Datei geschrieben, so dass diese Datei ein durch $\text{\LaTeX}$ verarbeitbares Dokument wird. Dieses Dokument wird anschließend extern formatiert und das Ergebnis als EPS-Graphik in das Buch eingebunden. An seiner Seite wird der Beispielcode dargestellt, indem die generierte Datei noch einmal eingelesen wird, wobei jedoch nur jene Zeilen dargestellt werden, die sich zwischen den Zeichenketten `\begin{document}` und `\end{document}` befinden. Dies entspricht den Beispielzeilen, die in schwarz gesetzt sind. Der blau dargestellte Präambelteil wird auf ähnliche Weise erzeugt. Hierfür werden die Start- und Ende-Zeichenketten neu definiert, so dass nur die Zeilen zwischen den Zeichenketten `\StartShownPreambleCommands` und `\StopShownPreambleCommands` berücksichtigt werden. Bei der externen Verarbeitung dieses Beispiels werden diese beiden Befehle einfach ignoriert, d.h. sie werden von der im Beispiel verwendeten Klasse (die ansonsten der Dokumentenklasse `article` ähnelt) so definiert, dass sie nichts tun. Damit entspricht der Beispielcode (glücklicher oder unglücklicher Weise) immer dem angezeigten Ergebnis.¹

*Wie die Beispiele im
Buch erzeugt wurden*

Um Daten wortwörtlich in eine Datei zu schreiben, kann man die Umgebung `VerbatimOut` verwenden. Sie hat ein obligatorisches Argument: den Namen der Datei, in welche die Daten geschrieben werden sollen. Es gibt jedoch ein logisches Problem, wenn man versucht, solch eine Umgebung innerhalb der Definition einer neuen Umgebung zu verwenden. Sobald

¹In der ersten Ausgabe entstanden leider eine Reihe von Fehlern dadurch, dass Code im Text angezeigt wurde, der nicht direkt verwendet wurde.

`\begin{VerbatimOut}` ausgeführt ist, wird das gesamte nachfolgende Material ohne weitere Verarbeitung eingelesen, so dass das Ende der neuen Umgebung (in der `\end{VerbatimOut}` versteckt ist) nicht erkannt wird. Als Lösung bietet das Paket `fancyvrb` den Befehl `\VerbatimEnvironment`. Wenn er innerhalb des `\begin`-Codes einer Umgebung ausgeführt wird, stellt er sicher, dass die korrespondierende Ende-Marke innerhalb „wortwörtlicher Ausgabe“ erkannt wird und ihr Code ausgeführt wird.

Um Daten wortwörtlich aus einer Datei auszulesen, kann man den Befehl `\VerbatimInput` verwenden. Er hat ein optionales Argument, das dem der Umgebung `Verbatim` ähnelt (d.h. er akzeptiert alle zuvor besprochenen Schlüsselwörter) sowie ein obligatorisches Argument, das angibt, aus welcher Datei die Daten ausgelesen werden sollen. Die Variante `\BVerbatimInput` gibt den darzustellenden Text in einer Box ohne Leerraum oben und unten aus. Das nächste Beispiel zeigt einige der Möglichkeiten: Es definiert die Umgebung `example`, die zunächst ihren Inhalt in eine Datei schreibt, die erste Zeile wieder liest und diese blau darstellt, die Datei ein zweites Mal liest, wobei sie dieses Mal in der zweiten Zeile startet, und die Zeilen beginnend bei 1 durchnummeriert. Wie oben erläutert, wird zum Erstellen der Beispiele in diesem Buch eine ähnliche, wenn auch kompliziertere Definition verwendet.

```
\usepackage{fancyvrb,color}
\newenvironment{example}
{
  \VerbatimEnvironment\begin{VerbatimOut}{test.out}
  {\end{VerbatimOut}\noindent
   \BVerbatimInput[lastline=1,%
                    formatcom=\color{blue}]{test.out}%
   \VerbatimInput[numbers=left,%
                  firstnumber=1,firstline=2]{test.out}}
\begin{example}
Eine blaue Zeile.
Zwei Zeilen
mit Nummern.
\end{example}
```

Eine blaue Zeile.

Eine blaue Zeile.

Zwei Zeilen

mit Nummern.

1 Zwei Zeilen

2 mit Nummern.

mit Nummern.

Bsp.
3-4-22

Eine Anzahl interessanter Beispielumgebungen enthält das Paket `fvr-ex` von Denis Girou, das auf den Funktionen von `fancyvrb` aufsetzt.

Umgebungs- und Befehlsvarianten

Bisher haben alle Beispiele die `Verbatim`-Umgebung verwendet. Es gibt jedoch noch eine Reihe von Varianten, die in bestimmten Situationen nützlich sind. `BVerbatim` ähnelt `Verbatim`, setzt die wortwörtliche Ausgabe aber in eine Box. Einige der oben besprochenen Schlüsselwörter (vorzugsweise jene, die sich mit Rahmen befassen) werden nicht unterstützt. Dafür gibt es zwei zusätzliche Schlüsselwörter. Das erste, `baseline`, gibt den Justierungspunkt für die Box an. Es kann die Werte `t` (für top, zu deutsch: oben), `c` (für center, zu deutsch: zentriert) und `b` (für bottom, zu deutsch: unten) annehmen. Letzterer ist der voreingestellte Wert. Das zweite Schlüsselwort `boxwidth` gibt die gewünschte Breite der Box an. Wenn es fehlt oder der angegebene Wert `auto` ist, wird die Box so breit wie die längste Zeile in der Umgebung. Auch

das schon angesprochene `\BVerbatimInput` unterstützt diese zusätzlichen Schlüsselwörter.

Bsp.
3-4-23

```

\usepackage{fancyvrb}
\begin{BVerbatim}[boxwidth=4pc,baseline=t]
erste Zeile
zweite Zeile
\end{BVerbatim}
\begin{BVerbatim}[baseline=c]
erste Zeile
zweite Zeile
\end{BVerbatim}

```

Alle Umgebungen und Befehle für den Satz von wortwörtlicher Ausgabe gibt es auch in Sternform, die, wie die Standardumgebungen von $\text{\LaTeX}$, Leerzeichen als `\_` darstellen. Mit anderen Worten: Sie setzen intern das Schlüsselwort `showspaces` auf `true`.

Definition eigener Varianten

Benutzerdefinierte Varianten von Verbatim-Befehlen und -Umgebungen zu definieren, ist recht einfach. Zunächst können die Standardeinstellungen des Paketes mithilfe des Befehls `\fvset` verändert werden. Er hat ein Argument: eine durch Kommas getrennte Liste von Schlüssel-Wert-Paaren. Diese werden auf alle Verbatim-Umgebungen oder -Befehle angewendet. Natürlich können diese neuen Standardwerte immer noch mit dem optionalen Argument des Befehls oder der Umgebung überschrieben werden. Wenn z.B. alle Verbatim-Umgebungen um zwei Leerzeichen eingerückt sind, möchte man diese vielleicht global entfernen, anstatt bei jeder Instanz `gobble` zu verwenden.

Bsp.
3-4-24

```

\usepackage{fancyvrb} \fvset{gobble=2}
\noindent Eine Textzeile, die den linken Rand zeigt.
\begin{Verbatim}
  Der neue 'Normal'-Fall.
\end{Verbatim}
Der neue 'Normal'-Fall.
\begin{Verbatim}[gobble=0]
Jetzt muss gobble bei Bedarf
explizit deaktiviert werden!
\end{Verbatim}

```

Möglicherweise ist es jedoch gar nicht erwünscht, dass `\fvset` auf alle Umgebungen und Befehle angewendet wird. Daher bietet das Paket vier Deklarationen, mit denen eigene Verbatim-Befehle und -Umgebungen definiert werden können bzw. das Verhalten von vordefinierten verändert werden kann.

<code>\CustomVerbatimEnvironment</code>	<code>{neu-umg}{basis-umg}{schlüssel-wert-liste}</code>
<code>\RecustomVerbatimEnvironment</code>	<code>{ändere-umg}{basis-umg}{schlüssel-wert-liste}</code>
<code>\CustomVerbatimCommand</code>	<code>{neu-befehl}{basis-befehl}{schlüssel-wert-liste}</code>
<code>\RecustomVerbatimCommand</code>	<code>{ändere-befehl}{basis-befehl}{schlüssel-wert-liste}</code>

Diese Deklarationen haben jeweils drei Argumente: den Namen der zu definierenden beziehungsweise zu ändernden Umgebung, den Namen der Umge-

bung, auf der diese basieren soll, und eine durch Kommas getrennte Liste von Schlüssel-Wert-Paaren, die das neue Verhalten definiert. Die Deklarationen für Befehle sind analog aufgebaut, man achte aber darauf, dass *basis-befehl* den Befehlsnamen ohne vorangestellten Backslash erwartet.

Zur Definition neuer Strukturen kann `\CustomVerbatimEnvironment` beziehungsweise `\CustomVerbatimCommand` verwendet werden. Um das Verhalten bestehender Befehle und Umgebungen (vordefinierter als auch selbstdefinierter) zu ändern, stehen `\RecustomVerbatimEnvironment` bzw. `\RecustomVerbatimCommand` zur Verfügung. Wie im folgenden Beispiel zu sehen, können die Standardwerte, die im dritten Argument gesetzt sind, wie üblich mit dem optionalen Argument überschrieben werden, wenn die Umgebung oder der Befehl instantiiert wird.

<pre> 1 Der Normalfall mit dicken 2 Linien und Nummern links. </pre>	<pre> Der Normalfall mit dicken Linien und Nummern links. </pre>	<pre> \usepackage{fancyvrb} \CustomVerbatimEnvironment{Xverbatim}{Verbatim} {numbers=left,frame=lines,framerule=2pt} \begin{Xverbatim} Der Normalfall mit dicken Linien und Nummern links. \end{Xverbatim} \begin{Xverbatim}[numbers=none,framerule=.6pt] Die Ausnahme ohne Nummern und dünneren Linien. \end{Xverbatim} \RecustomVerbatimEnvironment{Xverbatim}{Verbatim} {numbers=left,frame=none,showspaces=true} \begin{Xverbatim} Und ab hier verhält sich die Umgebung wieder anders. \end{Xverbatim} </pre>	<pre> Die Ausnahme ohne Nummern und dünneren Linien. </pre>	<pre> 1 Und ab hier verhält sich 2 die Umgebung wieder anders. </pre>	<pre> Und ab hier verhält sich die Umgebung wieder anders. </pre>	Bsp. 3-4-25

Verschiedene Funktionen

L^AT_EXs Standardbefehl `\verb` kann normalerweise nicht in Argumenten verwendet werden, weil dort die Analyse fehlschlägt und zu falschen Ergebnissen oder Fehlermeldungen führt. Eine Lösung dieses Problems besteht darin, die Daten für die wortwörtliche Ausgabe außerhalb des Argumentes zu verarbeiten, diese abzuspeichern und die analysierten Daten später an solchen gefährlichen Stellen zu verwenden. Zu diesem Zweck bietet das Paket `fancyvrb` die Befehle `\SaveVerb` und `\UseVerb`.

```

\SaveVerb [schlüssel-wert-liste] {label}=daten=
\UseVerb* [schlüssel-wert-liste] {label}

```

Der Befehl `\SaveVerb` hat ein obligatorisches Argument, ein *label*, das den Speicherbehälter angibt, in dem die analysierten Daten gesichert werden sollen. Dieses wird gefolgt von den *daten* der wortwörtlichen Ausgabe, die in zwei identische Zeichen eingeschlossen werden (in der Beispielsyntax oben =), auf dieselbe Weise, in der `\verb` sein Argument begrenzt. Um diese Daten zu verwenden, muss `\UseVerb` mit dem obligatorischen Argument *label* aufgerufen werden. Da die Daten von `\SaveVerb` zwar analysiert, aber nicht gesetzt

wurden, kann die Formatierung mithilfe einer Liste von Schlüssel-Wert-Paaren oder, wie bei anderen Verbatim-Befehlen und -Umgebungen, einer Sternform geändert werden. Natürlich ist nur ein Teil der Schlüsselwörter sinnvoll, die irrelevanten werden stillschweigend ignoriert. Der Befehl `\UseVerb` ist (unnötigerweise) zerbrechlich, daher muss er mit `\protect` in bewegten Argumenten geschützt werden.

Inhaltsverzeichnis

1 Reale \Gefahr

6

1 Reale \Gefahr

Die `Reale_\Gefahr` ist nicht länger problematisch und kann beliebig oft wiederverwendet werden.

```
\usepackage{fancyvrb}
\SaveVerb{danger}=Reale \Gefahr=
```

```
\tableofcontents
```

```
\section{\protect\UseVerb{danger}}{}
```

Die `\UseVerb*{danger}` ist nicht länger problematisch und kann `\marginpar{\UseVerb[fontsize=\tiny]{danger}}` beliebig oft wiederverwendet werden.

Bsp.
3-4-26

Reale \Gefahr

Solche Speicherbehälter können wieder verwendet werden, wenn sie nicht mehr benötigt werden. Bei Verwendung von `\UseVerb` in Befehlen, die ihre Argumente über eine große Entfernung verteilen, muss man jedoch aufpassen, dass der Speicherbehälter tatsächlich noch den gewünschten Inhalt enthält, der schließlich gesetzt werden soll. Im vorangegangenen Beispiel wurde `\SaveVerb` in die Präambel eingefügt, weil die Verwendung des Speicherbehälters im `\section`-Befehl letzten Endes dazu führt, dass `\UseVerb` im Befehl `\tableofcontents` ausgeführt wird.

`\SaveVerb` besitzt außerdem ein optionales Argument, in dem eine Schlüssel-Wert-Liste eingegeben werden kann. Aber auch hier sind nur ein paar Schlüssel relevant (z.B. jene, die sich mit der Analyse befassen). Es gibt ein zusätzliches Schlüsselwort namens `aftersave`, das den Code enthält, der unmittelbar nach dem Abspeichern der wortwörtlichen Ausgabe in den Speicherbehälter ausgeführt wird. Das nächste Beispiel zeigt eine Anwendung dieses Schlüsselwortes: die Definition einer speziellen Variante des Befehls `\item`, in welchem eine wortwörtliche Ausgabe angegeben werden kann, die in einer `description`-Umgebung dargestellt wird. Außerdem unterstützt er ein optionales Argument, das aus einer Schlüssel-Wert-Liste besteht, welche die Formatierung steuert. Es lohnt sich, diese Definition genauer zu betrachten, auch wenn die Unmenge an Klammern zunächst kompliziert erscheint. Sie sind notwendig, um sicherzustellen, dass die richtigen eckigen Klammern von `\SaveVerb`, `\item` und `\UseVerb` als Argumentbegrenzer erkannt werden – das übliche Problem, dass sich eckige Klammern in $\text{\TeX}$ nicht wie geschweifte verschachteln lassen.¹ Man beachte auch die Verwendung von `\textnormal`, welcher notwendig ist, um `\bfseries` zu deaktivieren, der implizit von `\item` verwendet wird. Ansonsten hätte der Befehl `\emph` keine

¹Der Autor gesteht, dass er drei Anläufe brauchte (bis kurz vor Mitternacht), bevor dieses Beispiel funktionierte.

Wirkung, da es keinen fettkursiven Computer Modern Schriftschnitt gibt.

<code>\ddanger</code>	Gefährliches Biest; zu finden im $\text{\TeX}$ book.	<code>\usepackage{fancyvrb}</code> <code>\newcommand\vitel[1][\]{\SaveVerb[commandchars=\\<>.% aftersave={\item[\textnormal{\UseVerb[#1]{vsave}}]}{vsave}}</code>
<code>\danger</code>	Sein kleiner Bruder, immer noch gefährlich.	<code>\begin{description}</code> <code>\vitel+\ddanger+ Gefährliches Biest; zu finden im \TeX{book}.</code> <code>\vitel[fontsize=\tiny]+\danger+ Sein kleiner Bruder,</code> <code>immer noch gefährlich.</code> <code>\end{description}</code>
<code>\dddanger{arg}</code>	Der ultimative Horror.	<code>\vitel+\dddanger{\emph{arg}}+ Der ultimative Horror.</code>

Bsp.
3-4-27

Auf dieselbe Weise können komplette Verbatim-Umgebungen gespeichert werden, indem man die Umgebung `SaveVerbatim` verwendet. Sie hat ein obligatorisches Argument, in dem der Name des Speicherbehälters angegeben wird. Um die Verbatim-Umgebung zu setzen, kann man `\UseVerbatim` oder `\BUseVerbatim` (für die Ausgabe in Boxen) mit den üblichen Schlüssel-Wert-Paaren verwenden.

In Fußnoten sind Verbatim-Befehle und -umgebungen normalerweise nicht zugelassen, trotzdem benötigt man kein `\SaveVerb` oder Ähnliches, um an solchen Stellen wortwörtliche Ausgabe zu erzeugen. Stattdessen wird der Befehl `\VerbatimFootnotes` am Anfang des Dokumentes (nach der Präambel) eingefügt, und von da ab können Verbatim-Befehle in Fußnoten verwendet werden. Dies gibt es jedoch nur für Fußnoten – für andere Befehle, wie `\section`, wird immer noch das oben beschriebene, kompliziertere Verfahren mit Speicherbehälter benötigt.

Ein bisschen Text als Anlass ¹ für eine Fußnote. War dies gut genug?	<code>\usepackage{fancyvrb}</code> <code>\VerbatimFootnotes</code> Ein bisschen Text als Anlass\footnote{Hier ist der Beweis: <code>\verb=\danger{%-}=</code> für eine Fußnote. War dies gut genug?

¹Hier ist der Beweis: `\danger{%-}`

Bsp.
3-4-28

Die `fancyvrb`-Version von `\verb` heißt `\Verb`. Sie unterstützt alle anwendbaren Schlüsselwörter, die ihr wie üblich mittels eines optionalen Argumentes übergeben werden können. Das Beispiel erzeugt `\verbx` als Variante von `\Verb` mit einer speziellen Einstellung von `commandchars`, so dass Befehle innerhalb seines Argumentes ausgeführt werden können. Zu diesem Zweck muss `\CustomVerbatimCommand` verwendet werden, da `\verbx` ein neuer Befehl ist, der in Standard- $\text{\LaTeX}$ nicht verfügbar ist.

<code>\dddanger{\emph{arg}}</code> <code>\dddanger{arg}</code>	<code>\usepackage{fancyvrb}</code> <code>\CustomVerbatimCommand\verbx{Verb}{commandchars=\\<>}</code> <code>\Verb [fontfamily=courier]+\dddanger{\emph{arg}}+ \par</code> <code>\verbx[fontfamily=courier]+\dddanger{\emph{arg}}+</code>

Bsp.
3-4-29

Wie bereits erwähnt, bietet `fancyvrb` die Möglichkeit, den Anfang und das Ende einer wortwörtlichen Ausgabe mit einem bestimmten Zeichen zu kennzeichnen, ohne ihr den Befehl `\verb` voranzustellen. Der Befehl zur Deklaration dieser Begrenzungssymbole heißt `\DefineShortVerb`. Wie ande-

re fancyvrb-Befehle akzeptiert er ein optionales Argument, in dem Schlüssel-Wert-Paare spezifiziert werden können. Sie wirken sich auf die Formatierung und die Analyse aus, wobei dieses Mal die gewählten Einstellungen in den einzelnen Instanzen nicht überschrieben werden können. Alternativ kann `\fvset` verwendet werden, da dieser Befehl für alle Verbatim-Befehle und -Umgebungen in seinem Gültigkeitsbereich funktioniert. Um die spezielle Bedeutung eines Zeichens aufzuheben, das mit `\DefineShortVerb` deklariert wurde, ist `\UndefineShortVerb` zu verwenden.

Die Verwendung von `\DefineShortVerb` kann Quellen viel lesbarer machen!

Und mit `\UndefineShortVerb{\|}` erhält das Zeichen wieder seine normale Bedeutung.

```
\usepackage{fancyvrb}
```

```
\DefineShortVerb[fontsize=\tiny]{\|}
```

```
Die Verwendung von \|DefineShortVerb|
```

```
kann Quellen viel lesbarer machen! \par
```

```
\UndefineShortVerb{\|}\DefineShortVerb{\|+}
```

```
\fvset{fontfamily=courier}
```

```
Und mit +\UndefineShortVerb{\|}+ erhält
```

```
das Zeichen wieder seine normale Bedeutung.
```

Es gibt die Möglichkeit, die beliebtesten Erweiterungen oder Anpassungen in einer Datei namens `fancyvrb.cfg` zusammenzufassen. Sobald `fancyvrb` geladen ist, sucht das Paket automatisch nach dieser Datei. Für die Verwendung einer solchen Datei spricht, dass die Erweiterungen nicht mehr in alle Dokumente eingefügt werden müssen, wenn die Datei an zentraler Stelle platziert wird. Der Nachteil ist, dass die Dokumente nicht mehr portierbar sind, es sei denn, man legt die Datei bei der Verteilung der Dokumente stets bei.

3.4.4 listings – Schön gesetzter Programmcode

Eine wichtige Anwendung der wortwörtlichen Ausgabe ist die Darstellung von Programmcode. Man kann zwar einfach und erfolgreich ein Paket wie `fancyvrb` einsetzen, um diese Aufgabe zu bewältigen, es ist jedoch häufig vorteilhaft, die Darstellung durch besondere Formatierung bestimmter Programmelemente (wie Schlüsselwörter, Bezeichner und Kommentare) zu verbessern.

Hierfür gibt es zwei Hauptansätze: Entweder kennzeichnet man mit Befehlen die logischen Aspekte des Algorithmus oder der Programmiersprache, oder die Anwendung analysiert den Programmcode hinter den Kulissen. Der Vorteil des ersten Verfahrens besteht darin, dass man im Prinzip eine bessere Kontrolle über die Darstellung hat. Auf der anderen Seite ist der Programmcode mit $\TeX$ -Befehlen durchgesetzt und damit schwer zu pflegen, für die direkte Verarbeitung nicht zu gebrauchen und häufig im Quelltext schlecht lesbar. Pakete, die in diese Kategorie fallen, sind beispielsweise `alg` und `algorithmic`. Hier ein Beispiel:

```
if  $i \leq 0$  then
```

```
   $i \leftarrow 1$ 
```

```
else
```

```
  if  $i \geq 0$  then
```

```
     $i \leftarrow 0$ 
```

```
  end if
```

```
end if
```

```
\usepackage{algorithmic}
```

```
\begin{algorithmic}
```

```
\IF  $\{i \leq 0\}$  \STATE  $i \leftarrow 1$  \ELSE
```

```
\IF  $\{i \geq 0\}$  \STATE  $i \leftarrow 0$  \ENDIF
```

```
\ENDIF
```

```
\end{algorithmic}
```

Bsp.
3-4-30

Bsp.
3-4-31

ABAP (R/2 4.3, R/2 5.0, R/3 3.1, R/3 4.6C, R/3 6.10)	Haskell	Perl
ACSL	HTML	PHP
Ada (83, 95)	IDL (<i>leer</i> , CORBA)	PL/I
Algol (60, 68)	Java (<i>leer</i> , AspectJ)	POV
Assembler (x86masm)	ksh	Prolog
Awk (<i>gnu</i> , POSIX)	Lisp (<i>leer</i> , Auto)	Python
Basic (Visual)	Logo	R
C (ANSI, Objective, Sharp)	Make (<i>leer</i> , <i>gnu</i>)	Reduce
C++ (ANSI, GNU, ISO, Visual)	Mathematica (1.0, 3.0)	S (<i>leer</i> , PLUS)
Caml (<i>light</i> , Objective)	Matlab	SAS
Clean	Mercury	Scilab
Cobol (1974, 1985, ibm)	MetaPost	SHELXL
Comal 80	Miranda	Simula (67, CII, DEC, IBM)
csh	Mizar	SQL
Delphi	ML	tcl (<i>leer</i> , tk)
Eiffel	Modula-2	TeX (AlLaTeX, common, LaTeX, <i>plain</i> , primitive)
Elan	MuPAD	VBScript
erlang	NASTRAN	Verilog
Euphoria	Oberon-2	VHDL (<i>leer</i> , AMS)
Fortran (77, 90, 95)	OCL (decorative, OMG)	VRML (97)
GCL	Octave	XML
Gnuplot	Pascal (Borland6, <i>Standard</i> , XSC)	

blau gibt den Standarddialekt an

Tabelle 3.7: Von listings unterstützte Sprachen (Winter 2003)

Der zweite Ansatz wird exemplarisch am Paket listings¹ von Carsten Heinz veranschaulicht. Dieses Paket analysiert zunächst den Code, zerlegt ihn in seine Bestandteile und formatiert diese dann nach selbst definierbaren Regeln. Der Parser ist recht allgemein und kann so angepasst werden, dass er die Syntax vieler verschiedenen Sprachen erkennt (siehe Tabelle 3.7). In regelmäßigen Abständen kommen neue Sprachen hinzu. Wenn die gewünschte Zielsprache nicht aufgeführt ist, empfiehlt es sich, auf CTAN die neueste Version des Paketes zu überprüfen. Alternativ kann man die erforderlichen Deklaration auch selbst bereitstellen. Das kostet etwas Arbeit, ist aber nicht sehr schwierig.

Die Anwenderbefehle und -Umgebungen in diesem Paket haben viele Gemeinsamkeiten mit jenen in fancyvrb. Die Aspekte zur syntaktischen Analyse und zur Formatierung werden über Schlüssel-Wert-Paare gesteuert, die in einem optionalen Argument angegeben werden und die Einstellungen für das gesamte Dokument oder größerer Teile davon können über \lstset definiert werden (der entsprechende fancyvrb-Befehl heißt \fvset). Wann immer möglich, verwenden beide Pakete dieselben Schlüsselwörter, so dass Benutzer des einen Paketes recht leicht auch das andere beherrschen lernen.

Nach dem Laden des Paketes empfiehlt es sich, mit \lstloadlanguages alle im Dokument benötigten Programmiersprachen anzugeben (als durch

¹Die hier beschriebene Paketversion ist 1.0. Frühere Versionen verwendeten eine etwas andere Syntax. Falls Funktionen nicht so arbeiten wie angegeben, sollte man das Paket aktualisieren.

Kommas getrennte Liste). Solch eine Deklaration wählt keine Sprache aus, sondern lädt nur die notwendige unterstützende Information und beschleunigt die Verarbeitung.

Programmteile werden in einer `lstlisting`-Umgebung eingebettet. Die Sprache des Programmteils wird mit dem Schlüsselwort `language` angegeben. Im folgenden Beispiel wird das Schlüsselwort per `\lstset` auf `C` gesetzt und später im optionalen Argument der zweiten `lstlisting`-Umgebung überschrieben.

Eine „for“-Schleife in C:

```
int sum;
int i; /* Schleifenvariable */
sum=0;
for (i=0;i<n;i++) {
    sum += a[i];
}
```

Nun dieselbe Schleife in Ada:

```
Sum: Integer;
-- keine Deklaration für I nötig
Sum := 0;
for I in 1..N loop
    Sum := Sum + A(I);
end loop;
```

```
\usepackage{listings}
\lstloadlanguages{C,Ada}
\lstset{language=C,commentstyle=\scriptsize
        ,extendedchars=true}
```

Eine "for"-Schleife in C:

```
\begin{lstlisting}[keywordstyle=\underbar]
int sum;
int i; /* Schleifenvariable */
sum=0;
for (i=0;i<n;i++) {
    sum += a[i];
}
\end{lstlisting}
```

Nun dieselbe Schleife in Ada:

```
\begin{lstlisting}[language=Ada]
Sum: Integer;
-- keine Deklaration für I nötig
Sum := 0;
for I in 1..N loop
    Sum := Sum + A(I);
end loop;
\end{lstlisting}
```

Bsp.
3-4-32

Das Schlüsselwort `extendedchars` wird benötigt, um die Umlaute in den Kommentaren korrekt darzustellen; weitere Informationen dazu finden sich auf Seite 184. Obiges Beispiel verwendet auch das Schlüsselwort `commentstyle`, welches das generelle Erscheinungsbild von Kommentaren steuert. Das Paket erkennt die sprachabhängigen Syntaxstile für Kommentare automatisch. Es gibt weitere solcher Schlüsselwörter - u.a. `basicstyle`, welches das gesamte Erscheinungsbild des Programmlistings steuert, `stringstyle`, für Zeichenketten in der Programmiersprache, und `directivestyle`, das Compilerdirektiven formatiert.

Um die Schlüsselwörter einer Programmiersprache zu formatieren, werden `keywordstyle` und `ndkeywordstyle` (zweiter Ordnung) verwendet. Weitere Bezeichner werden gemäß der Einstellung von `identifiestyle` formatiert. Die Werte aller „Stil“-Schlüsselwörter (außer `basicstyle`) können als letztes Element einen L^AT_EX-Befehl mit einem Argument enthalten, wie etwa `\textbf`. Dies funktioniert, weil der „Bezeichnername“ intern in geschweifte Klammern gesetzt wird und deshalb als Argument eines Befehls interpretiert werden kann.

Auf diese Weise können Schlüsselwörter, Bezeichner und andere Elemente automatisch und doch ganz individuell hervorgehoben werden. Manchmal sollen zusätzlich bestimmte Variablen, Funktionen oder Schnittstellen

betont werden. Zu diesem Zweck kann man die Schlüsselwörter `emph` und `emphstyle` einsetzen. Ersteres erhält eine Liste der Namen, die hervorgehoben werden sollen, letzteres gibt an, wie diese gesetzt werden sollen.

```

\usepackage{listings,color}
\lstset{emph={Sum,N},emphstyle=\color{blue},
        emph=[2]I,emphstyle=[2]\underbar}

\begin{lstlisting}[language=Ada]
Sum: Integer;   Sum := 0;
for I in 1..N loop
    Sum := Sum + A(I);
end loop;
\end{lstlisting}

```

Bsp.
3-4-33

Wenn Codefragmente im normalen Text gesetzt werden sollen, kann man den Befehl `\lstinline` verwenden. Der Code wird auf dieselbe Weise begrenzt wie beim `\verb`-Befehl, was bedeutet, dass jedes Zeichen (außer der öffnenden eckigen Klammer) als Begrenzungssymbol ausgewählt werden kann, das nicht im Codefragment vorkommt. Die öffnende eckige Klammer kann nicht verwendet werden, weil der Befehl auch ein optionales Argument akzeptiert, in dem eine Liste von Schlüssel-Wert-Paaren angegeben werden kann.

Die `for`-Schleife ist definiert als
`i=0;i<n;i++.`

```

\usepackage{listings} \lstset{language=C}
Die \lstinline[keywordstyle=\underbar]!for!-Schleife
ist definiert als \lstinline!i=0;i<n;i++!.

```

Bsp.
3-4-34

Natürlich ist es auch möglich, den Inhalt kompletter Dateien zu formatieren; zu diesem Zweck gibt es den Befehl `\lstinputlisting`. Er hat ein optionales Argument, in dem Schlüssel-Wert-Paare angegeben werden können, sowie ein obligatorisches Argument, in dem die zu verarbeitende Datei angegeben wird. Im folgenden Beispiel identifiziert das Paket Schlüsselwörter einer Sprache korrekt, die Groß- und Kleinschreibung nicht unterscheidet – selbst wenn sie eine ungewöhnliche Schreibweise haben.

```

\usepackage{listings}
\begin{filecontents*}{pascal.src}
for i:=1 to maxint do
begin
    WritE('Dies ist albern');
end.
\end{filecontents*}

\lstinputlisting[language=Pascal]{pascal.src}

```

Bsp.
3-4-35

Leerzeichen in Zeichenketten werden standardmäßig als `_` dargestellt. Dieses Verhalten kann durch Setzen des Schlüsselwortes `showstringspaces` auf `false` deaktiviert werden, wie das nächste Beispiel zeigt. Die gleiche Darstellung aller Leerzeichen erreicht man auch, indem man das Schlüsselwort `showspaces` auf `true` setzt. Auf ähnliche Weise können Tabzeichen sichtbar gemacht werden, und zwar mithilfe des booleschen Schlüsselwortes `showtabs`.

Auch Zeilennummerierung ist möglich; dafür werden dieselben Schlüsselwörter verwendet wie bei `fancyvrb`: `numbers` kann entweder den Wert `left`,

right oder none annehmen (welches die Nummerierung ein- oder ausschaltet). numberblanklines legt fest, ob Leerzeilen in Hinsicht auf die Nummerierung berücksichtigt werden (voreingestellt ist false), numberstyle definiert das gesamte Erscheinungsbild der Nummern, stepnumber gibt an, welche Zeilennummern angezeigt werden (0 bedeutet keine Nummerierung), und numbersep legt den Abstand zwischen Nummer und dem Zeilenanfang fest. Standardmäßig beginnt die Zeilennummerierung mit jedem \lstinputlisting bei 1, diese Einstellung kann jedoch mithilfe des Schlüsselwortes firstnumber geändert werden. Wenn last als spezieller Wert von firstnumber angegeben wird, beginnt die Nummerierung dort, wo die letzte Nummerierung aufhört.

Vorangehender Text ... 10 for i:=1 to maxint do begin 12 Write('Dies ist albern'); end.	<pre>\usepackage{listings} % pascal.src wie vorher definiert \lstset{numberstyle=\tiny,numbers=left, stepnumber=2,numbersep=5pt,firstnumber=10, xleftmargin=12pt,showstringspaces=false} \noindent Vorangehender Text \ldots \lstinputlisting[language=Pascal]{pascal.src}</pre>

Bsp.
3-4-36

Ein globaler Einzug kann, wie im vorigen Beispiel gezeigt, mit dem Schlüsselwort xleftmargin definiert werden. Eine bestimmte Anzahl von Zeichen (hoffentlich nur Leerzeichen) können mit gobble am Anfang jeder dargestellten Zeile entfernt werden. In der Regel werden Einzüge von übergeordneten Umgebungen, wie itemize, berücksichtigt. Diese Funktion kann mit dem booleschen Schlüsselwort resetmargin ausgeschaltet werden. Die Schlüsselwörter lassen sich natürlich kombinieren. Wenn nur ein Teil der Programmcode-Zeilen formatiert werden soll, kann man mit firstline und lastline die erste und/oder letzte Zeile definieren; zum Beispiel würden bei lastline=10 höchstens 10 Programmcode-Zeilen gesetzt.

Eine andere Möglichkeit, eine fortlaufende Nummerierung zu erhalten, besteht über das Schlüsselwort name. Wenn eine Umgebung mit diesem Schlüsselwort einen „Namen“ erhalten hat, wird die Nummerierung automatisch fortgesetzt, d.h. sie startet dort, wo die letzte Umgebung mit diesem Namen endete. Dies ermöglicht bei Bedarf eine unabhängige Nummerierung einzelner Codefragmente.

Sum: Integer; Das zweite Fragment setzt die Nummerierung fort. Sum := 0; for I in 1..N loop Sum := Sum + A(I); end loop;	<pre>\usepackage{listings} \lstset{language=Ada,numbers=right, numberstyle=\tiny,stepnumber=1,numbersep=5pt} \begin{lstlisting}[name=Test] Sum: Integer; \end{lstlisting} Das zweite Fragment setzt die Nummerierung fort. \begin{lstlisting}[name=Test] Sum := 0; for I in 1..N loop Sum := Sum + A(I); end loop; \end{lstlisting}</pre>

Bsp.
3-4-37

Wenn ein Programmlisting sehr lange Zeilen enthält, passen diese möglicherweise nicht in den verfügbaren Platz. In diesem Fall erzeugt listings überlaufende Zeilen, die in den rechten Seitenrand hineinragen, genauso wie bei einer verbatim-Umgebung. Mit dem Schlüsselwort `breaklines` kann man jedoch Zeilen auch an Leerzeichen oder Satzzeichen umbrechen lassen. Die Folgezeilen werden um 20pt eingerückt, wobei dieser Wert mit dem Schlüsselwort `breakindent` verändert werden kann.

Bei Bedarf kann man Material vor (Schlüsselwort `prebreak`) und/oder nach (Schlüsselwort `postbreak`) dem Umbruch einfügen, das darauf hinweist, dass das Programmlisting künstlich umbrochen wurde. Im folgenden Beispiel wird diese Funktionalität verwendet, um mit kleinen Pfeilen und später mit der Zeichenkette „(Forts.)“ in kleiner Schrift zu experimentieren. Beide Schlüsselwörter sind intern als T_EX-Befehle `\discretionary` realisiert, was bedeutet, dass sie nur bestimmte Eingaben akzeptieren (Zeichen, Boxen und Unterschneidungen¹). Komplexeres Material sollte besser in einer `\mbox` eingebettet werden (die Material in `\discretionary` verbirgt), wie es im Beispiel geschehen ist. Bei Farbbänderungen reicht selbst das nicht aus. Hier benötigt man ein weiteres Klammernpaar, um zu verhindern, dass die Wirkung des `\special`-Befehls zur Farbeinstellung über die Box hinausgeht (siehe Besprechung in Anhang A.2.5).

Das Beispiel zeigt eine weitere Eigenschaft der Umbruchfunktion: Leerzeichen oder Tabzeichen, die vor dem umbrochenen Material stehen, werden standardmäßig in den Folgezeilen wiederholt. Wenn dies nicht erwünscht ist, muss das Schlüsselwort `breakautoindent` auf `false` gesetzt werden (wie im zweiten Teil des Beispiels).

Text am linken ↘	
→ Seitenrand	<code>\usepackage{color,listings}</code>
/* Ein ↘	<code>\lstset{breaklines=true,breakindent=0pt,</code>
→ mehrfach ↘	<code>prebreak=\mbox{\tiny\$\searrow\$},</code>
→ umbrochener ↘	<code>postbreak=\mbox{{\color{blue}\tiny\$\rightarrow\$}}}</code>
→ Kommentar ↘	<code>\begin{lstlisting}</code>
→ ! */	Text am linken Seitenrand
	/*Ein mehrfach umbrochener Kommentar! */
	<code>\end{lstlisting}</code>
/* Ein ↘	<code>\begin{lstlisting}[breakautoindent=false,</code>
(Forts.) mehrfach ↘	<code>postbreak=\tiny (Forts.)\,,</code>
(Forts.) umbrochener ↘	/*Ein mehrfach umbrochener Kommentar! */
(Forts.) Kommentar! */	<code>\end{lstlisting}</code>

Bsp.
3-4-38

Mit dem Schlüsselwort `frame` können Rahmen oder Linien um den Programmcode herum gezogen werden; es hat dieselben Werte wie in `fancyvrb` (z.B. `single`, `lines`). Zusätzlich kann es eine Untermenge von `trblTRBL` als Wert annehmen. Die großgeschriebenen Buchstaben stehen für doppelte Linien, die kleingeschriebenen für einfache Linien. Es gibt ein halbes Dutzend weiterer Schlüsselwörter: zur Festlegung der Linienbreite, des Abstandes vom Text, Erzeugung von abgerundeten Ecken usw. Sie sind alle kompatibel mit `fancyvrb`, wenn die Funktionalität dieselbe ist.

¹Leerraum, der durch den T_EX-Befehl `\kern` erzeugt wird.

Bsp.
3-4-39

```

for i:=1 to maxint do
  begin
    WrItE( ' Dies_ist_albern ' )
  end .

```

```

\usepackage{listings}
% pascal.src wie vorher definiert
\lstset{frame=trBL,framerule=2pt,framesep=4pt,
        rulesep=1pt,showspaces=true}
\lstinputlisting[language=Pascal]{pascal.src}

```

Einzelne Programmlistings können mithilfe des Schlüsselwortes `caption` eine Legende erhalten. Die Legenden werden standardmäßig nummeriert und durch die Zeichenkette `Listing` eingeführt, die in `\lstlistingname` gespeichert wird. Als Zähler wird `lstlisting` verwendet, dessen Erscheinungsbild mit `\thelstlisting` verändert werden kann. Die Legende wird entweder über (voreingestellt) oder unter dem Listing platziert, wobei diese Einstellung mit dem Schlüsselwort `captionpos` geändert werden kann.

Eine formatierte Liste aller Legenden erhält man, indem man den Befehl `\lstlistoflistings` an geeigneter Stelle im Dokument einfügt. Er erzeugt eine Überschrift aus den Wörtern, die in `\lstlistlistingname` gespeichert sind (voreingestellt ist `Listings`). Wenn sich der Legendentext im Dokument vom Text im Listingverzeichnis unterscheiden soll, muss ein optionales Argument verwendet werden (siehe nächstes Beispiel). Zu beachten ist, dass in diesem Fall der Wert in geschweifte Klammern einzufassen ist, um die rechte eckige Klammer zu verbergen. Wenn die Legende nicht im Listingverzeichnis erscheinen soll, muss dem Schlüsselwort `no1ol` der Wert `true` zugewiesen werden. Durch Verwenden des Schlüsselwortes `label` kann ein Label definiert werden, um per `\ref` auf die Listingnummer verweisen zu können. Voraussetzung ist, dass die Nummer nicht unterdrückt wird.

Listings

1 Pascal-Listing 6

Der Pascalcode in Listing 1 zeigt...

```

for i:=1 to maxint do
  begin
    WrItE( ' Dies_ist_albern ' );
  end .

```

Bsp.
3-4-40

Listing 1: Pascal

```

\usepackage{listings}
% pascal.src wie vorher definiert
\lstset{frame=single,frameround=tftt,
        language=Pascal,captionpos=b}
\lstlistoflistings
%
\bigskip % normalerweise steht obiges
\noindent % im Vorspann, aber hier ...
%
Der Pascalcode in Listing~\ref{foo} zeigt\ldots
\lstinputlisting
[caption={ [Pascal-Listing]Pascal },label=foo]
{pascal.src}

```

Mit dem Schlüsselwort `frameround`, das im vorangegangenen Beispiel verwendet wurde, können Ecken abgerundet werden, indem für jede Ecke, beginnend bei der rechten oberen Ecke, im Uhrzeigersinn der Wert `t` für „true“ (wahr) und `f` für „false“ (falsch) eingegeben wird. Diese Funktion ist nicht für `fancyvrb`-Rahmen verfügbar.

Anstatt Programmlistings im Text zu formatieren, können sie mithilfe des Schlüsselwortes `float` in Gleitobjekte umgewandelt werden, typischerweise in Kombination mit dem Schlüsselwort `caption`. Der Wert von `float` ist eine Teilmenge von `htbp`, die angibt, wohin sich das Gleitobjekt bewegen

darf (ohne Angabe eines Wertes entspricht er `tbp`). Man sollte jedoch vermeiden, gleitende und nicht gleitende Listings zu vermischen, weil dadurch die Nummerierung durcheinander geraten könnte, wie im Beispiel 6-3-5 auf Seite 305 zu sehen ist.

8-Bit-Eingabe

Standardmäßig behandelt listings nur Eingabezeichen im ASCII-Bereich; bei unerwarteten 8-Bit-Eingaben können sehr seltsame Ergebnisse entstehen, wie die Buchstabendreher im folgenden Beispiel. Wenn man `extendedchars` auf `true` setzt, werden 8-Bit-Zeichen zugelassen, was dem Paket mehr Leistung abverlangt, aber (normalerweise) zu richtigen Ergebnissen führt. Bei Verwendung eines erweiterten Zeichensatzes empfiehlt es sich natürlich, das Schlüsselwort in die Deklaration `\lstset` einzufügen, anstatt es bei jeder Umgebung erneut anzugeben, wie in Beispiel 3-4-32 auf Seite 179 geschehen. Spezielle Eingabekodierungen für Programmfragmente, die sich von der Eingabekodierung für das restliche Dokument unterscheiden, kann man auch über das Schlüsselwort `inputencoding` definieren. Dieses Schlüsselwort kann nur verwendet werden, wenn das Paket `inputenc` geladen ist.

```

\usepackage[ansinew]{inputenc}
\usepackage{listings}
\lstset{language=C,commentstyle=\scriptsize}
\begin{lstlisting}
int i; /*für die äußere Schleife*/
\begin{lstlisting}[extendedchars=true]
int i; /*für die äußere Schleife*/
\end{lstlisting}
\end{lstlisting}

```

Bsp.
3-4-41

Das Paket bietet viele weitere Schlüssel, um die Art der Darstellung festzulegen. Für spezielle Formatierungstricks kann man beliebigen L^AT_EX-Code ausführen, man kann Tabulatoren oder Zeichen für den Papiervorschub anzeigen, bestimmte Bezeichner indexieren oder ein Interface zum `hyperref`-Paket aktivieren, so dass beim Klicken auf einen Bezeichner auf die vorherige Stelle gesprungen wird, an der dieser auftrat. Einige dieser Funktionen werden noch als experimentell betrachtet und müssen über ein optionales Argument angefordert werden, wenn das Paket geladen wird. Alle diese Schlüssel sind ausführlich im (etwa 50 Seiten umfassenden) Handbuch beschrieben, das zusammen mit dem Paket verteilt wird.

Als abschließendes Beispiel für die Art von Schätzen, die in diesem Handbuch zu finden sind, siehe auch das folgende Beispiel. Es zeigt die Formatierung des Programmcodes im Stil von Donald Knuths Veröffentlichungen zu „Literate Programming“.

```

\usepackage{listings}
\lstset{literate={:=}{\$\gets\$}1
{<=}{\$\leq\$}1 {>=}{\$\geq\$}1 {<>}{\$\neq\$}1}
\begin{lstlisting}[gobble=2]
var i:integer;
if (i<=0) i ← 1;
if (i>=0) i ← 0;
if (i<>0) i := 0;
\end{lstlisting}

```

Bsp.
3-4-42

3.5 Zeilen und Spalten

Im letzten Teil dieses Kapitels werden ein paar Pakete vorgestellt, die bei der Behandlung des Textstroms in seiner Gesamtheit helfen. Das erste Paket befasst sich mit dem Anhängen von Zeilennummern an Absätze und dem automatischen Verweis auf diese. Dies kann in kritischen Editionen und anderen wissenschaftlichen Arbeiten nützlich sein.

Das zweite Paket befasst sich mit dem Problem, Quelltextströme nebeneinander darzustellen – z.B. das Original und seine Übersetzung. Es wird gezeigt, wie beide Pakete in Standardfällen kombiniert werden können.

Das dritte Paket befasst sich mit mehrspaltigen Layouts. Es ermöglicht, die Anzahl der Spalten auf einer Seite zu variieren, und unterstützt das Ausbalancieren von textuellen Daten. Standard- $\text{\LaTeX}$ bietet bereits die Möglichkeit, Text in ein- oder zweispaltigem Modus zu setzen, aber ein- und zweispaltige Ausgaben können nicht auf derselben Seite vermischt werden.

Zum Abschluss wird ein Paket vorgestellt, mit dessen Hilfe Änderungen im Quelldokument durch vertikale Balken markiert werden können.

3.5.1 lineno – Nummerieren von Textzeilen

In einigen Anwendungen ist es nützlich oder gar notwendig, Absatzzeilen zu nummerieren, um auf diese verweisen zu können. Da $\text{\TeX}$ den Zeilenumbruch über den gesamten Absatz optimiert, ist es nicht sehr geeignet, eine solche Funktion bereitzustellen, da, technisch gesprochen, der Zeilenumbruch in einem sehr späten Stadium der Verarbeitung durchgeführt wird, nämlich direkt vor der endgültigen Zusammenstellung der Seiten. Zu diesem Zeitpunkt hat die Verarbeitung von Makros, welche die richtigen Zeilennummern hinzufügen oder automatische Verweise handhaben könnten, bereits stattgefunden. Die einzige Möglichkeit, eine Zeilennummerierung zu erhalten, besteht daher darin, die fertige Seite in der „Ausgaberoutine“ (d.h. dem Teil von $\text{\LaTeX}$, der normalerweise den Text in Seiten aufteilt und Kopf- und Fußzeilen hinzufügt) zeilenweise zu zerlegen und entsprechende Zeilennummern anzufügen.

Diesen Ansatz verfolgt das lineno-Paket von Stephan Böttcher. Auch wenn man erwarten würde, dass ein solches Unterfangen nur in begrenztem Maße funktioniert, ist sein Paket erstaunlich robust und arbeitet problemlos mit vielen anderen Paketen zusammen – selbst jenen, die $\text{\LaTeX}$ s Ausgaberoutine verändern, wie `ftnright`, `multicol` und `wrapfig`. Außerdem unterstützt es Layouts, die mit der Option `twocolumn` der Standardklassen von $\text{\LaTeX}$ erzeugt wurden.

<code>\linenumbers*[start-nummer]</code>	<code>\nolinenumbers</code>

Das Laden des lineno-Paketes hat noch keine direkte Auswirkung. Um die Zeilennummerierung zu aktivieren, muss der Befehl `\linenumbers` in der Präambel oder an anderer Stelle im Dokument eingefügt werden. Durch den Befehl `\nolinenumbers` wird die Zeilennummerierung wieder deaktiviert. Die Zeilennummerierung funktioniert absatzbezogen, d.h. $\text{\LaTeX}$ prüft am Absatzende, ob gerade eine Zeilennummerierung angefordert wird, und hängt dann an alle Zeilen des Absatzes Nummern an. Es empfiehlt sich daher, diese Befehle zwischen den Absätzen zu platzieren, anstatt in diese.

Der Befehl `\linenumbers` hat ein optionales Argument, das die Nummer angibt, die für die erste Zeile verwendet werden soll. Ohne dieses Argument setzt der Befehl die Nummerierung dort fort, wo sie beim letzten Mal endete. Man kann auch die Sternform verwenden, welches eine Abkürzung für `\linenumbers[1]` ist.

Hier keine Zeilennummern. Text zum Experimentieren mit Zeilennummern.

```
\usepackage{lineno}
\newcommand{\para}{ Text zum Experimentieren
mit Zeilennummern.\par}
```

Hier keine Zeilennummern.\para
`\linenumbers`

Aber hier gibt es Zeilennummern.\para
Und hier auch.\para

`\linenumbers[-10]`

Neustart mit einer negativen Zahl.\para

Bsp.
3-5-1

Anstatt die Zeilennummerierung mit den obigen Befehlen zu starten und zu beenden, kann auch die Umgebung `linenumbers` eingesetzt werden, um einen Bereich zu definieren, der Zeilennummern erhalten soll. Diese Umgebung gibt am Ende automatisch den Befehl `\par` aus, um den aktuellen Absatz abzuschließen. Für die Zeilennummerierung von kurzen Passagen eignet sich eher die Umgebungsform (oder eine der speziellen Umgebungen `numquote` und `numquotation`, die später beschrieben werden).

Absätze in Boxen
nummerieren

Da die Produktion der Zeilennummern die Ausgaberroutine einbezieht, wird die Nummerierung nur für jene Absätze durchgeführt, die im Fließtext der Kolumne zusammengestellt und gesetzt werden, nicht jedoch für Absätze, die in Boxen platziert sind (z.B. nicht innerhalb einer `\marginpar`-Umgebung oder in einem Gleitobjekt). Über den Befehl `\internallinenumbers` bietet das Paket begrenzte Unterstützung für die Nummerierung von Zeilen in einer solchen Anordnung. Eine Einschränkung ist, dass die Grundlinien in solchen Absätzen einen festen Abstand voneinander haben müssen (ansonsten werden die Zeilen nicht korrekt positioniert) und dass sie explizit mit dem Befehl `\par` beendet werden müssen. Der Befehl `\internallinenumbers` kann auch als Sternform und mit optionalem Argument verwendet werden, so wie `\linenumbers`. Die Sternform stellt nicht nur sicher, dass die Zeilennummerierung (wieder) bei 1 startet, sondern auch, dass die Zeilennummern nicht die Zeilennummerierung in der Kolumne beeinflussen; vergleiche hierzu auch die Ergebnisse in den beiden `\marginpars` unten.

1 Text im vertikalen
2 Hauptstrang! Text zum
3 Experimentieren mit
4 Zeilennummern.

```
\usepackage{lineno}
%\para wie zuvor definiert
\linenumbers
```

Text im vertikalen Hauptstrang!

`\marginpar{\footnotesize`

`\internallinenumbers* \para}`

`\para` In diesem Absatz wird eine zweite

Marginalie verwendet, welche diesmal

`\marginpar{\footnotesize`

`\internallinenumbers \para}`

die Zeilennummern beeinflusst. \para

Bsp.
3-5-2

8 In diesem Absatz
9 wird eine zweite Margi-
10 nalie verwendet, welche
11 diesmal die Zeilennum-
12 mern beeinflusst. Text
13 zum Experimentieren
14 mit Zeilennummern.

1 Text zum Experi-
2 mentieren mit Zei-
3 lennummern.

5 Text zum Experi-
6 mentieren mit Zei-
7 lennummern.

Die Zeilennummern im zweiten `\marginpar` setzen die Nummerierung der Kolumne fort (die letzte Zeile des vorangegangenen Absatzes war 4), und der zweite Absatz beginnt dann mit der Zeilennummer 8. Die `\marginpar`-Befehle werden verarbeitet, bevor der Absatz, in dem sie stehen, umbrochen wird, welches die Reihenfolge der Nummern erklärt.

Da bei Verwendung des `lineno`-Paketes zur Zeilennummerierung beim Aufruf der Ausgaberroutine der Befehl `\par` ausgeführt werden muss, entsteht ein $\TeX$ nisches Problem, wenn bestimmte abgesetzte mathematische Konstrukte verwendet werden: Der Teilabsatz über dieser abgesetzten Formel wird von $\TeX$ ohne Ausgabe von `\par` umbrochen. Ohne weitere Hilfe erhält solch ein Teilabsatz also keine Zeilennummern. Die Lösung besteht hier, wie das nächste Beispiel zeigt, darin, die Umgebung `linenomath` bereitzustellen, welche die Zeilennummerierung übernimmt, wenn sie eine solch abgesetzte Formel umfasst. Zu ihr gibt es eine Sternform, die auch die Zeilen mit den abgesetzten Formeln nummeriert.

*Abgesetzte Formeln
handhaben*

Keine Zeilennummer vor der abgesetzten
Formel:

$$x \neq y$$

1 Text zum Experimentieren mit Zeilennummern.

2 Aber hier Zeilennummern:

$$x \neq y$$

3 Text zum Experimentieren mit Zeilennummern.

```
\usepackage{lineno} \linenumbers
\newcommand\sample{ Text zum
  Experimentieren mit Zeilennummern.}
Keine Zeilennummer vor der abgesetzten
Formel:
\[ x \neq y \] \sample \par
Aber hier Zeilennummern:
\begin{linenomath}
  \[ x \neq y \]
\end{linenomath}
\sample\par
```

Bsp.
3-5-3

Wenn es viele solcher abgesetzten Formeln gibt, ist es sehr mühselig, all diese Formeln in eine `linenomath`-Umgebung einzubetten. Aus diesem Grund bietet das Paket die Option `displaymath`, die die Basisumgebungen von $\LaTeX$ für abgesetzte Formeln neu definieren, so dass sie intern `linenomath`-Umgebungen verwenden. Die Option `mathlines` bewirkt, dass sich `linenomath` wie seine Sternform verhält, so dass die abgesetzten mathematischen Formeln ebenfalls Zeilennummern erhalten.

1 Text zum Experimentieren mit Zeilennum-
2 mern.

$$x \neq y$$

4 Text zum Experimentieren mit Zeilennummern.

5 Text zum Experimentieren mit Zeilennum-
6 mern.

$$x \neq y$$

8 Text zum Experimentieren mit Zeilennummern.

```
\usepackage[displaymath,mathlines]
{lineno}
\linenumbers
% \sample wie zuvor definiert
\sample \[ x \neq y \] \sample\par
\sample
\begin{displaymath}
  x \neq y
\end{displaymath}
\sample
```

Bsp.
3-5-4

Um auf Zeilennummern zu verweisen, muss in der entsprechenden Zeile ein `\lineatop` eingefügt werden und dann, wie bei anderen Verweisen, die mit `\label` definiert sind, mit `\ref` oder `\pageref` auf sie verwiesen werden. `\lineatop` kann jedoch nur im Fließtext verwendet werden und sollte nur

*Querverweise auf
Zeilennummern*

innerhalb von Absätzen eingesetzt werden, die tatsächlich Nummern haben. An anderer Stelle entsteht entweder ein falscher Verweis (wenn die aktuelle Zeile keine Zeilennummer hat) oder es wird eine Fehlermeldung ausgegeben (an Stellen, an denen `\linelabel` nicht zugelassen ist).

```
\usepackage{lineno}
\linenumbers
% \sample wie zuvor definiert
```

```
\sample\linelabel{first}\sample\sample
\sample\linelabel{second}\sample
```

Im Text soll in den Zeilen~\ref{first},
`\lineref[1]{first}`, bis einschließlich
 Zeile~\ref{second} auf einzelne Zeilen
 verwiesen werden `\ldots`

Bsp.
3-5-5

1 Text zum Experimentieren mit Zeilen-
 2 nummern. Text zum Experimentieren mit
 3 Zeilennummern. Text zum Experimentie-
 4 ren mit Zeilennummern. Text zum Experi-
 5 mentieren mit Zeilennummern. Text zum
 6 Experimentieren mit Zeilennummern.
 7 Im Text soll in den Zeilen 2, 3, bis
 8 einschließlich Zeile 5 auf einzelne Zeilen
 9 verwiesen werden ...

Es ist auch möglich, auf eine Zeile zu verweisen, die keinen `\linelabel` hat, hierzu verwendet man den Befehl `\lineref` mit einem optionalen Argument, der den Versatz angibt. Dies kann nützlich sein, wenn man auf eine Zeile verweisen muss, die nicht so einfach mit einer Zeilennummer versehen werden kann, wie eine mathematische Formel, oder wenn man auf eine Zeilensequenz verweisen möchte, wie im vorangegangenen Beispiel.

*Nur manche Zeilen
beschriften*

Es gibt verschiedene Möglichkeiten, um das optische Erscheinungsbild der Zeilennummern zu verändern. Bei Angabe der Option `modulo` werden nur manche Zeilennummern angezeigt (standardmäßig jede fünfte). Dieser Effekt lässt sich auch mit dem Befehl `\modulolinenumbers` erzielen. Bei Aufruf dieses Befehls mit einem optionalen Argument werden alle Zeilen nummeriert, die ein Vielfaches der angegebenen Zahl darstellen (der Wert 1 entspricht der normalen Nummerierung). Da weder der Befehl noch die Option den Zeilennummerierungsmodus starten, ist zusätzlich der Befehl `\linenumbers` erforderlich.

```
\usepackage{lineno}
\linenumbers
% \sample wie zuvor definiert
```

```
\sample \sample \sample \par
\modulolinenumbers[2]
```

Und nun ein Absatz mit Nummern an jeder
 zweiten Zeile. `\sample\sample\sample\par`

Bsp.
3-5-6

1 Text zum Experimentieren mit Zeilen-
 2 nummern. Text zum Experimentieren mit
 3 Zeilennummern. Text zum Experimentie-
 4 ren mit Zeilennummern.
 5 Und nun ein Absatz mit Nummern an
 6 jeder zweiten Zeile. Text zum Experi-
 7 mentieren mit Zeilennummern. Text zum
 8 Experimentieren mit Zeilennummern. Text
 9 zum Experimentieren mit Zeilennummern.

Der Font für Zeilennummern wird über den Parameter `\linenumberfont` gesteuert. Standardmäßig werden kleine serifenlose Ziffern verwendet. Die Nummern werden rechtsbündig in einer Box der Breite `\linenumberwidth` platziert. Der Abstand zwischen Box und der Zeile wird durch den Wert angegeben, der in `\linenumbersep` gespeichert ist. Um die Nummer linksbündig zu setzen, muss man tiefer graben, aber selbst für diesen Fall findet man Parameter, wie `\makeLineNumberRight`, im Paket. Einstellungen mitten im

Dokument zu ändern, ist in der Regel keine gute Idee, wurde aber zu Demonstrationszwecken im nächsten Beispiel gemacht.

Die Option „right“ ändert die Position der Zeilennummer. Text zum Experimentieren mit Zeilennummern. Text zum Experimentieren mit Zeilennummern.

Hier wird ein anderer Font und ein größerer Abstand verwendet. Text zum Experimentieren mit Zeilennummern.

Bsp.
3-5-7

```
\usepackage[right]{lineno}
\linenumbers
% \sample wie zuvor definiert
1 Die Option "right" ändert die Position
2 der Zeilennummer. \sample \sample \par
3 \renewcommand\linenumberfont
4 {\normalfont\footnotesize\ttfamily}
5 \setlength\linenumbersep{20pt}
6 Hier wird ein anderer Font und ein
7 größerer Abstand verwendet. \sample \par
```

Für spezielle Anwendungen bietet das Paket zwei Umgebungen, die automatisch Zeilennummern bereitstellen: `numquote` und `numquotation`. Sie ähneln ihren L^AT_EX-Verwandten `quote` und `quotation`, außer dass ihre Zeilen nummeriert werden. Sie haben ein optionales Argument, das die Zeilennummer angibt, mit der begonnen werden soll (fehlt dieses Argument, wird bei 1 gestartet). Ihre Sternformen unterdrücken ein Zurücksetzen von Zeilennummern.

Der Hauptunterschied zu ihren L^AT_EX-Gegenstücken (bei gleichzeitiger Verwendung des Befehls `\linenumbers`) besteht in der Positionierung der Nummern, die eingerückt werden. Man verwendet diese beiden Umgebungen daher eigentlich nur, wenn der angegebene Text Zeilennummern erhalten soll, auf die separat verwiesen werden kann.

```
1 Text zum Experimentieren mit
2 Zeilennummern.

3 Text zum Experimentieren mit Zeilennum-
4 mern. Text zum Experimentieren mit Zei-
5 lennummern.
```

```
1 Text zum Experimentieren mit
2 Zeilennummern.
```

```
3 Noch mehr Text.
```

```
\usepackage{lineno}
\linenumbers
% \sample wie zuvor definiert
\begin{quote}
\sample
\end{quote}
\sample \sample
\begin{numquote}
\sample
\end{numquote}
Noch mehr Text.
```

Bsp.
3-5-8

Mithilfe der Möglichkeiten, die das Paket bietet, ist es recht einfach, eigene Umgebungen zu entwickeln, die spezielle Objekte an einzelne Zeilen anhängen. Das wichtigste Makro zur Anpassung heißt `\makeLineNumber`. Es wird innerhalb einer Box der Breite null am linken Rand jeder Zeile ausgeführt (wenn der Zeilennummerierungsmodus angeschaltet ist). Der Code sollte keinen zusätzlichen Leerraum generieren, weswegen man am besten mit `\llap` oder `\rlap` arbeitet. Ansonsten kann man praktisch alles benutzen. Man sollte nur daran denken, dass der Inhalt erst verarbeitet und angehängt wird, nachdem der Absatz umbrochen wurde und die normale Makroverarbeitung abgeschlossen ist. Daher kann der Code nicht mit Daten interagieren, die mitten im Absatz stehen. Die aktuelle Zeilennummer kann mit dem Befehl

*Eigene Erweiterungen
entwickeln*

`\LineNumber` erzeugt werden, der die Nummer oder nichts ausgibt, je nachdem, ob der Zeilennummerierungsmodus aktiv ist.

Das folgende Beispiel zeigt die Definition und Verwendung zweier neuer Umgebungen, die (wenngleich grob, weil sie sich nicht um Formatierung und dergleichen kümmern) einige der Möglichkeiten aufzeigen. Zu beachten ist, dass obwohl in der zweiten Umgebung keine Zeilennummern ausgegeben werden, die Zeilen intern gezählt werden, so dass die Zeilennummerierung hinterher mit dem korrekten Wert fortgeführt wird.

1→	Text zum Experimentie-	<code>\usepackage{lineno} \linenumbers</code>
2→	ren mit Zeilennummern.	<code>% \sample wie zuvor definiert</code>
	Text zum Experimentie- ←	<code>\newenvironment{numarrows}</code>
	ren mit Zeilennummern. Text ←	<code>{\renewcommand\makeLineNumber</code>
	zum Experimentieren mit ←	<code>{\llap{\LineNumber\$\rightarrow\$ }}}{\par}</code>
	Zeilennummern. ←	<code>\newenvironment{arrows}{\renewcommand\makeLineNumber</code>
7→	Text zum Experimentie-	<code>{\rlap{\hspace{\textwidth} \$\leftarrow\$}}}{\par}</code>
8→	ren mit Zeilennummern. Text	<code>\begin{numarrows} \sample \end{numarrows}</code>
9→	zum Experimentieren mit	<code>\begin{arrows} \sample \sample \end{arrows}</code>
10→	Zeilennummern.	<code>\sample</code>
		<code>\begin{numarrows} \sample \end{numarrows}</code>

Bsp.
3-5-9

Das Erscheinungsbild und Verhalten der Zeilennummern kann durch eine Reihe von Optionen oder Befehlen, die äquivalent zu den Optionen sind, weiter differenziert werden (siehe Paketdokumentation zu Einzelheiten der Befehlsformen). Mit den Optionen `left` (voreingestellt) und `right` gibt man an, in welchem Rand die Zeilennummern erscheinen soll. Mithilfe der Option `switch` oder `switch*` werden sie in den äußeren bzw. inneren Rand gesetzt.

Bis die Zeilennummern an der richtigen Stelle erscheinen, sind mindestens zwei $\text{\LaTeX}$ -Kompilierläufe erforderlich. Leider wird keine Warnung ausgegeben, die dazu auffordert, das Dokument ein zweites Mal kompilieren zu lassen. Daher muss man selbst darauf achten.

Mit der Option `pagewise` kann man eine seitenweise Nummerierung anfordern. Diese Option muss als Letzte angegeben werden.

3.5.2 parallel – Zwei synchronisierte Textstränge

Manchmal ist es notwendig, etwas in zwei parallelen Spalten darzustellen, z.B. bei der Ausgabe von Text und der zugehörigen Übersetzung. Parallel bedeutet in diesem Kontext, dass an bestimmten Synchronisierungspunkten zwei Textstränge (wieder) aufeinander abgestimmt werden. Diese Art von Layout wird normalerweise von $\text{\LaTeX}$ nicht unterstützt (das standardmäßig nur mit einem Textstrang arbeitet). Aber es lässt sich mit dem `parallel`-Paket von Matthias Eckermann erreichen.

Dieses Paket stellt die Umgebung `Parallel` zur Verfügung, in welche das gesamte `parallel` zu setzende Material eingebettet wird. Die Umgebung hat zwei obligatorische Argumente: die Breite der linken und rechten Spalte. Beide zusammen sollten schmaler als `\textwidth` sein; ansonsten werden sich die Texte in den beiden Spalten berühren oder gar überlappen. Um die Eingabe zu vereinfachen, können ein oder beide Argumente leer gelassen

werden. In diesem Fall wird die entsprechende Breite für die Spalte(n) automatisch berechnet, wobei der aktuelle Wert von `\ParallelUserMidSkip` als Wert für den Abstand der beiden Spalten gilt. Zur Kennzeichnung des linken und rechten Textstrangs wird `\ParallelLText` bzw. `\ParallelRText` verwendet. Beide Befehle erwarten den Text als Argument. Erstaunlicherweise ist es trotzdem möglich, den Befehl `\verb` oder eine verbatim-Umgebung in den Argumenten zu verwenden, wie das folgende Beispiel zeigt.

\verb ist zugelassen

Bsp.
3-5-10

This is text in the English language explain- ing the com- mand <code>\foo</code>.	Dies ist Text in deutscher Spra- che, der das Kom- mando <code>\foo</code> er- läutert.	<pre>\usepackage{parallel} \begin{Parallel}{}{} \ParallelLText{This is text in the English language explaining the command \verb=\foo=.} \ParallelRText{Dies ist Text in deutscher Sprache, der das Kommando \verb=\foo= erläutert.} \end{Parallel}</pre>

Um bestimmte Textzeilen auszurichten, unterteilt man die beiden Textstränge an geeigneten Stellen mit dem Befehlspaar `\ParallelLText` und `\ParallelRText` und trennt jedes Paar vom nächsten mit `\ParallelPar` ab. Wird einer der `\ParallelPar`-Befehle vergessen, geht ein Teil des Textes ohne Warnung verloren. Wie der Name suggeriert, führt `\ParallelPar` einen Absatzumbruch durch, so dass die Ausrichtung nur auf Absatzebene möglich ist. Innerhalb des Argumentes eines `\ParallelLText`-Befehls sind weitere Absatzumbrüche möglich, wobei hier jedoch kein Text ausgerichtet wird.

Im nächsten Beispiel, das ein paar „direkte“ Übersetzungen von Computer-Termini ins Deutsche (entnommen aus [56] mit freundlicher Genehmigung des Eichborn-Verlages) enthält, wird ein Kurzbefehl `\LR` definiert, der die Texteingabe vereinfacht. Bei Verwendung eines solchen Kurzbefehls kann `\verb` nicht mehr im Argument verwendet werden. Wenn `\verb` benötigt wird, sollte man die Paketbefehle direkt verwenden. Außerdem wurde im Beispiel das `lineno`-Paket eingesetzt, da bei einem Vergleich von Text und seiner Übersetzung Zeilennummern hilfreich sein können.

Bsp.
3-5-11

I just go online 2 and download an 4 update. 6 This laptop is 8 missing several 10 interfaces. 12 Microsoft 14 Office on floppy disks. 16	Ich geh mal eben auf den Strich und lade mir ein Auffrisch herunter. Dieser Schoßspitze fehlt so manches Zwischenge- sicht. Kleinweich Büro auf Schlabber- scheiben.	<pre>\usepackage{parallel,lineno} \linenumbers \modulolinenumbers[2] \setlength\linenumbersep{1pt} \newcommand\LR[2]{\ParallelLText{#1}% \ParallelRText{#2}\ParallelPar} \begin{Parallel}{.45\linewidth}{ \raggedright \setlength\leftskip{10pt} \setlength\parindent{-\leftskip} \LR{I just go online and download an update.}{Ich geh mal eben auf den Strich und lade mir ein Auf\frisch herunter.} \LR{This laptop is missing several interfaces.}{Dieser Schoß\spitze fehlt so manches Zwi\schen\ge\sicht.} \LR{Microsoft Office on floppy disks.}{Kleinweich Büro auf Schlabberscheiben.} \end{Parallel}</pre>

Wie man sehen kann, ist es möglich, Absatzparameter innerhalb des Gültigkeitsbereiches der `Parallel`-Umgebung zu verändern. Der negative `\parindent` hebt den positiven `\leftskip` auf, so dass alle Absätze linksbündig beginnen, die nachfolgenden Zeilen aber um `\leftskip` eingerückt werden (beide müssen verändert werden, *nachdem* `\raggedright` aufgerufen wurde, da dieser Befehl diese Register ebenfalls setzt).

Die Umgebung `Parallel` richtet den Text zeilenweise aus, welches unvorhergesehene Folgen hat, wenn ein Block ungewöhnlich große Objekte, wie etwa abgesetzte Formeln, enthält. Daher eignet sich diese Methode nur für normale Textzeilen.

Dieser Text enthält:

$$\sum_{n=1}^x 2a_n$$

Und hier die Erklä-

rung, die einen überraschenden Effekt zeigt.

```
\usepackage{parallel}
\begin{Parallel}{}{}
\ParallelLText{Dieser Text enthält:
\[\sum_{n=1}^x 2a_n \]}
\ParallelRText{Und hier die Erklärung,
die einen überraschenden Effekt zeigt.}
\end{Parallel}
```

Bsp.
3-5-12

*Fußnoten in
parallelem Text*

Fußnoten innerhalb von parallelem Text werden nicht am Fuß der aktuellen Seite platziert, sondern direkt am Ende der aktuellen `Parallel`-Umgebung ausgegeben und von dieser durch das Resultat von `\ParallelAtEnd` getrennt. Dieser Befehl bewirkt per Voreinstellung nichts. Er kann jedoch so umdefiniert werden, dass zwischen Fußnote und vorangegangenem Text etwas eingefügt wird. Wenn die neue Definition nur für eine einzige `Parallel`-Umgebung gelten soll, muss sie in den Gültigkeitsbereich der Umgebung platziert werden.

Die Darstellung der Fußnoten wird über vier Paketooptionen gesteuert: `OldStyleNums` formatiert die Fußnotennummern mit Mediävalziffern, `RaiseNums` erzeugt hochgestellte Fußnotennummern und `ItalicNums` produziert kursive Nummern. Wenn keine dieser Optionen angegeben ist, werden arabische Ziffern auf der Grundlinie (Versalziffern) erzeugt. Die Optionen wirken sich nur auf die Nummern vor dem Fußnotentext aus; die Fußnotenzeichen innerhalb des parallelen Textes werden stets in hochgestellten, arabischen Ziffern dargestellt. Die vierte Option, `SeparatedFootnotes`, kann mit jeder der anderen drei Optionen kombiniert werden und gibt an, dass die Fußnoten in beiden Spalten unabhängig voneinander nummeriert werden. In diesem Fall erhalten die Nummern der rechten Spalte ein mit `\ParallelDot` erzeugtes, nachgestelltes Zeichen (standardmäßig einen zentrierten Punkt). Im nächsten Beispiel ist die Definition leicht abgeändert, so dass der Punkt selbst keinen Platz einnimmt.

```
\usepackage[OldStyleNums,SeparatedFootnotes]{parallel}
\renewcommand\ParallelAtEnd{\vspace{7pt}\footnoterule}
\renewcommand\ParallelDot{\makebox[0pt][l]{\textperiodcentered}}
\begin{Parallel}[v]{}{} \raggedright
\ParallelLText{This is text\footnote{We hope!} in the
English language explaining the command \verb=\foo=.}
\ParallelRText{Dies ist Text\footnote{Ein Satz.} in
deutscher Sprache\footnote{Schlechter Stil!}, der das
Kommando \verb=\foo= erläutert.}
\end{Parallel}
```

This is text ¹ in the English language explaining the command <code>\foo</code> .	Dies ist Text ¹ in deutscher Sprache ² , der das Kommando <code>\foo</code> erläutert.

 Bsp.
3-5-13

- 1 We hope!
- 1· Ein Satz.
- 2· Schlechter Stil!

Die `Parallel`-Umgebung akzeptiert ein optionales Argument vor den obligatorischen, dessen Wert `c` (zwei Spalten erstellen – voreingestellter Wert), `v` (separate Spalten mit einer vertikalen Linie, wie im vorangegangenen Beispiel) oder `p` (linken Text auf linke Seiten und rechten Text auf rechte Seiten verteilen) sein kann. Bei Wahl der „page“-Variante ist es möglich, dass leere Seiten entstehen. Wenn man sich beispielsweise auf einer Verso-Seite befindet, muss die Umgebung auf die nächste Recto-Seite springen, um den Text auf gegenüberliegenden Seiten darzustellen.

3.5.3 multicol – Setzen in mehreren Spalten

Mit Standard- $\text{\LaTeX}$ lassen sich ein- oder zweispaltig gesetzte Dokumente erzeugen (durch Verwendung der Klassenoption `twocolumn`). Es ist jedoch nicht möglich, nur einen Teil einer Seite zweispaltig zu formatieren, da die Befehle `\twocolumn` und `\onecolumn` immer eine neue Seite beginnen. Außerdem werden die Spalten nicht ausbalanciert, was hin und wieder zu einer etwas seltsamen Verteilung des Textmaterials führt.

Das Paket `multicol`¹ von Frank Mittelbach löst diese Probleme, indem es die Umgebung `multicols` mit folgenden Eigenschaften definiert:

- Sie kann 2–10 Spalten über mehrere Seiten erzeugen.
- Die Spalten am Ende der Umgebung auf der letzten Seite werden ausbalanciert, so dass sie nahezu gleich lang sind.
- Die Umgebung kann auch innerhalb anderer Umgebungen, wie `figure` oder `minipage`, verwendet werden, wo sie eine Box mit dem in der gewünschten Anzahl an Spalten formatierten Text erstellt. Das bedeutet, dass man in derartigen Fällen das Layout nicht mehr von Hand nacharbeiten muss.
- Zwischen den einzelnen Spalten können vertikale Linien eingefügt werden, deren Stärke der Benutzer festlegt.
- Die Formatierung kann global oder für einzelne Umgebungen angepasst werden.

¹Aus historischen Gründen enthält das Copyright des `multicol`-Paketes, obwohl es unter LPPL ($\text{\LaTeX}$ Project Public License) [111] verbreitet wird, eine zusätzliche Klausel zur „moralischen Verpflichtung“, die kommerzielle Anwender auffordert, für die Verwendung des Paketes eine Gebühr an den Autor oder den $\text{\LaTeX}$ 3-Fonds zu bezahlen. Einzelheiten siehe Kopf des Paketes.

<code>\premulticols</code>	50.0pt	<code>\postmulticols</code>	20.0pt
<code>\columnsep</code>	10.0pt	<code>\columnseprule</code>	0.0pt
<code>\multicolsep</code>	12.0pt plus 4.0pt minus 3.0pt		

Tabelle 3.8: Von multicols verwendete Längenparameter

```
\begin{multicols}{spalten}[vortext][abstand]
```

Normalerweise beginnt die Umgebung einfach mit der Angabe der gewünschten Spaltenanzahl. Standardmäßig werden Absätze im Blocksatz gesetzt. Bei schmalem Satzspiegel – wie in den Beispielen – ist allerdings Flattersatz vorzuziehen, wie später gezeigt werden wird.

Dieser Text soll über mehrere Spalten verteilt werden. Wenn man es mit Flattersatz versuchen.

```
\usepackage{multicol}
\begin{multicols}{3}
Dieser Text soll über mehrere Spalten
verteilt werden. Wenn die Spalten
sehr schmal sind, sollte man es mit
Flattersatz versuchen.
\end{multicols}
```

Bsp.
3-5-14

Es kann aber natürlich auch vorkommen, dass man dem mehrspaltigen Text einige einspaltige Zeilen voranstellen möchte. Dies lässt sich mit dem optionalen Argument *vortext* bewerkstelligen. L^AT_EX versucht, den *vortext* und den Beginn des mehrspaltigen Textes immer auf einer Seite zu halten.

Einige nützliche Ratschläge

Dieser Text soll über mehrere Spalten verteilt werden. Wenn die Spalten sehr schmal sind, sollte man es mit Flattersatz versuchen.

```
\usepackage{multicol}
\begin{multicols}{2}[\section*{Einige
nützliche Ratschläge}]
Dieser Text soll über mehrere Spalten
verteilt werden. Wenn die Spalten
sehr schmal sind, sollte man es mit
Flattersatz versuchen.
\end{multicols}
```

Bsp.
3-5-15

Wenn auf der aktuellen Seite nicht mehr genügend Platz ist, beginnt die Umgebung `multicols` eine neue Seite. Dies wird durch einen globalen Parameter gesteuert. Wenn man das optionale Argument verwendet, kann es vorkommen, dass der voreingestellte Wert für diesen Parameter zu klein ist. In einem solchen Fall kann man entweder die *globale* Einstellung (siehe unten) oder mithilfe des zweiten optionalen Argumentes *abstand* den Wert für die *aktuelle* Umgebung wie folgt ändern:

```
\begin{multicols}{3}[\section*{Index}][7cm]
Text Text Text Text ...
\end{multicols}
```

<code>\multicolpretolerance</code>	<code>-1</code>	<code>\multicoltolerance</code>	<code>9999</code>
<code>columnbadness</code>	<code>10000</code>	<code>finalcolumnbadness</code>	<code>9999</code>
<code>collectmore</code>	<code>0</code>	<code>unbalance</code>	<code>0</code>
<code>tracingmulticols</code>	<code>0</code>		

Tabelle 3.9: Von multicols verwendete Zähler

Wenn der verfügbare vertikale Leerraum weniger als 7cm beträgt, wird eine neue Seite begonnen.

Die multicols-Umgebung gleicht die Spalten auf der letzten Seite aus (ursprünglich wurde sie genau für diesen Zweck entwickelt). Wenn dies nicht gewünscht ist, kann man stattdessen die Variante multicols* verwenden. Natürlich funktioniert diese Umgebung nur in der Kolumne, da man innerhalb einer Box Spalten ausgleichen muss, um die Spaltenhöhe festzulegen.

Ausgleich vermeiden

Die Umgebung multicols kennt verschiedene Parameter, mit denen die Formatierung gesteuert wird. Ihre Bedeutung wird in den folgenden Abschnitten beschrieben. Die Standardwerte sind in Tabelle 3.8 (Dimensionen) und Tabelle 3.9 (Zähler) aufgeführt. Soweit nicht anders angegeben, müssen alle Parameteränderungen vor dem Anfang der Umgebung vorgenommen werden, für die sie gelten sollen.

Die Umgebung multicols überprüft zunächst, ob der verfügbare Leerraum auf einer Seite mindestens ebenso groß ist wie der Parameter \premulticols oder der Wert des zweiten optionalen Argumentes, sofern dieses angegeben wurde. Wenn der benötigte Leerraum nicht vorhanden ist, wird der Befehl \newpage ausgegeben. Auch am Ende der Umgebung wird eine neue Seite begonnen, wenn der verbleibende Leerraum geringer ist als \postmulticols. Vor und hinter der Umgebung wird ein Leerraum der Größe \multicolsep erzeugt.

Notwendiger
Leerraum

Die Spaltenbreite in der multicols-Umgebung wird automatisch berechnet, basierend auf der Anzahl der gewünschten Spalten und dem aktuellen Wert von \linewidth. Anschließend wird sie in \columnwidth gespeichert. Zwischen den Spalten wird ein Leerraum von \columnsep eingefügt.

Spaltenbreite und
Abstand

Zwischen zwei Spalten wird jeweils eine vertikale Linie der Stärke \columnseprule platziert. Wenn dieser Parameter auf 0pt gesetzt ist (dem voreingestellten Wert), wird die Linie unterdrückt. Ist die Linienstärke größer als der Spaltenabstand, überdrückt die Linie den Spaltentext.

Vertikale Linien
hinzufügen

```
\usepackage{multicol,ragged2e}
\setlength\columnseprule{0.4pt}
\addtolength\columnsep{2pt}
\begin{multicols}{3}
\begin{RaggedRight}
  Dieser Text wird über mehrere
  Spalten verteilt. In diesem
  Beispiel wird
  Flattersatz verwendet.
\end{multicols}
```

Bsp.
3-5-16

Dieser Text wird über mehrere	Spalten verteilt. In diesem Bei-	spiel wird Flattersatz verwendet.

Spaltenformatierung

Entsprechend der Voreinstellung (der Einstellung `\flushcolumns`) versucht die Umgebung `multicols`, alle Spalten durch Dehnen der vertikalen Abstände innerhalb der Spalten in der gleichen Länge zu formatieren. Bei Einsatz von `\raggedcolumns` wird der zusätzliche Freiraum stattdessen an den Fuß jeder Spalte gesetzt.

Absätze werden mit den Standard-Parametereinstellungen formatiert (wie in den Abschnitten 3.1.11 und 3.1.12 beschrieben). Eine Ausnahme bilden `\pretolerance` und `\tolerance`, für welche die aktuellen Werte von `\multicolpretolerance` bzw. `\multicoltolerance` verwendet werden. Voreingestellt sind die Werte `-1` und `9999`, so dass der Zeilenumbruch ohne Silbentrennung übersprungen wird und relativ schlechte Absätze zugelassen werden (angesichts der Tatsache, dass die Spalten in der Regel sehr schmal sind). Wenn die Spalten breit genug sind, sollte diese Voreinstellung möglicherweise durch restriktivere Werte ersetzt werden, wie etwa

```
\multicoltolerance=3000
```

Man beachte die etwas ungewöhnliche Zuweisungsform des internen $\TeX$ -Zählers `\multicoltolerance`, er wird genauso wie `\tolerance` gesteuert.

Kontrolliertes Ausbalancieren

Am Ende der `multicols`-Umgebung wird der verbleibende Text verwendet, um die Spalten auf gleiche Länge auszubalancieren. Wenn mehr Text in der linken Spalte eingefügt werden soll, kann der Zähler `unbalance` hochgesetzt werden. Dieser Zähler bestimmt die Anzahl der zusätzlichen Zeilen in den Spalten, verglichen mit der Anzahl, welche der Balancierungsalgorithmus berechnet hat. Am Ende der Umgebung wird er automatisch auf null zurückgesetzt. Um den Effekt zu demonstrieren, verwendet das nächste Beispiel den Text aus Beispiel 3-5-16 auf der vorherigen Seite, fordert aber eine zusätzliche Zeile an.

Dieser Text wird über mehrere Spalten

verteilt. In diesem Beispiel wird Flattersatz

verwendet.

```
\usepackage{multicol,ragged2e}
\addtolength\columnsep{2pt}
\begin{multicols}{3}
\RaggedRight
\setcounter{unbalance}{1}
  Dieser Text wird über mehrere Spalten
  verteilt. In diesem Beispiel wird
  Flattersatz verwendet.
\end{multicols}
```

Bsp.
3-5-17

Der Ausgleich der Spaltenlänge wird außerdem durch die beiden Zähler `columnbadness` und `finalcolumnbadness` gesteuert. Immer wenn $\LaTeX$ Boxen (wie z.B. eine Spalte) erstellt, wird ein Badness-Wert für die Qualität der Box errechnet, d.h. die Höhe des überschüssigen Weißraums. Ein Wert von null ist optimal, während ein Wert von 10000 in $\LaTeX$ s Augen schlecht ist.¹

¹Wenn eine Box überläuft, wird ihr Badness-Wert von $\TeX$ auf 100000 gesetzt, um diesen speziellen Fall zu markieren.

Während die Spalten ausgeglichen werden, vergleicht der verwendete Algorithmus den `Badness`-Wert der möglichen Lösungen. Wenn eine Spalte einen höheren Wert als `columnbadness` hat (mit Ausnahme der letzten), wird die Lösung ignoriert. Hat der Algorithmus schließlich eine Lösung gefunden, prüft er den `Badness`-Wert der letzten Spalte. Ist dieser größer als `finalcolumnbadness`, wird der überschüssige Leerraum der letzten Spalte an den Fuß der Spalte gesetzt, so dass diese kürzer wird als die anderen.

Sammeln von Material

Um Spalten richtig setzen zu können, muss die `multicols`-Umgebung genügend Material sammeln, um den übrigen Teil der Seite zu füllen. Erst dann unterteilt sie das gesammelte Material in einzelne Spalten. Dabei geht sie davon aus, dass durch den Umbruch pro Spalte nicht mehr als das Äquivalent einer Textzeile an vertikalem Leerraum im Seitenrand verschwindet. In einigen Fällen ist diese Annahme falsch und es wird erforderlich, mehr oder weniger Material zu sammeln. Hier hat man die Möglichkeit, die Standardeinstellung des Zählers `collectmore` zu verändern. Erhöhung oder Verringerung dieses Zählers um eins bedeutet, dass Material für einen `\baselineskip` mehr bzw. weniger gesammelt wird.

Tatsächlich gibt es Gründe, warum man dieses Sammeln von Material reduzieren sollte. Wenn das Dokument viele Fußnoten enthält und sehr viel zusätzliches Material gesammelt wird, ist die Wahrscheinlichkeit höher, dass der ungenutzte Teil noch Fußnoten enthält, die dann auf der falschen Seite ausgegeben werden. Der kleinste vernünftige Wert für den Zähler ist die negative Anzahl von verwendeten Spalten. Bei diesem Wert sammelt `multicols` genau soviel Material, um alle Spalten zu füllen, solange kein Leerraum an einem Spaltenumbruch verloren geht. Leerräume, die in dieser Einstellung verworfen werden, tauchen jedoch als Weißraum in der letzten Spalte wieder auf.

Überwachung des Algorithmus

Man kann das `multicol`-Paket bei der Arbeit beobachten, indem man es mit einer der folgenden Optionen lädt. Der Standardwert `errorshow` gibt nur die tatsächlichen Fehler aus. Bei `infoshow` wird `multicol` auskunftsfreudiger und gibt grundlegende Verarbeitungsinformation aus, wie etwa

```
Package multicol: Column spec: 185.0pt = indent + columns + sep =
(multicol)          0.0pt + 3 x 55.0pt + 2 x 10.0pt on input line 32.
```

Hier zeigt das Paket die berechnete Spaltenbreite an.

Bei `balancingshow` werden zusätzliche Informationen über die einzelnen Schritte geliefert, die `multicols` durchführt, um die optimale Spaltenhöhe für den Ausgleich festzustellen, einschließlich der resultierenden `Badness`-Werte der Spalten, Gründe für den Abbruch eines Versuchs usw.

Bei `markshow` wird zusätzlich gezeigt, welche Textmarken für die lebenden Kolumnentitel auf jeder Seite generiert werden. Anstatt die Optionen zu verwenden, kann vorübergehend der Zähler `tracingmulticols` auf einen positiven Wert gesetzt werden (je höher der Wert, desto mehr Informationen werden ausgegeben).

Manueller Umbruch von Spalten

Manchmal ist es notwendig, den Spaltenumbruchsalgorithmus zu überschreiben. Es wurde bereits gezeigt, wie der Zähler `unbalance` verwendet wird, um das Ausbalancieren zu beeinflussen. In manchen Fällen möchte man vielleicht eine Spalte nach einer bestimmten Zeile ausdrücklich beenden. In Standard- $\text{\LaTeX}$ lässt sich dies durch einen `\pagebreak`-Befehl erreichen, aber dieser Ansatz funktioniert nicht innerhalb einer `multicols`-Umgebung, denn dadurch wird das Sammeln von Material abgeschlossen und damit *alle* Spalten auf der Seite beendet.

Als Alternative wird der Befehl `\columnbreak` bereitgestellt. Wenn er innerhalb eines Absatzes verwendet wird, markiert er das Ende der aktuellen Zeile als bevorzugte Umbruchstelle. Zwischen Absätzen eingesetzt, bewirkt er, dass der nächste Absatz in die nächste Spalte (oder Seite) verschoben wird, wie das folgende Beispiel zeigt. Falls `\flushcolumns` eingestellt ist, wird das Material in der Spalte vertikal gedehnt (wenn möglich), um die gesamte Spaltenhöhe auszufüllen. Wenn dies nicht gewünscht ist, kann man einen `\vfill`-Befehl voranstellen, um den Fuß der Spalte mit Weißraum zu füllen.

Dieser Text soll
über mehrere Spal-
ten verteilt werden.

Mithilfe des Befehls
`\columnbreak`
wurde dieser Absatz
in die zweite Spalte
verschoben.

```
\usepackage{multicol,ragged2e}
\begin{multicols}{2} \RaggedRight
Dieser Text soll über mehrere Spalten verteilt
werden. \par \vfill\columnbreak
Mithilfe des Befehls \verb=\columnbreak= wurde
dieser Absatz in die zweite Spalte verschoben.
\end{multicols}
```

Bsp.
3-5-18

Gleitobjekte und Fußnoten in multicols-Umgebungen

Gleitobjekte (wie z.B. Abbildungen und Tabellen) werden von `multicols` nur zum Teil unterstützt. Man kann die Sternform der Gleitumgebungen verwenden, um Gleitobjekte zu erzeugen, die sich über alle Spalten erstrecken. Gleitobjekte in Spalten und `\marginpar`-Befehle werden jedoch nicht unterstützt.

Fußnoten werden am Fuß der Seite (über die gesamte Breite) gesetzt, nicht unter einzelnen Spalten (ein Zugeständnis an die Tatsache, dass auf einer einzelnen Seite unterschiedliche Spaltenbreiten unterstützt werden).

Unter gewissen Umständen kann es vorkommen, dass Fußnoten nicht auf derselben Seite erscheinen wie ihr Verweis im Text. In diesem Fall gibt `multicols` eine Warnung aus, und die fragliche Seite muss überprüft werden. Wenn Fußnote und Verweis tatsächlich auf verschiedenen Seiten erscheinen, muss dieses Problem lokal durch Setzen eines `\pagebreak`-Befehls an einer strategisch günstigen Stelle gelöst werden. Der Grund für dieses Verhalten ist, dass `multicols` beim Ansammeln von Material vorausschauen muss und später unter Umständen nicht alles gesammelte Material verwenden kann. Wie weit $\text{\LaTeX}$ vorausschaut, wird durch den Zähler `collectmore` gesteuert.

3.5.4 changebar – Hinzufügen von Revisionsbalken

Wenn ein Dokument entwickelt wird, ist es manchmal notwendig, Änderungen im Text optisch hervorzuheben. Eine übliche Methode ist das Hinzufügen von Balken im Seitenrand, sogenannten „Revisionsbalken“. Diese Funktion wird

vom Paket `changebar` unterstützt, das ursprünglich von Michael Fine und Neil Winton entwickelt wurde und nun von Johannes Braams gepflegt wird. Dieses Paket arbeitet mit den meisten PostScript-Treibern zusammen, insbesondere aber mit `dvips`, dem Standardtreiber beim Laden des Paketes. Andere Treiber können über die Paketoption ausgewählt werden. Unterstützt werden die Optionen `dvitooln03`, `dvitops`, `dvips`, `emtex`, `textures` und `vtex`.

*Unterstützte
Druckertreiber*

```
\begin{changebar}[balkenbreite] \cbstart[balkenbreite] ... \cbend
```

Wenn Text im Dokument hinzugefügt wird und diese Tatsache angezeigt werden soll, sollte er in die Umgebung `changebar` eingebettet werden. Dies gewährleistet, dass $\text{\LaTeX}$ eine Warnung ausgibt, wenn das Ende einer Änderung nicht markiert wird. Diese Umgebung lässt sich gut in andere Umgebungen einbetten. Wenn die Änderungen jedoch in einer Umgebung beginnen und in einer anderen enden, kann diese eine Umgebung nicht verwendet werden, da dies zu falsch verschachtelten Umgebungen führen würde. Für diese Situation bietet das Paket die Befehle `\cbstart` und `\cbend`. Sie sollten jedoch mit Vorsicht verwendet werden, denn es gibt keine Überprüfung, ob sie korrekt eingesetzt wurden. Leerzeichen nach den Befehlen werden möglicherweise ignoriert.

Wenn ein einzelner Balken eine andere Breite erhalten soll, kann man mithilfe des optionalen Argumentes die Breite als normale $\text{\LaTeX}$ -Länge angeben.

```
\cbdelete[balkenbreite]
```

Entfernter Text kann über den Befehl `\cbdelete` markiert werden. Auch hier lässt sich die Breite des Balkens ändern.

```
\usepackage{changebar}
```

```
\cbstart
```

```
Dies ist der Text im ersten Absatz. Dies  
ist weiterer Text im ersten Absatz.\cbend
```

```
Dies ist der Text im zweiten Absatz.
```

```
\cbdelete
```

```
Dies ist der Text im zweiten Absatz.
```

```
\setcounter{changebargrey}{35}
```

```
\begin{changebar}[4pt]
```

```
Dies ist Absatz Drei.
```

```
Dies ist Absatz Vier.
```

```
\end{changebar}
```

Dies ist der Text im ersten Absatz. Dies ist weiterer Text im ersten Absatz.

Dies ist der Text im zweiten Absatz. Dies ist der Text im zweiten Absatz.

Dies ist Absatz Drei.

Dies ist Absatz Vier.

Bsp.
3-5-19

```
\nochangebars
```

Wenn das Dokument seine endgültige Form erhalten hat, können die Revisionsbalken unterdrückt werden, indem in der Präambel der Befehl `\nochangebars` eingefügt wird.

Anpassungen

Breite ändern Wenn die Breite *aller* Revisionsbalken geändert werden soll, muss der Wert von `\changebarwidth` über den Befehl `\setlength` geändert werden. Das Gleiche gilt auch für Löschbalken, hier ist der Wert des Parameters `\deletebarwidth` zu ändern.

Revisionsbalken positionieren Standardmäßig erscheinen Revisionsbalken im „inneren Seitenrand“; mit einer der folgenden Optionen kann jedoch eine andere Position ausgewählt werden: `outerbars`, `innerbars`, `leftbars` oder `rightbars`.

Der Abstand zwischen Text und Balken wird über `\changebarsep` gesteuert. Er ist nur in der Dokumentenpräambel abänderbar.

Revisionsbalken einfärben Auch die Farbe der Revisionsbalken kann der Benutzer abändern. Standardmäßig ist die Option `grey` ausgewählt, so dass die Revisionsbalken grau erscheinen (Grauwert 65%). Eine Ausnahme bilden die Treiber `dvitoln03` und `emtex`, sie erzeugen schwarze Revisionsbalken.

Der Grauwert der Revisionsbalken lässt sich mithilfe des L^AT_EX-Zählers `changebargrey` einstellen. Er wird als Prozentwert angegeben, wobei 0 schwarze Balken und 100 weiße Balken erzeugt. Somit liefert zum Beispiel `\setcounter{changebargrey}{85}` ein helles Grau.

Die Option `color` ermöglicht die Verwendung von farbigen Revisionsbalken. Intern wird dabei `dvipsnames` geladen, so dass die Auswahl einer Farbe über Namen erfolgen kann.

```
\cbcolor{name}
```

Die Farbe, die zur Ausgabe der Revisionsbalken verwendet werden soll, wird über den Befehl `\cbcolor` ausgewählt. Er akzeptiert die gleichen Argumentwerte wie der Befehl `\color` aus dem `color`-Paket [58, S. 317–326].

Dies ist der Text im ersten Absatz.
Dies ist der Text im ersten Absatz.
Dies ist der Text im zweiten Absatz.
Dies ist der Text im zweiten Absatz.
Dies ist Absatz Drei.
Dies ist Absatz Vier.

```
\usepackage[rightbars,color]{changebar}
\cbcolor{blue}
\setlength\changebarsep{10pt}

\cbstart
Dies ist der Text im ersten Absatz.
Dies ist der Text im ersten Absatz.\cbend
```

```
Dies ist der Text im zweiten Absatz.
\cbdelete
Dies ist der Text im zweiten Absatz.
```

```
\begin{changebar}
Dies ist Absatz Drei. \par
Dies ist Absatz Vier.
\end{changebar}
```

Bsp.
3-5-20

Algorithmus überwachen

Das Verhalten des Paketes `changebar` lässt sich überwachen, indem es mit einer der folgenden Optionen geladen wird. Die voreingestellte `traceoff` gibt die normale Information aus, die L^AT_EX immer anzeigt. Die Option `tracelon` informiert über den Anfangs- und Endpunkt von definierten Revisionsbalken. Die *zusätzliche* Option `tracestacks` fügt Informationen über die Verwendung von internen Stapeln hinzu.

KAPITEL 4

Das Seitenlayout

4.1 Geometrische Dimensionen des Layouts.	202
4.2 Verändern des Seitenlayouts.	205
4.3 Dynamische Seitendaten: Seitenzahlen und Textmarken	223
4.4 Layouts für Kolumnentitel	230
4.5 Visuelle Formatierung	242
4.6 Layouts mit Klasse	244

Unter einem Satzspiegel versteht man die Fläche auf dem Papier, die mit Text und Abbildungen gefüllt werden soll. Er ist in der Regel nicht auf dem Papier zentriert, und der freie Raum am Kopfsteg und am Bundsteg (Leerraum in der Mitte einer Doppelseite) ist normalerweise kleiner als der freie Raum am Fußsteg und am Seitensteg (Leerraum an der Außenseite). Ein klassisches Seitenverhältnis von *Bund*, *Kopf*, *Außenseite* und *Fuß* zueinander ist z.B. 2 : 3 : 4 : 6. In manchen Fällen enthält der Seitensteg kurze Texte, so genannte *Marginalien*, oder auch erläuternde Texte zu Abbildungen und Tafeln. Eine gesetzte Seite (bzw. Spalte) wird in der Fachwelt auch als *Kolumne* bezeichnet. Eventuelle Kopf- und Fußzeilen, welche die Seitennummer oder andere Informationen über die aktuelle Seite enthalten, nennt man deshalb auch *Kolumnentitel*.

Die Größe, Form und Position all dieser Felder sowie die Struktur der Kolumnentitel bilden zusammen das *Seitenlayout*. In diesem Kapitel wird gezeigt, wie sich verschiedene Seitenlayouts festlegen lassen. Häufig benötigt man schon für ein einziges Dokument unterschiedliche Seitenlayouts. So unterscheidet sich z.B. die erste Seite eines Kapitels, welche die Kapitelüberschrift enthält, meistens von den übrigen Kapitelseiten.

Zunächst werden die Dimensionsparameter vorgestellt, mit denen $\text{\LaTeX}$ das Seitenlayout steuert, und es wird erklärt, wie man sie verändern und ihre Werte bildlich darstellen kann. Danach folgt eine ausführliche Besprechung der Pakete `typearea` und `geometry`, mit deren Hilfe sich die Konfiguration von Seitenlayouts sehr genau steuern lässt.

Im dritten Abschnitt wird erklärt, nach welchen Verfahren $\text{\LaTeX}$ das Datenmaterial für die Kolumnentitel zusammenstellt. Der darauf folgende

Abschnitt legt anhand zahlreicher Beispiele, unter anderem mit dem Paket `fancyhdr`, dar, wie diese Elemente formatiert werden.

Der fünfte Abschnitt stellt Befehle vor, die hilfreich sind, wenn der Text nicht ins Layout passt und manuelle Änderungen erforderlich werden. Das Kapitel schließt mit einem kurzen Blick auf zwei generische Dokumentenklassen, die eine nahezu vollständige Kontrolle über den Prozess zum Festlegen des Seitenlayouts gewähren.

4.1 Geometrische Dimensionen des Layouts

Der Text eines Dokumentes nimmt normalerweise eine rechteckige Fläche auf dem Papier ein, den so genannten *Satzspiegel* oder *Textbereich*. Über und unter dem Text (am Kopf und Fuß der Seite) können sich *Kolumnentitel* befinden. Diese können aus einer oder mehreren Zeilen bestehen und die Seitennummer (Folio), Informationen über das aktuelle Kapitel, den Abschnitt, die Zeit und das Datum sowie möglicherweise weitere Informationen enthalten.

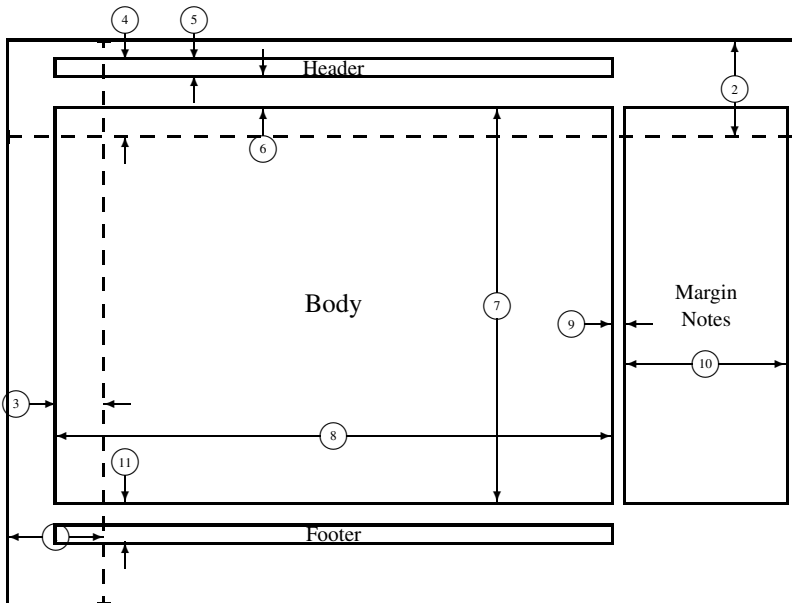
Man spricht von einem *lebenden Kolumnentitel*, wenn dieser Informationen enthält, die sich mit dem Inhalt der aktuellen Seite verändern, wie z.B. den Überschriftentext des derzeitigen Kapitels. Kolumnentitel, die nur aus einer Seitennummer oder aus unveränderlichem Text bestehen, bezeichnet man als *tote Kolumnentitel*. Wenn die Kolumnentitel sehr ins Auge fallen und eng mit dem Text verbunden sind, werden diese Elemente dem Satzspiegel zugerechnet. Das gilt häufig für lebende Kolumnentitel am Seitenkopf, insbesondere wenn diese unterstrichen sind. Andernfalls werden sie zum *Kopf-* oder *Fußsteg* (oberer und unterer Seitenrand) gezählt. Diese Unterscheidung ist wichtig für die Interpretation der Größenangaben.

Die Ränder links und rechts des Textbereiches sind normalerweise leer; in manchen Fällen enthalten sie jedoch kurze Texte, wie z.B. Anmerkungen oder Kommentare, die so genannten *Marginalien*. Im Allgemeinen spricht man hier vom inneren Rand oder *Bundsteg* und vom äußeren Rand oder *Seitensteg*. Beim zweiseitigen Druck ist der Bundsteg der mittlere Rand – d.h. auf rechten (ungeraden) Seiten der linke und auf linken (geraden) Seiten der rechte. Beim einseitigen Druck ist der Bundsteg immer der linke Rand. Bei den Doppelseiten eines Buches liegen ungerade Seiten immer rechts.

Mithilfe der Standarddokumentenklassen von $\text{\LaTeX}$ lassen sich Dokumente für den Recto-Verso-Druck (zweiseitiger Druck) formatieren. Zweiseitige Layouts können entweder asymmetrisch oder symmetrisch ($\text{\LaTeX}$ -Voreinstellung) sein. Bei einem symmetrischen Layout sind die Satzspiegel der rechten und linken Seiten deckungsgleich, wenn man ein bedrucktes Blatt gegen das Licht hält. Auch die Marginalien wechseln normalerweise ihre Position je nach linker oder rechter Seite.

Die Dimensionsparameter für das Seitenlayout sind in Abbildung 4.1 auf der nächsten Seite schematisch dargestellt.¹ Die Standardwerte dieser Parameter sind abhängig vom Papierformat. Die Klassendateien von $\text{\LaTeX}$ unterstützen eine Anzahl verschiedener Optionen, mit denen sich Ausdrücke leichter auf andere Papierformate anpassen lassen. Sie setzen `\paperwidth` und

¹Die graphische Darstellung wurde mit dem Paket `layouts` erzeugt, siehe Abschnitt 4.2.1.



1	one inch + \hoffset	2	one inch + \voffset
3	\oddsidemargin = -36pt	4	\topmargin = -58pt
5	\headheight = 12pt	6	\headsep = 25pt
7	\textheight = 296pt	8	\textwidth = 418pt
9	\marginparsep = 11pt	10	\marginparwidth = 121pt
11	\footskip = 30pt		\marginparpush = 5pt (not shown)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 597pt		\paperheight = 423pt

Bsp.
4-1-1

\paperheight Papierhöhe.

\paperwidth Papierbreite.

\textheight Höhe des Satzspiegels (ohne Kolumnentitel).

\textwidth Breite des Satzspiegels.

\columnsep Spaltenzwischenraum bei Mehrspaltensatz (Zwischensteg).

\columnseprule Stärke der Spaltentrennlinie bei Mehrspaltensatz (Voreinstellung 0pt, d.h. unsichtbare Linie).

\columnwidth Spaltenbreite bei Mehrspaltensatz. Wird von L^AT_EX aus \textwidth und \columnsep passend berechnet.

\linewidth Breite der aktuellen Zeile; hat normalerweise den gleichen Wert wie \columnwidth, kann sich aber in Umgebungen ändern, die andere Ränder setzen.

\evensidemargin Bei zweiseitigem Druck zusätzlicher linker Rand auf Verso-Seiten.

\oddsidemargin Bei zweiseitigem Druck zusätzlicher linker Rand auf Recto-Seiten, sonst zusätzlicher linker Rand auf allen Seiten.

\footskip Vertikaler Abstand zwischen der Grundlinie der letzten Textzeile und der Grundlinie der Fußzeile.

\headheight Höhe der Kopfzeile.

\headsep Vertikaler Abstand zwischen Kopfzeile und Textbereich.

\topmargin Zusätzlicher vertikaler Abstand oberhalb der Kopfzeile.

\marginparpush Vertikaler Mindestabstand zwischen zwei aufeinander folgenden Marginalien (nicht abgebildet).

\marginparsep Horizontaler Abstand zwischen Textbereich und Marginalien.

\marginparwidth Breite der Marginalien.

Abbildung 4.1: Seitenlayoutparameter und ihre Darstellung (mit layouts generiert)

letterpaper	$8\frac{1}{2} \times 11$	Zoll		
legalpaper	$8\frac{1}{2} \times 14$	Zoll		
executivepaper	$7\frac{1}{4} \times 10\frac{1}{2}$	Zoll		
a4paper	$\approx 8\frac{1}{4} \times 11\frac{3}{4}$	Zoll	210×297	mm
a5paper	$\approx 5\frac{7}{8} \times 8\frac{1}{4}$	Zoll	148×210	mm
b5paper	$\approx 7 \times 9\frac{7}{8}$	Zoll	176×250	mm

Tabelle 4.1: Standardoptionen für Papierformate in L^AT_EX

`\paperheight` auf die erforderliche Papiergröße und passen die davon abhängigen Werte (wie z.B. `\textheight` und `\textwidth`) entsprechend an.

Tabelle 4.1 zeigt die Optionen für Papierformate, die den L^AT_EX-Standardklassen bekannt sind, zusammen mit den entsprechenden Papiermaßen. Tabelle 4.2 auf der nächsten Seite zeigt die Parameterwerte des Seitenlayouts für die Papierformat-Option `letterpaper`, die voreingestellt ist, wenn nicht explizit ein anderes Format gewählt wurde. Die aufgeführten Werte sind für die drei L^AT_EX-Standarddokumententypen (`article`, `book` und `report`) gleich. Andere Papierformat-Optionen können natürlich andere Werte ergeben. Mit Hilfe des folgenden Befehls kann man z.B. auf DIN A4-Papier drucken: `\documentclass[a4paper]{article}`.

Für andere Klassen mögen weitere oder andere Optionen zur Verfügung stehen. Es erscheint jedoch nicht sehr sinnvoll, beispielsweise für die Dokumententypklasse `book` eine Option `a0paper` zu definieren, die unglaublich breite Zeilen erzeugt.

Die meisten Layoutparameter in den Klassendateien von L^AT_EX sind direkt von der tatsächlichen Papiergröße abhängig. Daher ändern sie sich automatisch, wenn die Größen `\paperwidth` oder `\paperheight` über eine der Optionen für Papierformate modifiziert werden. Wenn man diese beiden Größen in der Dokumentenpräambel ändert, hat das nicht die gleiche Wirkung, da die anderen Parameter schon berechnet sind, wenn die Präambel gelesen wird.

Standardrand von
einem Zoll

Der Standard für dvi-Treiber sieht vor, dass der Referenzpunkt für T_EX ein Zoll unterhalb und rechts der oberen linken Papierecke liegt. Dieser durch räumliche Verschiebung des Referenzpunktes erzeugte zusätzliche Rand legt den *Druckbereich* einer Seite fest. Der Referenzpunkt lässt sich über die Längen `\hoffset` und `\voffset` verschieben. Sie sind auf null voreingestellt und sollten im allgemeinen nicht verändert werden. Man kann jedoch mit ihrer Hilfe auf einfache Weise den gesamten Satzspiegel (Text, Kopf, Fuß und Marginalien) auf der Ausgabefläche verschieben, ohne das Layout zu beeinträchtigen. Die vom Treiber erzeugten Ränder stammen aus T_EX und werden von L^AT_EX für die Parametrisierung des Seitenlayouts nicht benötigt. Über `\topmargin` verschiebt man den Satzspiegel vertikal, über `\oddsidemargin` oder `\evensidemargin` verschiebt man ihn horizontal.

Man sollte dabei allerdings beachten, dass einige dvi-Treiber den Satzspiegel um andere Werte verschieben. Um sicherzugehen, dass der Referenzpunkt richtig positioniert ist, kann man die Testdatei `testpage.tex` (von Leslie Lamport, mit Änderungen von Stephen Gildea) unter L^AT_EX mit dem fraglichen dvi-Treiber ausdrucken. Die ausgegebene Seite zeigt die Position des

<i>Parameter</i>	<i>zweiseitiger Druck</i>			<i>einseitiger Druck</i>		
	10pt	11pt	12pt	10pt	11pt	12pt
<code>\oddsidemargin</code>	44pt	36pt	21pt	63pt	54pt	39pt
<code>\evensidemargin</code>	82pt	74pt	59pt	63pt	54pt	39pt
<code>\marginparwidth</code>	107pt	100pt	85pt	90pt	83pt	68pt
<code>\marginparsep</code>	11pt	10pt	10pt	<i>wie links</i>		
<code>\marginparpush</code>	5pt	5pt	7pt	<i>wie links</i>		
<code>\topmargin</code>	27pt	27pt	27pt	<i>wie links</i>		
<code>\headheight</code>	12pt	12pt	12pt	<i>wie links</i>		
<code>\headsep</code>	25pt	25pt	25pt	<i>wie links</i>		
<code>\footskip</code>	30pt	30pt	30pt	<i>wie links</i>		
<code>\textheight</code>	$\underbrace{43 \quad 38 \quad 36}_{\times \backslash baselineskip}$			<i>wie links</i>		
<code>\textwidth</code>	345pt	360pt	390pt	<i>wie links</i>		
<code>\columnsep</code>	10pt	10pt	10pt	<i>wie links</i>		
<code>\columnseprule</code>	0pt	0pt	0pt	<i>wie links</i>		

Tabelle 4.2: Voreinstellungen der Seitenlayoutparameter (letterpaper)

Referenzpunktes relativ zu den Papierrändern. Diese Datei wurde von Rainer Schöpf für L^AT_EX 2_ε so umgeschrieben, dass man eine Papierformat-Option angeben kann.

4.2 Verändern des Seitenlayouts

Die Werte der Parameter für das Seitenlayout sollte man nur über die Befehle `\setlength` und `\addtolength` verändern. Dabei sollten Änderungen an den geometrischen Parametern nur in Klassen- oder Paketdateien und/oder in der Präambel (d.h. vor dem `\begin{document}`-Befehl) erfolgen. Auch wenn es nicht völlig unmöglich ist, sie mitten in einem Dokument zu ändern, wird daraus doch mit ziemlicher Sicherheit nichts Gutes entstehen, da die internen T_EX-Abläufe sehr komplex und zeitlich voneinander abhängig sind. Wenn man z.B. den Wert von `\textwidth` ändert, kann sich das ganz unerwartet auf den lebenden Kolumnentitel auf der vorhergehenden Seite auswirken.

Vertikale Abstände sollte man möglichst als Vielfaches des T_EX-Parameters `\baselineskip` initialisieren. Dieser Parameter gibt den Abstand zwischen den Grundlinien zweier aufeinanderfolgender Zeilen (Durchschuss) in einem Absatz mit „normalem“ Dokumentenschriftgrad an. Daher kann `\baselineskip` auch als Höhe einer Zeile betrachtet werden. Somit bezieht sich die folgende Einstellung immer auf „zwei Zeilen Text“.

```
\normalsize                                % normales \baselineskip
\setlength\headheight{2\baselineskip} % Kolumnentitel einstellen
```

 *Parameter nur
in der Präambel
ändern*

Damit `\baselineskip` wirklich richtig eingestellt ist, sollte man zunächst (falls erforderlich) die in einem Dokument verwendeten Fonts festlegen und dann den Befehl `\normalsize` aufrufen, um den Schriftgrad auf die im Dokument verwendete Standardgröße, die so genannte Grund- oder Brotschrift, zu initialisieren.

In manchen Fällen ist es einfacher, die Parameter für das Seitenlayout in Anlehnung an bestimmte typographische Vorgaben zu errechnen. Die Vorgabe „der Text sollte 50 Zeilen enthalten“ kann z.B. durch den unten angegebenen Befehl ausgedrückt werden. Dabei geht man davon aus, dass die Höhe aller Zeilen außer der ersten dem Wert `\baselineskip` entspricht, während die Höhe der obersten Zeile des Textbereiches `\topskip` beträgt (das ist die T_EX-Entsprechung zu `\baselineskip` für die erste Zeile, mit einem Standardwert von 10pt).

```
\setlength\textheight{\baselineskip*49+\topskip}
```

Man beachte, dass in den Beispielen in diesem Kapitel das L^AT_EX-Paket `calc` (welches die Notation von Berechnungen erleichtert) und die erweiterten Steuerfunktionen von L^AT_EX 2_ε verwendet werden (siehe Anhang A, Abschnitte A.3.1 und A.3.2).

Eine Vorgabe wie „der Text sollte eine Höhe von 198mm haben“ kann auf ähnliche Weise erfüllt werden. Die entsprechende Berechnung ist weiter unten aufgeführt. Zunächst ist die Anzahl der Zeilen zu berechnen, die ein Textbereich der gewünschten Höhe enthalten kann: Dazu muss ein Dimensionsparameter durch den anderen geteilt werden, um die entsprechende ganze Zahl zu erhalten. Da T_EX diese Operation jedoch nicht direkt durchführen kann, werden die Dimensionsparameter zunächst Zählerregistern zugeordnet. Die dabei benutzte Konvertierung ist sehr präzise, da intern sp-Einheiten (skalierte Punkte, siehe Tabelle A.1 auf Seite 888) verwendet werden.

```
\newcounter{tempc} \newcounter{tempcc} % zwei temp. Zähler
\setlength\textheight % Zielhöhe ohne
    {198mm-\topskip} % erste Zeile
\setcounter{tempc}{\textheight} % Zähler 1 zuweisen
\setcounter{tempcc}{\baselineskip} % Zähler 2 zuweisen
\setcounter{tempc}% % Zähler teilen
    {\value{tempc}/\value{tempcc}}
\setlength\textheight{\baselineskip*\value{tempc}+\topskip}
```

Der vertikale Abstand `\topmargin` für den oberen Rand kann ebenfalls angepasst werden, wenn z.B. der obere Rand nur halb so groß sein soll wie der verbleibende Raum unterhalb des Textbereiches. Die folgende Berechnung zeigt, wie der gewünschte Wert für DIN A4-Papier errechnet wird (Papierhöhe ist 297mm).

```
\setlength\topmargin
    {(297mm-\textheight)/3 - \lin - \headheight - \headsep}
```

Generell gilt: wenn man das Seitenlayout ändert, sollte man einige Grundregeln der Lesbarkeit beachten (siehe z.B. [151]). Untersuchungen gedruckter Texte im englischen Sprachraum haben gezeigt, dass eine Zeile nicht mehr als

zehn bis zwölf Wörter enthalten sollte, was einem Maximum von 60 bis 70 Zeichen pro Zeile entspricht.

Die Anzahl der Zeilen pro Seite ist vom verwendeten Schriftgrad abhängig. Der unten angegebene Code zeigt, wie man die Texthöhe `\textheight` in Abhängigkeit vom normalen Schriftgrad eines Dokumentes berechnet. Dabei ist die Tatsache hilfreich, dass in den meisten Dokumentenklassen der interne L^AT_EX-Befehl `\@ptsize` die jeweilige Grundschrift von 10pt, 11pt oder 12pt in der Form 0, 1 oder 2 kodiert. Dieser Befehl wird gesetzt, wenn man eine Option wie etwa 11pt wählt.

```
\ifthenelse{\@ptsize = 0}%      10 pt-Schrift als Grundschrift
  {\setlength\textheight{53\baselineskip}}{}
\ifthenelse{\@ptsize = 1}%      11 pt-Schrift als Grundschrift
  {\setlength\textheight{46\baselineskip}}{}
\ifthenelse{\@ptsize = 2}%      12 pt-Schrift als Grundschrift
  {\setlength\textheight{42\baselineskip}}{}
\addtolength\textheight{\topskip}
```

Ein weiterer wichtiger Parameter ist die Größe des Weißraums um den Text. Da Druckschriften sehr wahrscheinlich auch gebunden oder geheftet werden, sollte am inneren Rand, eben dem Bundsteg, immer genug Raum dafür gelassen werden. Wenn `\oddsidemargin` festgelegt ist, berechnet sich beim zweiseitigen Druck der Wert für `\evensidemargin` aus der Gleichung:

```
Papierbreite =
  1in + \oddsidemargin + \textwidth + \evensidemargin + 1in
```

In den meisten Klassen wird der zweiseitige Druck durch die Klassenoption `twoside` aktiviert, welche die boolesche Variable `@twoside` auf `true` (wahr) setzt. Mithilfe des Paketes `ifthen` kann man in Abhängigkeit vom Wert dieser Variablen und unter Berücksichtigung der gewählten Grundschrift weitere Parameter einstellen:

```
\ifthenelse{\@ptsize = 0}%      10 pt-Schrift als Grundschrift
  {\setlength\textwidth{5in}%
   \setlength\marginparwidth{1in}%
   \ifthenelse{\boolean{@twoside}}%
     {\setlength\oddsidemargin {0.55in}%      zweiseitig
      \setlength\evensidemargin{0.75in}}%
     {\setlength\oddsidemargin {0.55in}%      einseitig
      \setlength\evensidemargin{0.55in}}%
   }{}
\ifthenelse{\@ptsize = 1}{...}%   11 pt-Schrift als Grundschrift
\ifthenelse{\@ptsize = 2}{...}%   12 pt-Schrift als Grundschrift
```

Wenn ein Dokument viele Marginalien enthält, kann es außerdem angebracht sein, die Ränder des Layouts zu vergrößern. Das (veraltete) Paket `a4` enthält für diese Situation z.B. den Befehl `\WideMargins`. Es stellt die geometrischen Parameter so ein, dass auf Kosten der Textbreite ein Rand von 1,5 Zoll für die Marginalien freigehalten wird (Marginalsatzspalte).

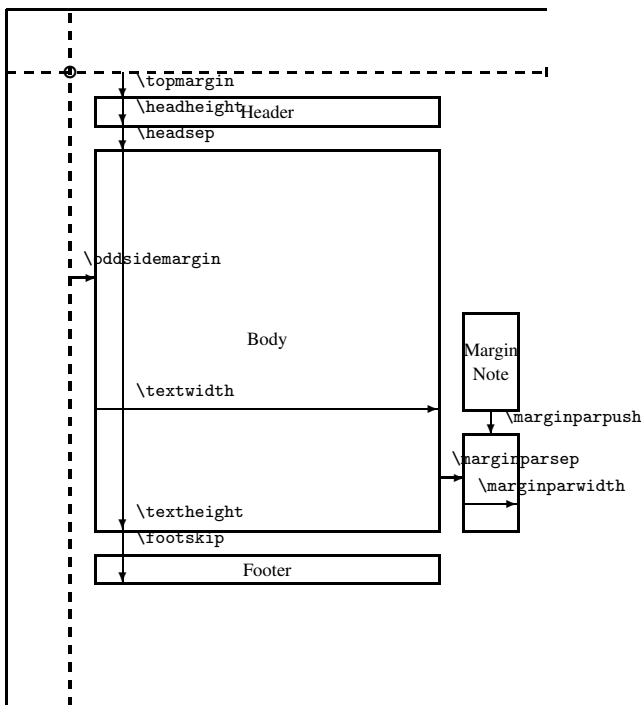
4.2.1 layouts – Darstellen des Layouts

Es gibt zwei Pakete, mit deren Hilfe man die Einstellungen der Layoutparameter visualisieren und mit verschiedenen Werten experimentieren kann: `layout` und `layouts`.

Das Paket `layout` (ursprünglich von Kent McPherson, von Johannes Braams nach $\text{\LaTeX 2}_{\epsilon}$ konvertiert) generiert über den Befehl `\layout` aus den aktuellen Seitenparametern eine um die Hälfte reduzierte Abbildung des Seitenlayouts; mit der Klassenoption `twoside` werden zwei Seiten ausgegeben.

Das Paket `layouts` von Peter Wilson ist flexibler. Es bietet zwei Möglichkeiten: Zum einen lässt sich über den Befehl `\pagediagram` (wie im nächsten Beispiel) eine abstrakte Darstellung der Layoutparameter (ohne Berücksichtigung der aktuellen Werte) anfertigen; zum anderen kann es über den Befehl `\pagedesign` Testlayouts erzeugen, die zeigen, wie sich Testwerte für die einzelnen Parameter auswirken. In beiden Modi kann man mit einer `\setlayoutscales`-Deklaration den Skalierungsfaktor verändern.

The circle is at 1 inch from the top and left of the page. Dashed lines represent `(\hoffset + 1 inch)` and `(\voffset + 1 inch)` from the top and left of the page.



```
\usepackage{layouts}
\setlayoutscales{0.33}
\setparameter{textfont}
{\scriptsize}
\setlabelfont{\scriptsize}
\pagediagram
```

Bsp.
4-2-1

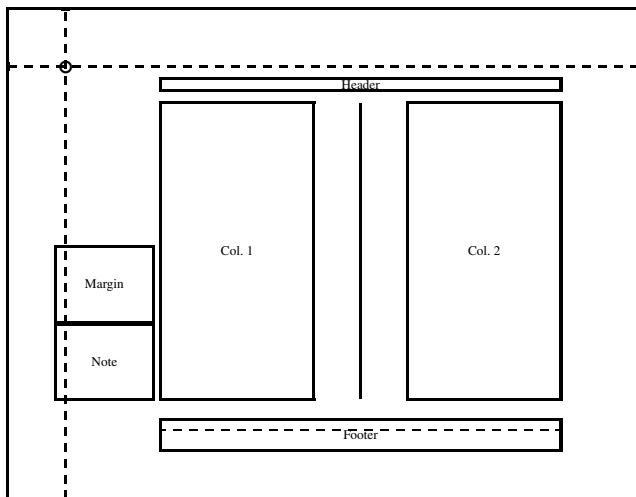
Für ein Testlayout müssen zunächst allen Seitenlayoutparametern geeignete Werte zugewiesen werden. Für jeden Parameter *param* gibt es eine Deklaration `\try{param}`, welche die Testwerte für diesen Parameter als Argument aufnimmt. Der Befehl `\tryheadsep{18pt}` würde z.B. ein Layout erzeugen, in dem `\headsep` auf 18pt eingestellt ist. Außerdem gibt es vier Deklarationen für boolesche Variablen: `\oddpagelayoutfalse`

erzeugt eine „gerade Seite“ (voreingestellt sind ungerade Seiten), die Deklaration `\twocolumnlayouttrue` generiert einen Zweispaltensatz (voreingestellt ist einspaltiger Satz). Der Befehl `\reversemarginpartrue` ahmt die Auswirkungen des $\text{\LaTeX}$ -Befehls `\reversemarginpar` nach und `\marginparswitchfalse` bewirkt, dass Marginalien auf linken und rechten Seiten immer auf der gleichen Seite des Textbereiches bleiben (eine nützliche Einstellung für asymmetrische Layouts, die sich leicht mit dem Paket `geometry` erzeugen lassen; siehe Seite 217).

Um die Eingabe geeigneter Testwerte zu erleichtern, kann man zunächst den Befehl `\currentpage` nutzen. Er übernimmt die aktuell im Dokument verwendeten Werte als Testwerte und stellt auch die booleschen Schalter entsprechend ein.

Der Seitenfuß wird auf die Höhe einer Zeile voreingestellt, da $\text{\LaTeX}$ über keinen expliziten Parameter verfügt, mit dem sich die Boxgröße des unteren Kolumnentitels ändern lässt. Je nachdem welches Kolumnentitel-Layout verwendet wird, kann dieser Wert jedoch ungeeignet sein, wenn das Layout für die Box des unteren Kolumnentitels eine besonders große Tiefe vorgibt. Um in diesem Fall ein (einigermaßen) korrektes Diagramm zu erhalten, kann man mit `\setfootbox`, wie im folgenden Beispiel, die Höhe und Tiefe der Box explizit festlegen.

Das Beispiel zeigt auch, dass man dieses Paket mit dem Paket `calc` kombinieren kann, so dass arithmetische Ausdrücke in den Testdeklarationen möglich sind.



```
\usepackage{calc,layouts}
\setlayoutscales{0.3}
\currentpage
\oddpagelayoutfalse
\twocolumnlayouttrue

\trypaperwidth{11in}
\trypaperheight{8.5in}
\trytextwidth{500pt}
\trytextheight{\topskip
+ 30\baselineskip}
\trycolumnsep{120pt}
\trycolumnseprule{3pt}

\tryheadheight{12pt}
\tryheadsep{18pt}
\tryfootskip{40pt}

\tryevensidemargin{120pt}

\setfootbox{12pt}{24pt}

\setlabelfont{\tiny}
\drawdimensionsfalse
\printheadingsfalse
\pagedesign
```

Lengths are to the nearest pt.

page height = 614pt	page width = 795pt
\hoffset = 0pt	\voffset = 0pt
\evensidemargin = 120pt	\topmargin = 16pt
\headheight = 12pt	\headsep = 18pt
\textheight = 370pt	\textwidth = 500pt
\footskip = 40pt	\marginparsep = 11pt
\marginparpush = 5pt	\columnsep = 120pt
\columnseprule = 3.0pt	

Steuern der Darstellung

Die visuelle Darstellung der gedruckten Seitenlayouts wird von einer Reihe von Befehlen gesteuert, von denen einige schon im vorigen Beispiel verwendet wurden. Die wichtigsten von ihnen werden hier besprochen; weitere Erläuterungen findet man in der Dokumentation des Paketes.

Mit der Deklaration `\setlabelfont` kann man den Schriftgrad für die Textlabel ändern. Mit `\setparameterfont` lässt sich auf ähnliche Weise der Schriftgrad für angezeigte Parameter beeinflussen (z.B. Beispiel 4-2-1 auf Seite 208).

Der vorgegebene englische Text über dem Beispiel kann mit der Deklaration `\printheadingsfalse` unterdrückt werden. Der boolesche Schalter `\printparametersfalse` blendet die tabellarische Auflistung von Parameterwerten unterhalb des Diagramms aus. Eine ähnliche Tabelle kann mit dem Befehl `\pagevalues` einzeln generiert werden.

Mit `\drawdimensionstrue` werden Pfeile eingezeichnet, die zeigen, wo sich die Parameter auswirken (normalerweise ist diese Eigenschaft bei `\pagediagram` ein- und bei `\pagedesign` ausgeschaltet).

Andere Layoutobjekte abbilden

Das Paket `layouts` beschränkt sich nicht auf das Seitenlayout. Es unterstützt auch die Darstellung anderer Objekte. Mithilfe von acht „Diagramm“-Befehlen können weitere L^AT_EX-Layoutparameter visualisiert werden. Der Befehl `\listdiagram` bildet die Parameter ab, die bei der Darstellung von Listen Verwendung finden (er wird in Abbildung 3.3 auf Seite 153 verwendet). Der Befehl `\tocdiagram` zeigt, welche Befehle Inhaltsverzeichnisse und Ähnliches beeinflussen und wie diese zusammenhängen. Parameter für Gleitobjekte werden über `\floatdiagram` und `\floatpagediagram` dargestellt. Parameter für Gliederungsbefehle lassen sich mit `\headingdiagram`, solche für Fußnoten und Absätze mit `\footnotediagram` und `\paragraphdiagram` veranschaulichen. Und der Befehl `\stockdiagram` schließlich erzeugt ein Seitenlayout-Diagramm, ähnlich wie `\pagediagram`, wobei jedoch nur die Parameter der Dokumentenklasse `memoir` und ihrer Abkömmlinge dargestellt werden (vgl. Abschnitt 4.6.2 auf Seite 245).

Es gibt außerdem entsprechende „Design“-Befehle, wie zum Beispiel `\listdesign`, `\tocdesign`, `\floatdesign`, `\floatpagedesign`, `\headingdesign` usw., mit deren Hilfe man verschiedene Parametereinstellungen ausprobieren kann. Jeder Parameter kann über eine Deklaration `\try{param}` eingestellt werden. Eine vollständige Liste der so unterstützten Parameter ist in der Dokumentation zum Paket enthalten. Wenn man die entsprechenden L^AT_EX-Parameter bereits kennt (oder sie aus dem Ergebnis der „Diagramm“-Befehle abliest), kann man sofort anfangen zu experimentieren.

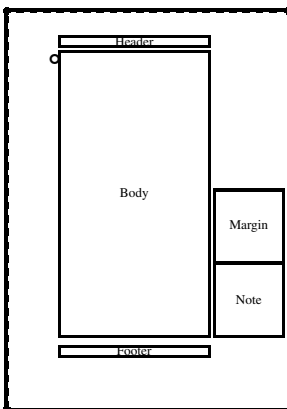
4.2.2 Eine Sammlung von Paketen für Seitenlayouts

Da die Originalklassen von L^AT_EX auf amerikanischen Papierformaten basieren, haben europäische Anwender verschiedene Pakete entwickelt, mit denen die Seitenlayoutparameter auf metrische Formate angepasst werden können. Alle diese Pakete wurden durch das `typearea`- und das `geometry`-Paket (die in den nächsten zwei Abschnitten beschrieben werden) abgelöst. Neue Dokumente sollten immer mit diesen Paketen erstellt werden.

Da frühere Ansätze immer noch in den Archiven zu finden sind, werden sie hier kurz umrissen. Beispiele für derartige Pakete sind `a4` (das relativ

kleine Seiten generiert), das gut dokumentierte Paket `a4dutch` (von Johannes Braams und Nico Poppelier) sowie `a4wide` (von Jean-François Lamy), das etwas längere Zeilen erzeugt. Zudem existieren häufig lokal definierte Dateien unter diesen Namen, was zu einem heillosen Durcheinander führt. Für DIN A5-formatige Seiten gibt es die Pakete `a5` und `a5comb` von Mario Wolczko. Bei all diesen Paketen bestand das Problem darin, dass sie kaum oder gar nicht erlaubten, die Größe und Position des Textbereiches anzupassen. Von einigen gibt es sogar inkompatible Implementierungen.

Das Paket `vmargin` von Volker Kuhlmann folgt einem allgemeineren Ansatz. Sein Paket unterstützt eine Vielzahl von Papierformaten und ermöglicht dem Anwender eine Reihe von Layoutparametern mit einer einzigen Deklaration festzulegen. Die verbleibenden Werte werden aus diesen Angaben errechnet (es gibt verschiedene Deklarationsvarianten). Im folgenden Beispiel werden die Ränder festgelegt und der Satzspiegel berechnet.



Bsp.
4-2-3

```
\usepackage{vmargin}
\setpapersize[portrait]{A5}
\setmarginsrb{80pt}{40pt}% links, oben
                        {120pt}{80pt}% rechts, unten
                        {12pt}{10pt}% Kopf: Höhe, Abst.
                        {12pt}{30pt}% Fuß: Höhe, Abst.
\setlength\marginparwidth{100pt}
% Programmcode um das erhaltene Layout darzustellen:
\usepackage{layouts}
\newcommand\showpage{%
  \setlayoutscales{0.25}\setlabelfont{\tiny}%
  \printheadingsfalse\printparametersfalse
  \currentpage\pagedesign}
\showpage
```

Das Paket hebt intern die voreingestellte Verschiebung des Referenzpunktes um einen Zoll (die normalerweise von den TeX-Ausgabetreibern hinzugefügt wird) auf, indem es negative Werte für `\hoffset` und `\voffset` verwendet. Das kann zu Überraschungen führen. Dieses Verhalten wird im Beispiel dadurch deutlich, dass die gestrichelte Linie, die diese Verschiebung normalerweise anzeigt, hinter dem Blattrand verschwunden ist und nur der Kreis bei (1 Zoll, 1 Zoll) verbleibt.

4.2.3 typearea – Ein traditioneller Ansatz

In Büchern zum Thema Typographie findet man normalerweise einen Abschnitt, der sich mit dem Seitenlayout befasst. Darin werden häufig Konstruktionsmethoden für die Positionierung des Textkörpers beschrieben und das ein oder andere Kriterium für Textbreite, Zeilenanzahl, Verhältnis der Seitenränder zueinander und weitere Gesichtspunkte aufgeführt.

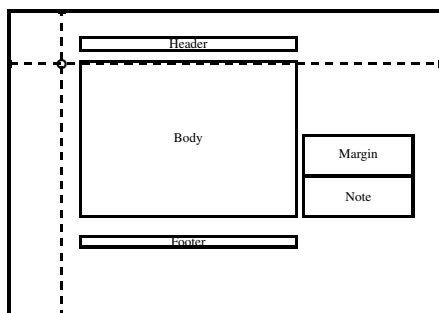
Das Paket `typearea` von Markus Kohm und Frank Neukam, das als Teil der KOMA-Script-Sammlung verteilt wird, stellt Funktionen zur Verfügung, mit deren Hilfe man auf einfache Weise eine der traditionelleren Konstruktionsmethoden für das Seitenlayout einsetzen kann, die seit den frühen Tagen des Buchdrucks in vielen Büchern zur Anwendung kam.

Das bedeutet, ein mithilfe von `typearea` generiertes Seitenlayout weist dem Textbereich das gleiche Größenverhältnis zu, wie es das Papierformat für die Ausgabe des Dokumentes vorgibt. Außerdem ist der Seitensteg doppelt so breit wie der Bundsteg und der Fußsteg doppelt so hoch wie der Kopfsteg.

Bei dieser Konstruktionsmethode wird das Papier horizontal und vertikal in n gleiche Stücke aufgeteilt, von denen jeweils eines für den Kopf- und den Bundsteg und zwei für den Fuß- und den Seitensteg verwendet werden. Normalerweise berechnet das Paket die Variable n automatisch. Sie kann aber auch mithilfe der Option `DIVcalc` explizit abgefragt werden (z.B. um die in der Datei `typearea.cfg` gespeicherte Konfiguration zu überschreiben). Diese Option prüft die Grundschrift und wählt für eine Seite im Hochformat einen Wert, der ungefähr 60–70 Zeichen pro Zeile zulässt. Alternativ dazu kann man, um eine vorgegebene Anzahl von Stücken zu erhalten, den Wert für n über die Option `DIVn` manuell festlegen. Die dritte Möglichkeit besteht in der Option `DIVclassic`, die ein Seitenlayout erzeugt, wie es in bestimmten mittelalterlichen Werken zu sehen ist.

Die Seitenhöhe, die sich aus dem gewählten oder berechneten `DIV`-Wert ergibt, wird automatisch auf eine ganze Anzahl von Zeilen gerundet. Damit dies funktioniert, muss vorher der effektive Wert von `\baselineskip` für das Dokument festgelegt werden. Wenn man also ein Paket wie `setspace` oder den Befehl `\linespread` einsetzt, dann muss das geschehen, bevor `typearea` geladen wird.

Als Papierformat lässt `typearea` alle entsprechenden Optionen der $\text{\LaTeX}$ -Standardklassen zu (siehe Tabelle 4.1 auf Seite 204), sowie alle ISO-A-, ISO-B- und ISO-C-Formate (z.B. `a0paper` oder `c5paper`). Die Orientierung des Textes kann, wie im folgenden Beispiel, mit `landscape` geändert werden.



```
\usepackage[a5paper,landscape,
             DIVcalc]{typearea}
% um das erhaltene Layout darzustellen:
\usepackage{layouts}
\newcommand\showpage{%
  \setlayoutscales{0.27}\setlabelfont{\tiny}%
  \printheadingsfalse\printparametersfalse
  \currentpage\pagedesign}
\showpage
```

Bsp.
4-2-4

Der errechnete `DIV`-Wert wird zusammen mit den für die anderen Seitenparameter gewählten Werten in der `.log`-Datei des $\text{\LaTeX}$ -Laufs gespeichert. Im vorigen Beispiel war dieser Wert 7. Anstelle von `DIVcalc` hätte man also auch `DIV7` verwenden können.

Bestimmen des Satzspiegels

Bisher wurde erklärt, wie das Paket die Größe des Satzspiegels wählt und wie es diesen auf der Seite positioniert. Es wurde noch nicht besprochen, ob die Kolumnentitel bei dieser Berechnung berücksichtigt werden. Die Antwort auf diese Frage hängt von ihrem Inhalt ab. Wenn zum Beispiel der lebende Kolumnentitel am Kopf der Seite sehr umfangreich ist, vielleicht zusätzlich mit einer Linie unterstrichen, und damit erheblich zum Grauwert der Seite beiträgt, sollte er als Teil des Satzspiegels betrachtet werden. In anderen

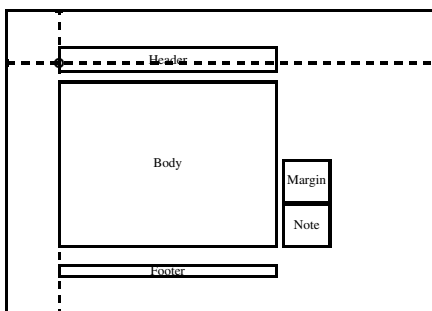
Fällen kann es zutreffender sein, ihn den Rändern zuzurechnen (z.B. wenn er aus sehr lichtem Text mit kleinem Schriftgrad besteht). Aus dem gleichen Grund sollte ein toter Kolumnentitel am Fuß der Seite, der nur eine Seitenzahl (genannt *Pagina*, *Folio* oder *Kolumnenziffer*) enthält, als außerhalb des Satzspiegels angesehen werden und damit nicht in die Berechnungen für dessen Positionierung einfließen.

Für ein bestimmtes Dokument können diese Optionen mit `headinclude`, `footinclude`, `headexclude` und `footexclude` explizit ausgewählt werden. Die letzten beiden Optionen entsprechen der Voreinstellung. Bei hohen `DIV`-Werten (d.h. schmalen Rändern) können die Kolumnentitel über die Seitenränder hinausgehen, wenn sie bei der Berechnung nicht berücksichtigt werden. In diesem Falle müssen die Einstellungen nachgebessert werden.

Genauso lässt sich der Marginalienbereich `\marginpar` (mithilfe von `mpinclude` und `mpexclude`) für den linken und rechten Rand ein- bzw. ausschließen. Er wird normalerweise ebenfalls nicht berücksichtigt, aber wenn ein Layout viele Marginalien enthält, kann es angebracht sein, ihn einzubeziehen.

Der obere Kolumnentitel hat eine Standardhöhe von 1,25 Zeilen. Dieser Wert lässt sich mit einer Option der Art *numheadlines* einstellen, wobei *num* eine Dezimalzahl (z.B. 2.3) ist, die angibt, über wie viele Zeilen sich der Kopf erstrecken soll.

Im nächsten Beispiel sind der obere Kolumnentitel und die Marginalien einbezogen, wobei der obere Kolumnentitel auf 2,5 Zeilen vergrößert ist. Zum Vergleich empfiehlt sich das in Beispiel 4-2-4 auf der gegenüberliegenden Seite dargestellte Layout, bei dem Kolumnentitel und Marginalien aus dem Satzspiegel ausgeschlossen sind.



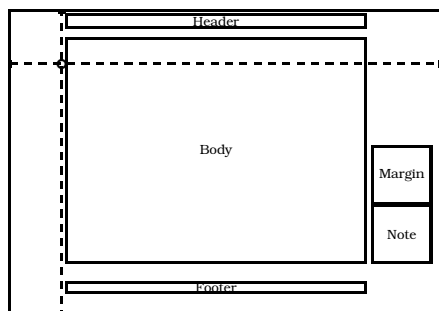
Bsp.
4-2-5

```
\usepackage[a5paper,landscape,
2.5headlines,
headinclude,mpinclude,
DIVcalc]{typearea}
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Je nach Art der Bindung des fertigen Werkes wird der innere Rand, der Bundsteg, noch mehr oder weniger verschmälert. Um diesen Verlust an Weißraum auszugleichen unterstützt das Paket die Option `BCOR{val}`, wobei *val* (in einer beliebigen $\text{\LaTeX}$ -Einheit) der Raum ist, den die Bindung einnimmt. Ein Wert von `BCOR1.2cm` würde die Seitenbreite z.B. um 1,2cm reduzieren, bevor das Seitenlayout berechnet wird.

Neben der Anpassung des Layouts durch Paketoptionen kann man die Parameter auch mit dem Befehl `\typearea` berechnen; die KOMA-Script-Dokumentation enthält hierzu nähere Details. Diese Möglichkeit ist insbesondere dann nützlich, wenn eine Dokumentenklasse der KOMA-Script-Sammlung das `typearea`-Paket bereits lädt, und man in der Präambel des Dokumentes eine ungewöhnliche Schriftart als Grundschrift angeben möchte. In

diesem Fall muss das Layout unter Berücksichtigung des ausgewählten Fonts neu berechnet werden.



```
\usepackage[a5paper,landscape]{typearea}
\usepackage{bookman}

% Syntax: \typearea[<bindungs-korr.>]{<stücke>}
\typearea[10mm]{11}

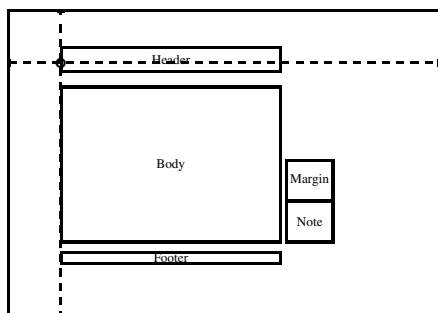
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Bsp.
4-2-6

4.2.4 geometry – Layouts mit Auto-Vervollständigung

Das geometry-Paket von Hideo Umeki bietet eine universelle und leicht zu bedienende Schnittstelle für alle geometrischen Belange des Seitenlayouts. Es nutzt das Paket keyval, so dass alle Parameter (und ihre Werte) als Optionen von \usepackage deklariert werden können.

Anders als das typearea-Paket verwendet geometry kein bestimmtes typographisches Konzept, sondern setzt die eingegebenen Werte wie gewünscht um. Es kennt jedoch das Verhältnis verschiedener Seitenparameter zueinander und kann bei unvollständigen Angaben die restlichen Parameter automatisch berechnen. Das nächste Beispiel zeigt ein Layout, das dem mit typearea erzeugten Beispiel 4-2-5 auf der vorherigen Seite sehr ähnelt. Hier werden eine Reihe von Werten explizit festgelegt (z.B. für den Kopf- und den Bundsteg), der Satzspiegel wird jedoch automatisch aus dem Papierformat (a5paper), den Werten für den Kopfsteg (tmargin) und den Bundsteg (lmargin) und einem expliziten Verhältnis der Seitenränder von 1:2 (marginratio) berechnet.



```
\usepackage[marginratio=1:2,
  paper=a5paper,landscape=true,
  tmargin=52pt,lmargin=74pt,
  headheight=30pt,marginparwidth=62pt,
  includehead,includemp]{geometry}
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Bsp.
4-2-7

Das Beispiel zeigt außerdem, dass boolesche Optionen auch ohne Wert verwendet werden können (so dass die Voreinstellung =true greift); für alle anderen Optionen müssen Werte angegeben werden.

Im Folgenden werden die verschiedenen Aspekte des Seitenlayouts erklärt, die von geometry unterstützt werden. In den meisten Fällen führen mehrere Wege zum gleichen Ergebnis, da einige Parameter bestimmten Beziehungen gerecht werden müssen. Verstößt eine Spezifikation gegen eine solche

Beziehung, dann gibt `geometry` eine Warnung aus und ignoriert dann die eine oder andere Einstellung.

Das Papierformat kann mit der Option `paper` eingestellt werden, welche die Werte `a0paper` bis `a6paper`, sowie `b0paper` bis `b6paper` akzeptiert. Daneben können auch die Werte `letterpaper`, `legalpaper` und `executivepaper` verwendet werden. Als Kurzform kann man auch einfach das gewünschte Papierformat als Optionsnamen verwenden; die Option `a5paper` entspricht zum Beispiel der Einstellung `paper=a5paper`.

Papierformate

Für die Ausgabe auf dem Bildschirm gibt es die Option `screen`. Bei Formaten, die keinem Standard entsprechen, können die Maße des Papiers explizit mit den Optionen `paperwidth` und `paperheight` spezifiziert werden.

An allgemeinen Seiteneigenschaften unterstützt das `geometry`-Paket die booleschen Optionen `twoside` (zweiseitiger Satz), `landscape` (Papierhöhe und -breite vertauschen), und `portrait`. Dabei ist `portrait=false` das gleiche wie `landscape`.

*Allgemeine
Seiteneigenschaften*

Wenn durch die Bindemethode ein Teil der Seite verschwindet, kann man diesen Verlust an Weißraum über die Option `bindingoffset` ausgleichen. Der angegebene Wert wird zum Bundsteg hinzugefügt.

Die boolesche Option `twocolumn` bewirkt, dass der Satzspiegel für Zweispaltensatz eingerichtet wird. In diesem Falle wird der Spaltensteg (der Abstand zwischen den Spalten) mithilfe der Option `columnsep` festgelegt.

In Abschnitt 4.2.3, in dem das `typearea`-Paket beschrieben wird, wurde erklärt, dass es je nach Art eines Dokumentes angebracht sein kann, die lebenden Kolumnentitel (und in manchen Fällen sogar die Marginalsatzspalte) dem Satzspiegel zuzurechnen. Das Paket `geometry` ist so voreingestellt, dass es Kolumnentitel und Marginalien nicht mit einbezieht. Da diese Einstellungen das Verhältnis von Textbereich und Rändern zueinander verändern und sich damit auf automatisch berechnete Werte auswirken, sollten sie passend festgelegt werden. Die Voreinstellungen lassen sich mithilfe einer Reihe boolescher Optionen¹ ändern: `includemp` schließt die Marginalien mit ein, was jedoch selten erforderlich ist; `includehead` wird bei schweren lebenden Kolumnentiteln am Seitenkopf verwendet; `includefoot` ist kaum je erforderlich, da der Kolumnentitel am Fuß der Seite normalerweise nur die Kolumnenziffer (Seitenzahl) enthält; und `includeheadfoot` und `includeall` sind Abkürzungen für Kombinationen der anderen Optionen.

*Bestandteile des
Satzspiegels*

Fußnoten werden immer als Teil des Satzspiegels angesehen. Mit der Option `footnotesep` wird nur der Abstand zwischen der letzten Textzeile und den Fußnoten festgelegt; die Berechnung der Ränder bleibt davon unbeeinträchtigt.

Die Größe des Satzspiegels lässt sich nach mehreren Methoden festlegen; es ist weitgehend eine Frage der persönlichen Vorliebe, welche dieser Methoden man verwendet. Man kann sie bestimmen, indem man explizit Werte für `textwidth` und `textheight` vorgibt. In diesem Fall muss `textheight` normalerweise eine ganzzahlige Anzahl an Textzeilen enthalten, damit es bei Seiten, die ausschließlich Text enthalten, nicht zu „Underfull box“-Warnungen kommt. Für diesen Zweck lässt sich sehr gut die `lines`-Option einsetzen, die mithilfe von `\baselineskip` und `\topskip` einen geeigneten Wert für `\textheight` errechnet.

Satzspiegel

¹Das Paket `typearea` bietet die gleiche Funktionalität mit ähnlichen (aber effektiv doch anderen) Optionsnamen, wie `headinclude` anstatt `includehead`.

Alternativ kann man die boolesche Option `heightrounded` verwenden, die `geometry` veranlasst, den Wert von `\textheight` passend zu runden. Diese boolesche Option ist besonders nützlich, wenn der Satzspiegel automatisch berechnet wird – zum Beispiel, wenn man nur für einige der Ränder Werte angibt und die Größe der übrigen dem Paket überlässt.

Mit den zuvor genannten Optionen wird die Größe des Textbereiches vorgegeben und das Paket errechnet durch Hinzufügen der Kolumnentitel und/oder Marginalien daraus die Werte für den Satzspiegel. Stattdessen kann man auch Werte für den gesamten Satzspiegel vorgeben und das Paket daraus den Textbereich berechnen lassen. Dazu werden die Optionen `width` und `height` verwendet (dieser Ansatz macht natürlich nur einen Unterschied zum vorhergehenden, wenn man die Kolumnentitel in den Satzspiegel einschließt). Bei dieser Methode sollte man das Paket mithilfe von `heightrounded` den erforderlichen Wert für `\textheight` berechnen lassen.

Wenn man nicht gerne feste Werte vorgibt, sondern lieber den Satzspiegel vom Papierformat abhängig macht, kann man dazu die Optionen `hscale` und `vscale` verwenden. Sie bestimmen den horizontalen bzw. vertikalen Anteil der Seitengröße, der vom Satzspiegel eingenommen werden sollte.

Ränder Die Größe der Ränder kann mithilfe der Optionen `lmargin`, `rmargin`, `tmargin` und `bmarg` (für den Bund-, Seiten-, Kopf- bzw. Fußsteg) explizit festgelegt werden. Wenn die boolesche Option `twoside` den Wert `true` hat, beziehen sich `lmargin` und `rmargin` auf den mittleren und den äußeren Rand, so dass die Optionsnamen etwas irreführend sind. Um dem gerecht zu werden unterstützt das Paket alternativ auch die Optionsnamen `inner` und `outer` – sie beziehen sich jedoch auf die gleichen Optionen. Zusammen mit der Option `asymmetric` für asymmetrische Layouts, die im Folgenden beschrieben wird, wären sie erneut missverständlich. Um dem Anwender noch mehr freie Auswahl zu bieten, gibt es zusätzlich einen weiteren Satz von Optionsnamen: `left`, `right`, `top` und `bottom`. Wenn man nur Vorgaben für Verso-Seiten (linke oder gerade Seiten) machen will (und die Recto-Seiten [rechte oder ungerade Seiten] automatisch mithilfe der Optionen `twoside` oder `asymmetric` erzeugen lässt), ist die erste oder die letzte Gruppe von Optionsnamen wahrscheinlich die beste Wahl.

Wenn keine oder nicht alle Werte für die Ränder vorliegen, werden die fehlenden errechnet. Mit der Gleichung

$$\text{paperwidth} = \text{left} + \text{width} + \text{right} \quad (4.1)$$

$$\text{paperheight} = \text{top} + \text{height} + \text{bottom} \quad (4.2)$$

lässt sich aus zwei Werten auf der rechten Seite der dritte Wert bestimmen (anstatt mit `width` oder `height` kann der Satzspiegel auch mit einer der zuvor besprochenen Methoden festgelegt werden). Wenn auf der rechten Seite nur ein Wert vorliegt, verwendet das Paket zwei weitere Gleichungen um die Anzahl der freien Variablen zu reduzieren:

$$\text{left/right} = \text{hmarginratio} \quad (4.3)$$

$$\text{top/bottom} = \text{vmarginratio} \quad (4.4)$$

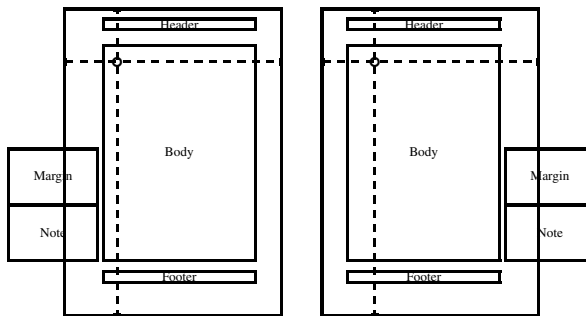
Dabei ist die Option `hmarginratio` auf 2:3 eingestellt, wenn `twoside true` ist, sonst auf 1:1. Voreinstellung für `vmarginratio` ist immer 2:3.

Die Werte für diese „Verhältnis“-Optionen unterliegen folgenden Beschränkungen: Beide Werte müssen positive ganze Zahlen unter 100 sein, die durch einen Doppelpunkt getrennt sind; man schreibt also beispielsweise 4:5, nicht jedoch 1:1.25.

Mit der Option `centering` wird der Satzspiegel zentriert. Das ist ein schneller Weg `hmarginratio` und `vmarginratio` jeweils auf 1:1 zu setzen.

In den L^AT_EX-Standardklassen erfüllt die Option `twoside` zwei Funktionen: Sie bereitet die Kolumnentitel auf unterschiedliche Inhalte für gerade und ungerade Seiten vor und stellt automatisch ein symmetrisches Layout ein, bei dem auf geraden Seiten der linke und rechte Rand (inklusive der Marginalien) vertauscht werden. Das Ergebnis ist aus dem nächsten Beispiel ersichtlich, das auch zeigt, dass die `geometry`-Voreinstellungen einen sehr großen Textbereich erzeugen, dabei jedoch die Größe der Marginalboxen nicht an den verbleibenden Rand anpassen.

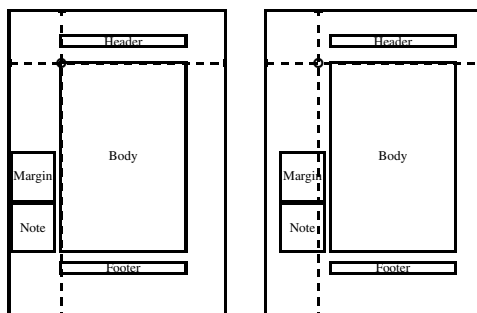
Asymmetrische und symmetrische Layouts



```
\usepackage[a6paper,twoside]
{geometry}
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage \newpage \showpage
```

Bsp.
4-2-8

Das `geometry`-Paket erlaubt es, asymmetrische Layouts einfach mithilfe der Option `asymmetric` festzulegen. Der Einsatz von `bindingoffset` im nächsten Beispiel beweist, dass wirklich ein asymmetrisches zweiseitiges Layout erzeugt wird, da der innere und nicht immer der linke Rand verändert wird, obwohl die Marginalien immer links erscheinen. Da der größere Rand links sein soll, muss `hmarginratio` entsprechend geändert werden. Auf den ersten Blick erscheint der rechte Rand der Verso-Seiten im Verhältnis zur Marginalspalte bei einem Wert von 2:1 für `hmarginratio` vielleicht als zu groß; das liegt jedoch daran, dass dort der zusätzliche Bundsteg von `bindingoffset` hinzugefügt wird.



```
\usepackage[a6paper,asymmetric,
bindingoffset=18pt,
marginparwidth=.8in,reversemp,
hmarginratio=2:1,vmarginratio=4:5,
left=1in,top=1in]{geometry}
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage \newpage \showpage
```

Bsp.
4-2-9

Lebende Kolumnentitel

Die Maße des oberen Kolumnentitels und sein Abstand zum Text können mit den Optionen `headheight` und `headsep` festgelegt werden. Der Abstand zwischen Textbereich und unterem Kolumnentitel wird über `footskip` gesteuert. Außerdem kann man die entsprechenden Werte mithilfe der booleschen Optionen `nohead`, `nofoot` und `noheadfoot` auf null setzen. In den meisten Fällen ist es jedoch besser, `ignorehead` und die verwandten Optionen zu verwenden, da diese erlauben, auf einzelnen Seiten Kolumnentitel hinzuzufügen, ohne dass die Berechnung der Ränder davon beeinträchtigt wird.

Marginalien

Da die meisten Dokumente nur wenige Marginalien enthalten, wird der Raum, den sie einnehmen, normalerweise bei der Berechnung der Ränder nicht berücksichtigt. Dieser Raum lässt sich mit `marginparwidth` und sein Abstand zum Text mit `marginparsep` festlegen. Solange `includemp` nicht verwendet wird, liegt es beim Anwender, sicherzustellen, dass dieser Bereich in den errechneten oder festgelegten Rand passt.

Die Marginalien sind so voreingestellt, dass sie im Seitensteg erscheinen. Diese Einstellung lässt sich mit der booleschen Option `reversemp` umkehren.

Verschiedene Funktionen

Anstelle eines externen Paketes wie `layouts` kann `geometry` die erzeugten Seiten auch selbst über die eingebaute Option `showframe` darstellen. Alle Einstellungen, einschließlich der berechneten Werte, werden normalerweise in der Protokolldatei des aktuellen $\text{\LaTeX}$ -Laufes gespeichert. Mithilfe der booleschen Option `verbose` werden diese Einstellungen außerdem zusätzlich am Bildschirm angezeigt.

Einige $\text{\TeX}$ -Erweiterungen oder Gerätetreiber wie `pdf $\text{\TeX}$` oder `V $\text{\TeX}$` benötigen die Maße des Papierformates, auf dem die Ausgabe erfolgen soll. Dem trägt das `geometry`-Paket durch die Optionen `pdftex`, `vtex`, `dvipdfm` und `dvips` Rechnung, von denen eigentlich immer eine ausgewählt werden sollte. Wird ein Dokument mit dem Programm `pdf $\text{\TeX}$` generiert, so wird automatisch die Option `pdftex` gewählt (und alle anderen ausgeschaltet).

Wie die meisten modernen Pakete unterstützt `geometry` die erweiterte Syntax des Paketes `calc`, wenn letzteres vor `geometry` geladen wird.

Um ein ungewöhnliches Druckerverhalten auszugleichen, verfügt $\text{\LaTeX}$ über die zwei Dimensionsparameter `\hoffset` und `\voffset`, welche die Ausgabe (für alle Seiten) um den angegebenen Wert horizontal nach rechts und vertikal nach unten verschieben. Das Paket spricht diese Variablen über die Optionen `hoffset` und `voffset` an. Sie haben keinerlei Auswirkungen auf die Berechnung der übrigen Seitenmaße.

Vergrößerung

$\text{\TeX}$ verfügt über eine Vergrößerungsfunktion, die alle verwendeten Maße und Fonts um einen angegebenen Faktor vergrößert. In $\text{\LaTeX}$ ist diese Funktion normalerweise deaktiviert, doch das `geometry`-Paket stellt sie dem Anwender über die Option `mag` wieder zur Verfügung. Ihr Wert muss eine ganze Zahl sein, wobei 1000 keine Vergrößerung bedeutet. Der Wert `mag=1414` würde bei dem Papierformat `a5paper` einen Ausdruck im Format `a4paper` erzeugen, da alle Maße um $1,414(=\sqrt{2})$ vergrößert werden, also um den Faktor, um den sich die aufeinanderfolgenden Formate der ISO-A-Serie voneinander unterscheiden. Dies kann nützlich sein, wenn man zum Beispiel den Ausdruck nachträglich photomechanisch verkleinern möchte, um eine höhere Auflösung für den Druck zu erhalten. Da diese Option Fonts skaliert, anstatt Originalfonts in der entsprechenden Entwurfsgröße einzusetzen, ist es normalerweise nicht sinnvoll, die Vergrößerung als Endergebnis weiter zu verwenden.

Beim Vergrößern kann man $\TeX$ anweisen, bestimmte Dimensionen in ihrer Originalgröße zu belassen, indem man den Maßeinheiten die Zeichenfolge `true` voranstellt. Die Eingabe `left=1truein` würde beispielsweise einen linken Rand in der exakten Größe von einem Zoll belassen, egal welcher Vergrößerungsfaktor angegeben wurde. Implizit festgelegte Maße (wie z.B. Papierformate, die über die Option `paper` definiert wurden) werden normalerweise ebenfalls vergrößert, es sei denn, man wählt die Option `true`dimensionen.

Mit den zuvor beschriebenen Optionen kann man individuelle Werte festlegen. Für die häufigsten Fälle bietet `geometry` außerdem kombinierte Optionen. Mit ihrer Hilfe lassen sich mehrere Werte auf einmal einstellen: Entweder durch einen einzelnen Wert (der mehrfach verwendet wird) oder als eine durch Kommas getrennte Liste von Werten (die in geschweiften Klammern stehen muss, damit die Kommas nicht fälschlicherweise als Begrenzungszeichen für Optionen interpretiert werden).

Kurzbefehle

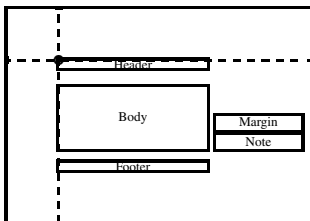
Die Option `papersize` erhält eine Liste von zwei Maßen für die horizontalen und vertikalen Abmessungen der Seite.

Die Option `hmargin` stellt den Bund- und den Seitensteg ein, entweder auf denselben Wert (wenn nur einer angegeben ist) oder auf unterschiedliche Werte (wenn eine Liste angegeben ist). Auf die gleiche Weise stellt `vmargin` den Kopf- und Fußsteg ein. Dieses Vorgehen kann mit der Option `margin` manchmal noch mehr verkürzt werden. Sie gibt ihren Wert (oder die Liste) an `hmargin` und `vmargin` weiter. Genauso gibt `marginratio` seinen Wert zur weiteren Verarbeitung an `hmarginratio` und `vmarginratio` weiter.

Die Abmessungen des Textbereiches können mit der Option `body` festgelegt werden, die einen oder zwei Werte erhält, mit denen sie `textwidth` und `textheight` einstellt. Alternativ dazu kann man auch die Option `total` verwenden, die `width` und `height` bestimmt. Außerdem kann man mit der Option `scale` einen oder zwei Skalierungsfaktoren für `hscale` und `vscale` zur Verfügung stellen.

Wenn das Paket `geometry` als Teil einer Klasse zum Einsatz kommt, möchte man vielleicht einige seiner Einstellungen in der Präambel des eigenen Dokumentes überschreiben. Hier ist die `\usepackage`-Option keine große Hilfe, da das Paket ja bereits geladen ist. Für diese Fälle hält es den Befehl `\geometry` bereit, dessen Argument eine durch Kommas getrennte Liste von Optionen erhält. Der Befehl kann immer wieder aufgerufen werden und überschreibt jedes Mal die zuvor getroffenen Einstellungen. Im nächsten Beispiel wird gezeigt, wie man ihn einsetzt: Zunächst wird das Paket geladen, alle Ränder werden auf einen Zoll gesetzt und die Kolumnentitel und Marginalien als Teil des Satzspiegels festgelegt; danach wird der rechte Rand auf zwei Zoll geändert und die Marginalien werden aus der Berechnung herausgenommen.

Verwendung in der Präambel



```
\usepackage[a6paper,landscape,
            margin=1in,includeall]{geometry}
```

```
% Überschreiben einiger Werte:
\geometry{right=2in,ignoremp}
```

```
\usepackage{layouts}
% \showpage wie zuvor definiert
\showpage
```

Zwei weitere Optionen können sich beim Umgang mit der `\geometry`-Schnittstelle als nützlich erweisen: Mit `reset` stellt man die Voreinstellungen des Paketes wieder her, und bei `pass` wird das Paket deaktiviert.

4.2.5 lscape – Setzen einzelner Seiten im Querformat

Bei den meisten Dokumenten ist die längere Papierseite die Vertikale (beim so genannten *Hochformat*). Für manche Dokumente, wie z.B. Folien oder Tabellen, ist jedoch das *Querformat*, bei dem die längere Seite horizontal liegt, besser geeignet. Moderne Drucker und dvi-Treiber können normalerweise beide Seitenformate drucken.

Hoch- und Querformat benötigen unterschiedliche Seitenlayouts, und Pakete wie `geometry` liefern die Werkzeuge, um diese passend zu erzeugen. Manchmal ist es jedoch wünschenswert, nur für einzelne Seiten zwischen Hoch- und Querformat zu wechseln. In diesem Fall helfen die bisher besprochenen Pakete nicht weiter, da sie ein Seitenlayout für das gesamte Dokument festlegen.

Stattdessen kann man hier das Paket `lscape` von David Carlisle einsetzen, das die Umgebung `landscape` definiert, mit deren Hilfe man eine ausgewählte Gruppe von Seiten im Querformat setzen kann, ohne dass die Kolumnentitel ihre Position verändern. Es beendet zunächst mit `\clearpage` die aktuelle Seite, so dass zuerst alle aufgelaufenen Gleitobjekte gesetzt werden. Dann vertauscht es intern die Werte von `\textheight` und `\textwidth` und dreht alle erzeugten Satzspiegel in seinem Geltungsbereich um 90 Grad. Für die Drehung verwendet es das Paket `graphics`, so dass es mit jedem von diesem Paket unterstützten Gerätetreiber funktioniert, der Rotationen ausführen kann. Wenn die Umgebung endet, wird ein weiterer `\clearpage`-Befehl ausgegeben, bevor wieder zum Hochformat zurückgekehrt wird.

Um Gleitobjekte mit oder ohne ihre Legenden zu drehen, ist das Paket `rotating` die bessere Wahl. Es wird in Abschnitt 6.3.3 beschrieben.

4.2.6 crop – Erzeugen von Beschnittmarken

Wenn man eine reprofähige Vorlage erzeugt, wird der endgültige Druck normalerweise auf überformatigem Papier, dem sogenannten „Rohbogen“, aufgebracht. In diesem Fall muss das bedruckte Papier noch zugeschnitten werden, bevor es gebunden werden kann. Damit der Schnitt exakt wird, benötigen Druckereien normalerweise so genannte Beschnittmarken auf jeder Seite. Wenn bei Mehrfarbdrucken mehrere logische (einfarbige) Seiten zu einer mehrfarbigen Seite übereinander gedruckt werden, benötigt man ebenfalls Markierungen, die in diesem Fall Passermarken genannt werden.

Das Paket `crop` von Melchior Franz bietet für diese Anforderungen eine einfache Schnittstelle, mit der verschiedene Arten von Beschnitt- und Passermarken gesetzt werden können. Mit seiner Hilfe lassen sich auch wahlweise nur der Text oder die Abbildungen eines Dokumentes drucken, und die Ausgabe unter anderem invertieren, spiegeln oder drehen – Funktionen, die in dieser Phase des Druckprozesses nützlich sein können.

Beschnitt- oder Passermarken können mit einer der nachfolgend beschriebenen Optionen angefordert werden.

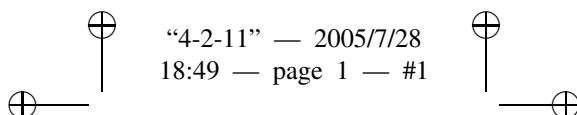
cam Erzeugt acht Marken, welche die Maße der Nettoseite anzeigen, ohne sie zu berühren (siehe Beispiel 4-2-11). Sie dienen hauptsächlich der Ausrichtung der Kamera.

cross Erzeugt vier große Kreuze an den Ecken der logischen Seite, welche die Kanten der logischen Seite berühren.

frame Erzeugt einen Rahmen um die Nettoseite; dieser soll hauptsächlich die Abmessungen der Seite verdeutlichen.

Das Paket geht davon aus, dass `\paperheight` und `\paperwidth` die korrekten Maße der *logischen* Seite (Nettoseite) wiedergeben. Die *tatsächliche* Papiergröße (des Rohbogens) für den Druck wird dem Paket als Option übergeben. Diese Optionen sind `a0`, `a1`, `a2`, `a3`, `a4`, `a5`, `a6`, `b0`, `b1`, `b2`, `b3`, `b4`, `b5`, `b6`, `executive`, `legal` und `letter`. Wenn das Druckpapier im Querformat verwendet wird, kann man dies zusätzlich mit der Option `landscape` angeben. Wenn keine dieser Optionen zu den tatsächlichen Papiermaßen passt, lassen sich die genauen Maße mit den Optionen `width` und `height` festlegen, die beide Dimensionswerte aufnehmen.

Das nächste Beispiel erzeugt mithilfe des `geometry`-Paketes eine unnatürlich kleine Seite (die in den Beispielbereich in diesem Buch passt) und zentriert diese auf einer Rohseite im Format DIN A5. Da aber alle Beispiele nachträglich auf ihre „sichtbare“ Größe zugeschnitten werden und die Ränder der DIN A5-Seite aus ersichtlichem Grund nicht Teil des Beispiels werden, kann man nicht sehen, dass die Beispielseite in einem Arbeitsgang korrekt zentriert wurde – man muss es entweder glauben oder selbst ausprobieren.



Text für den Satzspiegel
 um sein Verhältnis zu den Beschnittmarken aufzuzeigen.



```
\usepackage{graphicx,geometry}
\geometry{paperwidth=2in,
  paperheight=1.3in,
  margin=5mm}
\usepackage[cam,a5,center]{crop}
Text für den Satzspiegel
\includegraphics[width=8mm]
  {cat.ps}
um sein Verhältnis zu den
Beschnittmarken aufzuzeigen.
```

Bsp.
4-2-11

Die Beschreibung und das Beispiel sollten deutlich machen, dass `crop` erst *nach* Festlegung des Layouts für das Dokument geladen werden sollte.

Der Informationstext zwischen den oberen Beschnittmarken wird automatisch eingefügt. Er kann mit der Option `noinfo` unterdrückt werden, aber normalerweise ist es nützlich, ihn beizubehalten. Er enthält sowohl die Seitenanzahl (wie sie $\text{\LaTeX}$ bekannt ist) und einen Seitenindex, der mit 1 beginnt und für jede gedruckte Seite hochgezählt wird. Vor allem bei umfangreichen Publikationen, die verschiedene Methoden zur Seitennummerierung verwenden, kann dieser Ausdruck verhindern, dass die Seiten durcheinander geraten.

Verschiedene Optionen des `crop`-Paketes verlassen sich auf Unterstützung durch den Druckertreiber. Wenn eine explizite Angabe zum Treiber fehlt, versucht das Paket, den Druckertreiber aus den Installationseinstellungen für das `graphics`- oder das `color`-Paket zu bestimmen. Es ist aber auch möglich, den Treiber mit Optionen wie `dvips`, `pdflatex` oder `vtex` explizit anzugeben. Ist eine dieser Optionen ausgewählt, dann wird das Papierformat an den externen Treiber übergeben; das ist wichtig, wenn man das Dokument mit `ghostview` oder ähnlichen Programmen ansehen möchte.

Wenn man Graphiken gesondert ausdrucken will – zum Beispiel, weil der Ausdruck des gesamten Dokumentes auf einem Farbdrucker nicht sinnvoll ist – kann man unterschiedliche Versionen des gleichen Dokumentes erzeugen: eine Version, die nur Text und keine Graphiken enthält (genauer gesagt, ohne Graphiken, die mit `\includegraphics` eingebunden sind) und eine, die nur die Graphiken enthält (da der gesamte Text in der Farbe „weiß“ formatiert wurde). Diese Effekte werden mit den Optionen `nographics` bzw. `notext` erreicht. Die letzte Option kann natürlich nur genutzt werden, wenn der verwendete Treiber Farbbefehle unterstützt, da intern das `color`-Paket eingesetzt wird. Das nächste Beispiel¹ zeigt, wie sich die Optionen `nographics` und `cross` auswirken; vergleiche die Ausgabe von Beispiel 4-2-11.

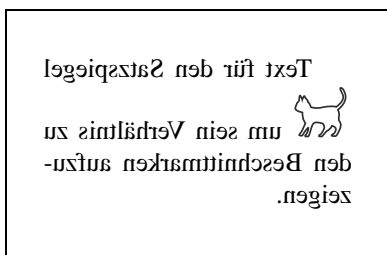
	“4-2-12” — 2005/7/28 18:49 — page 1 — #1	
	Text für den Satzspiegel um sein Verhältnis zu den Beschnittmarken aufzu- zeigen.	<pre>\usepackage{graphicx,geometry} \geometry{paperwidth=2in, paperheight=1.3in, margin=5mm} \usepackage[cross,a5,nographics] {crop}</pre>
		Text für den Satzspiegel <pre>\includegraphics[width=8mm] {cat.ps}</pre> um sein Verhältnis zu den Beschnittmarken aufzuzeigen.

Bsp.
4-2-12

Drei weitere Optionen erfordern, dass der Gerätetreiber die erweiterten Befehle der Pakete `graphics` und `color` zum Drehen und Spiegeln und zum Einfärben des Hintergrundes unterstützt. Mithilfe der Option `rotate` werden die Seiten um 180 Grad gedreht. Die Option `mirror` spiegelt jede Seite, wie im nächsten Beispiel. Und die Option `invert` vertauscht weiß und schwarz, so dass der Text weiß auf schwarzem Hintergrund erscheint.

¹Die kreuzförmigen Beschnittmarken sehen in dieser Größe zugegebenermaßen ziemlich seltsam aus.

82\72002\7\52 — “4-2-13”
 1# — 1 page — 18:49



```
\usepackage{graphicx,geometry}
\geometry{paperwidth=2in,
          paperheight=1.3in,
          margin=5mm}
\usepackage[frame,a5,mirror]{crop}
Text für den Satzspiegel
\includegraphics[width=8mm]
{cat.ps}
um sein Verhältnis zu den
Beschnittmarken aufzuzeigen.
```

Bsp.
4-2-13

4.3 Dynamische Seitendaten: Seitenzahlen und Textmarken

Die Ausgaberroutine von $\text{\LaTeX}$, welche die formatierten Seiten erzeugt, arbeitet asynchron. Das heißt, $\text{\LaTeX}$ sammelt genug Material, um gut eine Seite zu füllen, bereitet es auf und baut daraus die Seite auf. Dabei bleibt normalerweise etwas Material übrig, das für die nächste(n) Seite(n) weiter verwendet wird. Während $\text{\LaTeX}$ also Überschriften, Absätze und andere Elemente aufbereitet, ist normalerweise noch nicht abzusehen, auf welcher Seite diese letztendlich erscheinen werden, da sich möglicherweise noch herausstellen wird, dass nicht alles auf die aktuelle Seite passt. Dieses Problem wurde bereits im Abschnitt 3.2.2 im Zusammenhang mit der seitenweisen Nummerierung von Fußnoten angesprochen.

Wenn die fertige Seite gesetzt wird, möchte man vielleicht einige Informationen aus dem Seiteninhalt in lebende Kolumnentitel aufnehmen (z.B. die aktuelle Überschrift), um dem Leser eine zusätzliche Orientierungshilfe zu geben. Solange das Material gesammelt wird, lassen sich keine korrekten Informationen in Befehlen abspeichern: In dieser Phase liest $\text{\LaTeX}$ oft zu weit voraus und der Befehl könnte Daten enthalten, die gar nicht auf der fertigen Seite auftauchen. $\text{\LaTeX}$ löst dieses Problem mithilfe eines Textmarkenmechanismus, mit dessen Hilfe sich „interessante“ Daten markieren lassen. Alle Marken der Seite werden in der Ausgaberroutine gesammelt, welche dann die erste und die letzte der Marken bereitstellt. Dieser Abschnitt erklärt, wie das genau funktioniert, und verweist auf einige nützliche Erweiterungspakete.

4.3.1 Seitenzahlen in $\text{\LaTeX}$

Die Seitenzahlen werden durch einen Zähler namens `page` gesteuert. Dieser Zähler wird von $\text{\LaTeX}$ automatisch hochgezählt, sobald eine Seite fertiggestellt ist – also *nachdem* er bereits verwendet wurde. Daher muss er mit 1 initialisiert werden, während die meisten anderen $\text{\LaTeX}$ -Zähler bei 0 beginnen und erst kurz vor ihrer Verwendung hochgezählt werden.

Entsprechend der $\text{\LaTeX}$ -Standardkonventionen lautet der Befehl zur typographischen Darstellung der Seitenzahl `\thepage`. Es gibt jedoch noch einen weiteren feinen Unterschied zu anderen $\text{\LaTeX}$ -Zählern: Der Befehl `\thepage` wird nicht vom $\text{\LaTeX}$ -Kern definiert. Er entsteht in dem Moment, wenn zum

ersten Mal die Deklaration `\pagenumbering` ausgeführt wird, was normalerweise in der Dokumentenklassendatei geschieht.

Der beste (wenn vielleicht auch nicht bequemste) Weg, mitten im Text der aktuellen Seite an die Seitenzahl zu gelangen, ist eine Kombination der Befehle `\label` und `\pageref`, die direkt hintereinander stehen sollten, so dass kein Seitenumbruch zwischen ihnen erfolgen kann.

Hier ist Seite 6. Diese Kodierung erzeugt immer ein korrektes Ergebnis, während „Seite 6“ zwar hier funk-

6

tionieren würde, jedoch nicht hier: „Seite 6“, da $\LaTeX$ beschlossen hat, den Absatz über drei Seiten zu umbrechen.

7

Hier ist Seite~`\label{p1}\pageref{p1}`. Diese Kodierung erzeugt immer ein korrektes Ergebnis, während "`Seite \thepage{}`" zwar hier funktionieren würde, jedoch nicht hier: "`Seite \thepage`", da $\LaTeX$ beschlossen hat, den Absatz über drei Seiten zu umbrechen.

Bsp.
4-3-1

Aufgrund des asynchronen Verfahrens der Ausgaberroutine kann man `\thepage` innerhalb des Dokumentes nicht ohne Risiko einsetzen. Der Befehl lässt sich nur in den Deklarationen zuverlässig verwenden, die das Erscheinungsbild der von der Ausgaberroutine fertiggestellten Seite beeinflussen.

`\pagenumbering{layout}`

Der Befehl `\pagenumbering` setzt den Zähler `page` auf 1 zurück und definiert den Befehl `\thepage` in `\layout{page}` um. Gebrauchsfertige Layouts für Seitenzahlen sind: `Alph`, `alph`, `Roman`, `roman` und `arabic` (siehe auch Abschnitt A.1.4).

Bei Büchern etwa ist es üblich, die Seiten des Vorspanns mithilfe von `roman` in römischen Zahlen zu setzen. Für das erste Kapitel des Hauptteils beginnt die Seitennummerierung dann in arabischen Ziffern von vorne (mit `arabic`). Diesen Effekt kann man auch manuell erzielen, indem man den Befehl `\pagenumbering` zweimal einsetzt; die Befehle `\frontmatter` und `\mainmatter` der Klasse `book` setzen dieses Verfahren implizit im Hintergrund ein.

4.3.2 lastpage – Verweise auf die letzte Seite

Standard- $\LaTeX$ kennt keinen Weg, auf die Anzahl der Seiten eines Dokumentes zu verweisen. Man kann also nicht schreiben: „Dieses Dokument besteht aus 6 Seiten“ oder den Text „Seite 5 von 10“ generieren, ohne zuvor die Seiten von Hand zu zählen. Das Paket `lastpage` von Jeffrey Goldberg umgeht dieses Problem, indem es automatisch auf der letzten Seite ein Label namens `LastPage` verwendet, auf dessen Seitenzahl man sich dann mit `\pageref{LastPage}` beziehen kann. Beispiel 4-4-5 auf Seite 234 zeigt, wie man es verwendet.

Die Zeichenfolge, die durch den Aufruf von `\pageref` erzeugt wird, entspricht dem Inhalt von `\thepage`, wie er auf der letzten Seite erscheinen würde. Wenn innerhalb des Dokumentes die Nummerierung neu begonnen wird – z.B. weil der Vorspann separat nummeriert ist – gibt die Zeichenfolge nicht die Anzahl aller Seiten an.

Das Paket generiert das Label mithilfe von `\AtEndDocument`, um sicherzugehen, dass zunächst alle aufgelaufenen Gleitobjekte platziert werden. Da dieser Befehl jedoch auch von anderen Paketen dazu verwendet werden kann, Textmaterial ans Ende des Dokumentes zu setzen, bleibt immer noch das Risiko, dass das Label zu früh platziert wird. In diesem Fall kann man versuchen, `lastpage` nach dem Paket zu laden, das dieses zusätzliche Material erzeugt.

4.3.3 chappg – Kapitelweise Nummerierung der Seiten

Manche Werke erfordern, dass die Seiten in jedem Kapitel separat nummeriert werden und die Seitenzahl zusammen mit der Kapitelnummer auf jeder Seite steht. Das lässt sich mit den bereits verfügbaren Befehlen erreichen, indem man folgenden Code nach jedem `\chapter`-Befehl wiederholt:

```
% Kapitelweise Nummerierung (nach jedem \chapter-Befehl wiederholen:
\pagenumbering{arabic} % zuerst die Seitenzahlen zurücksetzen und dann das ...
\renewcommand\thepage{\thechapter--\arabic{page}} % ... Layout überschreiben
```

Das ist jedoch sehr umständlich und zwingt dazu, sehr viel Layoutinformationen in das Dokument einzubringen, was man besser vermeiden sollte.

Das Paket `chappg`, ursprünglich von Max Hailperin und später von Robin Fairbairns reimplementiert und erweitert, bietet einen besseren Weg. Es unterstützt jede Dokumentenklasse, die über den Befehl `\chapter` verfügt, und führt für die gewünschte Seitennummerierung das Layout `bychapter` ein. Außerdem erweitert es den `\pagenumbering`-Befehl um ein optionales Argument, mit dessen Hilfe man der Seitenzahl ein beliebiges Präfix anstelle der Kapitelnummer voranstellen kann. Das ist zum Beispiel im Vorspann nützlich, da die Überschriften dort normalerweise nicht nummeriert sind.

Bsp.
4-3-2

... hier befinden wir
uns mitten im Vorspann,

Vorwort-1

wo Kapitel normalerweise
nicht nummeriert

Vorwort-2

```
\usepackage{chappg}
```

```
% \chapter*{Vorwort} % -- nicht gezeigt
\pagenumbering[Vorwort]{bychapter}
\ldots hier befinden wir uns mitten im
Vorspann, wo Kapitel normalerweise
nicht nummeriert sind.
```

Mit etwas Fingerspitzengefühl kann man dieses Paket sogar mit Dokumentenklassen verwenden, die nicht über einen `\chapter`-Befehl verfügen. Angenommen die höchste Gliederungsebene ist `\section` und jeder Abschnitt beginnt automatisch auf einer neuen Seite (das ist eine wichtige Voraussetzung). Dann bewirkt die Deklaration

```
\makeatletter \@addtoreset{page}{section} \makeatother
\pagenumbering[\thesection]{bychapter}
```

dass die Seiten abschnittsweise nummeriert werden. Wenn jedoch nicht jeder Abschnitt auf einer neuen Seite beginnt, kann dieses Verfahren scheitern, da L^AT_EX möglicherweise bereits den Anfang eines neuen Abschnitts gefunden

und den `section`-Zähler hochgezählt hat, ohne die vorherige Abschnittsnummer auf der aktuellen Seite zu vermerken. Dadurch ergibt sich das gleiche Problem, das bereits zuvor für `\thepage` beschrieben wurde.

Auch das Zeichen zwischen dem Präfix und der Seitenzahl lässt sich anpassen, da er mit dem Befehl `\chappgsep` erzeugt wird. Damit generiert

```
\renewcommand\chappgsep{/}
```

in Kapitel „2“ Seiten mit der Nummerierung 2/1, 2/2, 2/3 und so weiter.

4.3.4 Textmarkenbefehle

In Paketdateien, die sich mit Seitenlayouts oder Ausgaberroutinen befassen, findet sich der $\text{\TeX}$ -Basisbefehl `\mark`. Er ist dafür zuständig, irgendwelche Informationen (sein Argument) mit einer Position auf einer Seite zu verknüpfen (d.h. mit der Position, an welcher der `\mark`-Befehl ausgeführt wurde). Wenn die fertige Seite erzeugt wird, speichert $\text{\TeX}$ die erste Marke auf der zusammengestellten Seite in `\firstmark`, die letzte in `\botmark` und die `\botmark` der vorigen Seite als `\topmark`. Wenn die Seite keine Marken enthält, übernehmen auch `\firstmark` und `\botmark` den Wert der vorherigen `\botmark`. Wenn also jeder Gliederungsbefehl intern mit `\mark` eine Marke setzt, die als Argument den Text der Überschrift aufnimmt, könnte man mithilfe dieser Befehle die erste oder letzte Überschrift der Seite in einem lebenden Kolumnentitel anzeigen.

Diese Befehle können jedoch *nicht* direkt in $\text{\LaTeX}$ eingesetzt werden, da $\text{\LaTeX}$ ein darauf aufbauendes Protokoll verwendet, welches die Argumente der Befehle intern strukturiert, um mehr als eine Marke zu ermöglichen. Sie werden hier nur erwähnt, um den grundlegenden Mechanismus dahinter zu erklären; ein direkter Einsatz dieser Befehle würde sehr wahrscheinlich zu seltsamen Fehlermeldungen führen.

Anstelle des `\mark`-Befehls stellt Standard- $\text{\LaTeX}$ die beiden folgenden Befehle zum Erzeugen von Textmarkenpaaren zur Verfügung:

<code>\markboth{hauptmarke}{untermarke}</code>	<code>\markright{untermarke}</code>

Der erste Befehl platziert ein Textmarkenpaar an der aktuellen Stelle im Dokument. Der zweite erzeugt intern ebenfalls ein Markenpaar, wobei er aber nur die *untermarke* verändert, während er die *hauptmarke* vom vorherigen `\markboth`-Befehl übernimmt.

Ursprünglich wollte man damit einigermaßen voneinander unabhängige Marken erzeugen – zum Beispiel Kapitelüberschriften als *hauptmarken* und Abschnittsüberschriften als *untermarken*. Der Befehlsname `\markright` deutet jedoch schon darauf hin, dass Leslie Lamport ein ganz bestimmtes Markierungsschema vor Augen hatte, als er diese Befehle entwarf. Das wird noch deutlicher, wenn man sich die Befehle ansieht, mit deren Hilfe die Ausgaberroutine die Werte der Marken ausliest.

In der Ausgaberroutine enthält `\leftmark` das Argument *hauptmarke* des letzten `\markboth`-Befehls vor dem Ende der Seite. Der Befehl `\rightmark` enthält die *untermarke* des ersten `\markright`- oder `\markboth`-Befehls auf

Keine Low-Level
 $\text{\TeX}$ -Marken in $\text{\LaTeX}$
verwenden

Textmarkenbefehle	Markenpaar	verwendete Marken	
		\leftmark	\rightmark
\markboth{L1}{}	{L1}{}		
\newpage% -- Seitenumbruch --		L1	
\markright{R1.1}	{L1}{R1.1}		
\markboth{L2}{}	{L2}{}		
\markright{R2.1}	{L2}{R2.1}		
\newpage% -- Seitenumbruch --		L2	R1.1
\markright{R2.2}	{L2}{R2.2}		
\markright{R2.3}	{L2}{R2.3}		
\markright{R2.4}	{L2}{R2.4}		
\newpage% -- Seitenumbruch --		L2	R2.2
\markboth{L3}{}	{L3}{}		
\markright{R3.1}	{L3}{R3.1}		
\newpage% -- Seitenumbruch --		L3	
\newpage% -- Seitenumbruch --		L3	R3.1
\markright{R3.2}	{L3}{R3.2}		
\markboth{L4}{}	{L4}{}		
\markboth{L5}{}	{L5}{}		
\newpage% -- Seitenumbruch --		L5	R3.2
\markright{R5.1}	{L5}{R5.1}		
\end{document}		L5	R5.1

Abbildung 4.2: Arbeitsweise des Textmarkenmechanismus von L^AT_EX

der Seite, sofern dort einer dieser Befehle vorkommt; ansonsten enthält es diejenige, die zuletzt definiert wurde.

Diese Textmarkenbefehle funktionieren ziemlich gut, wenn die rechten Marken in Abhängigkeit von den linken Marken nummeriert sind – daher auch die Namen (wenn also z.B. die linken Marken durch den Befehl `\chapter` und die rechten durch den Befehl `\section` geändert werden). Die Ergebnisse werden jedoch etwas ungewöhnlich, wenn einem `\markboth`-Befehl auf der gleichen Seite ein anderer Markenbefehl vorangeht – vgl. die Seiten mit L2 R1.1 und L5 R3.2 in Abbildung 4.2. Diese Abbildung zeigt schematisch, welche linken und rechten Marken für die Ausgabeseiten erzeugt werden. Bei einigen Arten von Kolumnentiteln wäre die Anzeige der ersten *hauptmarke* oder der letzten *untermarke* besser geeignet. Zu diesem Zweck lässt sich das Paket `extramarks`, das im Folgenden noch beschrieben wird, heranziehen, da Standard-L^AT_EX diese Möglichkeit nicht bietet. Es sollte auch beachtet werden, dass man keine *hauptmarke* setzen kann, ohne gleichzeitig eine *untermarke* zu setzen und damit zu überschreiben.

Bei Layouts, deren lebende Kolumnentitel aus dem Text von Überschriften erzeugt werden, wäre es schön, wenn die Marken automatisch aus den entsprechenden Gliederungsbefehlen generiert werden. Glücklicherweise gibt es eine Schnittstelle, mit deren Hilfe man festlegen kann, welche Befehle Marken setzen und welcher Text dabei an die Marke übergeben wird. Das funktioniert folgendermaßen: Alle Standard-Gliederungsbefehle rufen intern den Befehl `\namemark` auf, wobei *name* der Name des Gliederungsbefehls ist (z.B. `\chaptermark`, `\sectionmark`). Diese Befehle besitzen ein Argument, das

den Text der Überschrift oder seine Kurzform aus dem optionalen Argument des Gliederungsbefehls aufnimmt.

Sie haben normalerweise keinerlei Auswirkung. Wenn sie jedoch in geeigneter Weise umdefiniert werden, können sie ein Textmarkenpaar erzeugen, wie L^AT_EX es benötigt. In der Klasse `book` werden diese Befehle z.B. (ungefähr) folgendermaßen definiert:

```
\renewcommand\chaptermark[1]{\markboth{\chaptername\
\thechapter. #1}{}}
\renewcommand\sectionmark[1]{\markright{\thesection. #1}}
```

Im Falle eines Kapitels wird dadurch der Inhalt von `\chaptername` (Voreinstellung „Chapter“¹), gefolgt von der Gliederungsnummer des Kapitels (die in `\thechapter` gespeichert ist), sowie der Inhalt (oder die Kurzversion) der Kapitelüberschrift als Argument *hauptmarke* von `\markboth` gespeichert; gleichzeitig wird der Inhalt der *untermarke* gelöscht. Bei einem Abschnitt wird die Abschnittsnummer (aus dem Zähler `section`), gefolgt vom Inhalt (oder der Kurzversion) der Abschnittsüberschrift, an den Befehl `\markright` übergeben, der daraus ein Markenpaar mit einer neuen *untermarke* erzeugt.

4.3.5 extramarks – Eine neue Art von Marken

Wie bereits zu sehen war, wurde der Textmarkenmechanismus von L^AT_EX für ein ganz bestimmtes Layout entworfen und ist daher für andere Anwendungen nur bedingt geeignet. Daher wurden einige Versuche unternommen ihn durch Funktionen zu erweitern oder zu ersetzen, die komplexere Markensysteme unterstützen.

Teilweise liegen die Grenzen in T_EX selbst, das nur eine Art von Marken kennt und es damit schwierig (wenn auch nicht unmöglich) macht, verschiedene, voneinander unabhängige Marken zu implementieren. Dieses Problem ist in eT_EX behoben, das unabhängige Textmarkenklassen zu Verfügung stellt. Da dieses Programm jedoch bis vor kurzem noch nicht sehr weit verbreitet war, gibt es bisher noch keine Pakete, welche die neuen Möglichkeiten des erweiterten Textmarkenmechanismus verwenden.

Das Paket `extramarks` von Piet van Oostrum (das als Teil des `fancyhdr`-Paketes verteilt wird) stellt einen erweiterten Mechanismus innerhalb des zentralen L^AT_EX-Modells zur Verfügung. Es bietet zwei zusätzliche, (teilweise) voneinander unabhängige Marken, sowie weitere Steuerungsmöglichkeiten für die L^AT_EX-Standardmarken, indem es ermöglicht, dass man auf jeder Seite für beide Markentypen *hauptmarke* und *untermarke* sowohl die erste als auch die letzte Textmarke ansprechen kann.

Das Paket bietet die Befehle `\firstleftmark` und `\lastleftmark`, um die erste bzw. letzte *hauptmarke* auf einer Seite anzusprechen. Auf die gleiche Weise kann man mit `\firstrightmark` und `\lastrightmark` auf die erste bzw. letzte *untermarke* zugreifen.² Beispiel 4-4-9 auf Seite 237 zeigt, wie sie verwendet werden.

¹Für Sprachanpassungen siehe Tabelle 9.2 auf Seite 561.

²Genaugenommen sind die Befehle `\lastleftmark` und `\firstrightmark` Synonyme für die L^AT_EX-Befehle `\leftmark` und `\rightmark`, nur dass ihre Namen ihre Funktion genauer beschreiben.

```
\extramarks{linke-xmarke}{rechte-xmarke}
```

Mithilfe des Befehls `\extramarks` des Paketes lassen sich zusätzliche Textmarken in das Dokument einfügen. Er besitzt zwei obligatorische Argumente: die Texte für zwei Marken an der aktuellen Position. Während des Seitenaufbaus verweist dann der Befehl `\firstleftxmark` auf die erste *linke-xmarke* der Seite und `\lastleftxmark` auf die letzte. Genauso können `\firstrightxmark` und `\lastrightxmark` in der Ausgaberroutine verwendet werden, um auf die *rechte-xmarke* zuzugreifen.

Das nächste Beispiel zeigt die praktische Anwendung dieser Befehle. Mithilfe des Paketes `fancyhdr` (das in Abschnitt 4.4.2 beschrieben ist) wird ein Seitenlayout erzeugt, bei dem die erste *linke-xmarke* am Kopf einer Seite und die letzte *rechte-xmarke* rechts unten auf jeder Seite dargestellt wird. Dabei ist der Einsatz des Befehls `\extramarks` besonders interessant. Das Beispiel beginnt mit einem `\extramarks`-Befehl, der den Text „Eine Geschichte“ als *linke-xmarke* und eine leere *rechte-xmarke* besitzt. Unmittelbar darauf folgt ein weiteres Markenpaar mit den Werten „... geht weiter“ und „bitte umblättern“. Das führt dazu, dass die erste *linke-xmarke* auf der ersten Seite den Text „Eine Geschichte“ enthält, auf den folgenden Seiten jedoch „... geht weiter“. Die letzte *rechte-xmarke* auf jeder Seite enthält immer den Text „bitte umblättern“. Solange die Geschichte weitergeht erhält man also immer die richtigen Fortsetzungsmarken am Kopf und Fuß jeder Seite. Am Ende der Geschichte sollte jedoch nicht „bitte umblättern“ stehen. Dazu enthält das Beispiel am Ende einen weiteren `\extramarks`-Befehl, dessen *rechte-xmarke* leer ist. Seine *linke-xmarke* enthält weiterhin den Text „... geht weiter“, damit auch am Kopf der letzten Seite der richtige Text steht.

Eine Geschichte

Text für unsere
Seite, der immer wie-
der verwendet wird.
Text für unsere Seite,
der immer wieder

bitte umblättern

... geht weiter

verwendet wird.

```
\usepackage{fancyhdr,extramarks}
\pagestyle{fancy} \cfoot{}
\lhead{\firstleftxmark}
\rfoot{\lastrightxmark}
\newcommand\sample{ Text für unsere Seite,
  der immer wieder verwendet wird.}
\extramarks{Eine Geschichte}{}
\extramarks{\ldots\ geht weiter}
  {bitte umblättern}
\sample \sample
\extramarks{\ldots\ geht weiter}{}

```

Bsp.
4-3-3

Die zusätzlichen Marken können mit den Standardmarken kombiniert werden, die $\LaTeX$ mit Gliederungsbefehlen oder mit `\markboth` und `\markright` erzeugt. Dabei ist zu beachten, dass sie nicht völlig unabhängig voneinander sind. Immer wenn der Befehl `\extramarks` oder einer der Standard-Textmarkenbefehle von $\LaTeX$ verwendet wird, generiert $\LaTeX$ alle vier Marken (wobei die Werte für nicht explizit gesetzte Marken beibehalten werden). Daher kann es sein, dass die erste Marke einer bestimmten Art einen unerwarteten Inhalt hat. Wenn ein Dokument z.B. mit einem `\extramarks`-Befehl anfängt, generiert es implizit eine leere *hauptmarke* und eine leere *untermarke*.

Der Textmarkenmechanismus von $\TeX$ kennt noch einen dritten Basisbefehl namens `\topmark`, der normalerweise nicht in $\LaTeX$ verfügbar ist. Dieser

enthält den Wert des Befehls `\botmark` der vorhergehenden Seite, so dass er im Grunde die Marken unmittelbar zu Beginn der Seite wiedergibt – daher auch sein Name. In $\text{\LaTeX}$ ist er normalerweise nicht verfügbar, da er dort mit den Mechanismen für Gleitobjekte und mit `\marginpar` Konflikte erzeugt. Anders ausgedrückt löst jedes dieser Objekte intern die Ausgaberroutine aus, so dass die `\topmark`-Werte für die aktuelle Seite verschwinden.

Wenn aber weder Gleitobjekte noch `\marginpar` zum Einsatz kommen, kann man die Informationen von `\topmark` durchaus verwenden, und genau für diese Fälle bietet `extramarks` eine entsprechende Schnittstelle. Wenn man Verwendung für eine solche Textmarke hat, kann man mit den Befehlen `\topleftxmark` und `\toprightxmark` auf die mit `\extramarks` erzeugte *linke-xmarke* und *rechte-xmarke* zugreifen.

4.4 Layouts für Kolumnentitel

Während der Seitenspiegel bei fast allen Seiten gleich bleibt, kann sich das Format der lebenden Kolumnentitel im Verlauf eines Dokumentes ändern. In $\text{\LaTeX}$ werden Layouts für Kolumnentitel *page style* genannt. Sie enthalten verschiedene Formatierungen und ihre Namen wie `empty` oder `plain` lassen schon auf ihre Verwendung schließen.

Mit den Befehlen `\pagestyle` oder `\thispagestyle` können neue Layouts ausgewählt werden. Beide erwarten den Namen des Layouts als obligatorisches Argument. Der erste Befehl legt das Layout für die aktuelle und alle folgenden Seiten fest, der zweite nur für die aktuelle Seite.

Bei kürzeren Dokumenten ist es normalerweise nicht erforderlich, zwischen verschiedenen Layouts für die Kolumnentitel zu wechseln. Normalerweise reicht hier das voreingestellte Layout der Dokumentenklasse aus. Bei umfangreicheren Dokumenten, wie etwa ganzen Büchern, kann man durch typographische Gepflogenheiten, Vorgaben der Verlage oder aus anderen Gründen gezwungen sein, das Layout der Kolumnentitel an bestimmten Stellen manuell zu verändern.

*$\text{\LaTeX}$ -Standardlayouts
für Kolumnentitel*

$\text{\LaTeX}$ verfügt über vier grundlegende Kolumnentitel-Layouts; besondere Pakete oder Dokumentenklassen können weitere bereitstellen.

`empty` Beide Kolumnentitel sind leer.

`plain` Der obere Kolumnentitel ist leer und der untere enthält die Seitenzahl (Folio).

`headings` Der obere Kolumnentitel enthält durch die Dokumentenklasse festgelegte Informationen sowie die Seitenzahl; der untere ist leer.

`myheadings` Ähnlich wie `headings`, jedoch kann der obere Kolumnentitel vom Anwender bestimmt werden.

Die Standardklassen verwenden die ersten drei Layouts. Normalerweise wird die Titelseite intern mit dem Befehl `\thispagestyle{empty}` gestaltet. Für die erste Seite der Hauptgliederungsbefehle (wie `\part` oder `\chapter`, aber auch `\maketitle`), verwenden die $\text{\LaTeX}$ -Standardklassen den Befehl `\thispagestyle{plain}`. Deshalb erhält man weiterhin Seitenzahlen auf

*Alle Seitenzahlen
unterdrücken*

	Befehl	Dokumentenklasse	
		book, report	article
Zweiseitiger Druck	<code>\markboth^a</code>	<code>\chapter</code>	<code>\section</code>
	<code>\markright</code>	<code>\section</code>	<code>\subsection</code>
Einseitiger Druck	<code>\markright</code>	<code>\chapter</code>	<code>\section</code>

^aLegt eine leere rechte Marke an (siehe Abbildung 4.2 auf Seite 227).

Tabelle 4.3: Layout-Befehle für Kolumnentitel in $\LaTeX$

den Seiten, die durch Befehle wie `\chapter` oder `\maketitle` gebildet werden, auch wenn am Anfang des Dokumentes das Layout `\pagestyle{empty}` eingestellt ist. Um die Seitenzahlen überall zu unterdrücken, muss entweder auf jeden dieser Befehle die Anweisung `\thispagestyle{empty}` folgen, oder das Layout `plain` muss in einem eigenen Anpassungspaket des Anwenders durch den Befehl `\let\ps@plain=\ps@empty` auf das Layout `empty` umdefiniert werden.

Im Layout `headings` erzeugen die Gliederungsbefehle die Kolumnentitel automatisch mithilfe von `\markboth` und `\markright`, wie in Tabelle 4.3 angezeigt.

Das Standardlayout `myheadings` ähnelt `headings`, wobei der Anwender hier die Informationen für die Kopfzeile mithilfe der oben beschriebenen Befehle `\markboth` und `\markright` selbst angibt. Außerdem lassen sich damit auch Überschriften von anderen Elementen wie Inhalts- und Abbildungs- oder Stichwortverzeichnissen übernehmen. Diese Befehle (`\tableofcontents`, `\listoffigures` und `\listoftables`) und Umgebungen (`thebibliography` und `theindex`) verwenden eigentlich den `\chapter*`-Befehl, der nicht `\chaptermark` aufruft, sondern stattdessen den Befehl `\@mkboth` verwendet. Das Layout `headings` definiert `\@mkboth` als `\markboth`, im Layout `myheadings` wird `\@mkboth` jedoch nicht aktiviert, so dass die Entscheidung dem Anwender überlassen bleibt.

4.4.1 Die Low-Level-Schnittstelle für Kolumnentitel-Layouts

Der $\LaTeX$ -Kern verwendet vier interne Befehle als Schnittstelle für das Kolumnentitel-Layout. Zwei dieser Befehle werden auf jeder Seite dazu verwendet, die lebenden Kolumnentitel zu formatieren. Durch Umdefinieren dieser Befehle kann man verschiedene Aktionen anstoßen.

`\@oddhead` erzeugt bei zweiseitigem Druck den oberen Kolumnentitel für ungerade Seiten, ansonsten für alle Seiten.

`\@oddfoot` erzeugt bei zweiseitigem Druck den unteren Kolumnentitel für ungerade Seiten, ansonsten für alle Seiten.

`\@evenhead` erzeugt bei zweiseitigem Druck den oberen Kolumnentitel für gerade Seiten und wird bei einseitigem Druck ignoriert.

`\@evenfoot` erzeugt bei zweiseitigem Druck den unteren Kolumnentitel für gerade Seiten und wird bei einseitigem Druck ignoriert.

Ein benanntes Layout besteht lediglich aus geeigneten Umdefinitionen für diese Befehle, die in einem Makro namens `\ps@<stil>` gespeichert werden; man muss also dieses Makro umdefinieren, um das Verhalten eines Layouts *stil* zu verändern. Beispielsweise sieht die Definition für das Layout `plain`, das lediglich eine zentrierte Seitenzahl am Fuß der Seite ausgibt, im $\text{\LaTeX}$ -Kern ungefähr folgendermaßen aus:

```
\newcommand\ps@plain{%
  \renewcommand\@oddhead{}%           % leerer Recto-Kopf
  \let\@evenhead\@oddhead             % leerer Verso-Kopf
  \renewcommand\@evenfoot{\hfil
    \normalfont \textrm{\thepage}\hfil}% % zentrierte
  \let\@oddfoot\@evenfoot             %           Seitenzahl
}
```

4.4.2 fancyhdr – Anpassen von Kolumnentitel-Layouts

Da sich das Layout der Kolumnentitel in Standard- $\text{\LaTeX}$ nur über interne Befehle ändern lässt, ist es nicht überraschend, dass eine Reihe von Paketen für besondere Layouts entwickelt wurden. So ändert z.B. das Paket `rplain` das Layout `plain` dahingehend, dass die Seitenzahl rechts und nicht zentriert erscheint. Es gibt auch komplexere Pakete. Beispielsweise lohnt es sich, die entsprechenden Deklarationsmöglichkeiten des Paketes `titlesec` (zur Definition von Gliederungsbefehlen, siehe Abschnitt 2.2.6) näher zu erkunden.

Ein gut eingeführtes, eigenständiges Paket in diesem Bereich ist `fancyhdr`¹ von Piet van Oostrum, mit dessen Hilfe sich Kolumnentitel auf einfache Weise anpassen lassen. Das Standardlayout von `fancyhdr` heißt `fancy`. Es sollte mit `\pagestyle` aktiviert werden – gegebenenfalls *nach* dem Ändern bzw. Einstellen von `\textwidth`, da `fancyhdr` die Breite der Kolumnentitel in Abhängigkeit vom aktuellen Wert dieser Länge initialisiert.

Basisschnittstelle

Das Erscheinungsbild des Layouts `fancy` wird durch sechs Deklarationen bestimmt, die festlegen, welches Material in den Kolumnentiteln links, zentriert und rechts erscheint. Der Befehl `\lhead` legt z.B. fest, was links im oberen Kolumnentitel erscheint, während `\cfoot` die Mitte des unteren Kolumnentitels definiert. Im nächsten Beispiel werden die Ergebnisse aller sechs Deklarationen dargestellt.

LINKS	MITTE	RECHTS	
Text für unsere Seite, der immer und immer wieder verwendet wird.			<code>\usepackage{fancyhdr} \pagestyle{fancy}</code>
			<code>\lhead{LINKS} \chead{MITTE} \rhead{RECHTS}</code>
			<code>\lfoot{links-ganz-ganz-ganz-ganz-lang} \cfoot{}</code>
			<code>\rfoot{rechts-sehr-lang}</code>
			<code>\renewcommand\headrulewidth{2pt}</code>
			<code>\renewcommand\footrulewidth{0.4pt}</code>
Text für unsere Seite, der immer und immer wieder verwendet wird.			<code>\newcommand\sample{ Text für unsere Seite,</code>
			<code>der immer und immer wieder verwendet wird.}</code>
links-ganz-ganz-ganz-ganz-lang	rechts-sehr-lang		<code>\sample \par \sample</code>

Bsp.
4-4-1

¹In diesem Buch wird Version 2.0 von `fancyhdr` beschrieben. Frühere Versionen waren unter dem Namen `fancyheadings` bekannt.

Oft wird nur ein Bereich der Kolumnentitel mit Material gefüllt. Wenn allerdings mehr als eine Deklaration ein Argument besitzt, das nicht leer ist, muss man darauf achten, dass der gedruckte Text nicht zu breit wird. Ansonsten kommt es, wie im vorigen Beispiel, teilweise zu Überlappungen.

Die Stärke der Linien unter dem oberen und über dem unteren Kolumnentitel wird mit den Befehlen `\headrulewidth` (default 0.4pt) und `\footrulewidth` (default 0pt) eingestellt. Eine Linienstärke von 0pt blendet die Linie aus. Dabei ist zu beachten, dass es sich hier nicht um Längenparameter, sondern um Befehle handelt, die mit `\renewcommand` geändert werden müssen. Wenn man die Befehle `\headrule` und/oder `\footrule`, welche die eigentlichen Linien erzeugen, undefiniert, kann man die Änderungen noch aufwendiger gestalten, wie in Beispiel 4-4-6 auf Seite 235. Beim Umdefinieren muss man unter Umständen negative vertikale Abstände hinzufügen, da das Material normalerweise in einem Abstand von `\baselineskip` unter der Kopf- bzw. über der Fußzeile erscheint.

Das nächste Beispiel zeigt, wie man durch den Einsatz von `\\` in den Deklarationsbefehlen mehrzeilige Kolumnentitel erzeugen kann. In diesem Falle muss man allerdings die Länge `\headheight` (die Höhe der Box für die Kolumnentitel) vergrößern, da diese normalerweise nur groß genug für eine Zeile ist. Wenn `fancyhdr` feststellt, dass `\headheight` zu klein ist, gibt es eine Warnung aus und nennt den erforderlichen Mindestwert für das aktuelle Dokument.

Von: Frank
An: Michel

Seite: 6
29. Februar 2004

Text für unsere Seite, der immer und immer wieder verwendet wird.

Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{fancyhdr} \pagestyle{fancy}
\setlength\headheight{23pt}
\lhead{Von: Frank\ An: Michel}
\rhead{Seite: \thepage\ \today}
\chead{} \lfoot{} \cfoot{} \rfoot{}
\renewcommand\headrule{\vspace{-8pt}\dotfill}

% \sample definiert wie zuvor
\sample \par \sample
```

Im obigen Beispiel führt `\\` zu mehrzeiligen Texten, die entsprechend des Bereiches im Kolumnentitel, in dem sie erscheinen, ausgerichtet werden. In `\thead` sind sie z.B. linksbündig und in `\rhead` rechtsbündig. Wenn dieses Ergebnis nicht erwünscht ist, sollte man stattdessen eine einfache `tabular`-Umgebung in Betracht ziehen. Man beachte das Element `@{}` in der Spalten-deklaration für die Tabelle. Es unterdrückt den Standardleerraum hinter der Spalte. Andernfalls wäre der Kolumnentitel nicht bündig mit dem Rand ausgerichtet.

Von: Frank Seite: 6
An: Michel 29. Februar 2004

Text für unsere Seite, der immer
und immer wieder verwendet wird.

Text für unsere Seite, der immer
und immer wieder verwendet wird.

```
\usepackage{fancyhdr} \pagestyle{fancy}
\setlength\headheight{23pt}
\lhead{Von: Frank\ An: Michel}
\rhead{\begin{tabular}[b]{l@{}}
      Seite: \thepage\ \today
    \end{tabular}}
\chead{} \lfoot{} \cfoot{} \rfoot{}
% \sample definiert wie zuvor
\sample \par \sample
```

Genaue Steuerung

Mithilfe der bisher beschriebenen Deklarationen lässt sich das Layout der Kolumnentitel nicht in Abhängigkeit von der Art der aktuellen Seite ändern. Dies wird jedoch mit den allgemeineren Deklarationen `\fancyhead` und `\fancyfoot` möglich. Sie besitzen ein zusätzliches optionales Argument, mit dem man festlegen kann, für welchen Seitentyp und für welchen Bereich der Kolumnentitel die Deklaration angewendet werden soll. Der Seitentyp wird mit O oder E für ungerade bzw. gerade Seiten ausgewählt, der Bereich dagegen mit L, C oder R. Wenn kein Seitentyp oder Bereich ausgewählt ist, gilt die Deklaration für alle Seitentypen oder Bereiche. Das Kürzel LO steht also für den linken Bereich auf ungeraden Seiten, während C für den mittleren Bereich auf allen Seiten steht. Anders ausgedrückt waren die bisher besprochenen Deklarationen Kurzformen für die hier beschriebenen.

Wie das nächste Beispiel zeigt, können die Kürzel sogar aneinandergereiht werden. So bedeutet etwa RO,LE, dass diese Deklaration für den rechten Bereich auf ungeraden und den linken Bereich auf geraden Seiten gilt.

6	Memo
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Autor: Frank	

Memo	7
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Autor: Frank	

```
\usepackage{fancyhdr}\pagestyle{fancy}
\fancyhead{} % Kopfzeilen leeren
\fancyhead[RO,LE]{\thepage}
\fancyhead[LO,RE]{Memo}
\fancyfoot{} % Fußzeilen leeren
\fancyfoot[L]{Autor: Frank}
\renewcommand\headrulewidth{0.4pt}
\renewcommand\footrulewidth{0.4pt}

% \sample definiert wie zuvor
\sample \par \sample
```

Bsp.
4-4-4

Tatsächlich sind `\fancyhead` und `\fancyfoot` nur von einer noch allgemeineren Deklaration, nämlich `\fancyhf`, abgeleitet. Die Syntax ist ähnlich, wobei es eine weitere Art von Kürzeln gibt. Das optionale Argument kann zusätzlich H oder F für Kopf- oder Fußzeilen enthalten. Damit ist `\fancyfoot[LE]` gleichbedeutend mit `\fancyhf[FLE]`, wobei die zweite Form wohl schlechter zu lesen ist, weshalb sie in diesem Buch normalerweise nicht verwendet wird. Allerdings ist die Deklaration `\fancyhf` von Vorteil, wenn man alle Bereiche gleichzeitig leeren möchte.

Das nächste Beispiel zeigt eine Anwendung des `lastpage`-Paketes: Im unteren Kolumnentitel wird die aktuelle Seitenzahl und die Gesamtzahl aller Seiten angezeigt.

1	EIN TEST
1 Ein Test	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Seite 6 von 7	

1	EIN TEST
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Seite 7 von 7	

```
\usepackage{fancyhdr,lastpage}
\pagestyle{fancy}
\fancyhf{} % -- alle Bereiche leeren
\fancyhead[RO,LE]{\leftmark}
\fancyfoot[C]{Seite \thepage
              von \pageref{LastPage}}

% \sample definiert wie zuvor
\section{Ein Test}
\sample \par \sample
```

Bsp.
4-4-5

Die Kolumnentitel werden in Boxen gesetzt, die auf eine Breite von `\textwidth` voreingestellt sind. Ihre Breite kann soweit erforderlich mithilfe

der Deklaration `\fancyhoffset`¹ vergrößert oder verringert werden. Sie besitzt ein optionales Argument, das angibt, welche Box (Kopf- oder Fußzeile) geändert werden soll, wo sich diese befindet (links oder rechts) und auf welchem Seitentyp sie steht (gerade oder ungerade) – dazu wird eine Kombination der Buchstaben HFLREO eingesetzt. Das obligatorische Argument legt dann die Größe der Erweiterung (oder Verringerung) fest. Wie auch schon bei anderen Befehlen, gibt es wiederum zwei nützliche Kurzformen: `\fancyheadoffset` und `\fancyfootoffset` sind wie `\fancyhoffset` mit der Einstellung H bzw. F.

Breite und Position der Kolumnentitel

Wenn man beispielsweise am Kopf der Seite einen Kolumnentitel erzeugen will, der sich auch über die Marginalien erstreckt, muss man als obligatorisches Argument die Summe von `\marginparsep` und `\marginparwidth` angeben. Das `calc`-Paket ermöglicht hier die folgende elegante Deklaration

```
\fancyheadoffset[R0,LE]{\marginparsep+\marginparwidth}
```

sobald die Parameter ihre korrekten Werte erhalten haben. (Diese Methode wurde zum Beispiel auch für die Kolumnentitel-Layouts in diesem Buch verwendet.)

Im nächsten Beispiel erstreckt sich die Kopfzeile bis in den äußeren Rand, während die Seitenzahl innerhalb der Begrenzungen des Satzspiegels zentriert wird. Das beweist, dass die Einstellungen für den oberen und unteren Kolumnentitel tatsächlich voneinander unabhängig sind.

Innerhalb der Kolumnentitel ist ihre Gesamtbreite über das Register `\headwidth` verfügbar (das für Kopf- und Fußzeile einzeln neu berechnet wird). Es lässt sich zum Positionieren von Objekten in diesen Bereichen nutzen. Im nächsten Beispiel wird mithilfe des Befehls `\headrule` eine Schmucklinie aus zwei blauen Linien erzeugt, die sich über die gesamte Breite der Kopfzeile erstreckt.

OBJEKT	1	A-TITEL
1 A-Titel 1.1 B-Titel Text für unsere Seite, der immer und immer wieder verwendet wird. Text für un-		
	6	

1.1	B-Titel	OBJEKT
sere Seite, der immer und immer wieder verwendet wird.		
	7	

```
\usepackage{color,fancyhdr}
\pagestyle{fancy} \fancyhf{}
\fancyheadoffset[R0,LE]{30pt}
\fancyhead[R0,LE]{OBJEKT}
\fancyhead[L0]{\rightmark}
\fancyhead[RE]{\leftmark}
\fancyfoot[C]{\thepage}
\renewcommand\headrule
{{\color{blue}%
\hrule height 2pt
width\headwidth
\vspace{1pt}%
\hrule height 1pt
width\headwidth
\vspace{-4pt}}}}
% \sample definiert wie zuvor
\section{A-Titel}
\subsection{B-Titel}
\sample \sample
```

¹Diese Funktion wurde in Version 2.1 hinzugefügt. Frühere Versionen bedienten sich einer anderen Methode.

Die Voreinstellungen
von fancyhdr

Die bisherigen Beispiele ließen auf die ein oder andere Voreinstellung von fancyhdr schließen. Im nächsten Beispiel werden alle verwendet. Um sie besser erkennen zu können, sind sie im Programmcode des Beispiels in Kommentaren aufgeführt. Die Standardeinstellungen setzen eine schmale Linie unter der Kopfzeile und keine Linie über der Fußzeile; die Seitenzahl steht im Fuß und ist zentriert und der Kopf zeigt sowohl \leftmark als auch \rightmark an, wobei ihre Reihenfolge durch den Seitentyp festgelegt wird.

1 TEST
1 Test
1.1 B-1
Text für unsere Seite, der immer und immer wieder verwendet wird.
6

1 TEST	1.2 B-2
1.2 B-2	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
	7

```
\usepackage{fancyhdr}
\pagestyle{fancy}
%\fancyhead[LE,RO]
%{\slshape\rightmark}
%\fancyhead[LO,RE]
%{\slshape\leftmark}
%\fancyfoot[C]{\thepage}
%\renewcommand\headrulewidth{0.4pt}
%\renewcommand\footrulewidth{0pt}
% \sample definiert wie zuvor
\section{Test}
\subsection{B-1} \sample
\subsection{B-2} \sample
```

Bsp.
4-4-7

Der Abstand zwischen Zahl und Text im oberen Kolumnentitel ist sicher zu groß. Dies liegt an dem extrem schmalen Satzspiegel im Beispiel, daher soll dieses Problem zunächst vernachlässigt werden. Wie nützlich sind, davon einmal abgesehen, diese Voreinstellungen? Wie bereits erwähnt, wurden die L^AT_EX-Befehle \leftmark und \rightmark in erster Linie mit Blick auf „Abschnitte in Kapiteln“ entwickelt – also für den Fall, dass \leftmark immer mit einer Überschrift verknüpft ist, die eine neue Seite einleitet. Andernfalls kann man, wie im nächsten Beispiel, recht seltsame Kolumnentitel erhalten.

Ein Abschnitt wurde auf Seite 5 platziert (die Seite wird nicht angezeigt) und erstreckt sich bis auf Seite 6. Dadurch erscheint im oberen Kolumnentitel von Seite 6 der Unterabschnitt 1.1 zusammen mit Abschnitt 2, ähnlich wie auf Seite 7, wo man Abschnitt 3 zusammen mit Unterabschnitt 2.1 erhält.

1.1 B-1	2 A-2
1.1 B-1	
Text für die Seite, der wiederverwendet wird.	
2 A-2	
Text für die Seite, der wiederverwendet wird.	
6	

3 A-3	2.1 B-2
2.1 B-2	
Text für die Seite, der wiederverwendet wird.	
3 A-3	
Text für die Seite, der wiederverwendet wird.	
7	

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\newcommand\sample{ Text für die
Seite, der wiederverwendet wird.}

\setcounter{page}{5}
\section{A-1} \newpage
% Code oben setzt einen Abschnitt
% auf Seite 5 (nicht angezeigt)
\subsection{B-1} \sample
\section{A-2} \sample
\subsection{B-2} \sample
\section{A-3} \sample
```

Bsp.
4-4-8

Um dieses Verhalten nachvollziehen zu können, muss man sich noch einmal vor Augen führen, dass sich `\leftmark` auf die letzte von `\markboth` erzeugte Marke auf dieser Seite bezieht, und `\rightmark` auf die erste von `\markright` oder `\markboth` erzeugte.

Wer häufig Seiten wie die oben dargestellten erzeugt, z.B. in einem Dokument mit vielen Unterabschnitten, dem ist mit den Standardeinstellungen des `fancyhdr`-Paketes sehr wahrscheinlich nicht gedient. Stattdessen sollte man sie auf die ein oder andere Weise überschreiben, wie in den meisten Beispielen dieses Abschnitts geschehen. Man muss sich folgende Frage stellen: Welche Informationen möchte ich dem Leser in einem solchen Kolumnentitel geben? Will man beispielsweise bei geraden (linken) Seiten die Situation zu Beginn der Seite und bei ungeraden Seiten den Status am Ende der Seite darstellen, dann bietet sich der Einsatz von `\firstleftmark` und `\lastrightmark` des `extramarks`-Paketes als mögliche Lösung an.

<i>1.1</i>	<i>B-1</i>	1	<i>A-1</i>
1.1	B-1		
Text für die Seite, der wiederverwendet wird.			
2	A-2		
Text für die Seite, der wiederverwendet wird.			
		6	

3	A-3		
2.1	B-2		
Text für die Seite, der wiederverwendet wird.			
3	A-3		
Text für die Seite, der wiederverwendet wird.			
		7	

```
\usepackage{extramarks}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[RO]{\lastrightmark}
\fancyhead[RE]{\firstleftmark}
% \sample definiert wie zuvor
\setcounter{page}{5}
\section{A-1} \newpage
% Code oben setzt einen Abschnitt
% auf Seite 5 (nicht angezeigt)
\subsection{B-1} \sample
\section{A-2} \sample
\subsection{B-2} \sample
\section{A-3} \sample
```

Bsp.
4-4-9

Wer prüfen will, ob der Mechanismus nun klar ist, sollte erklären, warum Seite 7 jetzt nur den Titel „A-3“ anzeigt und versuchen sich vorzustellen, was passieren würde, wenn die Überschrift „B-1“ (nicht aber der Text des ganzen Abschnitts) schon auf Seite 5 aufgetaucht wäre.

Trotz der zuvor aufgestellten Behauptung, alle Voreinstellungen gezeigt zu haben, nimmt das `fancy`-Layout doch noch zwei weitere Voreinstellungen vor. Da sie sozusagen im Verborgenen wirken, wurden sie bisher übergangen. Es wurde noch nicht erklärt, wie `\leftmark` und `\rightmark` ihre Werte erhalten; aus den vorherigen Beispielen sollte ersichtlich sein, dass sie Daten erhalten. Wie in Abschnitt 4.3.4 erläutert, geben die Gliederungsbefehle ihr Titelfragment an Befehle wie `\sectionmark` weiter. Diese werden über `\markboth` oder `\markright` veranlasst, Marken oder eben keine Marken zu erzeugen. Das Kolumnentitel-Layout `fancy` definiert nun zwei derartige Befehle: `\chaptermark` und `\sectionmark`, wenn die aktuelle Klasse einen `\chapter`-Befehl definiert, oder andernfalls `\sectionmark` und `\subsectionmark`. Wenn man also einen anderen Textmarkenmechanismus verwenden möchte, oder selbst nur ein etwas abgewandeltes Kolumnentitel-Layout (z.B. ohne Gliederungsnummern in der Kopfzeile oder ohne `\MakeUppercase` für den Markentext), dann muss man diese Befehle unter Umständen selbst definieren.

Das nächste Beispiel ist eine Kopie von Beispiel 4-4-7 auf Seite 236, nur dass dieses Mal selbst definierte `\sectionmark`- und `\subsectionmark`-Befehle zum Einsatz kommen, die den Abstand zwischen der Zahl und dem Text verringern und auf den Befehl `\MakeUppercase` verzichten.

	<i>1 Test</i>
1 Test	
1.1 Titel B	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
6	

<i>1 Test</i>	<i>1.2 Titel B2</i>
1.2 Titel B2	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
7	

```
\usepackage{fancyhdr}
\pagestyle{fancy}
\renewcommand\sectionmark[1]
{\markboth{\thesection\ #1}{}}
\renewcommand\subsectionmark[1]
{\markright{\thesubsection\ #1}}
% \sample wie zuvor definiert
\section{Test}
\subsection{Titel B} \sample
\subsection{Titel B2}\sample
```

Bsp.
4-4-10

„Benannte“ Layouts
definieren

Bisher wurde das Layout fancy in allen Beispielen immer aufs Neue angepasst. Mit dem Paket fancyhdr kann man jedoch auch eigene Anpassungen unter einem Namen speichern und diese dann über die Befehle `\pagestyle` oder `\thispagestyle` auswählen. Das geschieht mithilfe der Deklaration `\fancypagestyle`. Sie verfügt über zwei Argumente: den Namen des Kolumnentitel-Layouts und die Anpassungen, die vorgenommen werden sollen, wenn das Layout später aufgerufen wird. Nicht definierte (oder gelöschte) Felder, sowie die Einstellungen für Linienstärken werden aus den fancyhdr-Standards übernommen. Das erklärt, warum zunächst alle Felder mit `\fancyhf` gelöscht werden.

6	Memo
Text für unsere Seite, der immer und immer wieder verwendet wird.	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
28. Juli 2005	

Memo	7
te, der immer und immer wieder verwendet wird.	
Text für unsere Seite, der immer und immer wieder verwendet wird.	
28. Juli 2005	

```
\usepackage{fancyhdr}
\fancypagestyle{memo}{\fancyhf{}}
\fancyhead[R0,LE]{\thepage}%
\fancyhead[LO,RE]{Memo}%
\fancyfoot[R]{\scriptsize\today}%
\renewcommand\headrulewidth{1pt}%
\pagestyle{memo}
% \sample definiert wie zuvor
\sample \par \sample\sample
```

Bsp.
4-4-11

Einige L^AT_EX-Befehle, wie `\chapter` und `\maketitle`, wechseln mithilfe von `\thispagestyle` automatisch auf das Layout plain, und ignorieren damit das aktuell gültige Layout. Um die Kolumnentitel für solche Seiten anzupassen, kann man entweder diese Befehle entsprechend umdefinieren (was ziemlichen Ärger machen kann) oder das plain-Layout über `\fancypagestyle` abändern. Es ist genaugenommen nicht wirklich der richtige Ansatz, ein Standardlayout wie plain zu verändern. Der eigentliche Fehler liegt aber bei den L^AT_EX-Standardklassen selbst,¹ die es versäumen, für solche Fälle speziell benannte Layouts zu verwenden, und stattdessen einfach den

¹Die KOMA-Script-Klassen verwenden für solche speziellen Kolumnentitel-Layouts beispielsweise Befehle wie `\chapterpagestyle`, die sich leicht anpassen lassen.

wahrscheinlichsten Kandidaten einsetzen. In der Praxis funktioniert die von fancyhdr bereitgestellte Lösung aber recht gut.

Manchmal ist es wünschenswert, das Layout in Abhängigkeit von Gleitobjekten auf der aktuellen Seite zu verändern. Für diesen Zweck sieht das Paket fancyhdr eine Reihe von Steuerbefehlen vor. Sie können in den Layoutdeklarationen eingesetzt werden, wodurch das Layout auf die An- oder Abwesenheit von Fußnoten (`\iffootnote`) sowie Gleitobjekten im oberen (`\iftopfloat`) oder unteren Bereich (`\ifbottomfloat`) auf der aktuellen Seite reagieren kann. Jeder von ihnen besitzt zwei Argumente: Das erste wird aktiv, wenn die Bedingung eintritt, das zweite, wenn nicht.

Im nächsten Beispiel fällt durch Umdefinieren von `\headrulewidth` die Linie unter der Kopfzeile weg, wenn im oberen Bereich Gleitobjekte erscheinen. Außerdem enthält der obere Kolumnentitel unterschiedliche Texte, je nachdem ob oben auf der Seite Gleitobjekte erscheinen oder nicht.

Von Gleitobjekten
abhängige
Kolumnentitel-Layouts

SPEZIAL
Abbildung oben Text für unsere Seite, der immer und immer wieder verwendet wird. 6

NORMAL
Text für unsere Seite, der immer und immer wieder verwendet wird. 7

```
\usepackage{fancyhdr}
\pagestyle{fancy} \fancyhf{}
\head{\iftopfloat{SPEZIAL}{NORMAL}}
\cfoot{\thepage}
\renewcommand\headrulewidth
    {\iftopfloat{0pt}{0.4pt}}

% \sample definiert wie zuvor
\sample
\begin{figure}[t]
  \centering
  \fbox{Abbildung oben}
\end{figure}
\sample
```

Bsp.
4-4-12

Zum Anpassen des Layouts für reine Gleitobjektseiten gibt es einen ähnlichen Befehl, `\iffloatpage` – um z.B. die oberen Kolumnentitel auf solchen Seiten zu unterdrücken. Wenn das Layout von mehreren Bedingungen abhängig sein soll, können die Steuerbefehle verschachtelt werden, auch wenn das mit der Zeit etwas unübersichtlich wird. Wenn z.B. die Linie unter der Kopfzeile auf allen Seiten, die nur Gleitobjekte oder die Gleitobjekte im oberen Bereich enthalten, ausgeblendet werden soll, müsste man `\headrulewidth` folgendermaßen definieren:

Layout für
Gleitobjektseiten

```
\renewcommand\headrulewidth
    {\iftopfloat{0pt}{\iffloatpage{0pt}{0.4pt}}}
```

In Wörterbüchern und ähnlichen Werken enthält der lebende Kolumnentitel häufig das erste und das letzte Wort, die auf einer Seite erklärt werden, um das Auffinden der Informationen zu erleichtern. Dieses Schema lässt sich leicht verwirklichen, indem man einen geeigneten Befehl definiert, der für jeden Wörterbucheintrag eine Marke setzt. Im nächsten Beispiel verwenden wir $\text{\LaTeX}$ s `right-mark` um solche Marken zu speichern und greifen mit `\firstrightmark` und `\lastrightmark` aus dem Paket `extramarks` darauf

Kolumnentitel für
Wörterbücher

zu. Auf Seiten, die nur einen einzigen Eintrag behandeln, erfolgt nur ein Eintrag im Kolumnentitel. Dazu wird mithilfe des `ifthen`-Paketes geprüft, ob beide Befehle den gleichen Wert enthalten. Die lebenden Kolumnentitel für den Index dieses Buches wurden mit einem ähnlichen Mechanismus erzeugt.

Fahne—Marke	
Fahne	Formatierter Text, nicht in Seiten aufgeteilt.
Kolumne	Textspalte.
Marke	Steht in der
6	

OR	
Fahne, um mit der OR zu kommunizieren.	
OR	Output-Routine.
7	

```
\usepackage{ifthen,fancyhdr,extramarks}
\pagestyle{fancy} \fancyhf{}
\newcommand\combinemarks{\ifthenelse
  {\equal{\firstrightmark}%
    {\lastrightmark}}%
  {\firstrightmark} gleiche Werte
  {\firstrightmark---\lastrightmark}}
\head{\combinemarks} \cfoot{\thepage}
\newcommand\idxitem[1]{\par\vspace{8pt}%
  \textbf{\#1}\markright{\#1}%
  \quad\ignorespaces}

\idxitem{Fahne} Formatierter Text,
  nicht in Seiten aufgeteilt.
\idxitem{Kolumne} Textspalte.
\idxitem{Marke} Steht in der Fahne,
  um mit der OR zu kommunizieren.
\idxitem{OR} Output-Routine.
```

Bsp.
4-4-13

*Probleme beim
Zweispaltensatz*

Wörterbücher werden oft mit zwei oder mehr Spalten pro Seite gesetzt. Unglücklicherweise behandelt der L^AT_EX-Standardmodus `twocolumn` Marken fehlerhaft - `\leftmark` bezieht sich stets auf die zweite Spalte, anstatt die erste Marke der ersten Spalte wiederzugeben. Wenn das ein Problem darstellt, sollte man die neue Version aus dem Paket `fixltx2e` verwenden. Oder man setzt das Paket `multicol` ein, das die Marken ebenfalls korrekt verarbeitet.

4.4.3 truncate – Texte auf eine bestimmte Länge kürzen

Der begrenzte verfügbare Platz ist immer ein potentielles Problem für lebende Kolumnentitel: Ist der Text zu lang, dann überschreibt er unter Umständen die Seitennummer oder anderes Material. Für solche Fälle eignet sich das Paket `truncate` von Donald Arseneau. Es stellt einen Befehl bereit, mit dem sich ein vorgegebener Text auf eine bestimmte Länge kürzen lässt.

```
\truncate[zeichen]{breite}{text}
```

Wenn das Argument *text* für die angegebene *breite* zu groß ist, wird es abgeschnitten und ein oder mehrere Fortführungszeichen an sein Ende gesetzt. Wenn das optionale Argument *zeichen* fehlt, werden in `\TruncateMarker` gespeicherte Standardzeichen eingesetzt (das Paket verwendet den Wert `\,``\dots`).

Normalerweise werden Kürzungen nur an Wortgrenzen vorgenommen, und nur da, wo Wörter nicht mit „~“, d.h. durch einen geschützten Leerraum verbunden sind. Aus diesem Grunde wird der Text im folgenden Beispiel nach dem Wort *hier* abgeschnitten. Das Beispiel veranschaulicht auch den

Gebrauch eines *zeichen*-Argumentes, das extra in geschweifte Klammern gesetzt werden muss, um die eckigen Klammern, die als Teil des Textes erscheinen sollen, zu verstecken. Damit man sich den Platz, der von dem gekürzten Text eingenommen werden darf, besser vorstellen kann, wurden links und rechts `|`-Zeichen angefügt.

```
\usepackage{truncate}
|Der Text hier wird~gekappt|

|\truncate{50pt}
{Der Text hier wird~gekappt}|

|\truncate[{\, [. . .]}]{100pt}
{Der Text hier wird~gekappt}|
```

Bsp.
4-4-14

Man kann den Text auch innerhalb eines Wortes abschneiden, wenn man eine der Paketooptionen `hyphenate`, `breakwords` oder `breakall` angibt. Die ersten beiden kappen Wörter entsprechend ihrer Silbentrennung, mit dem Unterschied, dass `breakwords` das Trennzeichen unterdrückt (die gängigere Variante). Mit der dritten Option kann man den Text an jeder beliebigen Stelle abschneiden. Mithilfe dieser Optionen würde das vorige Beispiel folgendes Ergebnis liefern:

Der Text hier wird ge-[..]	(hyphenate)
Der Text hier wird ge[..]	(breakwords)
Der Text hier wird gekap[.]	(breakall)

Der Text (ob gekürzt oder nicht) wird entsprechend der Voreinstellungen linksbündig in einer Box mit der angegebenen *breite* gesetzt. Die Paketooption `fit` sorgt dafür, dass der formatierte Text seine natürliche Breite bis zum Maximum der vorgegebenen *breite* behält.

Im nächsten Beispiel wird das Paket `truncate` mit `fancyhdr` kombiniert. Man beachte den Einsatz der Option `fit`. Ohne diese Option wäre die Kopfzeile immer linksbündig (um den Effekt deutlicher zu machen, wurde `\headwidth` etwas verkleinert).

1 ABSCHNITT ...

1 Abschnitt mit langem Titel

Text für unsere Seite,
der immer und immer

6

1 ABSCHNITT ...

wieder verwendet wird.

Text für unsere Seite,
der immer und immer
wieder verwendet
wird.

7

```
\usepackage[fit]{truncate}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhf{} % -- alle Bereiche leeren
\fancyhead[R0,LE]{\truncate
{.95\headwidth}{\leftmark}}
\fancyfoot[C]{\thepage}
% \sample definiert wie zuvor
\section{Abschnitt mit langem Titel}
\sample \par \sample
```

Bsp.
4-4-15

4.5 Visuelle Formatierung

In der letzten Bearbeitungsphase für ein wichtiges Dokument ist es oft notwendig, einige manuelle Formatierungen vorzunehmen, um ungünstige Seitenumbrüche zu vermeiden. Für diesen Zweck stellt $\text{\LaTeX}$ die Befehle `\pagebreak`, `\nopagebreak`, `\newpage` und `\clearpage` sowie die Deklaration `\samepage` zur Verfügung, wobei letztere in $\text{\LaTeX 2}_\epsilon$ als veraltet angesehen wird. Durch die Deklaration `\samepage` in Verbindung mit einer geeigneten Anzahl von `\nobreak`-Befehlen kann man erreichen, dass bestimmte Teile des Dokumentes zusammengehalten werden. Unglücklicherweise sind die Ergebnisse oft nicht zufriedenstellend; so verlängert $\text{\LaTeX}$ z.B. niemals eine Seite über ihre nominale Höhe (`\textheight`) hinaus, sondern bewegt eher alles, was sich im Gültigkeitsbereich der `\samepage`-Deklaration befindet, auf die nächste Seite. Die Alternative ist der $\text{\LaTeX 2}_\epsilon$ -Befehl `\enlargethispage*`, der im Folgenden beschrieben wird.

Bei der Buchproduktion ist es üblich, eine gewisse Anzahl von Seiten (normalerweise Doppelseiten) zu verkürzen oder zu verlängern, um dadurch spätere ungünstige Seitenumbrüche zu vermeiden. Das bedeutet, dass die nominale Höhe dieser Seiten um ein bestimmtes Maß, zum Beispiel eine Zeile (`\baselineskip`), vergrößert oder verkleinert wird. $\text{\LaTeX 2}_\epsilon$ unterstützt diese Praxis durch den Befehl `\enlargethispage{größe}`.

`\enlargethispage{größe}`

Wenn man also die Länge einiger Seiten um eine Zeile verlängern oder verkürzen möchte, kann man folgende Definition verwenden:

```
\newcommand\longpage[1][1]{\enlargethispage{#1\baselineskip}}
\newcommand\shortpage[1][1]{\enlargethispage{-#1\baselineskip}}
```

Diese Befehle müssen zwischen zwei Absätzen auf den betreffenden Seiten eingefügt werden.¹ Der Befehl `\enlargethispage` vergrößert die Höhe `\textheight` der aktuellen Seite, ändert ansonsten jedoch keinen der Formatierungsparameter. Wenn also `\flushbottom` eingestellt ist, füllt der Text die gesamte Länge der fraglichen Seite aus, falls erforderlich auch durch Dehnen oder Stauchen der vertikalen Leerräume auf der Seite. Dadurch setzen die obigen Definitionen genau eine Zeile Text mehr bzw. weniger auf eine Seite, während die Position der übrigen Zeilen unverändert bleibt. Das ist wichtig, denn auf diese Weise bleibt das einheitliche Erscheinungsbild gewahrt.

`\enlargethispage*{größe}`

Die Sternform des Befehls, `\enlargethispage*`, vergrößert oder verkleinert ebenfalls die Seitenlänge. Dabei wird jedoch die resultierende Seite so weit wie möglich gestaucht (d.h. soweit es der verfügbare Weißraum auf der Seite

¹Aufgrund der vielen Beispiele in diesem Buch war dieser Trick einige Male notwendig, um halbleere Seiten zu vermeiden. So wurden z.B. vier Seiten dieses Kapitels ab Seite 228 um eine Zeile verlängert oder die vorherige Doppelseite um eine Zeile gekürzt. Das erwies sich aufgrund der vielen (umfangreichen) Beispiele in Abschnitt 4.4.2 als erforderlich – alle anderen Formatierungen führten irgendwo zu einer halbleeren Seite.

zulässt). Dieses Verfahren kann hilfreich sein, wenn man einen bestimmten Teil seines Dokumentes auf einer Seite zusammenhalten möchte, auch wenn die Seite dadurch etwas zu lang wird. (Ansonsten kann man einfach die Umgebung `minipage` verwenden.) Der Trick dabei ist, im Voraus ausreichend zusätzlichen Platz anzufordern und den Seitenumbruch dann dort, wo er erfolgen soll, zu erzwingen. Zum Beispiel:

```
\enlargethispage*{100cm}      % absurde Forderung
\begin{center}
\begin{tabular}{llll}
....                          % etwas zu lange
\end{tabular}                 % Tabelle
\end{center}
\pagebreak                    % erzwungener Seitenumbruch
```

Aus der obigen Beschreibung ist ersichtlich, dass beide Befehle nur in der letzten Bearbeitungsphase verwendet werden sollten, da jegliche spätere Änderung im Text (im ungünstigsten Fall selbst das Hinzufügen oder Löschen eines einzigen Wortes) die manuelle Formatierung ad absurdum führen und damit sehr hässliche Seiten erzeugen kann.

Es kann hilfreich sein, sich die Gründe vor Augen zu führen, warum $\TeX$ Seiten an einer bestimmten Stelle umbricht, und wie viel Spielraum für bestimmte Seiten verfügbar ist, wenn man endgültige Seitenumbrüche manuell korrigiert. Dies gilt z.B. für Veröffentlichungen wie dieses Buch (das durch seine vielen Beispiele, in denen kein Seitenumbruch erfolgen darf, recht anspruchsvoll ist). Eine Beschreibung der Werkzeuge für diesen Zweck findet sich in Anhang B.3.2.

4.5.1 nextpage – Erweiterungen für `\clearpage`

Standard- $\LaTeX$ beendet den aktuellen Absatz und die laufende Seite mithilfe der Befehle `\clearpage` und `\cleardoublepage`, nachdem alle aufgelaufenen Gleitobjekte gesetzt wurden. (Falls erforderlich wird dazu eine Reihe von Gleitobjektseiten erzeugt.) Bei zweiseitigem Druck sorgt `\cleardoublepage` außerdem dafür, dass die nächste Seite immer eine rechte (ungerade) Seite ist, auch wenn dazu eine Seite ohne Text eingefügt werden muss. Diese zusätzliche Seite enthält immer noch die Kolumnentitel des momentan gültigen Layouts, was nicht immer wünschenswert ist.

1	2
1 Ein Test	1 EIN TEST
1.1 Ein Abschnitt	
Etwas Text für die Seite.	

```
\pagestyle{headings}
% rechte Seite, in diesem Beispiel
% links, aufgrund von:
\setcounter{page}{1}

\section{Ein Test}
\subsection{Ein Abschnitt}
Etwas Text für die Seite.
\cleardoublepage
\section{Noch ein Abschnitt}
Dieser gehört auf Seite 3.
```

Das Paket `nextpage` von Peter Wilson erweitert diesen Ansatz durch die Befehle `\cleartoevenpage` und `\cleartooddpage`. Beide verfügen über ein optionales Argument, das Text aufnehmen kann, welcher auf einer gegebenenfalls erzeugten Seite stehen soll. Im nächsten Beispiel wird mit ihrer Hilfe der Befehl `\myclearpage` definiert, der LEERE SEITE auf solche generierten Seiten schreibt.

1	2
1 Ein Test 1.1 Ein Abschnitt Etwas Text für die Seite.	1 EIN TEST LEERE SEITE

```

\usepackage{nextpage}
\pagestyle{headings}
\newcommand\myclearpage{%
  \cleartooddpage
  [\vspace*{\fill} \centering
  LEERE SEITE \vspace*{\fill}]}
\setcounter{page}{1} %rechte Seite
\section{Ein Test}
\subsection{Ein Abschnitt}
Etwas Text für die Seite.
\myclearpage
\section{Noch ein Abschnitt}
Dieser gehört auf Seite 3.

```

Bsp.
4-5-2

Mit dieser Befehlsfolge erscheint immer noch ein lebender Kolumnentitel, aber mittlerweile ist ja bekannt, wie sich das beheben lässt: Man muss der Definition lediglich `\thispagestyle{empty}` hinzufügen.

Das Paket `nextpage` stellt außerdem ähnliche Funktionen mit den Befehlen `\movetoevenpage` und `\movetooddpage` zur Verfügung – nur werden hier aufgelaufene Gleitobjekte nicht ausgegeben.

4.6 Layouts mit Klasse

Das Seitenlayout wird normalerweise durch die Dokumentenklasse festgelegt, so dass es nicht überraschen sollte, dass die Verfahren und Pakete, die in diesem Kapitel beschrieben werden, normalerweise im Verborgenen zur Anwendung kommen (innerhalb einer Dokumentenklasse).

Die Standardklassen bedienen sich unmittelbar der $\text{\LaTeX}$ -Parameter und -Schnittstellen, um Seitenverhältnisse, lebende Kolumnentitel und andere Elemente zu definieren. Neuere Klassen behandeln jedoch auch häufig bestimmte Aspekte des Seitenlayouts mithilfe von Paketen wie `geometry`.

In diesem Abschnitt werden zwei derartige Lösungen vorgestellt. Das CTAN-Archiv enthält möglicherweise noch mehr dieser Schätze.

4.6.1 KOMA-Script – Ein Ersatz für `article` et al.

Die KOMA-Script-Klassen von Markus Kohm, auf der Basis früherer Entwicklungen von Frank Neukam, sind ein passender Ersatz für die Standardklassen `article/report/book`. Sie arbeiten mit den typographischen Regeln nach Tschichold. Die Klasse `article` wird zum Beispiel zu `scrartcl`.

Die KOMA-Script-Klassen erzeugen ihr Seitenlayout mithilfe des Paketes `typearea` (siehe Abschnitt 4.2.3), wobei sie dessen Paketoptionen als Klassenoptionen anbieten. Ein erweitertes Seitendesign ist mit dem Paket `scrpage2`

möglich, das über ähnliche Funktionen verfügt wie das fancyhdr-Paket. Genau wie typearea kann es ebenfalls eigenständig in Kombination mit einer der Standardklassen eingesetzt werden. Layouteinstellungen, wie Fontkontrolle, Layout der Legenden usw., wurden um Anpassungsmöglichkeiten erweitert, die in der Präambel eines Dokumentes zum Einsatz kommen.

Neben allen Funktionen der Standardklassen bieten die KOMA-Script-Klassen dem Anwender zusätzliche Steuerungsmöglichkeiten im Vor- und Nachspann, sowie eine Reihe anderer nützlicher Erweiterungen.

Die Sammlung ist gut dokumentiert. Das Handbuch, das in deutscher und englischer Sprache vorliegt, erläutert ausführlich alle Funktionen. Die deutsche Dokumentation wurde von der deutschen T_EX-Anwendergruppe DANTE auch als nett gesetztes Buch [102] veröffentlicht.

4.6.2 memoir – Setzen komplexer Werke

Die Klasse memoir von Peter Wilson wurde ursprünglich als Alternative für die Standardklasse book entwickelt. Sie umfasst viele Funktionen, die man sonst nur als Erweiterungspakete findet. Die aktuelle Version eignet sich auch als Ersatz für die Klasse article und kann daher für alle Arten von Veröffentlichungen – von kurzen Vermerken bis hin zu komplexen Büchern – eingesetzt werden.

Es unterstützt unter anderem einen erweiterten Satz an Größen für Grundschriften (von 9pt bis 17pt), sowie die konfigurierbare Gestaltung von Abschnittsüberschriften, Kolumnentiteln und Legenden. Für alle diese Elemente gibt es vordefinierte Layouts und neue können nach Bedarf deklariert werden. Die Klasse unterstützt Deklarationen, mit denen sich alle Größeneinstellungen für Seiten, Texte und Seitenränder vornehmen lassen, sowie die Verwendung von Beschnittmarken. Viele Komponenten der Klasse sind für Anwender, die andere Klassen um bestimmte Funktionen erweitern wollen, auch als eigenständige Pakete verfügbar (z.B. für Epigraphie oder zum Formatieren von Legenden).

Wie die KOMA-Script-Klassen, enthält auch die Klasse memoir ein ausgezeichnetes, fast 200 Seiten umfassendes Handbuch, das sich mit allen Fragen des Dokumentendesigns befasst und zeigt, wie sich potentielle Schwierigkeiten mit memoir beheben lassen.

KAPITEL 5

Tabellen

5.1 $\text{\LaTeX}$ -Standardumgebungen	248
5.2 <code>array</code> – Erweiterung von <code>tabular</code> & Co.	252
5.3 Errechnen von Spaltenbreiten	257
5.4 Mehrseitige Tabellen	263
5.5 Farbige Tabellen	272
5.6 Anpassen von Linien und Abständen	273
5.7 Sonstige Erweiterungen	280
5.8 Tabellenfußnoten	285
5.9 Verwendungsmöglichkeiten	287

Daten lassen sich oftmals am besten in Tabellenform präsentieren. Daher verfügt $\text{\TeX}$ über sehr leistungsfähige Basisbefehle, um Text in Zeilen und Spalten anzuordnen. Da diese Befehle jedoch nur Low-Level-Funktionen für die Formatierung bereitstellen, wurden auf ihrer Grundlage zahlreiche Makropakete entwickelt, die eine High-Level-Befehlssprache mit benutzerfreundlicher Schnittstelle bilden.

$\text{\LaTeX}$ verfügt über zwei Umgebungen für Tabellen. Meistens wird die `tabular`-Umgebung oder ihr Äquivalent für Formeln, `array`, verwendet. Unter gewissen Umständen kann sich jedoch auch die Umgebung `tabbing` als hilfreich erweisen.

Tabellen sind normalerweise relativ umfangreich, so dass sie fähig sein müssen, innerhalb des Textes zu „gleiten“, damit die Seiten an geeigneter Stelle umbrochen werden können. Die in diesem Kapitel beschriebenen Umgebungen befassen sich hauptsächlich mit dem Tabellenlayout. Um korrekte Seitenumbrüche zu erhalten, werden sie deshalb oft innerhalb der `table`-Umgebung gesetzt, die in Kapitel 6 beschrieben ist. Hiervon ausgenommen sind die Umgebungen für mehrseitige Tabellen, die in Abschnitt 5.4 vorgestellt werden. Diese sollten nie zusammen mit dem Gleitobjektmechanismus von $\text{\LaTeX}$ verwendet werden.

Man darf die `tabular`- nicht mit der `table`-Umgebung verwechseln. Mit `tabular` lässt sich Tabellentext in Spalten anordnen, während `table`

Tabellen in Gleitobjektumgebungen

ein logisches Dokumentenelement (für Tabellen) darstellt, dessen Inhalt zusammengehalten wird und im Text gleiten kann. Dabei kann eine `table`-Umgebung auch mehrere `tabular`-Umgebungen enthalten.

Nach einem kurzen Blick auf die `tabbing`-Umgebung wendet sich dieses Kapitel den Erweiterungen der $\text{\LaTeX}$ -Basisumgebungen `tabular` und `array` durch das Paket `array` zu. Dieses Paket verfügt über einen erweiterten Funktionsumfang; vor allem lassen sich Absätze flexibler positionieren sowie Spalten- und Zeilenabstände besser einstellen. Außerdem können für die Tabellenpräambel neue Tabellenformatkürzel definiert werden. Eine Reihe von Paketen stellen auf Grundlage der Basisbefehle des `array`-Paketes noch weitere Sonderfunktionen zur Verfügung. Durch Kombinieren dieser Funktionen kann man selbst komplexe Tabellen auf einfache Weise zusammenstellen. Beispielsweise beinhalten die Pakete `tabularx` und `tabulary` zusätzliche Spaltentypen mit automatischer Berechnung der Spaltenbreite.

Die $\text{\LaTeX}$ -Standardumgebungen für Tabellen lassen keine Seitenumbrüche in Tabellen zu. Es werden jedoch einige Beispiele für mehrseitige Tabellen gezeigt, die man mithilfe der Umgebungen `supertabular` und `longtable` (aus den gleichnamigen Paketen) erzeugen kann.

Anschließend erfolgt ein kurzer Blick auf farbige Tabellen und auf mehrere Pakete, mit deren Hilfe sich Linien und der sie umgebende Leerraum in Tabellen genauer einstellen lassen. Als Nächstes werden Tabelleneinträge besprochen, die sich mithilfe des Paket `multirow` über mehrere Zeilen erstrecken. Darauf folgt das Paket `dcolumn`, dessen Mechanismus es erlaubt, Zahlenspalten an ihrem Dezimalzeichen auszurichten.

Auch der Einsatz von Fußnoten in Tabellen wird beleuchtet: Das Paket `threeparttable` kombiniert auf komfortable Weise Tabellenfußnoten und Legendes mit dem Tabellenlayout.

Der letzte Abschnitt enthält praktische Hinweise für den Umgang mit verschachtelten Tabellen und mit umfangreichen Einträgen, die sich über mehrere Spalten erstrecken.

Mathematisch orientierte Leser sollten das Kapitel über Höhere Mathematik, insbesondere den Abschnitt 8.2 auf Seite 481, konsultieren, in dem die Ausrichtungsstrukturen für Formeln beschrieben werden. Weitere Beispiele für Tabellenlayouts befinden sich bei den Erläuterungen zum Graphikpaket in Abschnitt 10.3 auf Seite 646.

5.1 $\text{\LaTeX}$ -Standardumgebungen

$\text{\LaTeX}$ verfügt über zwei Familien von Umgebungen, mit deren Hilfe sich Daten in Spalten anordnen lassen, und zwar die Umgebung `tabbing` sowie die Umgebungen `tabular` und ihr Gegenstück in Formeln `array`. Sie unterscheiden sich hauptsächlich in den folgenden Punkten:

- Die Umgebung `tabbing` ist nicht so flexibel wie die `tabular`-Umgebung. Erstere erzeugt immer einen eigenen Absatz, während Letztere beliebig innerhalb von Text oder Formeln positioniert werden kann.
- Die Umgebung `tabbing` erlaubt Seitenumbrüche, die Standardform der `tabular`-Umgebung nicht.

- Für die Umgebung `tabbing` muss jede Tabulatorposition explizit festgelegt werden. In der Umgebung `tabular` werden die Spaltenbreiten automatisch berechnet.
- `tabbing`-Umgebungen können im Gegensatz zu `tabular`-Umgebungen nicht verschachtelt werden, so dass sich Letztere für komplexe Tabellen besser eignen.

5.1.1 Die `tabbing`-Umgebung

Dieser Abschnitt befasst sich mit einigen der weniger bekannten Funktionen der `tabbing`-Umgebung. Zunächst ist es wichtig zu wissen, dass die Formatierung hier vollständig vom Anwender gesteuert wird. Bewegt man sich zu einem bestimmten Tabulator, so gelangt man immer genau an die horizontale Position, an welcher dieser gesetzt wurde. Anders als vielleicht erwartet, geschieht dies unabhängig von der aktuellen Startposition. Das bedeutet, dass sich die aktuelle Position auch zurückbewegen kann und man somit den vorherigen Text überschreibt. Der Geltungsbereich von Befehlen oder Parameter-einstellungen in den Zeilen ist normalerweise auf den Bereich zwischen zwei Tabulatoren begrenzt.

Außerdem ist zu beachten, dass die herkömmlichen L^AT_EX-Befehle für Akzente – `\'`, `\`` und `\=` – innerhalb der `tabbing`-Umgebung eine spezielle Bedeutung haben. Die Akzente sind stattdessen als `\a'`, `\a`` und `\a=` verfügbar. Der Befehl `\-`, der normalerweise eine mögliche Trennung kennzeichnet, ist ebenfalls undefiniert. Dies ist allerdings nicht weiter wichtig, da Zeilen in einer `tabbing`-Umgebung ohnehin nicht umbrochen werden.

*Andere Namen für
Akzentbefehle*

Der Stilparameter `\tabbingsep`, der zusammen mit dem Befehl `\'` verwendet wird, setzt Text rechtsbündig in einer bestimmten Entfernung vom nachfolgenden Tabulator. Die Voreinstellung für den Parameter entspricht dabei derjenigen von `\labelsep`, welche normalerweise 5pt beträgt.

Üblicherweise werden Tabulatorpositionen definiert, indem man entweder passende Stellen in einer zu setzenden Zeile markiert (Befehl `\=`) oder explizit den Abstand zum nächsten Tabulator angibt. Mit dem Befehl `\kill` kann man eine Zeile beenden, die nur zum Setzen von Tabulatoren benötigt wird: Die Zeile selbst wird dann nicht gesetzt. Dies wird im nächsten Beispiel deutlich, das in der fünften Zeile auch die Änderung eines Tabulators zeigt.

	<code>\begin{tabbing}</code>
	Erster \= Zweiter \= Dritter \= <code>\kill</code>
eins zwei drei vier	<code>eins \> zwei \> drei \> vier \\\</code>
eins zwei èè (Akzentbefehle)	<code>eins \> zwei \> \a'{e}\a'{e} \\' (Akzentbefehle) \\\[3mm]</code>
Ein (!) neuer Tabulator zwei	<code>Ein (!) neuer Tabulator\ \= zwei \\\</code>
eins drei ist jetzt links von zwei	<code>eins \> zwei \> drei ist jetzt links von zwei</code>
	<code>\end{tabbing}</code>

Bsp.
5-1-1

Akzente innerhalb von Befehlsdefinitionen, die möglicherweise in einer `tabbing`-Umgebung zum Einsatz kommen, muss man über ihre `\a...`-Formen

setzen, da die Standard-Akzentbefehle wie `\'` sonst, wie im folgenden Beispiel, als `tabbing`-Befehle interpretiert werden. Man kann auch das Paket `inputenc` verwenden, um Akzentbuchstaben direkt einzugeben.

	<pre>\usepackage[ansinew]{inputenc} \newcommand\acafe{Caf\'e} \newcommand\bcafe{Caf\'a\'e} \newcommand\ccafe{Caf\'e}</pre>	
	<pre>\begin{tabbing}</pre>	
Tab eins Tab zwei	<pre>Tab eins \= Tab zwei \\</pre>	
7 bit Caf e	<pre>7 bit \> \acafe \\</pre>	
7 bit Café	<pre>7 bit \> \bcafe \\</pre>	
8 bit Café	<pre>8 bit \> \ccafe \end{tabbing}</pre>	

Bsp.
5-1-2

Alternativ dazu bietet das `Tabbing`-Paket (von Jean-Pierre Drucbert) eine `Tabbing`-Umgebung, in welcher die Akzentbefehle *nicht* umdefiniert werden. Stattdessen werden die Namen der Tabulatorbefehle als `\TAB'`... gebildet.

	<pre>\usepackage[ansinew]{inputenc} \usepackage{Tabbing} % Definitionen wie zuvor</pre>	
	<pre>\begin{Tabbing}Tab eins \TAB= Tab zwei\\</pre>	
Tab eins Tab zwei	<pre>7 bit \TAB> \acafe \\</pre>	
7 bit Café	<pre>7 bit \TAB> \bcafe \\</pre>	
8 bit Café	<pre>8 bit \TAB> \ccafe \end{Tabbing}</pre>	

Bsp.
5-1-3

Die Umgebung `tabbing` eignet sich besonders, um Informationen in Spalten mit bekannter, gleich bleibender Breite anzuordnen. Das folgende Beispiel stammt aus Tabelle A.1 auf Seite 888.

	<pre>\newcommand\lenrule[1]{\makebox[#1]{% \rule{.4pt}{4pt}\hrulefill\rule{.4pt}{4pt}}}</pre>	
	<pre>\begin{tabbing}</pre>	
	<pre>dd\quad \= \hspace{.55\linewidth} \= \kill</pre>	
pc Pica = 12pt	<pre>pc \> Pica = 12pt \> \lenrule{1pc} \\</pre>	
cc Cicero = 12dd	<pre>cc \> Cicero = 12dd \> \lenrule{1cc} \\</pre>	
cm Zentimeter = 10mm	<pre>cm \> Zentimeter = 10mm \> \lenrule{1cm} \\</pre>	
	<pre>\end{tabbing}</pre>	

Bsp.
5-1-4

5.1.2 Die tabular-Umgebung

Für komplexere Tabellen sind meistens die `tabular`-ähnlichen Umgebungen von $\text{\LaTeX}$ besser geeignet. Sie ordnen Material horizontal in Reihen (getrennt durch `\\`) und vertikal in Spalten (getrennt durch `&`) an.

<pre>\begin{array}[pos]{spalten-dekl}</pre>	<pre>zeilen \end{array}</pre>
<pre>\begin{tabular}[pos]{spalten-dekl}</pre>	<pre>zeilen \end{tabular}</pre>
<pre>\begin{tabular*}[breite]{pos}{spalten-dekl}</pre>	<pre>zeilen \end{tabular*}</pre>

Die `array`-Umgebung ist im Wesentlichen eine `tabular`-Umgebung für Formeln. Zwar werden die Tabelleneinträge als Formeln gesetzt und der Spaltenabstand ist, wie im Folgenden beschrieben, anders voreingestellt, aber abgesehen davon verfügen beide Umgebungen über identische Funktionen.

<code>l</code>	Linksbündige Spalte.
<code>c</code>	Zentrierte Spalte.
<code>r</code>	Rechtsbündige Spalte.
<code>p{breite}</code>	Entspricht <code>\parbox[t]{breite}</code> .
<code> </code>	Fügt eine vertikale Linie zwischen zwei Spalten ein. Der Spaltenabstand bleibt dabei unverändert.
<code>@{dekl}</code>	Unterdrückt den Spaltenabstand und fügt stattdessen <i>dekl</i> ein.
<code>*{anz}{kürz}</code>	Entspricht <i>anz</i> Exemplaren von <i>kürz</i> .

Tabelle 5.1: Präambel-Formatkürzel der tabular-Umgebung

Die Umgebung `tabular*` verfügt über ein zusätzliches Argument für die erforderliche Gesamtbreite der Tabelle. Wie noch erklärt wird, kann T_EX die Spaltenzwischenräume so anpassen, dass diese Tabellenbreite erreicht wird.

Tabelle 5.1 zeigt die Formatkürzel für das Argument *spalten-dekl*, die in der Tabellenpräambel von Umgebungen der `tabular`-Familie zur Verfügung stehen. Das Paket `array`, das im nächsten Abschnitt vorgestellt wird, erweitert diese Liste noch einmal.

Das visuelle Erscheinungsbild `tabular`-ähnlicher Umgebungen ist von verschiedenen Layoutparametern abhängig. Diese Parameter lassen sich mithilfe der Befehle `\setlength` oder `\addtolength` an jeder beliebigen Stelle im Dokument festlegen. Sie können für das ganze Dokument oder lokal verändert werden. In letzterem Falle sollte man ihren Gültigkeitsbereich explizit mithilfe geschweifeter Klammern oder durch eine Umgebung begrenzen.

Layoutparameter

`\arraycolsep` Hälfte des Spaltenabstandes in einer `array`-Umgebung (voreingestellt auf 5pt).

`\tabcolsep` Hälfte des Spaltenabstandes in einer `tabular`-Umgebung (voreingestellt auf 6pt).

`\arrayrulewidth` Stärke der vertikalen Linie zwischen zwei Spalten (wenn in der Tabellenpräambel `|` angegeben wurde) oder der Linien, die durch `\hline`, `\cline` oder `\vline` erzeugt werden (voreingestellt auf 0.4pt).

Bei Verwendung des Paketes `array` wird diese Linienstärke auch beim Errechnen der gesamten Tabellenbreite berücksichtigt. (Standard-L^AT_EX setzt die Linien so, dass sie die endgültige Tabellenbreite nicht beeinflussen.)

`\doublerulesep` Abstand zwischen Linien, die durch zwei aufeinander folgende `||`-Zeichen in der Tabellenpräambel oder zwei aufeinander folgende `\hline`-Befehle erzeugt werden (voreingestellt auf 2pt).

`\arraystretch` Dezimalwert, mit dem die voreingestellte Minimalhöhe von Tabellenzellen multipliziert wird. Ein Wert von 1.5 würde z.B. normalgroße Zeilen um 50% weiter voneinander entfernen. Er wird mit dem Befehl `\renewcommand` eingestellt (Standardwert 1.0).

Geändertes Kürzel	
	Fügt eine vertikale Linie ein. Dabei wird der Spaltenabstand gegenüber der ursprünglichen L ^A T _E X-Definition um die Breite dieser Linie vergrößert.
Neue Kürzel	
m{breite}	Erzeugt eine Spalte dieser <i>breite</i> . Jeder Eintrag wird proportional zum Rest der Zeile vertikal zentriert – funktioniert ähnlich wie <code>\parbox{width}</code> .
b{breite}	Stimmt mit <code>\parbox[b]{breite}</code> überein.
>{dekl}	Kann vor den Kürzeln l, r, c, p{...}, m{...} oder b{...} verwendet werden. Es fügt <i>dekl</i> direkt vor dem Eintrag in der Spalte ein.
<{dekl}	Kann nach den Kürzeln l, r, c, p{...}, m{...} oder b{...} verwendet werden. Es fügt <i>dekl</i> direkt hinter dem Eintrag in der Spalte ein.
!{dekl}	Kann überall verwendet werden und entspricht dem Kürzel , allerdings mit dem Unterschied, dass anstelle der vertikalen Linie <i>dekl</i> eingefügt wird. Somit unterdrückt dieses Kürzel im Unterschied zu @{...} nicht den normalen Spaltenabstand.

Tabelle 5.2: Zusätzliche Formatkürzel des array-Paketes

5.2 array – Erweiterung von tabular & Co.

Die Familie der `tabular`-Umgebungen, wie sie im *L^AT_EX-Manual* beschrieben ist, wurde über die Jahre mehrfach erweitert.

Dieser Abschnitt befasst sich genauer mit der erweiterten Funktionalität des Paketes `array`, das von Frank Mittelbach, mit Beiträgen von David Carlisle, entwickelt wurde. Viele der später in diesem Kapitel beschriebenen Pakete setzen auf Funktionen des `array`-Paketes auf, mit denen sie die `tabular`-Umgebung erweitern oder anpassen.

Tabelle 5.2 zeigt die neuen Möglichkeiten für das Argument *spalten-dekl* in Umgebungen der `tabular`-Familie.

5.2.1 Einige Beispiele für Präambelbefehle

Um in einer linksbündigen Spalte einen besonderen Font wie z.B. `\bfseries` zu verwenden, kann man einfach `>{\bfseries}l` schreiben. Es muss nicht mehr jeder einzelne Eintrag mit dem Befehl `\bfseries` beginnen.

```
\usepackage{array}
```

```
\begin{tabular}{|>{\large}c|>{\large\bfseries}l|>{\itshape}c|}
\hline A & B & C\\ \hline 100 & 10 & 1 \\ \hline
\end{tabular}
```

Bsp.
5-2-1

A	B	C
100	10	1

Zusätzlicher Leerraum
zwischen Zeilen

Im nächsten Beispiel wird die Deklaration `\extrarowheight` verwendet. Sie fügt über jeder Zeile einen zusätzlichen vertikalen Leerraum von 4pt ein. Dieser zusätzliche Leerraum wird allerdings nur sichtbar, wenn das Produkt

aus `\arraystretch` und `(\extrarowheight + 0.7\baselineskip)` die Zellhöhe übersteigt. Genauer gesagt, im Falle der Spaltenformatkürzel `p`, `m` oder `b`, die Höhe der *ersten Zeile* der Zelle.

Dieser Leerraum ist vor allem für Tabellen mit horizontalen Linien wichtig, da bei diesen für ein optimales Erscheinungsbild oft der Abstand zwischen den Linien und dem Tabelleninhalt korrigiert werden muss. Der Standardwert von `\extrarowheight` beträgt `Opt`.

Bsp.
5-2-2

A	B	C
100	10	<i>1</i>

```
\usepackage{array}
\setlength\extrarowheight{4pt}
\begin{tabular}{|>{\large}c|>{\large\bfseries}l|>{\itshape}c|}
\hline A & B & C\\ \hline 100 & 10 & 1 \\ \hline
\end{tabular}
```

Für Deklarationen, die mit der Spaltenoption `>` verwendet werden dürfen, existieren wenige Einschränkungen. Aus technischen Gründen, deren Erläuterung die Grenzen dieses Buches sprengen würde, kann man die Fontkodierung für eine Tabellenspalte nicht über die Formatkürzel ändern. Wenn also die aktuelle Kodierung beispielsweise nicht `T1` ist, dann wird `>{\fontencoding{T1}}\selectfont` *nicht* funktionieren. Es wird keine Fehlermeldung erzeugt, aber am Anfang der Zellen dieser Spalte können falsche Zeichen erscheinen. Erfordert eine Textspalte eine besondere Kodierung, dann muss diese explizit am Anfang jeder Zelle dieser Spalte stehen.

*Keine Unterstützung
für Änderungen der
Fontkodierung in
einem
>{...}-Argument*

Der Unterschied zwischen den drei Optionen für Absätze `p` (die Absatzbox ist nach oben ausgerichtet), `m` (die Absatzbox ist mittig ausgerichtet) und `b` (die Absatzbox ist nach unten ausgerichtet) wird anhand der nächsten Beispiele schematisch dargestellt.

Bsp.
5-2-3

1 1 1 1	2 2 2 2	3 3 3 3
1 1 1 1	2 2 2 2	
1 1 1 1		

```
\usepackage{array}
\begin{tabular}{|p{1cm}|p{1cm}|p{1cm}|}
\hline 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabular}
```

Bsp.
5-2-4

1 1 1 1	2 2 2 2	3 3 3 3
1 1 1 1	2 2 2 2	
1 1 1 1		

```
\usepackage{array}
\begin{tabular}{|m{1cm}|m{1cm}|m{1cm}|}
\hline 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabular}
```

Bsp.
5-2-5

1 1 1 1		
1 1 1 1	2 2 2 2	
1 1 1 1	2 2 2 2	3 3 3 3

```
\usepackage{array}
\begin{tabular}{|b{1cm}|b{1cm}|b{1cm}|}
\hline 1 1 1 1 1 1 1 1 1 1 1 &
2 2 2 2 2 2 2 2 & 3 3 3 3 \\ \hline
\end{tabular}
```

In Spalten, die mit `p`, `m` oder `b` erzeugt wurden, beträgt der voreingestellte Wert für `\parindent` `Opt`. Er kann mithilfe des Befehls `\setlength` geändert

werden. Im nächsten Beispiel wird so die erste Spalte um 5mm eingerückt.

1 2 3 4 5 6	1 2 3 4 5 6 7 8
7 8 9 0 1 2 3 4	9 0 1 2 3 4 5 6
5 6 7 8 9 0	7 8 9 0

```
\usepackage{array}
\begin{tabular}
  {>{\setlength\parindent{5mm}}p{2cm}|p{2cm}|}
\hline 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 &
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 \\ \hline
\end{tabular}
```

Bsp.
5-2-6

Die Spaltenoption `<` wurde ursprünglich für folgenden Zweck entwickelt: `>{\$}c<{\$}` erzeugt in einer `tabular`-Umgebung eine Spalte im mathematischen Modus. In einer `array`-Umgebung hingegen erzeugt diese Art von Präambel eine Spalte im LR-Modus, da die zusätzlichen `$`-Zeichen die existierenden aufheben.

$10^{10!}$	eine große Zahl
10^{-999}	eine kleine Zahl

```
\usepackage{array}
\setlength\extrarowheight{4pt}
\begin{tabular}{>{\$}l<{\$}|l} \hline
10^{10!} & eine große Zahl \\
10^{-999} & eine kleine Zahl \\ \hline
\end{tabular}
```

Bsp.
5-2-7

Die Kürzel `!` und `@` werden oft benötigt, um mit dem `\extracolsep`-Befehl elastische Längen hinzuzufügen, so dass $\TeX$ die Tabelle in der `tabular`-Umgebung auf die gewünschte Breite dehnen kann. Im `array`-Paket unterliegt der Einsatz von `\extracolsep` dabei zwei Einschränkungen: Je `@`- oder `!`-Kürzel kann es nur höchstens einen `\extracolsep`-Befehl geben, und dieser muss direkt im Argument verwendet werden. Er darf also nicht in einer Befehlsdefinition versteckt sein. Dementsprechend funktioniert der Befehl `\newcommand\ef{\extracolsep{\fill}} ... @{\ef}` nicht in einer `tabular`-Präambel. Stattdessen kann man aber ein neues Kürzel, etwa `\newcolumntype{e}{@{\extracolsep{\fill}}}`, verwenden.

Setzen schmaler Spalten

$\TeX$ trennt das erste Wort in einem Absatz normalerweise nicht, so dass sehr schmale Zellen leicht übertoll werden können. Hier lässt sich Abhilfe schaffen, indem man dem Text den Befehl `\hspace{0pt}` voranstellt.

Charakteristika

Charakteristika

```
\fbox{\parbox{11mm}{Charakteristika}}%
\hfill
\fbox{\parbox{11mm}{\hspace{0pt}Charakteristika}}
```

Bsp.
5-2-8

Bei der Arbeit mit schmalen Spalten muss man nicht nur die Trennung des ersten Wortes ermöglichen, sondern auch berücksichtigen, dass kurze Texte besser im Flattersatz gesetzt werden (nicht bündig am rechten Rand). Dieser Effekt lässt sich erzielen, indem man dem Tabelleninhalt den Befehl `\raggedright` voranstellt (siehe Abschnitt 3.1.11). Da dieser Befehl allerdings den Zeilenumbruchbefehl `\\` undefiniert, muss für Zeilenumbrüche dann `\tabularnewline` verwendet werden. Genau wie in Standard- $\LaTeX$, entspricht dieser Befehl im `array`-Paket dem eigentlichen Zeilenende-

befehl `\` der Umgebungen `tabular` oder `array`. Alternativ könnte man nach `\raggedright` in der letzten Spaltendefinition zusätzlich den Befehl `\arraybackslash` verwenden. Er definiert `\` lokal so um, dass er die Tabellenzeile beendet (vgl. Beispiel 5-2-12 auf Seite 257).

Wie aus dem nächsten Beispiel deutlich wird, lässt sich das Tabellenmaterial in einer `tabular`-Umgebung dann linksbündig, rechtsbündig oder zentriert setzen, ohne dass man die Kontrolle über die Zeilenumbrüche verliert. Außerdem kann das erste Wort nun automatisch getrennt werden, wenn $\TeX$ auch im Falle des französischen und niederländischen Eintrags durch Angabe der möglichen Trennpositionen ein wenig nachgeholfen wurde.

Bsp.
5-2-9

Donau- dampf- schiff ist ein langes Wort	Possibili- tés et espéran- ces	Moge- lijkheden en hoop
Flutter- satz links in Spalte eins	Zentrier- ter Text in Spalte zwei	Flutter- satz rechts in Spalte drei

```
\usepackage{array}
\begin{tabular}%
  {|>{\raggedleft\hspace{0pt}}p{14mm}}%
  |>{\centering\hspace{0pt}}p{14mm}}%
  |>{\raggedright\hspace{0pt}}p{14mm}|}
\hline
  Donaudampfschiff ist ein langes Wort    &
  Possibi\li\t'es et esp'erances    &
  Moge\lijk\heden en hoop    \tabularnewline
\hline
  Flattersatz \ \ links \ \ in Spalte eins    &
  Zentrierter \ \ Text in Spalte zwei    &
  Flattersatz rechts \ \ in Spalte drei
                                     \tabularnewline
\hline
\end{tabular}
```

Einstellen von Spaltenabständen

Der Standardabstand zwischen Spalten wird durch die Parametereinstellungen für `\arraycolsep` (in `array`) und `\tabcolsep` (in `tabular`) gesteuert. Oft ist es jedoch wünschenswert, den Abstand zwischen einzelnen Spalten beziehungsweise vor der ersten und nach der letzten Spalte einer Tabelle zu ändern.

Bsp.
5-2-10

einszwei	drei	vier	–	fünf
1	2	3	– 4	– 5

```
\usepackage{array}
\begin{tabular}{c@{}c!{}c@{--}c!{--}c}
  eins&zwei&drei&vier&fünf\ \\
  1&2&3&4&5
\end{tabular}
```

Im vorigen Beispiel wurde `@{}` eingesetzt, um den Abstand zwischen den Spalten 1 und 2 zu entfernen. Eine leere Anweisung `!{}` hat keinerlei Auswirkung, wie man zwischen den Spalten 2 und 3 erkennen kann. Zwischen den Spalten 3 und 4 erscheint anstelle des Spaltenzwischenraums ein Gedankenstrich, der mit `@{--}` festgelegt wurde; zwischen den Spalten 4 und 5 wurde er mit `!{--}` definiert und erscheint dadurch in der Mitte des Spaltenzwischenraums.

Mit @{} Leerräume
neben Tabellen
entfernen

Mithilfe von @{} wird auch häufig der Leerraum der Breite \tabcolsep (in tabular) entfernt, der gemäß Voreinstellung an beiden Seiten der Tabelle erscheint. Dies wird im folgenden Beispiel deutlich.

Text Text Text Text
eins zwei
drei vier **weiteres Material ...**
Text Text Text Text
Text Text Text Text
eins zwei
drei vier **weiteres Material ...**
Text Text Text Text

```
\begin{flushleft} \textbf{Text Text Text Text}\\
\begin{tabular}{lr}
eins&zwei\\ drei&vier\\
\end{tabular}\textbf{weiteres Material \ldots}\\
\textbf{Text Text Text Text}\\
\begin{tabular}{@{}lr@{}}
eins&zwei\\ drei&vier\\
\end{tabular}\textbf{weiteres Material \ldots}\\
\textbf{Text Text Text Text} \end{flushleft}
```

Bsp.
5-2-11

5.2.2 Definieren neuer Spaltenformatkürzel

Um in einer Tabelle eine Spalte speziell zu formatieren, kann man ihr Layout mit > und < verändern:

>{Deklarationen}c<{weitere Deklarationen}

Dieses Verfahren ist jedoch ziemlich umständlich, wenn man solche Spalten häufiger verwendet. In solchen Fällen lässt sich mit dem folgenden Befehl ein neues Kürzel für das Spaltenformat festlegen:

```
\newcolumnntype{kürzel}[narg]{dekl}
```

Hierbei ist *kürzel* ein einzelner Buchstabe, der als Spezifikation für den Spaltentyp in der Präambel verwendet werden kann; *narg* ist ein optionales Argument, dessen Wert die Anzahl der Argumente für dieses Kürzel angibt und *dekl* sind zulässige Deklarationen, wie z.B.:

```
\newcolumnntype{x}{>{\bfseries\small}c}
```

Das neu definierte Spaltenformatkürzel x kann dann überall dort in den Präambelargumenten der array- und tabular-Umgebungen verwendet werden, wo diese Spaltenform erforderlich ist.

In einer tabular- oder array-Umgebung benötigt man recht häufig sowohl Spalten im mathematischen als auch im LR-Modus. Für diesen Fall eignen sich folgende Definitionen (Vorsicht, bei gleichzeitiger Verwendung des tabulary-Paketes kommt es zu Namenskonflikten mit dessen Kürzeln):

```
\newcolumnntype{C}{>{$}c<{$}}
\newcolumnntype{L}{>{$}l<{$}}
\newcolumnntype{R}{>{$}r<{$}}
```

Nach diesen Definitionen erzeugt z.B. C in der array-Umgebung eine zentrierte Spalte im LR-Modus beziehungsweise im mathematischen Modus in der tabular-Umgebung.

Der Befehl `\newcolumnntype` besitzt das gleiche erste optionale Argument wie `\newcommand`, das angibt, wie viele Argumente das neue Spaltenformatkürzel erhält. Anders als bei `\newcommand` besitzt er aber kein weiteres optionales Argument; d.h. in der derzeitigen Version können Spaltenformatkürzel nur obligatorische Argumente besitzen.

Bsp.
5-2-12

Donau- dampf- schiff ist ein langes Wort	Possibili- tés et espéran- ces	Moge- likheden en hoop
Flutter- satz links in Spalte eins	Zentrier- ter Text in Spalte zwei	Flutter- satz rechts in Spalte drei

```
\usepackage{array}
\newcolumnntype{P}[1]
  >{\#1\hspace{0pt}\arraybackslash}p{14mm}{}
\begin{tabular}
  {|P{\raggedleft}P{\centering}P{\raggedright}}
\hline
  Donaudampfschiff ist ein langes Wort      &
  Possibi\li\t'es et esp'erances            &
  Moge\li\k\heden en hoop                    \\ \hline
  Flattersatz links in Spalte eins            &
  Zentrierter Text in Spalte zwei             &
  Flattersatz rechts in Spalte drei           \\ \hline
\end{tabular}
```

Eine ganz andere Einsatzmöglichkeit von `\newcolumnntype` macht sich die Tatsache zunutze, dass sich der Ersetzungstext dieses Befehls auf mehr als eine Spalte beziehen kann. Das folgende Beispiel zeigt die Definition einer Spaltenoption Z. Wenn man ihre Definition in der Dokumentenpräambel ändert, würde sich das Layout aller Tabellen des Dokumentes ändern, in denen dieses Formatkürzel konsequent eingesetzt wird.

Bsp.
5-2-13

eins	zwei	drei
1	2	3

```
\usepackage{array} \newcolumnntype{Z}{clr}
\begin{tabular}{Z} eins&zwei&drei\1&2&3 \end{tabular}
```

Der Ersetzungstext in einem `\newcolumnntype`-Befehl kann jedes Basis- kürzel des `array`-Paketes enthalten sowie jeden neuen Buchstaben, der mit einem `\newcolumnntype`-Befehl definiert wurde.

Jede Spaltenformatierung in einer `tabular`-Umgebung, die auf einer sol- chen Definition für Spaltentypen basiert, wird im ersten Verarbeitungsschritt der Tabelle auf ihre Basisbefehle „erweitert“. Dadurch kommt es unter Um- ständen dazu, dass sich Fehlermeldungen, die bei der Verarbeitung der Spal- tenformatierung auftreten, auf das Präambelargument beziehen, das bereits vom `\newcolumnntype`-System *umgeschrieben* wurde, und nicht auf die vom Anwender verfasste Präambel.

*Fehlersuche bei
Deklarationen für
Spaltentypen*

Mithilfe des Befehls `\showcols` in der Präambel kann man sich am Bild- schirm eine Liste aller zurzeit aktiven `\newcolumnntype`-Definitionen anzei- gen lassen.

5.3 Errechnen von Spaltenbreiten

Wie in Anhang A.2 beschrieben, verfügt $\LaTeX$ über zwei verschiedene Modi zum Setzen von Text: Im LR-Modus wird der Text in eine einzige Zeile ge- schrieben, während er im Absatzmodus in Zeilen einer bestimmten Länge um- brochen wird. Diese Unterscheidung hat großen Einfluss auf die Gestaltung

der Tabellenbefehle in $\text{\LaTeX}$. Spaltentypen der Arten l, c und r nutzen den LR-Modus, während p und die Typen m und b des array-Paketes Tabelleneinträge im Absatzmodus erzeugen.

Im Absatzmodus muss die Zeilenbreite vorab festgelegt werden, was sich beim Setzen der Tabellen manchmal als schwierig erweisen kann. Im vorliegenden Abschnitt werden mehrere Ansätze beschrieben, wie sich die erforderlichen Spaltenbreiten auf Grundlage der benötigten Tabellenbreite und/oder ihrer Inhalte berechnen lassen.

5.3.1 Explizites Berechnen der Spaltenbreiten

Die im folgenden Beispiel definierte Umgebung `tabularc` kann eine Tabelle der Breite `\linewidth` mit einer vorgegebenen Anzahl gleich breiter Spalten erzeugen. Dazu bedient sie sich des Paketes `calc`, das in Anhang A.3.1 besprochen wird. Außerdem verwendet sie den Befehl `\tabularnewline`, der in Abschnitt 5.2.1 erwähnt wird. Die Anzahl der Spalten wird der Umgebung in ihrem Argument übergeben. Diese Anzahl, hier x genannt, wird verwendet, um die tatsächliche Breite jeder Spalte zu errechnen. Dazu wird zwei mal x mal der Spaltenabstand und $(x + 1)$ mal die Linienstärke von der Zeilenlänge abgezogen. Die verbleibende Länge wird durch x geteilt, um die Breite einer einzelnen Spalte zu erhalten. Der Spalteninhalt wird zentriert und die Trennung des ersten Wortes wird ermöglicht.

```
\usepackage{array,calc}    \newlength\mylen
\newenvironment{tabularc}[1]
  {\setlength\mylen
    {\linewidth/(#1)-\tabcolsep*2-\arrayrulewidth*(#1+1)/(#1)}%
   \par\noindent           % neuer Absatz, Anfang linksbündig
   \begin{tabular*}{\linewidth}%
     {*{#1}{|>\centering\hspace{0pt}}p{\the\mylen}}|}
  {\end{tabular*}\par}
\begin{tabularc}{3}
\hline
Inhalt von Spalte eins & Spalte zwei & Das ist Spalte drei
\tabularnewline\hline
... weitere Eingaben unterdrückt ...
```

Bsp.
5-3-1

Inhalt von Spalte eins	Spalte zwei	Das ist Spalte drei
Erneut Spalte eins	und Spalte zwei	Das ist Spalte drei
Noch mal Spalte eins	Spalte zwei	Letzte Spalte drei

Wenn man die Spaltenbreite auf diese Weise berechnet, kann man genau bestimmen, wie viel Platz jeder Spalte zugewiesen wird. Leider ist es schwierig, Informationen über Tabelleninhalte in die Berechnung einzubeziehen. Wenn z.B. einige Tabellenspalten zum Typ c gehören und so in ihrer natürlichen Breite gesetzt werden, möchte man den verbleibenden Platz für die anderen Spalten vielleicht im Absatzmodus zuweisen. Da die Breite solcher Spalten jedoch erst nach dem Setzen der Tabelle bekannt ist, kann man die übrigen nicht bereits im Voraus berechnen. Die zwei nachfolgenden Pakete verwenden unterschiedliche Algorithmen, um bestimmten Spalten geeignete Breiten

zuzuweisen. Das erste, `tabularx`, versucht im Wesentlichen den Raum gleichmäßig zwischen den Spalten aufzuteilen, die im Absatzmodus gesetzt werden. Das zweite, `tabulary`, versucht Spalten mit „mehr Inhalt“ mehr Platz zuzuweisen. Bei beiden Algorithmen wird die Tabelle mehrfach gesetzt, bis ein korrektes Ergebnis vorliegt.

5.3.2 `tabularx` – Spaltenbreiten automatisch berechnen

Das Paket `tabularx` (von David Carlisle) stellt die gleichnamige Umgebung zur Verfügung, die eine besondere Variante der `tabular*`-Umgebung darstellt. In dieser wird die Breite bestimmter Spalten automatisch aus der Gesamtbreite der Tabelle errechnet. Die automatisch berechneten Spalten werden in der Präambel mit dem Kürzel `X` angegeben. Das Kürzel wird später, sobald die korrekte Breite errechnet ist, zu `p{breite}` umgesetzt.

Bsp.
5-3-2

```
\usepackage{tabularx}
\newcolumnntype{Y}{>{\small\raggedright\arraybackslash}X}
\noindent\begin{tabularx}{100mm}{|Y|Y|Y|}
... weitere Eingaben unterdrückt ...
```

The Two Gentlemen of Verona	The Taming of the Shrew	The Comedy of Errors
Love's Labour's Lost	A Midsummer Night's Dream	The Merchant of Venice
The Merry Wives of Windsor	Much Ado About Nothing	As You Like It
Twelfth Night	Troilus and Cressida	Measure for Measure
All's Well That Ends Well	Pericles Prince of Tyre	The Winter's Tale
Cymbeline	The Tempest	

Wenn man `\linewidth` als Wert für das Argument *breite* einsetzt, ergibt sich folgendes Tabellenlayout:

Bsp.
5-3-3

```
\usepackage{tabularx}
\newcolumnntype{Y}{>{\small\raggedright\arraybackslash}X}
\noindent\begin{tabularx}{\linewidth}{|Y|Y|Y|}
... weitere Eingaben unterdrückt ...
```

The Two Gentlemen of Verona	The Taming of the Shrew	The Comedy of Errors
Love's Labour's Lost	A Midsummer Night's Dream	The Merchant of Venice
The Merry Wives of Windsor	Much Ado About Nothing	As You Like It
Twelfth Night	Troilus and Cressida	Measure for Measure
All's Well That Ends Well	Pericles Prince of Tyre	The Winter's Tale
Cymbeline	The Tempest	

Befehle zum Setzen von X-Spalten

In der Standardeinstellung des Paketes wird das Kürzel `X` in `p{berechnetebreite}` umgesetzt. Schmale Spalten erfordern dabei häufig ein besonderes Format, das sich mithilfe der `>`-Syntax erzielen lässt, z.B. mit einer Spezifikation wie `>{\small}X`.

Ein weiteres sehr nützliches Format zum Setzen schmaler Spalten ist der links bündig ausgerichtete Flattersatz (ragged right). Wie bereits erwähnt, muss man eine Tabellenzeile mit `\tabularnewline` beenden, wenn der letzte Eintrag der Zeile im Flattersatz gesetzt ist. Diese Befehlsfolge lässt sich mit `\newcolumnntype{Y}{>{\small\raggedright}X}` speichern. Man kann auch noch den Befehl `\arraybackslash` hinzufügen, damit die Tabellenzeilen wieder durch `\\` beendet werden. Danach kann man `Y` als Präambelargument für `\tabularx` verwenden.

Die `X`-Spalten werden also mithilfe einer `p`-Spalte gesetzt, die ihrerseits `\parbox[t]` entspricht. Womöglich möchte man die Spalten aber nun im `m`-Format setzen, das `\parbox[c]` entspricht. Da sich der Spaltentyp jedoch nicht mit der `>`-Syntax ändern lässt, kommt hier ein anderes Verfahren zum Einsatz. Der Befehl `\tabularxcolumn` kann als Makro mit einem Argument definiert werden, welches dann zu dem `\tabular`-Präambelbefehl umgesetzt wird, der in Zukunft für `X` verwendet werden soll. Beim Ausführen des Befehls wird als Argument die tatsächliche Spaltenbreite übergeben.

Die Paketdefinition lautet `\newcommand\tabularxcolumn[1]{p{#1}}`. Eine mögliche Alternative wäre

```
\renewcommand\tabularxcolumn[1]{>{\small}m{#1}}
```

Spaltenbreiten

Normalerweise haben alle `X`-Spalten einer Tabelle die gleiche Breite. Es ist allerdings auch möglich, ihnen mit `\tabularx` unterschiedliche Breiten zuzuweisen. Eine Präambel wie

```
>{\setlength\hsize{.5\hsize}}X>{\setlength\hsize{1.5\hsize}}X}
```

legt zwei Spalten fest, wobei die zweite Spalte dreimal so breit ist wie die erste. Wenn man diese Methode verwendet, muss man allerdings zwei Regeln beachten:

- Die Summe der Breiten aller `X`-Spalten darf sich nicht ändern. Im obigen Beispiel müssen die neuen Werte deshalb zusammen die Breite von zwei `X`-Standardspalten ergeben.
- Es können keine `\multicolumn`-Einträge verwendet werden, die sich mit `X`-Spalten überschneiden.

```
\usepackage{tabularx} \tracingtabularx
\noindent
\begin{tabularx}{\linewidth}%
  {|>{\setlength\hsize{.85\hsize}}X|}%
  >{\setlength\hsize{1.15\hsize}}X|}
  Donaudampfschiff ist ein langes Wort &
  Moge\~{l}ijk\~{h}eden en hoop \\
  Etwas Text in Spalte eins &
  Ein etwas längerer Text in Spalte zwei \\
\end{tabularx}
```

Bsp.
5-3-4

Wenn eine `\tracingtabularx`-Deklaration z.B. in der Dokumentenpräambel steht, dann schreiben alle nachfolgenden `tabularx`-Umgebungen Informationen über Spaltenbreiten auf den Bildschirm und in die Protokolldatei. Diese Daten entstehen dadurch, dass die Tabellen auf der Suche nach den optimalen Breiten immer wieder neu formatiert werden. So erzeugte das letzte Beispiel folgende Protokolldaten:

tabularx-
Berechnungen
protokollieren

Package tabularx Warning: Target width: \linewidth = 207.0pt..

```
(tabularx)      Table Width      Column Width      X Columns
(tabularx)      439.19998pt      207.0pt           3
(tabularx)      206.99998pt      90.90001pt       2
(tabularx) Reached target.
```

5.3.3 tabulary – Am Inhalt orientierte Spaltenbreiten

Ein alternativer Algorithmus für Spaltenbreiten findet sich im Paket `tabulary` (ebenfalls von David Carlisle), das die `tabulary`-Umgebung definiert. Es eignet sich sehr gut, wenn die Breite der Spalten in Abhängigkeit vom Inhalt der Tabelle berechnet werden muss. Das ist häufig der Fall, wenn man $\text{\LaTeX}$ benutzt, um SGML/XML- oder HTML-Dokumente zu formatieren. In diesen Sprachen wird normalerweise ein anderes Tabellensystem verwendet, in dem mehrzeilige Einträge keine feste Breite haben und das Layout mehr dem externen Formatierungsprogramm überlassen bleibt.

Das `tabulary`-Paket stellt folgende Kürzel zur Verfügung: die eigenen Formatkürzel wie beschrieben in Tabelle 5.3 auf der nächsten Seite, diejenigen des Paketes `array` aus Tabelle 5.2 auf Seite 252 sowie jedes andere Kürzel, das mit `\newcolumntype` definiert wurde.

```
\begin{tabulary}{breite}[pos]{spalten} zeilen \end{tabulary}
```

Die Hauptfunktion dieses Paketes besteht darin, verschiedene Versionen der Spaltenoption `p` bereitzustellen, in denen sich die Spaltenbreite automatisch am Tabelleninhalt orientiert.

Das folgende Beispiel ist nicht sehr realistisch, da die Tabelle nur eine Zeile umfasst. Es zeigt jedoch, dass `tabulary` darauf abzielt, die Spaltenbreite zugunsten einer gleichmäßigen Zeilenhöhe einzustellen. Normalerweise werden die umfangreichsten Einträge jeder Spalte natürlich nicht alle in der gleichen Zeile stehen, aber in vielen Tabellen verfügen die Einträge aller Zellen einer Spalte über ähnliche Charakteristika. Aus diesem Grund liefert der verwendete Algorithmus in den meisten Fällen vernünftige Ergebnisse.

[illegible]

Bsp.
5-3-5

J	Blocksatz-p-Spalte mit noch zu bestimmender Breite
L	Linksbündige p-Spalte mit noch zu bestimmender Breite
R	Rechtsbündige p-Spalte mit noch zu bestimmender Breite
C	Zentrierte p-Spalte mit noch zu bestimmender Breite

Tabelle 5.3: Zusätzliche Formatkürzel des tabulary-Paketes

Zuordnung der Spaltenbreite steuern

Das Paket `tabulary` verfügt über zwei Längenparameter, `\tymin` und `\tymax`, mit deren Hilfe die Zuordnung der Spaltenbreite gesteuert wird. Normalerweise werden die Breiten den L-, C-, R- oder J-Spalten proportional zur natürlichen Breite ihres längsten Eintrags zugewiesen. Um diese Breite festzustellen, setzt `tabulary` die Tabelle zweimal. Im ersten Durchgang wird der Text der L-, C-, R- und J-Spalten im LR-Modus gesetzt, ähnlich wie die Daten in Spalten mit Standardformatkürzeln wie z.B. `c`. Die Absätze in diesen Spalten werden einzeilig gesetzt und anschließend wird die Länge dieser Zeile gemessen. Danach wird die Tabelle ein zweites Mal gesetzt, diesmal in endgültiger Form. Dabei werden die Spalten wie p-Spalten behandelt, deren relative Breite proportional zu der im ersten Lauf bestimmten „natürlichen“ Länge ist. Damit sehr schmale Spalten durch diesen Vorgang nicht zu sehr „gequetscht“ erscheinen, werden alle Spalten mit einer Breite kleiner `\tymin` in ihrer natürlichen Breite gesetzt. Diese Größe lässt sich über `\setlength` einstellen und wird einfach mit `10pt` initialisiert. Ist abzusehen, dass eine Spalte ziemlich schmal sein wird, kann es günstiger sein, diese z.B. mit `c` anstatt mit `C` zu formatieren, so dass sie den `tabulary`-Mechanismus gar nicht erst durchläuft und ohnehin in ihrer natürlichen Breite gesetzt wird.

Ebenso kann ein sehr breiter Eintrag zu einer übertrieben breiten Spalte führen. Um diesem Problem vorzubeugen, wird für alle Spalten, deren natürliche Breite (im LR-Modus gemessen) `\tymax` überschreitet, dieser Wert in der weiteren Berechnung verwendet. Damit wird für diese Spalten immer dieselbe Breite eingestellt. Der Begrenzungswert `\tymax` für die natürliche Breite wird mit der doppelten Textbreite initialisiert.

Die Tabelle im vorigen Beispiel wird durch den langen Eintrag in der vierten Spalte dominiert. Durch einen `\tymin`-Wert von `30pt` lässt sich vermeiden, dass die ersten beiden Spalten zu schmal werden, und mit einem `\tymax`-Wert von `200pt` wird die Breite der vierten Spalte begrenzt. Daraus ergibt sich insgesamt eine gleichmäßigere Raumaufteilung innerhalb der Tabelle.

a	b b b b	c c c c c c c c c c c c c c c c	d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d d	<code>\usepackage{tabulary}</code> <code>\setlength\tymin{30pt}</code> <code>\setlength\tymax{200pt}</code> <code>\begin{tabulary}{200pt}{ C C C C }</code> <code>... weitere Eingaben unterdrückt ...</code>

Bsp.
5-3-6

Schmale p-Spalten können zu einer echten Herausforderung werden. Hier empfiehlt es sich, den Befehl `\tyformat` so umzudefinieren, dass er jeglicher Deklaration entsprechen kann, die unmittelbar auf `\centering` oder

`\ragged...` folgt. Normalerweise verändert `\tyformat \everypar` dahingehend, dass am Anfang jedes Absatzes ein Leerraum ohne vertikale Ausdehnung erscheint, so dass das erste Wort getrennt werden kann (siehe Abschnitt 5.2.1 auf Seite 254).

Genau wie `tabularx` unterstützt auch `tabulary` das optionale Ausrichtungsgesamtargument der `tabular`-Umgebung. Da die gesamte Umgebung zweimal gespeichert und ausgewertet wird, sollte man besonders auf L^AT_EX-Konstrukte mit Seiteneffekten achten, da diese dann mehrfach ausgeführt werden (z.B. Befehle, die in Dateien schreiben).

5.3.4 `tabular*`, `tabularx` und `tabulary` – Ein Vergleich

Alle drei Umgebungen besitzen die gleichen Argumente und dienen dazu, eine Tabelle mit einer vorgegebenen Breite zu erzeugen. Die wichtigsten Unterschiede zwischen ihnen sind:

- `tabularx` und `tabulary` verändern die Breite der *Spalten*, während `tabular*` den *Spaltenabstand* verändert.
- `tabular`- und `tabular*`-Umgebungen können ohne Einschränkung verschachtelt werden. Wenn aber eine `tabularx`- oder `tabulary`-Umgebung innerhalb einer anderen vorkommt, *muss* die innere in `{ }`-Klammern eingeschlossen sein.
- Die Inhalte von `tabularx`- und `tabulary`-Umgebungen sind intern eigentlich Argumente von Befehlen, so dass gewisse Einschränkungen gelten. Man kann zwar die Befehle `\verb` und `\verb*` verwenden, jedoch können Leerzeichen falsch verarbeitet werden und die Argumente dürfen keinen `%`-Befehl oder eine allein stehende `{`- oder `}`-Klammer enthalten.
- Die Umgebung `tabular*` verwendet eine T_EX-Basisfunktion zum Verändern der Spaltenabstände einer Ausrichtung. Dagegen muss eine `tabularx`-Tabelle wiederholt formatiert werden, bis die optimale Spaltenbreite gefunden ist – die Umgebung ist also erheblich langsamer. `tabulary` setzt Tabellen immer zweimal. Bei diesen beiden Umgebungen kann die wiederholte Verarbeitung des Tabelleninhalts dazu führen, dass bestimmte T_EX-Konstrukte beschädigt werden. Ein besonderes Problem sind Befehle, die in externe Dateien schreiben, da die entsprechenden Daten während der wiederholten Tabellenformatierung mehrmals geschrieben werden könnten.
- `tabularx` versucht den verfügbaren Platz gleichmäßig unter allen X-Spalten zu verteilen, um die gewünschte Breite zu erhalten; `tabulary` hingegen versucht Spalten mit längeren Einträgen mehr Raum zu geben.

\verb wird nur teilweise unterstützt

5.4 Mehrseitige Tabellen

Die `tabular`-Umgebung sowie ihre bisher betrachteten Varianten müssen immer auf eine Seite passen. Wenn die Tabelle zu lang wird, überschreibt der Text den unteren Seitenrand und man erhält die Fehlermeldung: `Overfull \vbox`.

Für mehrseitige Tabellen stehen zwei Pakete zur Verfügung: `supertabular` und `longtable`. Ihr Funktionsumfang ist ähnlich, jedoch verwenden sie eine unterschiedliche Syntax. Das Paket `longtable` setzt mit der $\TeX$ -Ausgaberoutine auf einen komplizierteren Mechanismus auf, um über alle Seiten hinweg optimale Seitenumbrüche und gleich bleibende Spaltenbreiten zu erhalten. Dadurch kann ein Dokument mehrere Durchläufe benötigen, bis die optimalen Spaltenbreiten errechnet sind. Im Gegensatz dazu teilt das `supertabular`-Paket eine Tabelle im Wesentlichen in eine Reihe von einseitigen `tabular`-Umgebungen auf und jede Seite wird dann einzeln gesetzt. Dieses Verfahren verzichtet auf mehrere Durchläufe und lässt sich vielfältiger einsetzen. Es funktioniert z.B. im Zwei- oder Mehrspaltensatz, welcher vom `longtable`-Paket nicht unterstützt wird.

5.4.1 `supertabular` – Erstellen mehrseitiger Tabellen

<code>\begin{supertabular}{spalten-dekl}</code>	<i>zeilen</i>	<code>\end{supertabular}</code>
<code>\begin{supertabular*}{breite}{spalten-dekl}</code>	<i>zeilen</i>	<code>\end{supertabular*}</code>
<code>\begin{mpsupertabular}{spalten-dekl}</code>	<i>zeilen</i>	<code>\end{mpsupertabular}</code>
<code>\begin{mpsupertabular*}{breite}{spalten-dekl}</code>	<i>zeilen</i>	<code>\end{mpsupertabular*}</code>

Das Paket `supertabular` (ursprünglich von Theo Jurriens, überarbeitet von Johannes Braams) definiert die Umgebung `supertabular`. Intern verwendet es die Standardumgebung `tabular`, wobei es allerdings bei jedem `\\`-Befehl den verbrauchten Platz berechnet. Wenn die Tabellenhöhe den Wert von `\textheight` erreicht, fügt es automatisch einen `\end{tabular}`-Befehl ein und beginnt eine neue Seite mit einem Tabellenkopf, der die `tabular`-Umgebung fortsetzt. Dadurch kann die Breite der Spalten und damit auch der gesamten Tabelle im Verlauf der Seiten variieren.

Es gibt noch drei weitere Varianten der Umgebung. Die `supertabular*`-Umgebung verwendet intern `tabular*` und stellt die Tabellenbreite über ihr obligatorisches Argument *breite* ein. Die Umgebungen `mpsupertabular` und `mpsupertabular*` verwenden die gleiche Syntax wie `supertabular` bzw. `supertabular*`; sie setzen die einzelnen Tabellenseiten jedoch in eine `minipage`-Umgebung. Dadurch können innerhalb der Tabellen `\footnote`-Befehle verwendet werden, deren Fußnoten auch wirklich am Ende der betreffenden Seite erscheinen.

Innerhalb einer `supertabular`-Umgebung werden neue Tabellenzeilen wie bisher mit `\\`-Befehlen erzeugt. Alle Formatkürzel einschließlich `@{...}` und `p{...}` sind zulässig. Wenn das `array`-Paket zusammen mit `supertabular` geladen wird, können seine zusätzlichen Tabellenformatkürzel ebenfalls eingesetzt werden. Allerdings ist es nicht möglich, die optionalen Positionierungsparameter, wie `t` und `b`, zu verwenden, die mit `\begin{tabular}` und `\begin{tabular*}` festgelegt werden.

Für `supertabular` & Co. stehen eine Reihe neuer Befehle zur Verfügung, die im Folgenden beschrieben werden. Diese Befehle sollten vor den Umgebungen verwendet werden, da sie sich auf alle nachfolgenden `supertabular`-Umgebungen auswirken.

```
\tablehead{zeilen} \tablefirsthead{zeilen}
```

Das Argument von `\tablehead` sind die Tabellenzeilen, die am Kopf jeder Seite wiederholt werden. Wenn auch `\tablefirsthead` verwendet wird, besteht die erste Tabellenüberschrift aus dem Argument dieses Befehls, anstatt aus den `\tablehead`-Zeilen. Das Argument kann ganze Zeilen (mit `\\` am Ende) sowie Füllmaterial zwischen den Zeilen (wie `\hline`) enthalten.

```
\tabletail{zeilen} \tablelasttail{zeilen}
```

Diese Befehle geben an, welches Material am Ende jeder Tabellenseite eingefügt wird. Wenn zusätzlich `\tablelasttail` verwendet wird, besteht das Tabellenende aus dem Argument dieses Befehls anstatt aus den `\tabletail`-Zeilen.

```
\topcaption[verzeichnis]{legende}
\bottomcaption[verzeichnis]{legende}
\tablecaption[verzeichnis]{legende}
```

Mit diesen Befehlen kann man eine Legende für die `supertabular`-Umgebung festlegen, die am Kopf oder Fuß der Tabelle erscheint. Das optionale Argument dient dem gleichen Zweck wie das des Standardbefehls `\caption`: Es enthält die Version der Legende, die im Tabellenverzeichnis aufgeführt wird. Durch `\tablecaption` erscheint die Legende an ihrer Standardposition am Kopf der Tabelle. Diese Voreinstellung kann innerhalb eines Paketes oder einer Klasse mit der Deklaration `\@topcaptionfalse` geändert werden.

Das Format der Legende lässt sich mithilfe des Paketes `caption` anpassen (vgl. Beispiel 5-4-4 auf Seite 270).

```
\shrinkheight{länge}
```

Die `supertabular`-Umgebung schätzt fortwährend, wie viel Platz noch auf der laufenden Seite vorhanden ist. Mithilfe des Befehls `\shrinkheight`, der am Anfang einer Tabellenzeile stehen muss, lässt sich dieser Schätzwert reduzieren. Dadurch kann man beeinflussen, wo `supertabular` einen Seitenumbruch setzt.

Beispiel für die `supertabular`-Umgebung

```
\usepackage{supertabular}
\tablecaption{ISOGRK3 Zeichen-Entity-Referenzen}
\tablehead {\bfseries Entity&\bfseries Unicode-Name&\bfseries Unicode\\ \hline}
\tabletail {\hline \multicolumn{3}{r}{\emph{Weiter auf der nächsten Seite}}\\}
\tablelasttail{\hline}
\begin{supertabular}{lll}
alpha          & & GREEK SMALL LETTER ALPHA          & & 03B1\\
beta           & & GREEK SMALL LETTER BETA           & & 03B2\\
chi            & & GREEK SMALL LETTER CHI            & & 03C7\\
Delta          & & GREEK CAPITAL LETTER DELTA        & & 0394\\
... weitere Eingaben unterdrückt ...
```

Bsp.
5-4-1

Seite 1

Tabelle 1: ISOGRK3 Zeichen-Entity-Referenzen

Entity	Unicode-Name	Unicode
alpha	GREEK SMALL LETTER ALPHA	03B1
beta	GREEK SMALL LETTER BETA	03B2
chi	GREEK SMALL LETTER CHI	03C7
Delta	GREEK CAPITAL LETTER DELTA	0394
delta	GREEK SMALL LETTER DELTA	03B4
epsi	GREEK SMALL LETTER EPSILON	03B5
epsis	GREEK LUNATE EPSILON SYMBOL	03F5
epsiv	GREEK SMALL LETTER EPSILON	03B5
eta	GREEK SMALL LETTER ETA	03B7
Gamma	GREEK CAPITAL LETTER GAMMA	0393
gamma	GREEK SMALL LETTER GAMMA	03B3
gammad	GREEK SMALL LETTER DIGAMMA	03DD
iota	GREEK SMALL LETTER IOTA	03B9
kappa	GREEK SMALL LETTER KAPPA	03BA
kappav	GREEK KAPPA SYMBOL	03F0
Lambda	GREEK CAPITAL LETTER LAMDA	039B
lambda	GREEK SMALL LETTER LAMDA	03BB
mu	GREEK SMALL LETTER MU	03BC
nu	GREEK SMALL LETTER NU	03BD
Omega	GREEK CAPITAL LETTER OMEGA	03A9
omega	GREEK SMALL LETTER OMEGA	03C9

Weiter auf der nächsten Seite

Seite 1

Seite 2

Entity	Unicode-Name	Unicode
Phi	GREEK CAPITAL LETTER PHI	03A6
phis	GREEK PHI SYMBOL	03D5
phiv	GREEK SMALL LETTER PHI	03C6
Pi	GREEK CAPITAL LETTER PI	03A0
pi	GREEK SMALL LETTER PI	03C0
piv	GREEK PI SYMBOL	03D6
Psi	GREEK CAPITAL LETTER PSI	03A8
psi	GREEK SMALL LETTER PSI	03C8
rho	GREEK SMALL LETTER RHO	03C1
rhov	GREEK RHO SYMBOL	03F1
Sigma	GREEK CAPITAL LETTER SIGMA	03A3
sigma	GREEK SMALL LETTER SIGMA	03C3
sigmav	GREEK SMALL LETTER FINAL SIGMA	03C2
tau	GREEK SMALL LETTER TAU	03C4
Theta	GREEK CAPITAL LETTER THETA	0398
thetas	GREEK SMALL LETTER THETA	03B8
thetav	GREEK THETA SYMBOL	03D1
Upsi	GREEK UPSILON WITH HOOK SYMBOL	03D2
upsi	GREEK SMALL LETTER UPSILON	03C5
Xi	GREEK CAPITAL LETTER XI	039E
xi	GREEK SMALL LETTER XI	03BE
zeta	GREEK SMALL LETTER ZETA	03B6

Weiter auf der nächsten Seite

Seite 2

Beispiel für die supertabular*-Umgebung

Die Breite einer supertabular-Umgebung kann auf einen bestimmten Wert wie z.B. die Textbreite `\textwidth` festgelegt werden. Im folgenden Beispiel wurde, zusätzlich zum Einsatz von `supertabular*`, eine elastische Länge zwischen den letzten beiden Spalten eingefügt, so dass die Tabelle auf die angegebene Breite gedehnt werden kann. Wie bei `supertabular` üblich, wird jede Seite einzeln gesetzt. Das Beispiel zeigt, dass auf diese Weise zwischen Spalten auf der ersten (linken) und der zweiten (rechten) Seite unterschiedliche Abstände auftreten können.

```
\usepackage{array,supertabular}
\tablecaption{ISOGRK3 Zeichen-Entity-Referenzen}
\tablefirsthead
    {\bfseries Entity & \bfseries Unicode-Name
      & \bfseries Unicode          \\ \hline}
\tablehead
    {\bfseries Entity & \bfseries Unicode-Name
      & \bfseries Unicode          \\ \hline}
\tabletail{\hline
    \multicolumn{3}{r}{\emph{Weiter auf der nächsten Seite}}\\}
\tablelasttail{\hline}
\centering
\begin{supertabular*}{\textwidth}{11!{\extracolsep{\fill}}1}
alpha          & GREEK SMALL LETTER ALPHA          & 03B1\\
beta           & GREEK SMALL LETTER BETA           & 03B2\\
chi            & GREEK SMALL LETTER CHI            & 03C7\\
Delta          & GREEK CAPITAL LETTER DELTA        & 0394\\
delta          & GREEK SMALL LETTER DELTA          & 03B4\\
... weitere Eingaben unterdrückt ...
```

Bsp.
5-4-2

Seite 1

Tabelle 1: ISOGRK3 Zeichen-Entity-Referenzen

Entity	Unicode-Name	Unicode
alpha	GREEK SMALL LETTER ALPHA	03B1
beta	GREEK SMALL LETTER BETA	03B2
chi	GREEK SMALL LETTER CHI	03C7
Delta	GREEK CAPITAL LETTER DELTA	0394
delta	GREEK SMALL LETTER DELTA	03B4
epsi	GREEK SMALL LETTER EPSILON	03B5
epsis	GREEK LUNATE EPSILON SYMBOL	03F5
epsiv	GREEK SMALL LETTER EPSILON	03B5
eta	GREEK SMALL LETTER ETA	03B7
Gamma	GREEK CAPITAL LETTER GAMMA	0393
gamma	GREEK SMALL LETTER GAMMA	03B3
gammad	GREEK SMALL LETTER DIGAMMA	03DD
iota	GREEK SMALL LETTER IOTA	03B9
kappa	GREEK SMALL LETTER KAPPA	03BA
kappav	GREEK KAPPA SYMBOL	03F0
Lambda	GREEK CAPITAL LETTER LAMDA	039B
lambda	GREEK SMALL LETTER LAMDA	03BB
mu	GREEK SMALL LETTER MU	03BC
nu	GREEK SMALL LETTER NU	03BD
Omega	GREEK CAPITAL LETTER OMEGA	03A9

Weiter auf der nächsten Seite

Seite 1

Seite 2

Entity	Unicode-Name	Unicode
omega	GREEK SMALL LETTER OMEGA	03C9
Phi	GREEK CAPITAL LETTER PHI	03A6
phis	GREEK PHI SYMBOL	03D5
phiv	GREEK SMALL LETTER PHI	03C6
Pi	GREEK CAPITAL LETTER PI	03A0
pi	GREEK SMALL LETTER PI	03C0
piv	GREEK PI SYMBOL	03D6
Psi	GREEK CAPITAL LETTER PSI	03A8
psi	GREEK SMALL LETTER PSI	03C8
rho	GREEK SMALL LETTER RHO	03C1
rhov	GREEK RHO SYMBOL	03F1
Sigma	GREEK CAPITAL LETTER SIGMA	03A3
sigma	GREEK SMALL LETTER SIGMA	03C3
sigmav	GREEK SMALL LETTER FINAL SIGMA	03C2
tau	GREEK SMALL LETTER TAU	03C4
Theta	GREEK CAPITAL LETTER THETA	0398
thetas	GREEK SMALL LETTER THETA	03B8
thetav	GREEK THETA SYMBOL	03D1
Upsi	GREEK UPSILON WITH HOOK SYMBOL	03D2
upsi	GREEK SMALL LETTER UPSILON	03C5
Xi	GREEK CAPITAL LETTER XI	039E
xi	GREEK SMALL LETTER XI	03BE

Weiter auf der nächsten Seite

Seite 2

5.4.2 longtable – Alternative mehrseitige Tabellen

Wie bereits zu Anfang dieses Abschnitts erwähnt, sollte für komplexere mehrseitige Tabellen, bei denen man auf eine gleich bleibende Tabellenbreite für mehrere Seiten Wert legt, das Paket `longtable` in Betracht gezogen werden. Es stammt von David Carlisle mit Beiträgen von David Kastrup. Genau wie die `supertabular`-Umgebung hat es einige Gemeinsamkeiten mit der Umgebung `table`. Es verwendet insbesondere den gleichen Zähler (`table`) und einen ähnlichen `\caption`-Befehl. Der Befehl `\listoftables` erstellt ein kombiniertes Verzeichnis der `table`- und `longtable`-Umgebungen.

Ein wichtiger Unterschied zwischen den Umgebungen `supertabular` und `longtable` liegt darin, dass letztere die Breite jeder `longtable`-Umgebung in der Hilfsdatei mit der Erweiterung `.aux` speichert. Diese Werte werden dann im nächsten Lauf verwendet, um die größten benötigten Spaltenbreiten für die jeweiligen Tabellen zu finden. Aufgrund eben dieser Verwendung der `.aux`-Datei muss man sehr vorsichtig sein, wenn man `longtable` zusammen mit dem Befehl `\nofiles` benutzen möchte. Unter anderem unterdrückt `\nofiles` Einträge in eben diese Datei, so dass der Befehl erst eingesetzt werden darf, wenn die letzten Änderungen an der jeweiligen Tabelle abgeschlossen sind und das Paket bereits die optimalen Spaltenbreiten in die Hilfsdatei geschrieben hat.

Zum Vergleich der beiden Pakete wird Beispiel 5-4-1 von Seite 265 auf der nächsten Seite noch einmal wiederholt, diesmal jedoch unter Verwendung der `longtable`- und nicht der `supertabular`-Umgebung. Man sieht, dass die Tabelle auf beiden Seiten (der rechte und linke Teil der Abbildung) gleich breit ist. Man beachte, dass bei `longtable` der größte Teil der Formatierung innerhalb der Umgebung erfolgt, während bei `supertabular` der Tabellenkopf über Befehle festgelegt wird, die vor der Umgebung ausgeführt werden.

Verwendung der
.aux-Datei

```
\usepackage{longtable}
\begin{longtable}{lll}
  \caption{ISOGRK3 Zeichen-Entity-Referenzen}\\
  \bfseries Entity & \bfseries Unicode-Name
                    & \bfseries Unicode           \\ \hline
\endfirsthead
  \bfseries Entity & \bfseries Unicode-Name
                    & \bfseries Unicode           \\ \hline
\endhead
  \hline \multicolumn{3}{r}{\emph{Weiter auf der nächsten Seite}}
\endfoot
  \hline
\endlastfoot
alpha              & GREEK SMALL LETTER ALPHA          & 03B1\\
beta              & GREEK SMALL LETTER BETA           & 03B2\\
... weitere Eingaben unterdrückt ...
```

Bsp.
5-4-3

Seite 1

Tabelle 1: ISOGRK3 Zeichen-Entity-Referenzen

Entity	Unicode-Name	Unicode
alpha	GREEK SMALL LETTER ALPHA	03B1
beta	GREEK SMALL LETTER BETA	03B2
chi	GREEK SMALL LETTER CHI	03C7
Delta	GREEK CAPITAL LETTER DELTA	0394
delta	GREEK SMALL LETTER DELTA	03B4
epsi	GREEK SMALL LETTER EPSILON	03B5
epsis	GREEK LUNATE EPSILON SYMBOL	03F5
epsiv	GREEK SMALL LETTER EPSILON	03B5
eta	GREEK SMALL LETTER ETA	03B7
Gamma	GREEK CAPITAL LETTER GAMMA	0393
gamma	GREEK SMALL LETTER GAMMA	03B3
gammad	GREEK SMALL LETTER DIGAMMA	03DD
iota	GREEK SMALL LETTER IOTA	03B9
kappa	GREEK SMALL LETTER KAPPA	03BA
kappav	GREEK KAPPA SYMBOL	03F0
Lambda	GREEK CAPITAL LETTER LAMDA	039B
lambda	GREEK SMALL LETTER LAMDA	03BB
mu	GREEK SMALL LETTER MU	03BC
nu	GREEK SMALL LETTER NU	03BD
Omega	GREEK CAPITAL LETTER OMEGA	03A9
omega	GREEK SMALL LETTER OMEGA	03C9
Phi	GREEK CAPITAL LETTER PHI	03A6

Weiter auf der nächsten Seite

Seite 2

Entity	Unicode-Name	Unicode
phis	GREEK PHI SYMBOL	03D5
phiv	GREEK SMALL LETTER PHI	03C6
Pi	GREEK CAPITAL LETTER PI	03A0
pi	GREEK SMALL LETTER PI	03C0
piv	GREEK PI SYMBOL	03D6
Psi	GREEK CAPITAL LETTER PSI	03A8
psi	GREEK SMALL LETTER PSI	03C8
rho	GREEK SMALL LETTER RHO	03C1
rhov	GREEK RHO SYMBOL	03F1
Sigma	GREEK CAPITAL LETTER SIGMA	03A3
sigma	GREEK SMALL LETTER SIGMA	03C3
sigmav	GREEK SMALL LETTER FINAL SIGMA	03C2
tau	GREEK SMALL LETTER TAU	03C4
Theta	GREEK CAPITAL LETTER THETA	0398
thetas	GREEK SMALL LETTER THETA	03B8
thetav	GREEK THETA SYMBOL	03D1
Upsi	GREEK UPSILON WITH HOOK SYMBOL	03D2
upsi	GREEK SMALL LETTER UPSILON	03C5
Xi	GREEK CAPITAL LETTER XI	039E
xi	GREEK SMALL LETTER XI	03BE
zeta	GREEK SMALL LETTER ZETA	03B6

Seite 2

```
\begin{longtable}[ausricht]{spalten} zeilen \end{longtable}
```

Die Syntax der longtable-Umgebung basiert auf der tabular-Umgebung. Der Hauptunterschied liegt darin, dass das optionale Argument *ausricht* die Tabelle *horizontal* ausrichtet, und nicht vertikal, wie es bei tabular der Fall ist. Das Argument *ausricht* kann die Werte [c], [l] oder [r] annehmen, um eine zentrierte, linke bzw. rechte Ausrichtung der Tabelle zu bewirken. Wenn das optionale Argument entfällt, wird die Ausrichtung der Tabelle von den beiden Längenparametern \LTleft und \LTRight gesteuert. Ihr Standardwert ist \fill, so dass die Tabellen normalerweise zentriert werden.

Diese beiden Parameter können beliebige Längen aufnehmen. Zumindest einer von ihnen sollte jedoch eine elastische Länge enthalten, um die Breite

Horizontale
Ausrichtung

der Seite füllen zu können, es sei denn, dass mit dem `\extracolsep`-Befehl schon elastische Längen zwischen den Spalten eingefügt wurden. So kann eine Tabelle z.B. mit den folgenden Definitionen linksbündig gesetzt werden:

```
\setlength\LTleft{0pt} \setlength\LTRight{\fill}
```

oder einfach mit `\begin{longtable}[l]`. Mithilfe der Parameter `\LTleft` und `\LTRight` kann man auch eine mehrseitige Tabelle über die gesamte Breite einer Seite setzen. Das Beispiel 5-4-2 von Seite 266, in dem `supertabular*` verwendet wurde, kann auch mit den Paketen `array` und `longtable` sowie den folgenden Deklarationen gesetzt werden:

*Tabellenbreiten über
Parameter steuern*

```
\setlength\LTleft{0pt} \setlength\LTRight{0pt}  
\begin{longtable}{ll!{\extracolsep{\fill}}l}
```

Im Normalfall, wenn es sich bei `\LTleft` und `\LTRight` um feste Größen handelt, wird die Tabelle mit der Breite `\textwidth - \LTleft - \LTRight` gesetzt.

Vor und nach der Tabelle fügt `longtable` vertikale Abstände ein, die sich über die Parameter `\LTpre` und `\LTpost` einstellen lassen. Beide sind auf `\bigskipamount` voreingestellt und können mit `\setlength` geändert werden.

*Vertikale Abstände um
Tabellen*

Jede Tabellenzeile endet mit einem `\\`-Befehl. Genau wie in der Standardumgebung `tabular` ist auch hier der Befehl `\tabularnewline` verfügbar; er ist hilfreich, wenn `\\` durch einen Befehl wie `\raggedright` umdefiniert wurde. Die ebenfalls verwendbare Sternform `\\*` verhindert einen Seitenumbruch an dem jeweiligen Zeilenumbruch. Sie kann auch in einer `tabular`-Umgebung eingesetzt werden, wo sie jedoch den gleichen Effekt hat wie `\\`. Andererseits kann direkt im Anschluss an einen `\\`-Befehl ein `\newpage`-Befehl erfolgen, der an dieser Stelle einen Seitenumbruch erzwingt.

Zeilenbefehle

Wenn man eine Tabellenzeile mit `\kill` anstatt mit `\\` beendet, wird diese Zeile nicht gesetzt. Ihre Einträge werden lediglich zur Bestimmung der Spaltenbreiten herangezogen. Dieses Verfahren ähnelt dem Einsatz des `\kill`-Befehls in der `tabbing`-Umgebung.

Das `longtable`-Paket unterscheidet sich in der Syntax hauptsächlich dadurch von `supertabular`, dass die Tabellenzeilen, die auf jeder Tabellenseite als Kopf- oder Fußzeile erscheinen sollen, *innerhalb* der Umgebung festgelegt werden und nicht davor, wie bei `supertabular`. In Beispiel 5-4-3 auf der vorherigen Seite werden Tabellenkopf und -fuß dadurch bestimmt, dass man den letzten `\\`-Befehl durch einen der unten aufgeführten ersetzt. Dabei ist zu beachten, dass alle diese Befehle, auch diejenigen für den Tabellenfuß, am *Anfang* der Umgebung erfolgen müssen. Der Befehl `\endhead` beendet die Zeilen, die am Kopf jeder Seite erscheinen. Mit `\endfirsthead` beendet man die Deklaration der Zeilen für den Tabellenanfang. Ohne diesen Befehl bilden die mit `\endhead` festgelegten Zeilen auch den Anfang der Tabelle. Ebenso beendet `\endfoot` die Zeilen, die am Fuß jeder Seite erscheinen sollen, und `\endlastfoot`, soweit verwendet, die Zeilen für das Tabellenende.

*Als Tabellenkopf und
-fuß verwendete Zeilen*

`\caption*[kurztitel]{langtitel}`

Der Befehl `\caption` und seine Variante `\caption*` entsprechen im Wesentlichen dem folgenden `\multicolumn`-Eintrag:

```
\multicolumn{n}{p{\LTcapwidth}}{...}
```

wobei n die Anzahl der Tabellenspalten angibt. Die Breite der Tabellenüberschrift wird mithilfe des Parameters `\LTcapwidth` gesteuert. So kann man z.B. den Eintrag `\setlength\LTcapwidth{breite}` in der Dokumentenpräambel vornehmen. Der Standardwert beträgt 4 Zoll. Genau wie bei den `\caption`-Befehlen der Umgebungen `figure` und `table`, enthält das optionale Argument den Text für das Tabellenverzeichnis, sofern sich dieser von der Tabellenüberschrift unterscheidet.

Sollen auf späteren Seiten andere Tabellenüberschriften erscheinen als auf der ersten, so kann man den `\caption`-Befehl mit vollständigem Text im ersten Tabellenkopf (`\endfirsthead`) benutzen und mit dem Befehl `\caption[ ]` im Haupttabellenkopf (`\endhead`) eine nachgeordnete Überschrift setzen, da bei leerem optionalen Argument kein Eintrag im Tabellenverzeichnis erfolgt. Wenn dabei nicht jedes Mal die Nummer der Tabelle wiederholt werden soll, kann man alternativ den Befehl `\caption*` verwenden. Genau wie bei der Umgebung `table` lassen sich Querverweise auf die Tabelle im Text mithilfe des `\label`-Befehls erzeugen.

Die Formatierung der Überschrift entspricht normalerweise derjenigen der Tabellenüberschriften in der $\text{\LaTeX}$ -Standardklasse `article`. Mithilfe des `caption`-Paketes (das in Abschnitt 6.5.1 beschrieben wird) lassen sich die Tabellenüberschriften in `longtable` und `table` leicht anpassen, wobei ihr Layout in beiden Umgebungen konsistent bleibt.

Tabelle 1: Eine Standardtabelle

1 2 3

```
\usepackage{longtable,supertabular}
\usepackage[font=sl,labelfont=bf]{caption}
\begin{table}[t]\centering
\caption{Eine Standardtabelle}
\begin{tabular}{ccc}1&2&3\end{tabular}
\end{table}
```

Tabelle 2: Mit `longtable`

1 2 3

```
\begin{longtable}{ccc}\caption{Mit longtable}\\
1&2&3\end{longtable}
```

Tabelle 3: Mit `supertabular`

1 2 3

```
\centering \tablecaption{Mit supertabular}
\begin{supertabular}{ccc}1&2&3\\ \end{supertabular}
```

Bsp.
5-4-4

*Fußnoten in
longtable*

Innerhalb der `longtable`-Umgebung lässt sich der Fußnotenbefehl verwenden. Der Text der Fußnoten erscheint am Fuß der jeweiligen Seite. Der Fußnotenzähler wird am Tabellenanfang nicht zurückgesetzt, sondern verwendet weiter die Standardnummerierung, die für die Fußnoten im übrigen Dokument zum Einsatz kommt. Wenn dieses Verfahren nicht erwünscht ist, kann man den Zähler `footnote` am Anfang jeder Tabelle auf Null stellen und nach der Tabelle wieder zurücksetzen, sofern die weiteren Fußnoten durchgehend nummeriert sein müssen.

Damit $\text{\TeX}$ sehr lange mehrseitige Tabellen setzen kann, muss man diese in kleinere Teile aufteilen, so dass nicht alle Daten gleichzeitig im Speicher verbleiben müssen. Normalerweise beträgt der Wert für `longtable` 20 Reihen pro Tabellenteil. Er lässt sich zum Beispiel mit dem Befehl `\setcounter{LTchunksize}{100}` ändern. Die Größe dieser Teile hat keine Auswirkungen auf den Seitenumbruch. Wenn $\text{\TeX}$ viel Speicherplatz zur Verfügung steht, kann `LTchunksize` auf einen hohen Wert eingestellt werden, so dass `longtable` die endgültigen Spaltenbreiten normalerweise in weniger Durchläufen bestimmen kann. Die meisten modernen $\text{\TeX}$ -Installationen können problemlos mit einem `LTchunksize`-Wert umgehen, der mehrere Tabellenseiten umfasst. Der Wert muss dabei mindestens so groß sein wie die Anzahl an Zeilen im Tabellentitel oder -fuß.

*LTchunksize erhöhen
um weniger
 $\text{\LaTeX}$ -Läufe zu
benötigen*

Probleme mit mehrseitigen Tabellen

Wenn ein Gleitobjekt auf der gleichen Seite wie der Anfang einer mehrseitigen Tabelle erscheint, kann das unerwartete Auswirkungen haben. Beide Pakete verfügen über Programmcode für eine solche Situation, aber unter gewissen Umständen können trotzdem Tabellen hinter eine mehrseitige Tabelle gleiten und damit die sequentielle Nummerierung zerstören. Dieses Problem lässt sich normalerweise beheben, indem man der Tabelle einen `\clearpage`-Befehl voranstellt, so dass ein Seitenumbruch erfolgt und zunächst alle aufgelaufenen Gleitobjekte gesetzt werden.

 *Schlechtes
Zusammenspiel von
Gleitobjekten und
mehrseitigen Tabellen*

Weder die `supertabular`- noch die `longtable`-Umgebung setzen nach einer Textzeile *innerhalb* einer Zelle einen Seitenumbruch. Dadurch werden die Seiten nur zwischen Tabellenzeilen (oder an einem `\hline`-Befehl) umbrochen. Wenn eine Tabelle umfangreiche mehrzeilige Zellen enthält, die mit dem Tabellenformatkürzel `p` gesetzt wurden, findet $\text{\LaTeX}$ unter Umständen keinen günstigen Ort für einen Seitenumbruch, so dass am Ende einer Seite unerwünschter Weißraum verbleibt.

 *In Zellen von
p-Spalten erfolgt
kein Seitenumbruch*

Das folgende Beispiel hat Platz für fünf Textzeilen auf jeder Seite. $\text{\LaTeX}$ bricht jedoch zwischen den Tabellenzeilen um, so dass die erste Seite nicht ausgefüllt wird.

```
\usepackage{longtable}
\begin{longtable}{llp{43mm}}
Zelle 1.1 & Zelle 1.2 & Zelle 1.3, ein langer Texteintrag über mehrere Zeilen.\\
Zelle 2.1 & Zelle 2.2 & Zelle 2.3, ein langer Texteintrag über mehrere Zeilen
in einer schmalen Spalte.
\end{longtable}
```

Bsp.
5-4-5

Seite 1			Seite 2		
Zelle 1.1	Zelle 1.2	Zelle 1.3, ein langer Texteintrag über mehrere Zeilen.	Zelle 2.1	Zelle 2.2	Zelle 2.3, ein langer Texteintrag über mehrere Zeilen in einer schmalen Spalte.
Seite 1			Seite 2		

In manchen Tabellen bilden die Zeilen eine wichtige logische Einheit und es ist durchaus erwünscht, dass diese normalerweise nicht umbrochen werden. In anderen Fällen kann dagegen der manuelle Umbruch einer Tabelle empfehlenswert sein, um einen günstigeren Seitenumbruch zu erhalten. Im vorigen Beispiel sollen die ersten beiden Zeilen von Seite 2 an das Ende von Seite 1 verschoben werden. Da $\TeX$ den Eintrag der dritten Zeile nach dem Wort „über“ umbrochen hat, kann man die Tabellenzeile einfach an dieser Stelle mit `\` beenden, leere Einträge in die ersten zwei Spalten einer neuen Zeile einfügen und den verbleibenden Inhalt der p-Spalte in der letzten Zeile dieser Reihe platzieren. Der erste Teil des getrennten Absatzes sollte mit einem `\parfillskip`-Wert von 0pt gesetzt werden, so dass die letzte Zeile sich über die gesamte Breite erstreckt, als wäre sie wie die ersten beiden Zeilen eines längeren Absatzes formatiert.

```
\usepackage{longtable}
\begin{longtable}{llp{43mm}}
Zelle 1.1 & Zelle 1.2 & Zelle 1.3, ein langer Texteintrag über
                                     mehrere Zeilen.\\
Zelle 2.1 & Zelle 2.2 & \setlength{\parfillskip}{0pt}%
                                     Zelle 2.3, ein langer Texteintrag über \\
                                     & & & Zelle 2.3, ein langer Texteintrag über \\
                                     & & & & mehrere Zeilen in einer schmalen Spalte.
\end{longtable}
```

Bsp.
5-4-6

Seite 1			Seite 2		
Zelle 1.1	Zelle 1.2	Zelle 1.3, ein langer Texteintrag über mehrere Zeilen.			
Zelle 2.1	Zelle 2.2	Zelle 2.3, ein langer Texteintrag über			mehrere Zeilen in einer schmalen Spalte.
Seite 1			Seite 2		

5.5 Farbige Tabellen

Die $\LaTeX$ -Farbbefehle aus dem Paket `color` sind den Fontbefehlen nachgebildet und können ohne Einschränkung in Tabellen eingesetzt werden. Besonders zeitsparend erfolgt das Zuweisen einer Farbe für eine ganze Spalte über das Tabellenformatkürzel `>` des `array`-Paketes.

Tag	Besucher	
Montag	57	<code>\usepackage{array,color}</code>
Dienstag	11	<code>\begin{tabular}{>{\color{blue}\bfseries}lr}</code>
Mittwoch	96	<code>Tag & \textcolor{blue}{\bfseries Besucher}\\\\\hline</code>
Donnerstag	122	<code>Montag & 57 \\ Dienstag & 11\\</code>
Freitag	210	<code>Mittwoch & 96 \\ Donnerstag & 122\\</code>
Samstag	198	<code>Freitag & 210 \\ Samstag & 198\\</code>
Sonntag	40	<code>Sonntag & 40</code>
		<code>\end{tabular}</code>

Bsp.
5-5-1

Es ist vielleicht üblicher, Farben als Hintergrund einzusetzen, um bestimmte Zeilen oder Spalten hervorzuheben. Für diesen Fall ist der Befehl `\fcolorbox` des `color`-Paketes leider ungeeignet, da der Hintergrund normalerweise die ganze Zelle ausfüllen soll. Deshalb verfügt das Paket `colortbl` von David Carlisle über eine Reihe von zusätzlichen Befehlen zum Einfärben von Tabellenhintergrund und -linien.

Bsp.
5-5-2

Tag	Besucher
Montag	57
Dienstag	11
Mittwoch	96
Donnerstag	122
Freitag	210
Samstag	198
Sonntag	40
Summe	724

```
\usepackage{colortbl}
\begin{tabular}
  >{\columncolor{blue}\color{white}\bfseries}lr}
\rowcolor[gray]{0.8}
  \color{black} Tag & \bfseries Besucher\\[2pt]
Montag& 57\\ Dienstag& 11\\
Mittwoch& 96\\ Donnerstag& 122\\
Freitag& 210\\ Samstag& 198 \\
Sonntag& 40 \\
\cellcolor[gray]{0.8}\color{black}Summe& 724
\end{tabular}
```

5.6 Anpassen von Linien und Abständen

In diesem Abschnitt werden eine Reihe von Paketen vorgestellt, die den Funktionsumfang der `tabular`-Umgebung um Befehle zum Zeichnen besonderer Tabellenlinien und zur Feineinstellung der Zeilenabstände ergänzen.

5.6.1 Farbige Tabellenlinien

Das Paket `colortbl` erweitert die Layoutparameter für Tabellenlinien um Farbeinstellungen für Linien und für den Freiraum zwischen Doppellinien. Die Deklarationen `\arrayrulecolor` und `\doublerulesepcolor` verfügen über die gleiche Art von Argumenten wie der `\color`-Befehl des `LATEX`-Standardpaketes `color`.

Normalerweise werden diese Deklarationen vor einer Tabelle oder in der Dokumentenpräambel verwendet, um die generelle Linienfarbe für eine Tabelle festzulegen. Durch Konstrukte, die denen im vorigen Beispiel sehr ähneln, lässt sich jedoch auch die Farbe einzelner Linien verändern.

Bsp.
5-6-1

A	B	C
X	Y	Z
100	10	1

```
\usepackage{colortbl} \setlength\arrayrulewidth{1pt}
\newcolumnmtype{B}{!{\color{blue}\vline}}
\newcommand\bhline
  {\arrayrulecolor{blue}\hline\arrayrulecolor{black}}
\newcommand\bcline[1]
  {\arrayrulecolor{blue}\cline{#1}\arrayrulecolor{black}}
\begin{tabular}{|cBc|c|}
\hline
A & B & C \\ \ccline{1-1}\bcline{2-3}
X & Y & Z \\ \bhline
100 & 10 & 1 \\ \hline
\end{tabular}
```

5.6.2 Linien variabler Stärke

Vertikale Linien mit variabler Stärke lassen sich mit der Deklaration `!{dekl}` und dem $\text{\TeX}$ -Basisbefehl `\vrule` mit einem Argument für die Linienstärke erzeugen. Dieser Befehl eignet sich sehr gut, da er automatisch die gesamte Spaltenhöhe ausfüllt, während man für den $\text{\LaTeX}$ -Linienbefehl `\rule` eine genaue Höhe angeben muss. Für horizontale Linien variabler Stärke eignet sich wiederum ein $\text{\TeX}$ -Befehl, und zwar `\noalign`, um den Layoutparameter `\arrayrulewidth` so einzustellen, dass er nur für eine einzelne `\hline`-Linie gilt. Anschließend setzt man die Linienstärke für den Rest der Tabelle zurück.

Im folgenden Beispiel wird ein neues Tabellenformatkürzel `I` definiert, das eine dickere vertikale Linie erzeugt. Ebenso wird der Befehl `\whline` definiert, der eine dickere horizontale Linie erzeugt.

```
\usepackage{array}
\newcolumntype{I}{!{\vrule width 3pt}}
\newlength\savedwidth
\newcommand\whline{\noalign{\global\savedwidth\arrayrulewidth
\global\arrayrulewidth 3pt}%
\hline
\noalign{\global\arrayrulewidth\savedwidth}}

\begin{tabular}{|c|c|c|} \hline
A & B & C \\ \hline
X & Y & Z \\ \hline
100 & 10 & 1 \\ \hline \end{tabular}
```

Bsp.
5-6-2

5.6.3 hline – Kombinieren horizontaler und vertikaler Linien

Das Paket `hline` (von David Carlisle) definiert den Befehl `\hhline`, der sich ähnlich verhält wie `\hline`, außer im Zusammenspiel mit vertikalen Linien.

`\hhline{dekl}`

Das Argument `dekl` besteht aus einer Liste der nachfolgenden Zeichen.

- Eine spaltenbreite, einfache `\hline` (horizontale Linie).
- = Eine spaltenbreite, doppelte `\hline`.
- ~ Eine Spalte ohne `\hline`, d.h. ein spaltenbreiter Leerraum.
- | Eine `\vline` (vertikale Linie), die horizontale Linien durchkreuzt.
- : Eine `\vline` die von einer doppelten `\hline` unterbrochen wird.
- # Segment einer doppelten `\hline` zwischen zwei `\vlines`.
- t Obere Linie eines doppelten `\hline`-Segmentes.
- b Untere Linie eines doppelten `\hline`-Segmentes.
- * `*{3}{==#}` wird wie in der `*`-Form für die Präambel zu `==#==#==#` umgesetzt.

Wenn eine doppelte `\vline` als `||` oder `::` angegeben ist, werden die horizontalen Linien, die durch `\hline` erzeugt werden, unterbrochen. Damit diese sich durchkreuzen, müssen sie mit `#` angegeben werden.

Die Zeichen `t` und `b` können zwischen zwei vertikalen Linien verwendet werden. Die Zeichenfolge `|tb|` liefert dabei das gleiche Ergebnis wie `#`, ist jedoch nicht so effizient. Diese Zeichen werden hauptsächlich für Konstruktionen wie `|t:` (obere linke Ecke) und `:b|` (untere rechte Ecke) verwendet.

Um mit `\hline` eine einzelne horizontale Linie zu erzeugen, sollte das Argument nur die Zeichen „-“, „~“ und „|“ (sowie *-Ausdrücke) enthalten. Im folgenden Beispiel werden die meisten dieser Funktionen demonstriert:

a	b	c	d	
e	f	g	h	
1	2	3	4	
i	j	k	l	?
w	x	y	z	

Bsp.
5-6-3

```
\usepackage{array,hhline}
\setlength\arrayrulewidth{.8pt}
\renewcommand\arraystretch{1.5}
\begin{tabular}{||cc||c|c|c}
    \hhline{|t:::t:::t|}
    a & b & c & d \\ \hhline{||~||~|~|}
    e & f & g & h \\ \hhline{||:::|~|~|}
    1 & 2 & 3 & 4 \\ \hhline{##==#~|=:b|-}
    i & j & k & l & \multicolumn{1}{c|}{?}
    \\ \hhline{||--||---}
    w & x & y & z \\ \hhline{|b:::b:::b|}
\end{tabular}
```

Die von `\hline` erzeugten Linien bestehen aus einer einzelnen `\hrule` (T_EX-Basisbefehl). Die mit `\hhline` erzeugten Linien bestehen aus vielen kleinen Segmenten. T_EX platziert diese in der `.dvi`-Datei sehr exakt, aber zur Ansicht oder zum Ausdruck eingesetzte dvi-Treiber reihen die Liniensegmente eventuell nicht richtig aneinander. Wenn durch einen solchen Effekt Probleme auftreten, kann man `\arrayrulewidth` erhöhen, um die Auswirkungen abzuschwächen.

*Probleme beim
Ausdruck*

5.6.4 arydashln – Gestrichelte Linien

Mithilfe des Paketes `arydashln` von Hiroshi Nakashima kann man in Tabellen mit gestrichelten Linien arbeiten. Es ist kompatibel zu dem Paket `array`, muss aber *nach* `array` geladen werden, wenn beide zum Einsatz kommen.

```
\hdashline[strich/lücke]
\cdashline[spalten-spez][strich/lücke]
\firsthdashline[strich/lücke]
\lasthdashline[strich/lücke]
```

Das Paket lässt sich sehr einfach anwenden. Das neue Tabellenformatkürzel „:“ wird zusammen mit den zwei neuen Befehlen `\hdashline` und `\cdashline` eingeführt. Diese Funktionen lassen sich genauso verwenden wie das L^AT_EX-Standardkürzel „|“ sowie `\hline` und `\cline`, nur dass gestrichelte Linien an Stelle der durchgezogenen erscheinen. Wenn das Paket

array auch geladen ist, sind außerdem die Befehle `\firsthdashline` und `\lasthdashline` verfügbar. Sie stellen gestrichelte Entsprechungen der Befehle `\firsthline` und `\lasthline` aus diesem Paket dar.

A	B	C
X	Y	Z
100	10	1

```
\usepackage{array,arydshln}
\setlength{\extrarowheight}{4pt} % zusätzlicher Raum am
                                   % Kopf der Zeilen

\begin{tabular}{|c::c|c|}
\hline
A & B & C \\ \hline
X & Y & Z \\ \hdashline
100 & 10 & 1 \\ \hline
\end{tabular}
```

Bsp.
5-6-4

Jeder der Befehle besitzt ein optionales Argument, mit dem das Layout der zu zeichnenden Linie festgelegt wird. Ein Wert von `[2pt/1pt]` bedeutet beispielsweise, dass die Linie aus 2pt langen Strichen mit 1pt langen Lücken dazwischen besteht. Die Syntax der `tabular`-Präambel erlaubt keine optionalen Argumente für Spaltenkürzel, so dass das Layout für das „:“-Kürzel nicht auf diese Weise festgelegt werden kann. Stattdessen existiert zusätzlich das Tabellenformatkürzel „;“ mit einem obligatorischen Argument der Form *strich/lücke*, wie im nächsten Beispiel zu sehen.

Die Standardgröße für die Striche und Lücken ist 4pt. Dieser Wert lässt sich ändern, indem man die Layoutparameter `\dashlinedash` und `\dashlinegap` über `\setlength` einstellt. Diese Möglichkeit wird im folgenden Beispiel gezeigt:

A	B	C
X	Y	Z
100	10	1

```
\usepackage{array,arydshln}
\renewcommand{\arraystretch}{1.3333} % zusätzlicher Raum
                                       % gleichmäßig verteilt

\setlength{\dashlinedash}{1pt}
\setlength{\dashlinegap}{1pt}

\begin{tabular}{|;{5pt/2pt}c::c;c;{5pt/2pt}|}
\hdashline
A & B & C \\ \hdashline
X & Y & Z \\ \hdashline[5pt/2pt]
100 & 10 & 1 \\ \hdashline
\end{tabular}
```

Bsp.
5-6-5

*Hässliche Lücken
vermeiden*

Es gibt drei Methoden, mit deren Hilfe das Paket die Striche innerhalb einer Zelle ausrichten kann. Es erzeugt manchmal am Rand der Tabelle eine übergroße Lücke, da nicht genügend Raum für einen weiteren „Strich“ bleibt. In diesem Fall lässt sich mit `\ADLdrawingmode{m}` ein anderer Algorithmus für die Platzierung der Striche einstellen, wobei *m* die Zahl 1 (Standard), 2 oder 3 sein kann.

Die Begleitdokumentation zum Paket gibt weiteren Aufschluss über die im jeweiligen Fall verwendeten Algorithmen. In der Praxis kann man jedoch einfach ausprobieren, welche Einstellungen für `\ADLdrawingmode` bei der gegebenen Tabelle und den Layouts für die gestrichelten Linien das ansprechendste Ergebnis erzielen.

5.6.5 `tabls` – Einstellen von Zeilenabständen

Eine der Schwierigkeiten, $\text{\LaTeX}$ -Tabellen mit außergewöhnlich großen Einträgen zu setzen, ist eine gute Raumverteilung um diese Einträge herum zu erreichen, insbesondere wenn horizontale Linien vorhanden sind. In diesem Fall können der $\text{\LaTeX}$ -Standardbefehl `\arraystretch` oder der Parameter `\extrarowheight` des Paketes `array` von Nutzen sein. Beide wirken sich jedoch auf alle Zeilen der Tabelle aus. Wenn man eine gezieltere Kontrolle benötigt, erweist sich das Paket `tabls` von Donald Arseneau als hilfreich. Man beachte, dass `tabls` zu dem Paket `array` und seinen Abkömmlingen nicht kompatibel ist. Das Paket führt drei neue Parameter ein:

`\tablinesep` Der Minstdurchschuss zwischen aufeinander folgenden Textzeilen in einer Tabelle. Negative Werte werden als 0 behandelt. Der Standardwert beträgt 1pt. Wenn dieser Parameter auf Opt gesetzt wird, überprüft das Programm nicht, wie hoch Tabelleneinträge sein müssen um keinen Text zu berühren (und emuliert damit das Verhalten der `tabular`-Umgebung).

`\arraylinesep` Die Entsprechung zu `\tablinesep` der `array`-Umgebung.

`\extrarulesep` Zusatzabstand, der über und unter allen `\hline`- und `\cline`-Befehlen eingefügt wird. Der Mindestabstand zwischen einer horizontalen Linie und dem Text der nachfolgenden Zeile beträgt `\extrarulesep + 0.5\tablinesep`. Negative Werte reduzieren den Abstand unterhalb der Linie, bis die Linie den Text berührt. Größere negative Werte führen *nicht* dazu, dass die Linie den Text überdrückt. Der Standardwert beträgt 3pt.

Außerdem wird der Befehl `\hline` durch ein optionales Argument, ähnlich dem von `\`, erweitert. Dieses Argument legt den Zusatzabstand unterhalb der Linie fest.

A	B	C
100	10	1

Bsp.
5-6-6

```
\usepackage{tabls} \setlength\tablinesep{2pt}
\begin{tabular}{|c|c|c|} \hline
\large A & \large B & \large C \\ \hline[5pt]
100 & 10 & 1 \\ \hline
\end{tabular}
```

5.6.6 `booktabs` – Formale Linien in Tabellen

Die vertikalen Linien in einer `tabular`-Umgebung werden aus einer Reihe von Liniensegmenten gebildet, jeweils eins pro Tabellenzeile. Befehle, mit deren Hilfe die vertikalen Abstände zwischen den Zeilen oder um horizontale Linien optimiert werden sollen, müssen sorgfältig darauf abgestimmt sein, vertikale Linien nicht zu „unterbrechen“, weil sie Leerräume zwischen den Segmenten einsetzen. Das Paket `booktabs` von Simon Fear verfolgt einen anderen Ansatz. Es erzeugt eher formale Tabellen, die einem traditionellen typographischen Stil entsprechen, der zwar horizontale Linien unterschiedlicher Stärke verwendet, um Tabellenüberschriften abzusetzen, dabei jedoch völlig auf vertikale Linien verzichtet. Das Tabellenformatkürzel `|` wird von diesem Paket zwar nicht deaktiviert, aber auch nicht unterstützt; zudem funktionieren die nachfolgend beschriebenen Zusatzbefehle für horizontale Linien nicht gut im

*Keine vertikalen Linien
verwenden*

Keine Doppellinien verwenden Zusammenspiel mit vertikalen Linien. Ebenso unterstützen `booktabs`-Befehle keine Doppellinien, wie sie durch `||` oder `\hline\hline` erzeugt werden.

Die `booktabs`-Befehle können zusammen mit den `tabular`-Standardumgebungen, den erweiterten Versionen des `array`-Paketes sowie in der `longtable`-Umgebung des `longtable`-Paketes verwendet werden. Das nächste Beispiel zeigt die am häufigsten genutzten Befehle des Paket.

Ware			Preis/kg		
Artikel	Kategorie	€			
Äpfel	Obst	1,50			
Orangen	Obst	2,00			
Steak	Fleisch	4,50			

```

\usepackage{euroman,booktabs}
\begin{tabular}{@{}llr@{}}
\toprule
\multicolumn{2}{c}{Ware}
&\multicolumn{1}{c}{Preis/kg}\\
\cmidrule(r){1-2}\cmidrule(l){3-3}
Artikel & Kategorie & \multicolumn{1}{c}{\euro}\\
\midrule
Äpfel & Obst & 1,50 \\
Orangen & Obst & 2,00 \\
Steak & Fleisch & 4,50 \\
\bottomrule
\end{tabular}

```

Bsp.
5-6-7

<code>\toprule[stärke]</code>	<code>\midrule[stärke]</code>	<code>\bottomrule[stärke]</code>

Das Paket `booktabs` verfügt über die Befehle `\toprule`, `\midrule` und `\bottomrule`. Sie können genauso verwendet werden wie der Standardbefehl `\hline`, bewirken jedoch eine bessere vertikale Raumaufteilung und die Liniestärke wird über die Längenparameter `\heavyrulewidth` (für Linien am Kopf und Fuß der Tabelle) und `\lightrulewidth` (für Linien inmitten der Tabelle) gesteuert. Diese Parameter sind auf 0,08em bzw. 0,05em voreingestellt (wobei em beim Laden des Paketes anhand der Grundschrift des Dokumentes festgelegt wird).

Die Abstände ober- und unterhalb der Linien werden durch folgende Längenparameter bestimmt: `\abovetopsep` (Standard 0pt) ist der Abstand oberhalb der oberen Linie, `\aboverulesep` (Standard 0,4ex) ist der Abstand oberhalb der inneren und unteren Linien, `\belowrulesep` (Standard 0,65ex) ist der Abstand unterhalb der oberen und inneren Linien und `\belowbottomsep` (Standard 0pt) ist der Abstand unterhalb der unteren Linie.

Für den Fall, dass man die Stärke einzelner Linien einstellen möchte, verfügen alle diese Befehle zu genau diesem Zweck über ein optionales Argument. So erzeugt `\midrule[0.5pt]` beispielsweise eine Linie mit der Stärke 0,5pt.

Wenn diese Befehle innerhalb einer `longtable`-Umgebung eingesetzt werden, können sie ein optionales (*Kürzungs*-)Argument erhalten, wie im Folgenden für `\cmidrule` beschrieben. Dieses Argument lässt sich verwenden, um die Linien gegenüber der Gesamtbreite der Tabelle leicht zu kürzen.

<code>\cmidrule[stärke] (kürzung) {start-stop}</code>

Der Befehl `\cmidrule` erzeugt ähnliche Linien wie der $\text{\LaTeX}$ -Standardbefehl `\cline`. Das Argument *start-stop* gibt an, über welche Spalten sich die Linie

erstrecken soll. Anders als bei `\cline`, verlaufen diese Linien normalerweise nicht bis an den Rand einer Spalte. Daher kann man mit `\cmidrule` in benachbarten Spalten Linien erzeugen, die einander nicht berühren, wie auch im nächsten Beispiel zu sehen ist.

Ohne das optionale Argument *stärke* entspricht die Linienstärke dem Wert des Längenparameters `\cmidrulewidth` (voreingestellt auf 0,03em).

Die Linie erstreckt sich normalerweise bis ganz nach links, wird dabei jedoch von der rechten Spalte her um den Wert des Längenparameters `\cmidrulekern` gekürzt. Das optionale Argument (*kürzung*) kann die Buchstaben `l` und `r` enthalten, die anzeigen, dass die Linie links bzw. rechts gekürzt wird. Diesen Buchstaben kann optional eine Längenangabe (in geschweiften Klammern) folgen, in welchem Fall die Länge der Linie um diesen Wert und nicht um den Standardwert `\cmidrulekern` vermindert wird.

Normalerweise bewirken zwei unmittelbar aufeinander folgende `\cmidrule`-Befehle, dass die Linien an der gleichen horizontalen Position durch die Spalten verlaufen. Mithilfe des Befehls `\morecmidrules` kann man eine Folge von inneren Tabellenlinien beenden. Die nachfolgenden inneren Linien erscheinen dann in einer neuen Zeile um den Wert von `\cmidrulesep` versetzt, der dem Wert von `\doublerulesep` entspricht.

Vor und nach jeder Linienfolge, die mit `\cmidrule` erzeugt wird, bleibt jeweils ein vertikaler Abstand der Höhe `\midrulesep`, so dass eine derartige Befehlsfolge die gleichen Abstände erzeugt wie `\midrule`. Die Linien des Befehls `\cmidrule` sind jedoch auf eine kleinere Stärke (leichter) voreingestellt als diejenigen von `\midrule`.

`\addlinespace[höhe]`

Mithilfe von `\addlinespace` lässt sich der Abstand zwischen zwei Zeilen vergrößern. Er unterscheidet sich dadurch von dem optionalen Argument des Befehls `\`, dass er auch unmittelbar vor oder nach Linienbefehlen eingesetzt werden kann.

An dieser Position überschreibt er den Standardabstand, der normalerweise von der Linie erzeugt wird. Ohne sein optionales Argument entspricht der zusätzliche Abstand dem Wert des Längenparameters `\defaultaddspace` (der auf 0,5em voreingestellt ist).

`\specialrule{stärke}{oberhalb}{unterhalb}`

Und schließlich, wenn keiner der anderen Befehle die gewünschte Linie erzeugen kann, bleibt noch der Befehl `\specialrule`. Er verfügt über drei obligatorische Argumente, mit deren Hilfe die Linienstärke sowie der Abstand oberhalb und unterhalb der Linie festgelegt werden.

Da dieses Paket darauf abzielt, „formale“ Tabellen mit gleichmäßig angeordneten Linien in aufeinander abgestimmter Stärke zu erzeugen, warnt sein Autor vor übermäßigem Gebrauch der optionalen Argumente und Sonderbefehle für individuelle Linien. Unter besonderen Umständen können sich diese Funktionen jedoch als sehr nützlich erweisen.

Das nachfolgende Beispiel zeigt die Auswirkungen vieler der genannten Optionen, wobei auch deutlich wird, dass ihr übermäßiger Einsatz dem Layout sehr abträglich ist.

Ware		Preis/kg
a	b	c
Artikel	Kategorie	€
Äpfel	Obst	1,50
Orangen	Obst	2,00
Steak	Fleisch	4,50
x	y	z

```

\usepackage{euroman,booktabs}
\begin{tabular}{@{}lrr@{}}
\toprule
\multicolumn{2}{c}{Ware}
& \multicolumn{1}{c}{Preis/kg}\\
\cmidrule(r){1-2}\cmidrule(l){3-3}
a & b & c \\
\cmidrule(l{2pt}r{2pt}){1-2}\cmidrule(l{2pt}r{2pt}){3-3}
\morecmidrules
\cmidrule(l{2pt}r{2pt}){2-3}
\addlinespace[5pt]
Artikel & Kategorie & \multicolumn{1}{c}{\euro}\\
\midrule
Äpfel & Obst & 1,50 \\
Orangen & Obst & 2,00 \\
\addlinespace
Steak & Fleisch & 4,50 \\
\specialrule{.5pt}{3pt}{3pt}
x & y & z \\
\bottomrule
\end{tabular}

```

Bsp.
5-6-8

5.7 Sonstige Erweiterungen

Zwei andere Paketdateien erweitern den Funktionsumfang des array-Paketes. Die erste ermöglicht Tabelleneinträge, die sich über mehrere Zeilen erstrecken. Die zweite erleichtert die Ausrichtung von Dezimalzahlen in einer Spalte. Man kann eine Zelle, die sich über mehrere Zeilen erstreckt, vortäuschen, indem man den Zellinhalt in eine Box der Höhe null setzt und diese dann anhebt.

100	qqq	
	A	B
20000000	10	10

```

\begin{tabular}{|c|c|c|}
& \multicolumn{2}{c}{qqq}\\
\hline
\raisebox{1.5ex}[0cm][0cm]{100}
& A & B \\
20000000 & 10 & 10 \\
\hline
\end{tabular}

```

Bsp.
5-7-1

Ebenso kann man eine tabular-Standardpräambel `r@{.}l` verwenden, um zwei Tabellenspalten so erscheinen zu lassen, als sei der Inhalt einer Spalte am Dezimalzeichen ausgerichtet; allerdings sieht die Eingabe dann recht seltsam aus. Eine alternative Lösung wird in Abschnitt 5.7.2 auf Seite 282 aufgezeigt.

```

1.2
1.23
913.17

\begin{tabular}{r@{.}l}
1 & 2 \\
1 & 23 \\
913 & 17
\end{tabular}

```

Bsp.
5-7-2

Diese Strategie ist nicht immer günstig, da man ständig darauf achten muss, dass diese „Spalte“ ja in Wirklichkeit aus zwei Tabellenspalten besteht. Das ist besonders beim Zählen der Spalten für die Befehle `\multicolumn` oder `\cline` wichtig. Außerdem muss man `\extracolsep` auf `Opt` setzen, wenn man dieses Konstrukt in einer `tabular*`-Umgebung verwenden will, da $\TeX$ ansonsten einen Leerraum hinter dem Dezimalzeichen einfügen kann, um die Tabelle auf die geforderte Breite zu dehnen.

5.7.1 multirow – Vertikale Ausrichtung in Tabellen

Das Paket `multirow` (von Jerry Leichter) definiert den Befehl `\multirow`. Mithilfe dieses Befehls lassen sich automatisch Tabellen mit mehrzeiligen Spalten konstruieren. Durch optionale Argumente ist eine sehr genaue Einstellung möglich. Dies kann besonders dann von Nutzen sein, wenn die mehrzeiligen Spalten besonders groß sind, wenn `\strut`-Befehle asymmetrisch verwendet werden oder bei übergroßen Unterlängen. In allen diesen Fällen kann die vertikale Zentrierung verschoben sein. Mit dem Korrekturargument `vbeweg` lassen sich dann zum Ausgleich manuell vertikale Verschiebungen einfügen.

$$\backslash\mathrm{multirow}\{n\mathrm{zeilen}\}[n\mathrm{jot}]\{breite\}[vbeweg]\{inhalt\}$$

Innerhalb von `array`-Umgebungen ist der Befehl nicht so nützlich, da Zeilen dort einen zusätzlichen Abstand von einem `\jot` haben. Diese Länge ist auf 3pt voreingestellt und wird verwendet, um abgesetzte Formeln einzuleiten. Der entsprechende Abstand wird von `\multirow` nicht berücksichtigt. Es ist fast unmöglich, dieses Problem (generell) zu lösen. Für eine halbautomatische Regelung kann man jedoch den Längenparameter `\bigstrutjot` auf `\jot` setzen und dann das zweite Argument `njot` von `\multirow` verwenden. Der Wert des Argumentes sollte dabei die Hälfte der Höhe der Zeilen betragen, die überspannt werden.

Mithilfe des Paketes lässt sich bis zu einem gewissen Grad auch die Formatierung innerhalb der Zellen steuern. Kurz vor der Formatierung des zu setzenden Textes wird der Befehl `\multirowsetup` ausgeführt, welcher eine besondere Formatierung für die Zelle festlegt. Als Voreinstellung enthält er den Befehl `\raggedright`; er kann jedoch mithilfe von `\renewcommand` umdefiniert werden.

Der Befehl `\multirow` kann gleichzeitig in mehreren Spalten angewendet werden, wie in dem folgenden Beispiel dargestellt:

Text in Spalte 1	C2a	C3a	C4a
	C2b	Text in Spalte 3	C4b
	C2c		C4c
	C2d		C4d

Bsp.
5-7-3

```
\usepackage{multirow}
\begin{tabular}{|l|l|l|l|} \hline
\multirow{4}{14mm}{Text in Spalte 1}
& C2a & C3a & C4a \\
& C2b & \multirow{3}{14mm}{Text in Spalte 3}
& C4b \\
& C2c & & C4c \\
& C2d & & C4d \\
\hline
\end{tabular}
```

Wir haben nun das Rüstzeug, das kleine Beispiel vom Beginn dieses Abschnitts auch ohne den `\raisebox`-Befehl zu erstellen. Dazu muss zunächst die Ausrichtung des ersten `\multirow`-Absatzes auf `\centering` (zentriert) umgestellt werden. Dann muss man die Breite des Textes in der Spalte berechnen lassen, da diese von dem Befehl `\multirow` benötigt wird. Wenn die Spalte mit der mehrzeiligen Zelle genau wie in den anderen Beispielen eine feste Breite hat, ist dieser Schritt überflüssig.

100	qqq	
	A	B
20000000	10	10

```
\usepackage{multirow}
\renewcommand\multirowsetup{\centering}
\newlength\LL \settowidth\LL{100}
\begin{tabular}{|c|c|c|} \hline
\multirow{2}{\LL}{100}&
\multicolumn{2}{c|}{qqq} \\
& A & B \\
20000000 & 10 & 10 \\
\end{tabular}
```

Bsp.
5-7-4

Die Auswirkungen des optionalen Argumentes für die vertikale Verschiebung, *vbeweg*, werden im folgenden Beispiel deutlich: Das untere Drittel der Tabelle wird um 2mm nach oben verschoben.

Einfacher Text in Spalte 1	Zelle 1a
	Zelle 1b
	Zelle 1c
	Zelle 1d
Einfacher Text in Spalte 1	Zelle 2a
	Zelle 2b
	Zelle 2c
	Zelle 2d
Einfacher Text in Spalte 1	Zelle 3a
	Zelle 3b
	Zelle 3c
	Zelle 3d

```
\usepackage{multirow}
\begin{tabular}{|l|l|} \hline
\multirow{4}{25mm}{Einfacher Text in Spalte 1}
& Zelle 1a \\
& Zelle 1b \\
& Zelle 1c \\
& Zelle 1d \\
\multirow{4}{25mm}{[-2mm]{Einfacher Text in Spalte 1}}
& Zelle 2a \\
& Zelle 2b \\
& Zelle 2c \\
& Zelle 2d \\
\multirow{4}{25mm}{[2mm]{Einfacher Text in Spalte 1}}
& Zelle 3a \\
& Zelle 3b \\
& Zelle 3c \\
& Zelle 3d \\
\end{tabular}
```

Bsp.
5-7-5

5.7.2 dcolumn – Ausrichtung am Dezimalpunkt

Das Paket `dcolumn` (von David Carlisle) eröffnet ein System, das innerhalb von `array`- und `tabular`-Umgebungen Spalten erzeugen kann, deren Inhalt an Dezimal- oder anderen Zeichen ausgerichtet wird. Dabei werden auch Einträge ohne Stellen vor oder hinter dem Komma und leere Einträge richtig verarbeitet.

Das Paket definiert für das Tabellenformat ein „Dezimal“-Kürzel `D` mit drei Argumenten.

`D{eing-zeichen}{ausg-zeichen}{kommastellen}`

eing-zeichen Ein einzelnes Zeichen, das als Trennzeichen (oder Dezimalzeichen) in der Quelldatei verwendet wird (z.B. „.“ oder „.“).

ausg-zeichen Das Trennzeichen, das bei der Ausgabe verwendet wird. Dabei kann es sich um das gleiche Zeichen wie bei dem ersten Argument handeln. Es kann aber auch jeder andere mathematische Ausdruck verwendet werden, wie z.B. `\cdot`.

kommastellen Die maximale Anzahl von Dezimalstellen in der Spalte. Bei einem negativen Wert sind in dieser Spalte beliebig viele Kommastellen erlaubt und alle Einträge werden am Dezimalzeichen zentriert. Dadurch kann eine Spalte allerdings auch zu breit werden (vgl. die ersten beiden Spalten im nächsten Beispiel). Man kann die Anzahl der Ziffern links *und* rechts des Dezimalzeichens mit einem Argument wie `{left.right}` festlegen, das im Folgenden noch beschrieben wird.

Wenn man nicht alle drei Einträge in der Präambel verwenden möchte, kann man mit `\newcolumnmtype` folgendermaßen eigene Kürzel deklarieren:

```
\newcolumnmtype{d}[1]{D{.}{\cdot}{#1}}
```

Das so definierte Spaltenformatkürzel „d“ besitzt als einziges Argument die Anzahl der Dezimalstellen. Als Dezimaltrennzeichen wird in der Quelldatei der normale Punkt „.“ und in der Ausgabe das mathematische Zeichen „ $\cdot$ “ verwendet.

```
\newcolumnmtype{.}{D{.}{.}{-1}}
```

In diesem Fall besitzt das Kürzel „.“ keine Argumente: Der normale Punkt wird bei der Ein- und Ausgabe verwendet und die Einträge werden an dem Punkt zentriert.

```
\newcolumnmtype{,}{D{,}{,}{2}}
```

Das hier definierte Formatkürzel „.“ verwendet in Ein- und Ausgabe das Komma „.“ als Dezimalzeichen und die Spalte erlaubt (höchstens) zwei Dezimalstellen.

Diese Definitionen kommen im folgenden Beispiel zur Anwendung. Dabei fällt auf, dass die erste Spalte, mit einem negativen Wert für *kommastellen* (das Dezimalzeichen wird also in der Spalte zentriert), breiter ist als die zweite Spalte, obwohl beide die gleichen Daten enthalten.

1.2	1.2	1.2	1,2
1.23	1.23	12.5	300,2
1121.2	1121.2	861.20	674,29
184	184	10	69
.4	.4		,4
		.4	

```
\usepackage{dcolumn}
\newcolumnmtype{d}[1]{D{.}{\cdot}{#1}}
\newcolumnmtype{.}{D{.}{.}{-1}}
\newcolumnmtype{,}{D{,}{,}{2}}
\begin{tabular}{|d{-1}|d{2}||.|,|}
  1.2 & & 1.2 & & 1.2 & & 1,2 & \\
  1.23 & & 1.23 & & 12.5 & & 300,2 & \\
  1121.2& & 1121.2& & 861.20 & & 674,29 & \\
  184 & & 184 & & 10 & & 69 & \\
  .4 & & .4 & & & & ,4 & \\
  & & & & & & & \\
  & & & & .4 & & & \\
\end{tabular}
```

Bsp.
5-7-6

Wenn in den Tabelleneinträgen nur numerische Daten ausgerichtet werden sollen, müssten die im vorigen Beispiel gezeigten Formatierungsarten ausreichen. Enthalten die Spalten aber auch Überschriften oder andere Einträge, die sich auf die Spaltenbreite auswirken, so werden die Zahlen möglicherweise nicht wie gewünscht positioniert. Im nächsten Beispiel erscheinen die Zahlen in der ersten Spalte nach links verschoben, obwohl das Dezimalzeichen zentriert ist. In der zweiten Spalte erscheinen die Zahlen rechtsbündig unter einer zentrierten Überschrift. Das ist manchmal wünschenswert, kann aber auch (insbesondere, wenn keine Linien vorhanden sind) suggerieren, dass die Überschrift nicht zu den Daten gehört. In der letzten Spalte sind die Zahlen am Dezimalzeichen ausgerichtet und als Block unter der Überschrift zentriert. Dieser Effekt wird durch ein drittes Argument mit dem Wert 4.2 für das Formatkürzel D erzielt. Dieses Argument legt fest, dass höchstens 4 Ziffern links und 2 Ziffern rechts des Punktes erscheinen dürfen.

```
\usepackage{dcolumn}
\begin{tabular}{|D..{-1}|D..{2}|D..{4.2}|}
\multicolumn{1}{|c|}{breiter Titel}&
\multicolumn{1}{c|}{breiter Titel}&
\multicolumn{1}{c|}{breiter Titel} \\
1000.20 & 1000.20 & 1000.20 \\
123.45 & 123.45 & 123.45
\end{tabular}
```

Bsp.
5-7-7

Das folgende Beispiel ist eine Abwandlung einer Tabelle aus dem *L^AT_EX-
Manual* [106, S. 204], die zeigt, dass D-Spalten auch anderen Zwecken dienen können als der Ausrichtung von numerischen Daten am Dezimalzeichen.

```
\usepackage{dcolumn}
\newcolumnntype{+}{D{/}{\mbox{--}}{4}}
\newcolumnntype{,}{D{,}{,}{2}}
\begin{tabular}{|r||+|
>\raggedrightp{2.2cm}|,|} \hline
\multicolumn{4}{|c|}{GG\&A Hoofed Stock} \\
\hline \hline
& \multicolumn{1}{c|}{Price}& & \\
\cline{2-2}
\multicolumn{1}{|c|}{Year}
& \mbox{low}/\mbox{high}
& \multicolumn{1}{c|}{Comments}
& \multicolumn{1}{c|}{Other} \\
\hline
1971 & 97-245 & Bad year for farmers in the West. & 23,45 \\
\hline
72 & 245-245 & Light trading due to a heavy winter. & 435,23 \\
\hline
73 & 245-2001 & No gnus was very good gnus this year. & 387,56 \\
\hline
\end{tabular}
```

Bsp.
5-7-8

GG&A Hoofed Stock			
hline Year	Price	Comments	Other
	low-high		
1971	97-245	Bad year for farmers in the West.	23,45
72	245-245	Light trading due to a heavy winter.	435,23
73	245-2001	No gnus was very good gnus this year.	387,56

5.8 Tabellenfußnoten

Wie bereits in Abschnitt 3.2.2 auf Seite 119 erklärt, gehören Tabellenfußnoten nicht zum $\text{\LaTeX}$ -Standard. Sie werden nur von den Umgebungen `tabularx`, `longtable`, `mpsupertabular` und `mpsupertabular*` automatisch gesetzt.

Da Tabellenfußnoten normalerweise direkt am Fuß einer Tabelle erscheinen sollen, kann man diese etwa mit Fußnotenmarken simulieren und die Fußnotentexte durch `\multicolumn`-Befehle am Ende der `tabular`-Umgebung platzieren.

5.8.1 minipage-Fußnoten in Tabellen

Wenn man eine `tabular`- oder `array`-Umgebung in eine `minipage`-Umgebung setzt, können in der Tabelle die Standardfußnotenbefehle verwendet werden. In diesem Fall werden die Fußnoten, wie in Abschnitt 3.2.1 auf Seite 117 erklärt, am Ende der `minipage`-Umgebung gesetzt.

Man beachte im nächsten Beispiel die Änderung des `\thefootnote`-Befehls, welche die Verwendung des Befehls `\footnotemark` innerhalb der `minipage`-Umgebung erlaubt. Ohne diese Änderung würde `\footnotemark` eine Fußnotenmarke im Stil des Haupttextes erzeugen, wie in Abschnitt 3.2.2 beschrieben.

		<pre>\begin{minipage}{\linewidth} \renewcommand\thefootnote{\thempfootnote} \begin{tabular}{ll} \multicolumn{2}{c}{\bfseries PostScript Type 1 Fonts} Courier\footnote{Gestiftet von IBM.} & cour, courb, courbi, couri Charter\footnote{Gestiftet von Bitstream.} & bchb, bchbi, bchr, bchri Nimbus\footnote{Gestiftet von URW GmbH.} & unmr, unmr URW Antiqua\footnotemark[\value{mpfootnote}] & uaqrcc URW Grotesk\footnotemark[\value{mpfootnote}] & ugqp Utopia\footnote{Gestiftet von Adobe.} & putb, putbi, putr, putri \end{tabular} \end{minipage}</pre>
PostScript Type 1 Fonts		
Courier ^a	cour, courb, courbi, couri	
Charter ^b	bchb, bchbi, bchr, bchri	
Nimbus ^c	unmr, unmr	
URW Antiqua ^c	uaqrcc	
URW Grotesk ^c	ugqp	
Utopia ^d	putb, putbi, putr, putri	
^a Gestiftet von IBM.		
^b Gestiftet von Bitstream.		
^c Gestiftet von URW GmbH.		
^d Gestiftet von Adobe.		

Bsp.
5-8-1

Dieser Lösungsansatz passt die Breite der Fußnoten allerdings nicht automatisch an die Breite der Tabelle an. Deshalb muss man normalerweise etwas experimentieren, um den richtigen Wert für das entsprechende Argument der `minipage`-Umgebung zu finden.

5.8.2 threeparttable – Tabellen und Fußnoten vereint

Eine andere Möglichkeit zum Setzen von Tabellenfußnoten bietet das Paket `threeparttable` von Donald Arseneau, das einen großen Vorteil hat: Es zeigt

eindeutig, dass es sich bei den Fußnoten um solche innerhalb von Tabellen handelt. Dabei bietet es vollständige Kontrolle über die Darstellung der Verweise sowie die Möglichkeit, der Tabelle eine Legende hinzuzufügen. Das Paket setzt die Tabellenfußnoten automatisch in einer Box mit der Breite der Tabelle.

*Tabellenfußnoten in
Tabellenbreite*

Normalerweise sollte die `threeparttable`-Umgebung innerhalb einer `table`-Umgebung verwendet werden, so dass die Tabelle gleiten kann. Man kann sie jedoch auch direkt verwenden. Dann ähnelt sie der nicht gleitenden `table`-Umgebung, die in Beispiel 6-3-4 auf Seite 304 beschrieben wird.

```
\usepackage{threeparttable}
\begin{threeparttable}
\caption[Beispiel einer
\texttt{threeparttable}-Umgebung]
{\textbf{PostScript Type 1 Fonts}}
\begin{tabular}{@{}ll@{}}
Courier\tnote{a} & cour, courb, courbi, couri & \\\
Charter\tnote{b} & bchb, bchbi, bchr, bchri & \\\
Nimbus\tnote{c} & unmr, unmr & \\\
URW Antiqua\tnote{c} & uaqrcc & \\\
URW Grotesk\tnote{c} & ugqp & \\\
Utopia\tnote{d} & putb, putbi, putr, putri & \\\
\end{tabular}
\begin{tablenotes}
\item[a] Gestiftet von IBM.
\item[b] Gestiftet von Bitstream.
\item[c] Gestiftet von URW GmbH.
\item[d] Gestiftet von Adobe.
\end{tablenotes}

\begin{tablenotes}[flushleft,online]
\item[a] Gestiftet von IBM.
\item[b] Gestiftet von Bitstream.
\item[c] Gestiftet von URW GmbH.
\item[d] Gestiftet von Adobe.
\end{tablenotes}

\begin{tablenotes}[para]
\item[] Gestiftet von:
\item[a] IBM, \item[b] Bitstream,
\item[c] URW GmbH,
\item[d] Adobe.
\end{tablenotes}
\end{threeparttable}
```

Tabelle 1: PostScript Type 1 Fonts

Courier ^a	cour, courb, courbi, couri
Charter ^b	bchb, bchbi, bchr, bchri
Nimbus ^c	unmr, unmr
URW Antiqua ^c	uaqrcc
URW Grotesk ^c	ugqp
Utopia ^d	putb, putbi, putr, putri

^a Gestiftet von IBM.

^b Gestiftet von Bitstream.

^c Gestiftet von URW GmbH.

^d Gestiftet von Adobe.

a Gestiftet von IBM.

b Gestiftet von Bitstream.

c Gestiftet von URW GmbH.

d Gestiftet von Adobe.

Gestiftet von: ^a IBM, ^b Bitstream,

^c URW GmbH, ^d Adobe.

Bsp.
5-8-2

Wie der Name schon andeutet, besteht die Umgebung `threeparttable` aus drei Teilen. Die **Legende** wird mit dem gewohnten `\caption`-Befehl erzeugt, der oberhalb oder unterhalb der Tabelle stehen kann. Für die **Tabelle** selbst kann man eine der `tabular`- oder `tabular*`-Standardumgebungen, eine der erweiterten Varianten des `array`-Paketes oder die `tabularx`-Umgebung

des `tabularx`-Paketes verwenden. Zukünftige Versionen des Paketes werden möglicherweise weitere Umgebungen unterstützen; die aktuelle Liste der zurzeit unterstützten Umgebungen findet man in der Paketdokumentation. Im dritten Teil schließlich befinden sich die Texte der **Fußnoten**, die mit einer oder mehreren `tablenotes`-Umgebungen gebildet werden.

Das Paket `threeparttable` bietet eine Reihe von Optionen zum Formatieren der Tabellenfußnoten:

`para` Die Fußnoten werden in einem Absatz gesetzt, ohne erzwungene Zeilenumbrüche.

`flushleft` Die Fußnoten erhalten keinen hängenden Einzug.

`online` Fußnotenmarker werden in normaler Größe gesetzt, nicht hochgestellt.

`normal` Die Standardformatierung wird wiederhergestellt.

Jede dieser Optionen lässt sich als Paketoption einsetzen, um so das Layout für alle Tabellen im Dokument festzulegen. Sie können jedoch auch, wie im Beispiel, für einzelne `tablenotes`-Umgebungen verwendet werden.

Zusätzlich zu diesen Optionen verfügt das Paket über eine Reihe von Befehlen, die undefiniert werden können, um speziellere Formatierungen durchzuführen, als sie mit den Paketoptionen möglich sind. Diese Möglichkeiten sind in der beiliegenden Dokumentation genauer beschrieben.

5.9 Verwendungsmöglichkeiten

Die folgenden Beispiele zeigen, wie komplex sich die Positionierung von Elementen gestalten kann, indem sie so anspruchsvolle Verfahren wie das Verschachteln von Tabellen darstellen. Dabei kommen viele der in diesem Kapitel beschriebenen Funktionen zur Anwendung.

5.9.1 Tabellen mit breiten Einträgen

Manchmal muss der Weißraum zwischen schmalen Spalten gleichmäßig über die gesamte Tabellenbreite verteilt werden. Die folgende Tabelle hat z.B. eine ziemlich breite erste Zeile, gefolgt von einer Reihe schmaler Spalten.

eine-ziemlich-lange-Zeile				<code>\begin{tabular}{ccc}</code>
C1	C2	C3		<code>\multicolumn{3}{c}{eine-ziemlich-lange-Zeile}\\</code>
2.1	2.2	2.3		<code>C1 &C2 &C3 \\ 2.1&2.2&2.3 \\ 3.1&3.2&3.3</code>
3.1	3.2	3.3		<code>\end{tabular}</code>

Bsp.
5-9-1

Mithilfe des Befehls `\extracolsep` kann man elastische Längen an den Anfang jeder Spalte setzen. Der tatsächliche Wert einer solchen Länge ist dabei irrelevant, solange diese stark genug schrumpfen kann, um gerade den benötigten Platz auszufüllen. In diesem Fall muss man natürlich eine Breite für die gesamte Tabelle angeben. Man könnte `\linewidth` verwenden und die Tabelle über die gesamte Breite setzen, aber hier lässt sich ein besseres

Ergebnis erzielen, wenn man die Breite des längsten Eintrags vorausberechnet und dann die Breite der gesamten `tabular*`-Umgebung darauf einstellt.

```

\usepackage{array}
\newlength\Mylen
\settowidth\Mylen{eine-ziemlich-lange-Zeile}
\addtolength\Mylen{2\tabcolsep}
\begin{tabular*}{\Mylen}%
    {!\extracolsep{4in minus 4in}}ccc
\multicolumn{3}{c}{eine-ziemlich-lange-Zeile}\\
C1 & C2 & C3 \\
2.1 & 2.2 & 2.3 \\
3.1 & 3.2 & 3.3
\end{tabular*}

```

Bsp.
5-9-2

Für eine korrekte Ausrichtung muss auch der Spaltenabstand (`\tabcolsep`) auf beiden Seiten eines Eintrags berücksichtigt werden. Alternativ dazu kann man die Spaltenabstände auf der linken und rechten Seite der `tabular*`-Umgebung mithilfe von `@{}`-Befehlen unterdrücken.

5.9.2 Tabellen in Tabellen

Das unten aufgeführte Beispiel zeigt, wie man in $\text{\LaTeX}$ mit wenig Mehraufwand komplexe Tabellenlayouts erzeugen kann.

```
\firstline \lastline
```

Die Familie der `tabular`-Umgebungen erlaubt eine vertikale Ausrichtung an der Grundlinie des Textes, in dem die Umgebung verwendet wird. Dabei wird sie normalerweise mittig gesetzt. Diese Voreinstellung kann auch so geändert werden, dass sich die Umgebung bündig an ihrer ersten oder letzten Zeile ausrichtet; dazu erhält das optionale Positionsargument den Wert `t` oder `b`. Dieses Verfahren funktioniert jedoch nicht, wenn es sich bei dem ersten oder letzten Element der Umgebung um einen `\hline`-Befehl handelt – dann wird die Umgebung bündig mit der horizontalen Linie gesetzt.

Tabellen	ohne <code>\hline</code> wie hier,	vs. Tabellen	mit <code>\hline</code> wie hier	<pre> \usepackage{array} Tabellen \begin{tabular}[t]{1} ohne \verb=\hline= \ wie hier, \end{tabular} vs. Tabellen \begin{tabular}[t]{ 1 } \hline mit \verb=\hline= \ wie hier \ \hline \end{tabular} </pre>
				Bsp. 5-9-3

Um dennoch eine korrekte Ausrichtung zu erzielen, kann man die beiden Befehle `\firstline` und `\lastline`, zwei besondere Versionen von

`\hline` aus dem `array`-Paket, verwenden. Sie sorgen dafür, dass die Tabelleninhalte korrekt ausgerichtet werden, solange die erste oder letzte Zeile nicht ausgesprochen große Objekte enthält.

Bsp.
5-9-4

Tabellen	ohne	vs. Tabellen	mit
	<code>\hline</code>		<code>\hline</code>
	wie hier,		wie hier

```
\usepackage{array}
Tabellen
\begin{tabular}[t]{l}
  ohne \\ \verb=\hline= \\ wie hier,
\end{tabular}
vs. Tabellen
\begin{tabular}[t]{|l|}
\firstline
  mit \\ \verb=\hline= \\ wie hier \\
\lasthline
\end{tabular}
```

`\setlength\extratabsurround{dim}`

Diese beiden Befehle benutzen den zusätzlichen Dimensionsparameter `\extratabsurround`, um am Anfang und Ende der Umgebung zusätzlichen Leerraum einzufügen. Diese Eigenschaft ist, wie im folgenden Beispiel, auch nützlich für die Ausrichtung verschachtelter Tabellen:

```
\usepackage{array}
\setlength\extratabsurround{5pt}
\begin{tabular}{|cc|}
\hline
\emph{Name} & \emph{Telefon}
\hline
  John & \begin{tabular}[t]{|cc|}
    \firstline
      \emph{Tag} & \multicolumn{1}{c|}{\itshape Telefon}
    \hline
      Mi & 5554434
    \hline
      Mo & \begin{tabular}[t]{|cc|}
        \firstline
          \emph{Zeit} & \emph{Telefon}
        \hline
          8--10 & 5520104 \\ 1--5 & 2425588
        \lasthline
      \end{tabular}
    \hline
    \end{tabular}
  \hline
Martin & \begin{tabular}[t]{|cp{4.5cm}|}
    \firstline
      \emph{Telefon} & \multicolumn{1}{c|}{\itshape Aufgaben}
    \hline
      3356677 & Mary soll die Nachricht beantworten.
    \lasthline
    \end{tabular}
  \hline
Peter & \begin{tabular}[t]{|cl|}
    \firstline
      \emph{Monat} & \multicolumn{1}{c|}{\itshape Telefon}
    \hline
      Sep--Mai & 5554434 \\ Jun & Kein Telefon \\
      Jul--Aug & 2211456 \\
    \lasthline
    \end{tabular}
  \hline
\end{tabular}
```

Name		Telefon	
John	Tag	Telefon	
	Mi	5554434	
	Mo	Zeit	Telefon
		8–10	5520104
		1–5	2425588
Martin	Telefon		Aufgaben
	3356677	Mary soll die Nachricht beantworten.	
Peter	Monat		Telefon
	Sep–Mai		5554434
	Jun		Kein Telefon
	Jul–Aug		2211456

Bsp.
5-9-5Abschließendes
Beispiel

Der folgende L^AT_EX-Code zeigt, wie sich verschiedene der zuvor in diesem Kapitel beschriebenen Techniken und Pakete kombinieren lassen. Mithilfe des Paketes `tabularx` wird eine Tabelle mit 12 Spalten erzeugt, wobei die Spalten 3 bis 12 die gleiche Breite haben. Mit dem Paket `multirow` wird der Zeilentitel „Präfix“ erzeugt, der sich in Spalte 1 über zwei Tabellenzeilen erstreckt. Damit dieser Zeilentitel korrekt positioniert werden kann, muss zunächst die Breite des Titels errechnet werden.

```
\usepackage{array,tabularx,multirow}
\newlength\Tl \settowidth{\Tl}{Präfix} \setlength\tabcolsep{1mm}
\newcommand\T[1]{${10}^{\#1}$}
\begin{tabularx}{\linewidth}{|l|l|*{10}{>{\small}X|}} \hline
\multicolumn{12}{|c|}{\textbf{Präfixe für SI-Einheiten}}\\ \hline
\multicolumn{2}{|c|}{Faktor} & \multicolumn{10}{c|}{} \\
\T{24}&\T{21}&\T{18}&\T{15}&\T{12}&\T{9}&\T{6}&\T{3}&\T{2}&\T{1}& & \\
\cline{1-2}
\multirow{2}{\Tl}{Präfix}&Name & & & & & & & & & & \\
Yotta &Zetta &Exa &Peta &Tera &Giga &Mega &Kilo &Hekto &Deka & & \\
& & & & & & & & & & & \\
& & & & & & & & & & & \\
Y &Z &E &P &T &G &M &k &h &da & & \\
\hline
... weitere Eingaben unterdrückt ...
```

Bsp.
5-9-6

Präfixe für SI-Einheiten											
Faktor		10 ²⁴	10 ²¹	10 ¹⁸	10 ¹⁵	10 ¹²	10 ⁹	10 ⁶	10 ³	10 ²	10
Präfix	Name Symbol	Yotta Y	Zetta Z	Exa E	Peta P	Tera T	Giga G	Mega M	Kilo k	Hekto h	Deka da
Präfix	Symbol Name	y Yocto	z Zepto	a Atto	f Femto	p Piko	n Nano	μ Mikro	m Milli	c Zenti	d Dezi
Faktor		10 ⁻²⁴	10 ⁻²¹	10 ⁻¹⁸	10 ⁻¹⁵	10 ⁻¹²	10 ⁻⁹	10 ⁻⁶	10 ⁻³	10 ⁻²	10 ⁻¹

KAPITEL 6

Gleitobjekte

6.1 Parameter für Gleitobjekte	292
6.2 Positionieren von Gleitobjekten.	295
6.3 Erweiterungen für L ^A T _E Xs Gleitobjektkonzept	300
6.4 Im Fließtext eingebettete Gleitobjekte	307
6.5 Gleitobjektlegenden	315

Dokumente wären viel leichter zu lesen, wenn die Informationen, die zusammengehören, auch zusammen auf einer Seite erschienen. Oft ist dies jedoch schon rein technisch unmöglich; zudem teilt T_EX das Textmaterial automatisch so in Seiten auf, dass möglichst keine unerwünschten Leerräume entstehen. Ist eine derartige Aufteilung nicht gewünscht oder nicht möglich (wie z.B. bei Abbildungen und Tabellen), muss man das betreffende Material an eine geeignete Position „gleiten“ lassen, um halbleere Seiten zu vermeiden. Das kann beispielsweise der Anfang oder das Ende der aktuellen oder nachfolgenden Seite sein.

Dieses Kapitel beschreibt, wie „größere Teile“ eines Dokumentes bequem als Gleitobjekte auf einer Seite zusammengehalten werden können. Zu Anfang werden die Parameter vorgestellt, die festlegen, wie L^AT_EX die Basis-Gleitumgebungen `figure` und `table` setzt. Außerdem werden einige der Pakete beschrieben, mit denen sich Gleitobjekte auf einfache Weise gezielt platzieren lassen (Abschnitt 6.2). Danach erklären wir, wie man eigene Gleitobjektumgebungen definiert (Abschnitt 6.3.1), wie man mithilfe des `caption`-Paketes nicht gleitende Elemente in die entsprechenden Verzeichnisse aufnimmt (Abschnitt 6.3.2) und wie man Gleitobjekte bei Bedarf teilweise oder vollständig dreht (Abschnitt 6.3.3).

Absätze lassen sich oft durch im Fließtext integrierte Bilder ansprechender gestalten. Um dies zu ermöglichen, wurde eine Reihe von Paketen entwickelt, von denen in Abschnitt 6.4 zwei ausführlicher behandelt werden.

Der letzte Abschnitt befasst sich mit der Formatierung von Legenden (Bildunterschriften und ähnliches). Es ist wichtig und notwendig, die Beschreibung für Abbildungs- und Tabelleninhalte auf unterschiedlichste Weise

gestalten zu können. Dazu gehört auch die Möglichkeit, Abbildungen und Tabellen, die mit eigener Legende und eigenem Label versehen sind, in einem größeren Gleitobjekt zusammenzufassen.

Im Laufe der Jahre wurden viele Pakete entwickelt, die sich mit Gleitobjekten beschäftigen. Nicht alle können hier erwähnt werden. Auch die besprochenen Pakete verfügen häufig über mehr als die hier dargestellten Funktionen. Ziel dieses Kapitels ist es, den Leser in die Lage zu versetzen, seine eigene Auswahl zu treffen, und zu zeigen, wie ein bestimmtes Verhalten unter den jeweils gegebenen Umständen erreicht werden kann. Darüber hinaus bietet die Originaldokumentation der Pakete immer die Möglichkeit ihren gesamten Funktionsumfang kennen zu lernen.

6.1 Parameter für Gleitobjekte

In der aktuellen Version von $\text{\LaTeX}$ stellen Gleitobjekte oftmals ein Problem dar, denn das System wurde zu einer Zeit entwickelt, als Dokumente noch erheblich weniger graphisches Material enthielten als heute. Die Gleitobjekte (Tabellen und Abbildungen) lassen sich im Allgemeinen leicht positionieren, solange sie im Vergleich zum Text nicht zu viel Raum einnehmen. Wenn jedoch viele Bilder oder Tabellen vorhanden sind, gleitet das gesamte Material häufig ab einem bestimmten Punkt an das Ende des gegebenen Kapitels oder Dokumentes. Um diesen Effekt zu vermeiden, kann man von Zeit zu Zeit alle noch nicht positionierten Gleitobjekte mit dem Befehl `\clearpage` platzieren. Man kann auch versuchen, die nachfolgend beschriebenen Parameter für Gleitobjekte in einem Dokument günstiger einzustellen, oder ein Paket verwenden, das dafür sorgt, dass alle Tabellen und Abbildungen stets genau dort gedruckt werden, wo sie im Quelldokument erscheinen.

In der folgenden Liste steht die Bezeichnung „Gleitobjekt“ immer für eine Tabelle oder Abbildung, und eine „Gleitobjektkolumne“ für eine Seite bzw. Spalte (bei mehrspaltigem Layout), die ausschließlich Gleitobjekte und keinen Fließtext enthält. Änderungen von Parametern treten grundsätzlich erst ab der nächsten Kolumne (nicht in der aktuellen) in Kraft.

`topnumber` Zähler, der die maximal zulässige Anzahl von Gleitobjekten am Kopf der Kolumne festlegt (voreingestellt ist der Wert 2). Wird mit dem Befehl `\setcounter` geändert.

`bottomnumber` Zähler, der die maximal zulässige Anzahl von Gleitobjekten am Fuß der Kolumne festlegt (voreingestellt ist der Wert 1). Wird mit `\setcounter` geändert.

`totalnumber` Zähler, der die maximal zulässige Anzahl von Gleitobjekten in einer Kolumne festlegt (voreingestellt ist der Wert 3). Wird mit `\setcounter` geändert.

`\topfraction` Maximaler Anteil der Kolumne, der von Gleitobjekten am Kopf der Kolumne eingenommen werden darf (z.B. bedeutet 0.2, dass 20% von Gleitobjekten belegt sein darf; voreingestellt ist der Wert 0.7). Wird mit `\renewcommand` geändert.

- `\bottomfraction` Maximaler Anteil der Kolumne, der von Gleitobjekten am Fuß der Kolumne eingenommen werden darf (voreingestellt ist der Wert 0.3). Wird mit `\renewcommand` geändert.
- `\textfraction` Mindestanteil einer normalen Kolumne, der von Fließtext eingenommen werden muss (voreingestellt ist 0.2). Wird mit `\renewcommand` geändert.
- `\floatpagefraction` Mindestanteil einer Gleitobjektkolumne, der mit Gleitobjekten gefüllt sein muss. Dadurch wird der Leerraum in der Kolumne begrenzt (voreingestellt ist 0.5). Wird mit `\renewcommand` geändert.
- `dbltopnumber` Analog zu `topnumber` die maximale Anzahl an zweispaltigen Gleitobjekten bei zweispaltigem Seitenlayout (voreingestellt ist 2). Wird mit `\setcounter` geändert.
- `\dbltopfraction` Analog zu `\topfraction` der maximale Anteil an zweispaltigen Gleitobjekten auf einer zweispaltigen Seite (voreingestellt ist 0.7). Wird mit `\renewcommand` geändert.
- `\dblfloatingpagefraction` Analog zu `\floatpagefraction` der Anteil an zweispaltigen Gleitobjekten auf einer Gleitobjektseite (voreingestellt ist 0.5). Wird mit `\renewcommand` geändert.
- `\floatsep` Elastische Länge, die den zusätzlichen vertikalen Abstand zwischen Gleitobjekten am Kopf oder Fuß einer Kolumne festlegt (voreingestellt sind 12pt plus 2pt minus 2pt für Dokumente mit einer Grundschrift von 10pt und 11pt, bzw. 12pt plus 2pt minus 4pt für Dokumente mit einer Grundschrift von 12pt). Wird mit `\setlength` geändert.
- `\textfloatsep` Elastische Länge, die den zusätzlichen vertikalen Abstand von Gleitobjekten am Kopf oder Fuß einer Kolumne zum Text festlegt (voreingestellt ist 20pt plus 2pt minus 4pt). Wird mit `\setlength` geändert.
- `\intextsep` Elastische Länge, die den zusätzlichen vertikalen Abstand vor und nach einem Gleitobjekt angibt, das mit der Option `h` mitten im Text positioniert ist (der voreingestellte Wert entspricht `\floatsep`). Wird mit `\setlength` geändert.
- `\dblfloatingsep` Zu `\floatsep` analoge elastische Länge für Gleitobjekte doppelter Breite auf einer zweispaltig gesetzten Seite (der voreingestellte Wert entspricht `\floatsep`). Wird mit `\setlength` geändert.
- `\dbltextfloatsep` Eine zu `\textfloatsep` analoge elastische Länge für Gleitobjekte doppelter Breite auf einer zweispaltig gesetzten Seite (der voreingestellte Wert entspricht `\textfloatsep`). Wird mit `\setlength` geändert.
- `\topfigrule` Befehl zum Platzieren eines Trennelementes (normalerweise einer Linie) zwischen Gleitobjekten am Kolumnenanfang und dem nachfolgenden Text. Es wird direkt vor dem Abstand `\textfloatsep` zwischen Text und Gleitobjekt eingefügt. Genau wie `\footnoterule` darf es keinen vertikalen Raum belegen.

`\botfigrule` Analog zu `\topfigrule`; wird jedoch nach dem Abstand `\textfloatsep` eingefügt, der Text von Gleitobjekten am Fuß einer Kolumne trennt.

`\dblfigrule` Funktioniert wie `\topfigrule`, jedoch für zweispaltige Gleitobjekte.

Durch Ändern der obigen Parameter lässt sich der L^AT_EX-Algorithmus zum Positionieren von Gleitobjekten steuern. Zum Optimieren der Ergebnisse sollte man jedoch die Abhängigkeiten der Parameter untereinander verstehen.

Wenn man in einem Dokument mit zahlreichen Gleitobjekten die voreingestellten Werte verwendet, wird das formatierte Dokument normalerweise eine Reihe von Gleitobjektkolumnen enthalten, d.h. Seiten oder Spalten, auf denen sich ausschließlich Gleitobjekte befinden. Diese bestehen häufig zu einem großen Teil aus Weißraum. Wenn sie z.B. nur ein einziges Gleitobjekt enthalten, das lediglich die Hälfte der Kolumne bedeckt, sähe es sicherlich besser aus, wenn L^AT_EX den verbleibenden Freiraum mit Text füllen würde. Die Ursache liegt darin, dass der Algorithmus angehalten ist, nach jeder Kolumne so viele aufgelaufene Gleitobjekte wie möglich zu platzieren. Dieser Prozess erzeugt solange Gleitobjektkolumnen, bis nicht mehr genügend Gleitobjekte für eine ganze Seite bzw. Spalte vorhanden sind. Das Erstellen dieser Seiten wird durch den Parameter `\floatpagefraction` gesteuert. Er gibt an, wie viel Raum auf einer Seite mindestens von Gleitobjekten eingenommen werden muss, und ist auf eine halbe Seite voreingestellt. Dem Standard entsprechend darf jedes Gleitobjekt in einer Gleitobjektkolumne platziert werden (durch das Argument `tbp`); damit kann es, sobald es etwas größer ist als eine halbe Kolumne, auch alleine in einer solchen stehen. Halbleere Seiten bzw. Spalten lassen sich also vermeiden, indem man den Wert für `\floatpagefraction` erhöht.

*Halbleere
Gleitobjektkolumnen*

Dadurch wird es allerdings auch schwieriger, Gleitobjektkolumnen zu erzeugen. Das kann dazu führen, dass die Positionierung einiger Gleitobjekte weiter aufgeschoben wird, was wiederum verhindert, dass andere Gleitobjekte platziert werden können. Darum ist es oft besser, die erlaubte Position für ein problematisches Gleitobjekt einfach explizit anzugeben (zum Beispiel mit `\begin{figure}[tb]`).

Eine weitere häufige Ursache dafür, dass alle Gleitobjekte erst am Ende eines Kapitels positioniert werden, ist die Verwendung des Positionsargumentes Seitenfuß, `[b]`. Dadurch wird der Fuß der Seite die einzig akzeptable Position für ein Gleitobjekt. Wenn dieses Objekt jedoch größer ist als der durch `\bottomfraction` zugelassene (normalerweise relativ kleine) Seitenanteil, kann es gar nicht positioniert werden. Damit können alle Gleitobjekte des gleichen Typs ebenfalls nicht positioniert werden. Das gleiche gilt, wenn nur das Argument `[h]` oder `[t]` angegeben wird, und das Objekt für den verbleibenden Platz auf der Seite oder für `\topfraction` zu groß ist.

Bei der Berechnung der erforderlichen Seitenanteile berücksichtigt L^AT_EX auch die Trennung (d.h. `\textfloatsep`) zwischen Gleitobjekten und Haupttext. Durch Vergrößern dieses Wertes verkleinert man automatisch die maximal zulässige Größe, die ein Gleitobjekt haben darf, um am Kopf oder Fuß einer Seite positioniert zu werden.

Wenn wirklich alle Gleitobjekte am Ende eines Kapitels landen, sollte man bei den ersten prüfen, ob vielleicht ihre Positionsargumente verhindern, dass sie ordentlich platziert werden.

6.2 Positionieren von Gleitobjekten

Der Algorithmus zum Positionieren von Gleitobjekten setzt die Objekte vorzugsweise an den Anfang einer Seite, selbst wenn sie dadurch vor ihrem eigentlichen Verweis erscheinen. Das ist nicht immer akzeptabel, aber es gibt keine einfache Lösung für dieses Problem, außer den Algorithmus grundlegend zu ändern. Diese Änderung wird durch das Paket `flatter` (von Frank Mittelbach) vorgenommen. Es stellt sicher, dass Gleitobjekte niemals vor ihrem Verweis erscheinen.

*Gleitobjekte immer
nach ihrem
Textverweis*

Manchmal mag man jedoch weniger drastische Lösungen bevorzugen. Wenn beispielsweise ein Gleitobjekt am Anfang einer Seite positioniert wird und zu einem Abschnitt gehört, der erst in der Mitte der Seite beginnt, sieht es so aus, als gehörte das Objekt zum vorhergehenden Abschnitt. Man möchte dieses Verhalten vielleicht vermeiden, aber trotzdem weiterhin zulassen, dass Gleitobjekte in anderen Situationen am Seitenanfang erscheinen können. Für diesen Zweck verfügt $\text{\LaTeX}$ über den folgenden Befehl:

`\suppressfloats[position]`

Das optionale Argument *position* kann entweder `t` oder `b` sein. Wenn der Befehl `\suppressfloats` irgendwo in einem Dokument erscheint, werden in den mit *position* festgelegten Bereichen auf derselben Seite keine weiteren Gleitobjekte mehr platziert. Ohne dieses Argument, werden alle verbleibenden Gleitobjekte aufgeschoben. Wenn man also verhindern möchte, dass Objekte vor den eigentlichen Beginn ihres Abschnitts gleiten, kann man die Gliederungsbefehle folgendermaßen umdefinieren:

```
\renewcommand\section
  {\suppressfloats[t]%
   \@startsection{section}{...}{...}{...} ... }
```

Die Argumente für `\@startsection` wurden in Abschnitt 2.2.2 behandelt.

Die Positionierung von Gleitobjekten in $\text{\LaTeX}$ lässt sich auch durch das Zeichen `!` in Verbindung mit den Positionsoptionen `h`, `t` und `b` beeinflussen. (In Gleitobjektspalten bleibt dieses Verfahren ohne Wirkung.) Das bedeutet, dass für dieses eine Gleitobjekt die Einschränkungen durch die zuvor beschriebenen Parametereinstellungen (z.B. `\textfraction`) ignoriert werden. Dadurch kann das betreffende Objekt immer an der vorgesehenen Stelle positioniert werden, solange die beiden folgenden Bedingungen erfüllt sind:

- Das Objekt passt auf die aktuelle Seite, d.h. seine Höhe plus die Höhe des bereits auf der Seite befindlichen Materials überschreitet nicht den Wert `\textheight`.
- Es existieren keine aufgelaufenen Gleitobjekte desselben Typs.

Alle anderen normalerweise gültigen Einschränkungen (wie z.B. die zulässige Anzahl von Gleitobjekten auf einer Seite) werden ignoriert. Wenn man zum Beispiel `[!b]` angibt, kann das Objekt am Ende der Seite positioniert werden, auch wenn es größer ist als die maximal durch `\bottomfraction` erlaubte Höhe. Beim Positionieren des betreffenden Gleitobjektes werden außerdem auch alle `\suppressfloats`-Befehle ignoriert.

*Algorithmus zum
Bestimmen der
zulässigen Position*

Die Reihenfolge der Positionsoptionen ist irrelevant und alle Optionen sollten in einem Befehl nur einmal verwendet werden. Das Argument `[bt]` ist z.B. identisch mit `[tb]`. Es bewirkt also *nicht*, dass $\text{\LaTeX}$ zunächst versucht, das Objekt am Fuß einer Seite zu positionieren, und erst bei Misslingen einen Platz am Kopf in Betracht zieht. Stattdessen verwendet $\text{\LaTeX}$ stets die folgende Testreihenfolge bis eine geeignete Position gefunden ist:

1. Bei `!` ignoriere die meisten Einschränkungen (wie oben beschrieben) und fahre fort.
2. Bei `h` versuche, das Objekt genau an dieser Stelle zu positionieren. Funktioniert das nicht und es ist keine weitere Position angegeben, ändere das Argument in `t` (um das Objekt möglicherweise auf der nächsten Seite zu positionieren).
3. Bei `t` versuche, das Objekt am Kopf der aktuellen Seite zu positionieren.
4. Bei `b` versuche, das Objekt am Fuß der aktuellen Seite zu positionieren.
5. Bei `p` versuche, das Objekt in einer Gleitobjektkolumne (oder in einer Gleitobjektspalte) zu positionieren, sobald die aktuelle Seite (oder Spalte) gesetzt ist.
6. Die Schritte 3 und 4 werden so oft wie nötig am Beginn jeder folgenden Seite wiederholt, gefolgt von 5 am Ende der Seite.

*`[h]` heißt
nicht „hier“* 

Manchmal können sich die Positionsargumente von $\text{\LaTeX}$ als zu unflexibel erweisen. Gleitobjekte sollen hin und wieder genau dort im Text erscheinen, wo sie auch in der Eingabedatei eingefügt wurden, das heißt sie sollen eigentlich gar nicht gleiten. Es ist ein weit verbreitetes Missverständnis, dass das Argument `[h]` „hier und sonst nirgendwo“ bedeutet. Tatsächlich hält das Argument $\text{\LaTeX}$ lediglich an, das Gleitobjekt *möglichst* an der aktuellen Position einzufügen. Wenn nicht mehr genug Platz auf der Seite ist, oder das Objekt aufgrund bestimmter Layoutparameter (siehe Abschnitt 6.1) nicht innerhalb des Textes positioniert werden darf, ignoriert $\text{\LaTeX}$ diese Anforderung und versucht, das Objekt in Übereinstimmung mit einem der weiteren angegebenen Positionsargumente zu platzieren. So bedeutet `[ht]` z.B., dass ein Gleitobjekt am Anfang einer späteren Seite erscheinen wird, wenn es nicht mehr auf die aktuelle Seite passt. Das kann ziemlich oft vorkommen, wenn die Gleitobjekte, die man mitten im Text positionieren möchte, vergleichsweise groß sind und daher auf der Seite nicht mehr genug Platz für sie ist. Durch Ignorieren von `h` und Ausprobieren der verbleibenden Positionsargumente verhindert $\text{\LaTeX}$ übermäßig leere Seiten, die ansonsten in solchen Fällen entstehen würden.

Manchmal nimmt man vielleicht tatsächlich große Lücken auf den Seiten in Kauf. Für diese Fälle stellt das Paket `float` die Option `[H]` zur Verfügung, welche bedeutet „genau hier, nirgendwo sonst“. Es wird in Abschnitt 6.3.1 auf Seite 300 beschrieben.

6.2.1 placeins – Gleitobjekte in Schranken

Donald Arseneau schrieb das Paket `placeins`, mit dem man Gleitobjekte mithilfe des Befehls `\FloatBarrier` daran hindern kann, über einen bestimmten Punkt im Ausgabedokument hinaus zu gleiten. Wenn man das `placeins`-Paket einsetzt und dieser Befehl auftaucht, werden alle aufgelaufenen Gleitobjekte unverzüglich ausgegeben. Dieses Verfahren ist nützlich, wenn man alle Objekte eines Abschnitts noch vor Beginn des nächsten Abschnitts positionieren möchte.

Man kann beispielsweise die entsprechenden Gliederungsbefehle umdefinieren, indem man den `\FloatBarrier`-Befehl folgendermaßen in ihren `@startsection`-Befehl (siehe Abschnitt 2.2.2) einbindet:

```
\makeatletter          % bei Einsatz in der Präambel erforderlich
\renewcommand\section{\@startsection
{section}{1}{0mm}%           name, ebene, einzug
{-\baselineskip}%          abstand-vor
{0.5\baselineskip}%        abstand-nach
{\FloatBarrier\normalfont\Large\bfseries}}% layout
\makeatother             % bei Einsatz in der Präambel erforderlich
```

Der Autor des `placeins`-Paketes ging davon aus, dass viele Anwender ihre Gleitobjekte vor Beginn des nächsten Abschnitts setzen wollen. Daher enthält das Paket die Option `section`, die automatisch `\FloatBarrier` zum `\section`-Befehl hinzufügt. Wird nur diese Option verwendet, erzwingt sie, dass alle Gleitobjekte des aktuellen Abschnitts *vor* jeglichem Material des nächsten Abschnitts gesetzt werden. `\FloatBarrier` verhindert nämlich, dass die entsprechenden Objekte nach dem Beginn des nächsten Abschnitts positioniert werden, selbst wenn noch Teile des vorigen Abschnitts auf der gleichen Seite erscheinen.

Mithilfe der Option `below` kann man zulassen, dass Gleitobjekte an `\FloatBarrier` vorbei an den Fuß der aktuellen Seite (d.h. in den neuen Abschnitt) gleiten. Durchlässigkeit in die entgegengesetzte Richtung (d.h. in den vorherigen Abschnitt) bewirkt die Option `above`.

*Begrenzungen
durchlässig machen*

Mit der Option `verbose` zeigt das Paket Verarbeitungsdaten auf dem Bildschirm und in der Protokolldatei an.

6.2.2 afterpage – Kontrolle am Seitenende

Das Paket `afterpage` (von David Carlisle) führt den Befehl `\afterpage` ein, dessen Argument erst ausgeführt wird, wenn die aktuelle Seite beendet ist. Auch wenn sein Autor es als „a hack that not even always works“¹ betrachtet (es versagt z.B. im Zweispaltensatz mit `twocolumn`), bietet es eine Reihe nützlicher Anwendungsmöglichkeiten.

Manchmal ist der $\text{\LaTeX}$ -Algorithmus zum Positionieren von Gleitobjekten überfordert, und alle gleitenden `figure`- und `table`-Umgebungen werden an das Kapitel- oder Dokumentenende verschoben. Mit einem `\clearpage`-Befehl kann man zwar alle bisher noch nicht positionierten Gleitobjekte ausge-

*Gleitobjekte am
Dokumentenende
vermeiden*

¹Siehe <http://jargon.watson-net.com/jargon.asp?w=hack> insbesondere Bedeutungen 1 & 2.

ben, allerdings mit dem Nebeneffekt, dass die aktuelle Seite vorzeitig beendet wird. Das `afterpage`-Paket ermöglicht stattdessen, `\afterpage{\clearpage}` zu verwenden. Dadurch wird die aktuelle Seite zunächst (wie üblich) mit Text gefüllt, und erst dann werden durch den Befehl `\clearpage` alle aufgelaufenen Gleitobjekte vor Beginn der nächsten Textseite positioniert.

Gleiten mehrseitiger Tabellen

Bei der mehrseitigen Umgebung `longtable` (siehe Abschnitt 5.4.2) können Probleme auftreten, wenn man den vorangehenden und nachfolgenden Text setzt. Für diesen Fall wird die Möglichkeit interessant, die Umgebung `longtable` „gleiten“ zu lassen. Da sich ihre Tabellen jedoch unter Umständen über mehrere Seiten erstrecken, kann es sein, dass es nicht möglich ist, sie im Speicher zu halten und genauso gleiten zu lassen wie eine `table`-Umgebung. Wenn die Tabelle jedoch in einer externen Datei (z.B. `lfile.tex`) gespeichert wird, kann man eine der folgenden Befehlszeilen verwenden:

```
\afterpage{\clearpage\input{lfile}}
\afterpage{\clearpage\input{lfile}\newpage}
```

Die erste Form lässt am Ende der `longtable`-Umgebung noch weiteren Text auf der gleichen Seite zu. Die zweite stellt sicher, dass der nachfolgende Text wieder auf einer neuen Seite beginnt.

Wie am Ende von Abschnitt 6.3.1 beschrieben, kann man den Befehl `\afterpage` auch zusammen mit dem Paket `float` und dem Positionsargument `[H]` verwenden.

6.2.3 endfloat – Gleitobjekte am Ende positionieren

Einige Zeitschriften erfordern, dass Tabellen und Abbildungen vom Text getrennt am Ende des Dokumentes gesammelt werden. Vielleicht soll ihnen noch ein Verzeichnis der Objekte vorangehen, und Markierungen im Text sollen anzeigen, wo sie ursprünglich positioniert waren. Dies lässt sich mithilfe des Paketes `endfloat` (von James Darrell McCauley und Jeffrey Goldberg) erreichen. Es setzt Abbildungen und Tabellen in getrennte Abschnitte mit den Titeln „Figures“ bzw. „Tables“ am Ende eines Artikels.

Das Paket enthält eine Reihe von Optionen, mit denen sich Abbildungs- und Tabellenverzeichnisse, ihre Überschriften und die Markierungen im Text steuern lassen. Es folgt eine Liste der verfügbaren Optionen:

- `figlist/nofiglist` Erzeugt (Voreinstellung) bzw. unterdrückt das Abbildungsverzeichnis.
- `tablist/notablist` Erzeugt (Voreinstellung) bzw. unterdrückt das Tabellenverzeichnis.
- `lists/nolists` Erzeugt bzw. unterdrückt beide Verzeichnisse (Kurzform für die Kombination der beiden vorhergehenden Optionsgruppen).
- `fighead/nofighead` Produziert bzw. unterdrückt (Voreinstellung) eine Überschrift vor den Abbildungen. Ihr Text wird in `\figuresection` festgelegt und ist auf die Zeichenfolge „Figures“ voreingestellt.
- `tabhead/notabhead` Produziert bzw. unterdrückt (Voreinstellung) eine Überschrift vor den Tabellen. Ihr Text wird in `\tablesection` festgelegt und ist auf die Zeichenfolge „Tables“ voreingestellt.

`heads/noheads` Erzeugt bzw. unterdrückt beide Überschriften (Kurzform für die Kombination der beiden vorhergehenden Optionsgruppen).

`markers/nomarkers` Setzt Markierungen (Voreinstellung) im Text oder lässt diese weg.

`figuresfirst/tablesfirst` Setzt alle Abbildungen vor den Tabellen (Voreinstellung) bzw. umgekehrt.

Das Paket enthält außerdem die Befehlsparameter `\AtBeginFigures`, `\AtBeginTables` und `\AtBeginDelayedFloats` als Schnittstelle zum Steuern der angesammelten Gleitobjekte. Beispielsweise sorgt die Anweisung `\AtBeginTables{\cleardoublepage}` dafür, dass die Ausgabe der aufgeschobenen Tabellen auf einer rechten Seite beginnt.

Befehlsparameter

Wenn die Gleitobjekte schließlich gesetzt werden, wird nach jedem der Objekte der Befehl `\efloatseparator` ausgeführt. Er besteht normalerweise aus dem Befehl `\clearpage`, so dass er ein Objekt je Seite erzwingt. Er kann mit `\renewcommand` umdefiniert werden.

Das Paket ist so voreingestellt, dass es die ursprüngliche Position eines Gleitobjektes im Text durch zusätzliche Zeilen wie „[Figure 4 about here.]“ ungefähr an der entsprechenden Stelle angibt. Das Setzen dieser Markierungen kann beim Laden des Paketes durch die Option `nomarkers` deaktiviert werden. Text und Format der Einträge werden mit den Befehlen `\figureplace` und `\tableplace` festgelegt, die sich über `\renewcommand` ändern lassen. So können sie z.B. an eine andere Sprache angepasst werden (das Paket unterstützt keine `babel`-Optionen). Eine Anpassung für die französische Sprache könnte folgendermaßen aussehen:

*Markierungen
im Text*

```
\renewcommand\figureplace
{\begin{center}[La figure~\thepostfig\ approx.\ ici.]\end{center}}
\renewcommand\tableplace
{\begin{center}[La table~\theposttbl\ approx.\ ici.]\end{center}}
```

Dabei verweisen `\thepostfig` und `\theposttbl` auf die aktuelle Abbildungs- bzw. Tabellenummer. Solche Definitionen können z.B. in die Konfigurationsdatei `endfloat.cfg` des Paketes eingebaut werden, die, sofern sie existiert, automatisch vom Paket geladen wird (mit allen daraus resultierenden Portabilitätsproblemen).

Normalerweise werden die aufgeschobenen Gleitobjekte am Ende des Dokumentes verarbeitet. Manchmal möchte man sie jedoch vielleicht bereits vorher ausgeben, beispielsweise am Ende jedes Kapitels. Durch den Befehl `\processdelayedfloats` gibt das Paket `endfloat` alle bis zu dem jeweiligen Punkt aufgeschobenen Gleitobjekte aus. Sie werden dabei weiterhin fortlaufend nummeriert. Wenn man die Nummerierung neu beginnen will, muss man die entsprechenden Zähler zurücksetzen. (Näheres dazu enthält die Dokumentation des Paketes.)

Vorgezogene Ausgabe

Zum Speichern der Abbildungen und Tabellen erzeugt das `endfloat`-Paket zwei zusätzliche Dateien mit den Erweiterungen `.fff` bzw. `.ttt`. Da der Inhalt der Umgebung wortwörtlich in diese Dateien geschrieben wird, muss der `\end`-Befehl (z.B. `\end{figure}`) im Quelldokument immer in einer eigenen Zeile (ohne Leerzeichen) stehen, ansonsten wird er nicht erkannt. Aus

 *Einschränkungen*

dem gleichen Grund werden auch die Namen der Standardumgebungen (d.h. `figure`, `table` und ihre Sternformen) nur erkannt, wenn sie direkt im Dokument erscheinen. Wenn sie in anderen Umgebungen verborgen sind, wird das Umgebungsende `\end` nicht erkannt.

Andere Gleitobjektumgebungen, wie etwa `sidewaysfigure` beziehungsweise `sidewaystable` aus dem `rotating`-Paket werden nicht automatisch unterstützt. Es ist jedoch möglich, das `endfloat`-Paket so zu erweitern, dass es auch solche Umgebungen erkennt. Dazu enthält die Softwaredistribution von `endfloat` die Beispieldatei `efxmpl.cfg`. Diese erweitert `endfloat` so, dass es auch die Umgebungen des `rotating`-Paketes unterstützt. Man muss die Datei in `endfloat.cfg` einbinden (hineinkopieren), damit der Code automatisch geladen wird.

6.3 Erweiterungen für L^AT_EXs Gleitobjektkonzept

Standard-L^AT_EX verfügt über zwei Klassen von Gleitobjektumgebungen: `figure` und `table`. Diese sind für viele Dokumente völlig ausreichend; manchmal werden jedoch weitere Klassen oder zusätzliche Funktionen benötigt. Dieser Abschnitt befasst sich mit Paketen, welche die Basiswerkzeuge für komplexere Erfordernisse erweitern.

Mit dem `float`-Paket lassen sich eigene Klassen von Gleitobjekten und die dazugehörigen Umgebungen definieren. Zudem liefert es eine Lösung für das Problem bei einzelnen Objekten jedwedes Gleiten zu verhindern. Mit einem anderen Ansatz wird dieses Problem auch durch das `caption`-Paket gelöst.

Mit den letzten beiden in diesem Abschnitt behandelten Paketen, `rotating` und `rotfloat`, lassen sich die Inhalte von Gleitobjekten drehen, was sich bei außergewöhnlich großen Objekten als notwendig erweisen kann.

6.3.1 float – Erstellen eigener Gleitobjektumgebungen

Das Paket `float` von Anselm Lingnau verbessert die L^AT_EX-Schnittstelle für die Definition von Gleitobjekten wie Abbildungen oder Tabellen. Es entwickelt das Konzept eines „Layouts für Gleitobjekte“, das die Darstellung von Gleitobjekten steuert. Neue Typen von Gleitobjekten können mit dem Befehl `\newfloat` definiert werden:

```
\newfloat{klasse}{position}{erw}[num-in]
```

Der Befehl `\newfloat` verfügt über drei obligatorische und ein optionales Argument mit folgender Bedeutung:

klasse Der Name für die neue Klasse von Gleitobjekten, wie zum Beispiel `program`. Eine `\newfloat`-Deklaration stellt die beiden Umgebungen `klasse` und `klasse*` zur Verfügung.

position Voreinstellung für die Platzierung der jeweiligen Klasse von Gleitobjekten (Kombination der L^AT_EX-Argumente `t`, `b`, `p` und `h` oder, alternativ dazu, `H`).

erw Dateierweiterung der Hilfsdatei, in der die Legenden der neuen Objektklasse gesammelt werden.

num-in Optionales Argument, das festlegt, ob die Gleitobjekte dieser Klasse innerhalb eines Gliederungselementes des Dokumentes nummeriert werden. Wenn z.B. *num-in* den Wert `chapter` enthält, werden die Gleitobjekte nach Kapiteln nummeriert (das entspricht dem L^AT_EX-Standard für Abbildungen und Tabellen der `report`- und der `book`-Klasse).

Die Deklaration `\floatstyle` richtet ein Layout ein, das für alle im Anschluss mit `\newfloat` definierten Gleitobjektklassen gilt, bis `\floatstyle` erneut verwendet wird. Als Argument erhält sie den Namen eines der folgenden vordefinierten Layouts:

Das Layout der Gleitobjektklasse

plain Ähnlich dem von Standard-L^AT_EX verwendeten Layout für Gleitobjekte, d.h. ohne spezielle Formatierungen. Der einzige Unterschied ist, dass die Legende immer direkt unter dem Gleitobjekt erscheint, egal, an welcher Stelle sie im Eingabetext steht.

plaintop Wie *plain*, nur dass die Legende oberhalb des Gleitobjektes erscheint.

boxed Das Gleitobjekt ist von einem Rahmen umgeben und die Legende steht unterhalb.

ruled Dieses Layout ist an das Tabellenformat des Buches *Concrete Mathematics* [61] angelehnt. Die Legende erscheint, von Linien umgeben, oberhalb des Gleitobjektes; eine weitere Linie begrenzt das Objekt nach unten.

Die Gleitobjektlayouts legen das generelle Erscheinungsbild der Gleitobjekte und die Formatierung der Legende fest. So setzt z.B. das Layout *ruled* die Legende linksbündig ohne Doppelpunkt, während sie bei anderen Layouts zentriert mit einem Doppelpunkt nach der Nummer erscheint. Wer mehrere Layouts in einem Dokument mischt, sollte deshalb sorgfältig darauf achten, keine typographischen Monstrositäten zu erzeugen. Da die Layouts auch die Platzierung der Legende festlegen, ist – im Gegensatz zu Standard-L^AT_EX – nur ein `\caption`-Befehl pro Gleitobjekt möglich.

 *Nur eine \caption pro Gleitobjekt*

Auch wenn das Paket keine direkte Benutzerschnittstelle zum Definieren eigener Layouts beinhaltet, ist es doch recht einfach, neue benannte Layouts hinzuzufügen. Eine genauere Erklärung befindet sich in der Dokumentation in `float.dtx`.

Das nächste Beispiel zeigt die Deklarationen für zwei neue, selbst entworfene Gleitobjektklassen: *Reihe* und *XMLexa*. Erstere werden nach Abschnitten nummeriert und sind "eingerahmt" und letztere werden unabhängig nummeriert und durch Linien abgesetzt (eine aus typographischer Sicht höchst fragwürdige Kombination).

Die automatisch generierte Bezeichnung, die L^AT_EX in den Legenden der jeweiligen Gleitobjektklasse verwendet, kann mithilfe der Deklaration `\floatname{klasse}{bezeichnung}` geändert werden. Im folgenden Beispiel wird für *XMLexa*-Objekte die Bezeichnung „XML-Listing“ gewählt. Soweit kein anderer Name definiert wird, verwendet der Befehl `\newfloat` sein Argument

Benennung der Gleitobjektklasse

klasse als Standardbezeichnung (wie im Beispiel bei der Gleitobjektumgebung Reihe zu sehen).

1 Neue Gleitobjektumgebungen

Text für unsere Seite, der immer und immer wieder verwendet wird.

XML Listing 1 Einfache XML-Datei

```
<XMLphrase>Viel Spaß!</XMLphrase>
```

Text für unsere Seite, der immer und immer wieder verwendet wird.

XML Listing 2 Verarbeitungsanweisung

```
<?xml version="1.0"?>
```

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

$$e = 1 + \sum_{k=1}^{\infty} \frac{1}{k!}$$

Reihe 1.1: Eulersche Konstante

```
\usepackage{float}
\floatstyle{boxed}
\newfloat{Reihe}{b}{los}[section]
\floatstyle{ruled}
\newfloat{XMLexa}{H}{lox}
\floatname{XMLexa}{XML Listing}
\newcommand\xmlcode[1]{\texttt{#1}}
\newcommand\sample{Text für unsere Seite,
  der immer und immer wieder verwendet wird. }
\section{Neue Gleitobjektumgebungen}
\sample
\begin{XMLexa} \caption{Einfache XML-Datei}
  \xmlcode{<XMLphrase>Viel Spaß!</XMLphrase>}
\end{XMLexa}
\sample
\begin{XMLexa}
  \caption{Verarbeitungsanweisung}
  \xmlcode{<?xml version="1.0"?>}
\end{XMLexa}
\sample
\begin{Reihe}
\caption{Eulersche Konstante}
  \[e = 1 + \sum^{\infty}_{k=1} \frac{1}{k!}\]
\end{Reihe}
\sample
```

Bsp.
6-3-1

Der Befehl `\listof{klasse}{titel}` erzeugt eine Liste aller Objekte der jeweiligen Klasse. Er entspricht den L^AT_EX-eigenen Befehlen `\listoffigures` und `\listoftables`. Das Argument *klasse* enthält den Namen der Objekt-klasse, der mit `\newfloat` festgelegt wurde. Das zweite Argument *titel* bestimmt die Überschrift für die Liste der Gleitobjekte, die aus den Legenden (d.h. aus den `\caption`-Befehlen) gebildet wird.

Das folgende Beispiel erweitert Beispiel 6-3-1 (die Quelle ist hier nur zum Teil wiedergegeben) um zwei `\listof`-Befehle.

Legenden einer Gleitobjektklasse auflisten

XML Listings

- 1

Einfache XML-Datei

1
- 2

Verarbeitungsanweisung . .

2

Liste der Serien

- 1.1

Eulersche Konstante

3

1 Neue Gleitobjektumgebungen

Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{float}
% Die Gleitobjektklassen "Reihe" und
% "XMLexa" und die Befehle \xmlcode und
% \sample aus dem letzten Beispiel
\listof{XMLexa}{XML Listings}
\listof{Reihe}{Liste der Serien}
\section{Neue Gleitobjektumgebungen}
\sample
\begin{XMLexa} \caption{Einfache XML-Datei}
  \xmlcode{<XMLphrase>Viel Spaß!</XMLphrase>}
\end{XMLexa}
... weitere Eingaben unterdrückt ...
```

Bsp.
6-3-2

Den L^AT_EX-Standardklassen `figure` und `table` kann mit `\newfloat` kein Layout zugewiesen werden, da diese schon existieren, wenn das `float`-Paket geladen wird. Deshalb stellt das Paket zusätzlich die Deklaration `\restylefloat{klasse}` zur Verfügung, mit der sich ein gültiges Layout für Gleitobjekte der jeweiligen *klasse* einstellen lässt (welches zuvor mit einem `\floatstyle`-Befehl festgelegt wurde).

Anpassen der
Gleitobjekt-
Standardklassen

Aus dem gleichen Grund gibt es auch die Deklaration `\floatplacement{klasse}{position}`, mit deren Hilfe man die Voreinstellung der möglichen Positionen für die jeweilige Gleitobjektklasse einstellen kann (zum Beispiel `\floatplacement{table}{tp}`). Im folgenden Beispiel wird die Funktionsweise dieser Deklarationen anhand von `figure` und `table` dargestellt, die hier (nicht unbedingt zu ihrem Vorteil) verändert werden.



Abb. 1: Beispiellabbildung

1 Anpassen der Standardklassen

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

Tabelle 1 Beispieltabelle

AAAA	BBBB	123
CCC	DDDD	45

Bsp.
6-3-3

```
\usepackage{graphicx,float}
\floatstyle{boxed} \restylefloat{figure}
\floatstyle{ruled} \restylefloat{table}
\floatplacement{table}{b}
% \sample wie zuvor definiert

\section{Anpassen der Standardklassen}
\sample
\begin{table}
\begin{tabular}{@{}llr}
AAAA&BBBB&123\end{tabular}
\caption{Beispieltabelle}
\end{table}
\sample
\begin{figure} \centering
\includegraphics[width=12mm]{cat.ps}
\caption{Beispiellabbildung}
\end{figure}
```

Dem `here`-Paket von David Carlisle nachempfunden, führt das Paket `float` die Positionsoption `[H]` ein, die besagt, „das Gleitobjekt soll unter allen Umständen `[H]`ier positioniert werden“. Es ist für alle Gleitobjektklassen, einschließlich der Standardumgebungen `figure` und `table`, verfügbar. Die Option `[H]` muss immer allein verwendet werden; eine Angabe wie `[Hbpt]` ist unzulässig.

Ein Gleitobjekt „hier“
positionieren

Wenn auf der aktuellen Seite nicht mehr genügend Platz ist, wird das Gleitobjekt am Anfang der nächsten Seite gedruckt. Dabei wird alles, was dem Objekt folgt, mit auf die nächste Seite verschoben, auch wenn auf der aktuellen Seite noch Platz gewesen wäre. Damit liegt es in der Verantwortung des Autors, die mit `H` positionierten Gleitobjekte so zu platzieren, dass keine großen Leerräume am Seitenende entstehen. Darüber hinaus muss man die Reihenfolge der Gleitobjekte überprüfen, wenn man sowohl Standard- als auch `[H]`-Parameter einsetzt. Ein mit `[t]` platziertes Gleitobjekt, das in der Eingabedatei dem mit `[H]` platzierten vorangeht, kann in der Ausgabe fälschlicherweise an Letzterem vorbei gleiten, so dass zum Beispiel Abbildung 3 nach Abbildung 4 erscheint.

Im folgenden Beispiel werden noch einmal alle Positionsoptionen zusammen dargestellt.

6

t Seitenkopf
b Seitenfuß
p Gleitobjekt-
kolumne
h Möglichst hier
H Immer hier

Tabelle 1: Gleitobjekt-
positionierung

Mit dem Wert „h“

7

```
\usepackage{float,array}
```

Im folgenden Beispiel werden noch einmal alle Positionsoptionen zusammen dargestellt.

```
\begin{table}[H]
\begin{tabular}>{\ttfamily}cl
t & Seitenkopf & \b & Seitenfuß \\
p & Gleitobjekt- & \\
& kolumne & \\
h & Möglichst hier & \H & Immer hier
\end{tabular}
\caption{Gleitobjektpositionierung}
\end{table}
```

Mit dem Wert "h" anstelle von "H" würde dieser Text im aktuellen Beispiel vor der Tabelle erscheinen.

Bsp.
6-3-4

In Kombination mit den Paketen `placeins` und `afterpage`, die in den Abschnitten 6.2.1 bzw. 6.2.2 beschrieben wurden, lassen sich die Gleitobjekte noch genauer positionieren. Manchmal, wenn man den Wert `H` verwendet, meint man z.B. gar nicht „genau an diesem Punkt“, sondern eher „hier in der Nähe“. Dieser Effekt lässt sich mit dem `\afterpage`-Befehl erzielen:

```
\afterpage{\FloatBarrier\begin{figure}[H]...\end{figure}}
```

Der Befehl `\FloatBarrier` sorgt dafür, dass zunächst alle aufgelaufenen Gleitobjekte an geeigneter Stelle positioniert werden (aufgrund von `\afterpage`, ohne große Lücken im Text zu erzeugen), wodurch das zuvor erwähnte Problem der Reihenfolge gelöst wäre. Unmittelbar danach wird das `[H]`-Gleitobjekt gesetzt. Wenn man anstelle von `\FloatBarrier` den Befehl `\clearpage` verwendet, erscheint es am Kopf der nächsten Seite.

6.3.2 caption – Nicht gleitende Abbildungen und Tabellen

Im vorigen Abschnitt wurde beschrieben, wie sich die verschiedenen Gleitumgebungen mithilfe der Option `[H]` festsetzen lassen. Alternativ dazu kann man normales Textmaterial mit Legenden versehen, die genauso gesetzt und in Abbildungs- oder Tabellenverzeichnisse aufgenommen werden, wie die $\text{\LaTeX}$ -Standardumgebungen `figure` und `table`. Eine solche Funktionalität stellt das Paket `caption` zur Verfügung, dessen andere Eigenschaften ausführlich in Abschnitt 6.5.1 besprochen werden.

```
\captionof{klasse}[kurztext]{text} \captionof*{klasse}{text}
```

Dieser Befehl verhält sich analog zu dem $\text{\LaTeX}$ -Befehl `\caption`, verfügt jedoch über ein zusätzliches obligatorisches Argument *klasse* für die Gleitobjektklasse, die er imitieren soll. Er kann für jede Art von nicht gleitendem Material eingesetzt werden, das eine (nummerierte) Legende erhalten und im Abbildungs- oder Tabellenverzeichnis erscheinen soll. Die Sternform unterdrückt sowohl die Nummerierung als auch den „...verzeichnis“-Eintrag.

Das folgende Beispiel zeigt eine normale Abbildung zusammen mit ihrer nicht gleitenden Variante. In einem solchen Fall kann es vorkommen, dass die gleitende Abbildung an der fest positionierten vorbei gleitet. Zur Verdeutlichung dieser Situation wird die gleitende Abbildung im Beispiel an den Fuß der Seite gedrängt. Dadurch gerät die Nummerierung durcheinander. Wenn man gleitende und nicht gleitende Objekte gemeinsam verwendet, muss man dieses Problem im Auge behalten.

 **Vorsicht**
vor falscher
Nummerierung

Abbildungen

2	Spezieller LOF Eintrag . . .	1
1	Standardabbildung	1

1 Gleitende und nicht gleitende Abbildungen

In diesem Abschnitt benutzen wir gleitende und nicht gleitende Objekte.

Abbildung II

Abb. 2: Nicht gleitende Abbildung

Abbildung I

Abb. 1: Standardabbildung

Bsp.
6-3-5

```
\usepackage{caption}
\listoffigures

\section{Gleitende und nicht gleitende
        Abbildungen}
In diesem Abschnitt benutzen wir gleitende
und nicht gleitende Objekte.
\begin{figure}[b]
\centering
\fbbox{Abbildung I}
\caption{Standardabbildung} \label{fig:I}
\end{figure}
\begin{center}
\fbbox{Abbildung II} \\
\captionof{figure}
        {Spezieller LOF Eintrag}
        {Nicht gleitende Abbildung}
\label{fig:II}
\end{center}
Abb.~\ref{fig:I} wird durch das optionale
Argument \texttt{[b]} nach unten gedrängt;
sie gleitet an Abb.~\ref{fig:II} vorbei und
die Nummerierung gerät durcheinander.
```

6.3.3 rotating – Drehen von Gleitobjekten

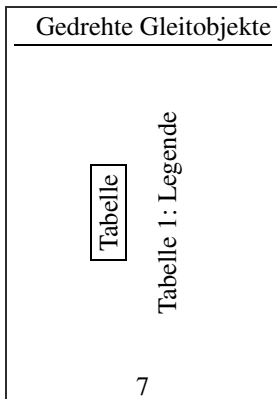
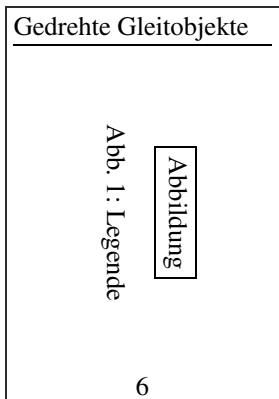
Manchmal möchte man den Inhalt eines Gleitobjektes um 90 oder 270 Grad drehen. Für diesen Vorgang benötigt T_EX die Unterstützung eines Ausgabebetreibers. Um so geräteunabhängig wie möglich zu sein, verpackt L^AT_EX die erforderlichen Abläufe in den Paketen `graphics` und `graphicx` (siehe Abschnitt 10.2). Eines der frühesten Pakete, das auf dieser Schnittstelle aufsetzte, war das Paket `rotating` von Sebastian Rahtz und Leonor Barroca.¹

Das `rotating`-Paket verfügt über die Umgebungen `sidewaysfigure` und `sidewaystable`, mit denen sich ganze Gleitobjekte drehen lassen. Diese Umgebungen erzeugen automatisch seitengroße, oder genauer gesagt spaltengroße Gleitobjekte (wenn sie im `twocolumn`-Modus eingesetzt werden). Die ebenfalls vorhandenen Sternformen der Umgebungen erstrecken sich auch im Zweispaltenmodus über beide Spalten.

Wie im Beispiel dargestellt, werden die Gleitobjekte normalerweise so gedreht, dass sie vom äußeren Rand lesbar sind. Wenn sie immer in die gleiche

¹Eigentlich wurde es sogar noch vor der `graphics`-Schnittstelle veröffentlicht, und erst nachträglich zu einer Erweiterung dieser Schnittstelle überarbeitet.

Richtung gedreht werden sollen, kann man die Drehrichtung über die Optionen `figuresright` bzw. `figuresleft` festlegen.

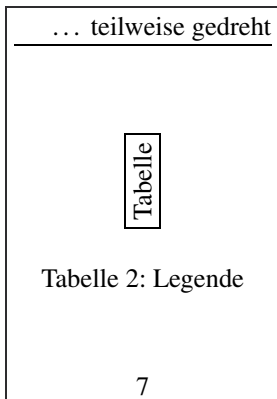
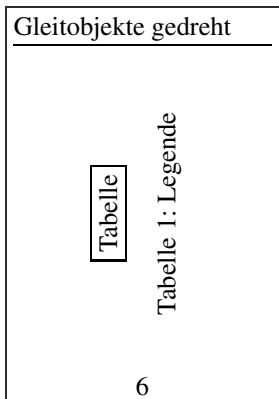


```
\usepackage{rotating}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[RO,LE]{Gedrehte
    Gleitobjekte}

\begin{sidewaysfigure}
  \centering \fbox{Abbildung}
  \caption{Legende}
\end{sidewaysfigure}
\begin{sidewaystable}
  \centering \fbox{Tabelle}
  \caption{Legende}
\end{sidewaystable}
```

Bsp.
6-3-6

Das Paket definiert auch eine Reihe von Umgebungen wie `turn` oder `rotate` zum Drehen beliebiger Objekte; diese sind in Abschnitt 10.3.4 beschrieben. Für Gleitobjekte ist insbesondere die `sideways`-Umgebung interessant, die es ermöglicht, den Inhalt des Objektes zu drehen, aber die Legende unverändert zu belassen. Sie wird im folgenden Beispiel eingesetzt, das auch die Ergebnisse der Option `figuresright` veranschaulicht (die sich ungeachtet ihres Namens sowohl auf `sidewaysfigure` als auch auf `sidewaystable` auswirkt).



```
\usepackage[figuresright]{rotating}
\usepackage{fancyhdr}
\pagestyle{fancy}
\fancyhead[LE]{Gleitobjekte gedreht}
\fancyhead[RO]{\ldots\ teilweise
    gedreht}

\begin{sidewaystable} \centering
  \fbox{Tabelle} \caption{Legende}
\end{sidewaystable}
\begin{table} \centering
  \begin{sideways}
    \fbox{Tabelle}
  \end{sideways}
  \caption{Legende}
\end{table}
```

Bsp.
6-3-7

Statt das gesamte Gleitobjekt oder den Inhalt zu drehen, ist es manchmal günstiger, nur die Legende zu drehen. Diese Funktion unterstützt das `rotating`-Paket durch den Befehl `\rotcaption`. Leider ist das Layout, dessen dieser Befehl sich bedient, fest eingestellt; es kann jedoch über das `caption`-Paket, das in Abschnitt 6.5.1 besprochen wird, angepasst werden.

6.3.4 rotfloat – float und rotating kombinieren

Nach Laden des Paketes `rotfloat` von Axel Sommerfeldt kann man die Gleitobjektlayouts des `float`-Paketes auch für die Umgebungen `sidewaysfigure`

und `sidewaystable` des `rotating`-Paketes benutzen. Zudem werden mit `\newfloat` erzeugte Gleitobjektklassen um Umgebungen ergänzt, die um 90 oder 270 Grad gedreht erscheinen.

Das Paket `rotfloat` verfügt über die gleichen Optionen wie das `rotating`-Paket. Intern erzeugt es für jede Gleitobjektklasse eine zusätzliche Umgebung mit dem Namen `sidewaysklasse` und ihrer entsprechenden Sternform. Durch die folgende Definition:

```
\newfloat{XMLexa}{tbp}{lox} \floatname{XMLexa}{XML Listing}
```

erhält man vier Umgebungen: `XMLexa`, `XMLexa*`, `sidewaysXMLexa` und `sidewaysXMLexa*`. Ebenso gestalten die Befehle zum Umdefinieren der Umgebungen `table` und `figure`, wie z.B.

```
\floatstyle{boxed} \restylefloat{table}
```

nicht nur die Umgebungen `table` und `table*` neu, sondern dann auch `sidewaystable` und `sidewaystable*`.

6.4 Im Fließtext eingebettete Gleitobjekte

Im Satzmodell von $\text{\TeX}$ werden Texte zunächst auf einer vertikal ausgerichteten Druckfahne in Absätze umbrochen. Sobald auf diese Weise genügend Material zusammengestellt wurde, ruft $\text{\TeX}$ seine Ausgaberoutine auf, die den ersten Teil der Druckfahne abschneidet, mit Kolumnentiteln versieht und das Ergebnis in die `.dvi`-Datei ausgibt. Dann beginnt der Prozess wieder von vorne, mit dem Sammeln von Textmaterial, Umbrechen in Absätze und Befüllen der Druckfahne.

Durch dieses Verarbeitungsverfahren kann man relativ einfach einen Mechanismus zur Positionierung von solchen Gleitobjekten aufsetzen, die sich über die gesamte Seitenbreite oder zumindest eine Spaltenbreite erstrecken. Leider ist es jedoch fast unmöglich, Gleitobjekte so zu setzen, dass sie nur einen Teil einer Textspalte ausfüllen, wobei der Text um sie herum fließt. Das liegt daran, dass ihre letztendliche Position zu dem Zeitpunkt, an dem die Absätze in Zeilen umbrochen werden, noch nicht bekannt ist. Damit ist es unmöglich, beim Umbruch der Absätze Lücken für die Gleitobjekte zu lassen, da ihre tatsächliche Position ja erst in einem späteren Arbeitsschritt festgelegt wird. Wenn man Gleitobjekte hingegen an den Kopf oder Fuß einer Seite (oder Spalte) setzt, muss die Ausgaberoutine lediglich weniger Material von der zusammengestellten Druckfahne übernehmen, ohne den Inhalt selbst zu verändern.

Aufgrund dieses Verfahrens muss die Positionierung von im Fließtext eingebetteten Gleitobjekten bereits während des Absatzumbruchs erfolgen. Das beste Ergebnis, das Pakete zurzeit erzielen können, ist die Gewähr, dass eingebettete Gleitobjekte nicht von der Seite fallen, d.h. wegen Platzmangels in den Fuß der Seite ragen. Dazu messen sie die Menge an bereits angesammeltem Material auf der Druckfahne, um zu prüfen, ob noch genügend Raum verbleibt, um das Gleitobjekt und die umgebenden Absätze einzupassen.

Ein solcher Algorithmus kommt z.B. in dem Paket `wrapfig` zum Einsatz. Da die eingebetteten Gleitobjekte dieses Paketes im Vergleich zu den Standardobjekten nur sehr begrenzt „gleiten“, kann die Vermischung beider Varianten die Nummerierung durcheinander bringen.¹ Die meisten Pakete überlassen die Positionierung völlig dem Anwender, da automatische Lösungen so häufig zu Fehlern führen, dass es sich gar nicht erst lohnt sie einzubauen.

In diesem Buch werden, stellvertretend für das Angebot in diesem Bereich, insgesamt drei Pakete vorgestellt. Von diesen dreien wurde das Paket `picinpar` bereits in Abschnitt 3.1.14 besprochen; hier folgen nun die beiden anderen. Das Paket `wrapfig` unterstützt Abbildungen und Tabellen und leistet etwas Hilfestellung bei der automatischen Positionierung. Mit dem Paket `picins` lassen sich Abbildungen sehr genau im Fließtext einbetten, so dass es für diesen Zweck ziemlich interessant sein kann. Anders als andere Pakete in diesem Bereich unterstützt es jedoch keine Tabellen.

Alle diese Pakete haben auch Schwachstellen, so dass es lohnenswert sein kann, noch weitere Möglichkeiten zu erkunden, wie sie sich zum Beispiel mit dem Paket `floatflt` von Mats Dahlgren (einer Erweiterung des `floatfig`-Paketes von Thomas Kneser) bieten, das mit dem `multicol`-Paket zusammenarbeitet. Ein guter Startpunkt für die Recherche nach weiteren Paketen ist der *T_EX-Online-Katalog* [171] von Graham Williams bzw. der *T_EX Topic Index* [48] von Jürgen Fenn.

6.4.1 `wrapfig` – Text um ein Bild fließen lassen

Das Paket `wrapfig` (von Donald Arseneau) definiert die Umgebungen `wrapfigure` und `wraptable`. Mithilfe dieser Umgebungen kann man ein schmales Gleitobjekt an den Textrand setzen und dann den Text um es herum fließen lassen. Beide erzeugen Legenden mit dem entsprechenden Standardlayout für Abbildungen und Tabellen. Auch wenn die Umgebungen über eine begrenzte „Gleitfähigkeit“ verfügen, lassen sie sich nicht mit anderen Gleitobjekten synchronisieren. Man muss also im Auge behalten, dass sie möglicherweise in falscher Reihenfolge zu den Standardgleitobjekten ausgegeben werden.

```
\begin{wrapfigure}[nzeilen]{position}[überhang]{breite}
```

Die Umgebungen `wrapfigure` und `wraptable` verfügen über zwei obligatorische und zwei optionale Argumente folgender Bedeutung:

nzeilen (optional) Die Anzahl schmaler Zeilen, die für das Gleitobjekt benötigt werden (normalerweise automatisch berechnet). Abgesetzte Formeln im Fließtext werden dabei als drei Textzeilen gezählt

position Horizontale Position des Objektes; wird durch *einen* der folgenden Buchstaben festgelegt: `r` oder `R` (rechte Seite des Textes) und `l` oder `L` (linke Seite des Textes). Es gibt keine Option zum Zentrieren des Gleitobjektes. Bei einem doppelseitig formatierten Dokument kann die Position auch alternativ mit `i` oder `I` (innerer Textrand) und `o` oder `O`

¹Theoretisch ließe es sich wohl noch besser machen, so dass beide Arten einwandfrei synchronisiert werden; programmiertechnisch wäre das jedoch ziemlich schwierig.

(äußerer Textrand) festgelegt werden. Diese Angaben beziehen sich auf den Bund- bzw. Außensteg der ganzen Seite und nicht einer einzelnen Spalte. Der Großbuchstabe erlaubt es der Abbildung oder Tabelle jeweils zu gleiten, während die kleingeschriebene Variante die Objekte „genau hier“ positioniert.

überhang (optional) Überhang des Gleitobjektes in den Rand (Standard 0pt).

breite Breite der Abbildung oder Tabelle. Ein Wert von 0pt bedeutet hier, dass die „natürliche Breite“ als die zu umfließende Breite gewählt wird. Die Legende wird dann in dieser Breite gesetzt. Wenn die Abbildung breiter ist als der ihr zugewiesene Freiraum, wird eine „Overfull Box“ (überlaufende Box) erzeugt, die dazu führen kann, dass die Abbildung oder Tabelle den sie umgebenden Text überschreibt.

L^AT_EX lässt den Text um die Abbildung oder Tabelle herumfließen, wobei oberhalb und unterhalb ein Freiraum der Höhe `\intextsep` und an der Seite ein Freiraum der Breite `\columnsep` verbleibt; dementsprechend wird eine Reihe verkürzter Textzeilen neben der Abbildung erzeugt. Die Größe der Lücke im Text ist die Breite des Gleitobjektes plus `\columnsep` minus dem Wert des Argumentes *überhang*.

L^AT_EX errechnet die Anzahl der erforderlichen verkürzten Zeilen anhand der Höhe der Abbildung und der Länge `\intextsep`. Dieser Näherungswert kann durch das erste optionale Argument *nzeilen* überschrieben werden, das die gewünschte Zahl der verkürzten Zeilen angibt. Dieses Vorgehen kann sich als nützlich erweisen, wenn der umlaufende Text zusätzliche vertikale Abstände enthält, die nicht automatisch berücksichtigt werden.

Das erste Beispiel zeigt eine eingebettete Tabelle von 4cm Breite, die auf der linken Seite des Absatzes gesetzt wird. Das Paket errechnete fünf umfließende Textzeilen, wodurch unterhalb der Legende ein großer Freiraum verblieben wäre. Daher wurden stattdessen explizit 4 Zeilen festgelegt. Der Verweis auf die Abbildung erfolgt mit den L^AT_EX-Standardbefehlen `\label` und `\ref`.

Eingebettete Tabelle

Tabelle 1: Die Legende

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird. Verweis auf Tabelle 1.

Text für unsere Seite, der immer und immer wieder verwendet

```
\usepackage{wrapfig}
% \sample wie zuvor
\begin{wraptable}[4]{1}{4cm}
\centering\fbbox{Eingebettete Tabelle}
\caption{Die Legende}\label{T}
\end{wraptable}
\sample \sample
\sample Verweis auf Tabelle~\ref{T}.
```

Bsp.
6-4-1

Die Umgebungen `wrapfigure` und `wraptable` sollten nicht innerhalb anderer Umgebungen (wie z.B. `list`) verwendet werden. Im zweispaltigen Seitenlayout (`twocolumn`) funktionieren sie nicht (wenn eine Spalte überhaupt breit genug für eingebettete Gleitobjekte ist).

Im Allgemeinen kann L^AT_EX diese Umgebungen nicht an ihre optimale Position verschieben, also bleibt es dem Anwender überlassen, sie so gut wie möglich zu platzieren. Sie sollten dazu erst kurz vor dem *endgültigen Ausdrucken* eingebunden werden, da jede Änderung des Dokumentes ihre sorgfältige

Positionierung wieder ruinieren kann. Mit der Option `verbose` werden die Verarbeitungsdaten von `wrapfig` in die Protokolldatei geschrieben. Hier folgen einige Regeln für eine günstige Positionierung:

- Die Umgebungen sollten nicht über einen Seitenumbruch hinweg verlaufen und nicht an besonderen Stellen, wie zum Beispiel in Listen, positioniert werden.
- Neben der Abbildung sollte nur normaler Text und keine Überschrift oder größere Formel stehen. Kleinere, nicht abgesetzte Formeln sind meistens akzeptabel.
- Es ist am einfachsten den Befehl `\begin{wrapfigure}` oder `\begin{wraptable}` direkt nach einem Absatz einzufügen. Wenn man in der Mitte eines Absatzes beginnen will, muss man den Befehl genau zwischen zwei Worten einfügen, die durch einen Zeilenumbruch voneinander getrennt sind; siehe nächstes Beispiel.

Das zweite Beispiel zeigt eine Abbildung, die in ihrer natürlichen Breite (letztes Argument `Opt`) gesetzt wird, dabei jedoch 20% in den linken Rand hineinragt (festgelegt durch das optionale Argument). Anstelle der speziellen Einheit `\width`, die hier für die natürliche Breite des Gleitobjektes steht, kann man natürlich auch eine explizite Länge, wie z.B. 30pt angeben. Der Effekt dieser Einstellungen lässt sich deutlich daran erkennen, wie der Absatz unterhalb des Bildes gesetzt wird, wo der umfließende Text endet. Wie das Beispiel außerdem zeigt, kann, falls erforderlich, auch mehr als ein Absatz das Objekt umfließen.

Die Formatierung der Legende lässt sich durch die Kombination von `wrapfig` mit Paketen wie `caption` beeinflussen, auch wenn eine Option wie `centerlast` in beengten Verhältnissen nicht gerade die richtige Wahl ist.

Der Beginn der `wrapfigure`-Umgebung im aktuellen Beispiel wurde explizit festgelegt, indem der Text zunächst

Anstelle eines Bildes.

Abb. 1: Ein Beispiel für die `wrapfigure`-Umgebung

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

ohne Abbildung gesetzt wurde um die Zeilenumbrüche zu ermitteln.

Text für unsere Seite, der immer und immer wieder

```
\usepackage{wrapfig}
\usepackage[labelfont={sf,bf},
justification=centerlast]{caption}
% \sample wie zuvor
Der Beginn der wrapfigure-Umgebung im
aktuellen Beispiel wurde explizit
festgelegt, indem der Text zunächst
\begin{wrapfigure}[7]{l}[0.2\width]{0pt}
\centering
\fbbox{Anstelle eines Bildes.}
\caption{Ein Beispiel für die
\texttt{wrapfigure}-Umgebung}
\end{wrapfigure}
ohne Abbildung gesetzt
wurde um die Zeilenumbrüche zu ermitteln.
\par \sample \sample
```

Bsp.
6-4-2

Im vorigen Beispiel wurde explizit ein Wert für das Argument *überhang* angegeben. Man kann den Überhang auch global für alle `wrapfig`-Umgebungen festlegen, indem man über den $\text{\LaTeX}$ -Befehl `\setlength` für die Länge `\wrapoverhang` einen anderen Wert einstellt als null. Der folgende Code be-

wirkt zum Beispiel, dass alle eingebetteten Abbildungen und Tabellen die Marginalspalte mitnutzen können:

```
\setlength \wrapoverhang{\marginparwidth}
\addtolength\wrapoverhang{\marginparsep}
```

Mithilfe der Umgebung `wrapfloat` lassen sich leicht neue „eingebettete“ Umgebungen für zusätzliche Gleitobjektklassen (wie sie mit dem `float`-Paket definiert werden) hinzufügen oder direkt aufrufen. Diese besitzen dann die gleiche Schnittstelle und verhalten sich genauso wie `wrapfigure` oder `wratable`.

```
\newfloat{XML}{tbp}{lox}
\newenvironment{wrapXML}{\begin{wrapfloat}{XML}}{\end{wrapfloat}}
```

Weitere Möglichkeiten, das Verhalten des `wrapfig`-Paketes genauer einzustellen, werden in den Anmerkungen zur Implementierung am Ende der `wrapfig.sty`-Paketdatei erläutert.

6.4.2 picins – Bilder im Fließtext positionieren

Das Paket `picins` von Joachim Bleser und Edmund Lang definiert den `\parpic`-Befehl, mit dessen Hilfe man ein „Bild“ so auf der linken oder rechten Seite eines oder mehrerer Absätze positionieren kann, dass der Text das Bild umfließt.

`\parpic (b,h) (x-o,y-o) [opt] [pos] {bild}`

b,h (optional) Breite und Höhe des Bildes. Die Textzeilen, die um das Bild fließen, werden in einen Absatz gesetzt, dessen Zeilen um den Wert *b* kürzer sind als die Textbreite. Anhand der Höhe *h* wird die Anzahl der so umlaufenden Zeilen errechnet.

Wenn das Argument nicht angegeben wird, verwendet $\text{\LaTeX}$ die tatsächliche Größe des Bildes (die „Bounding Box“), sofern es diese berechnen kann. Andernfalls kommt es zu einer Fehlermeldung.

x-o,y-o (optional) Die Werte, um welche das Bild entlang der *x*- und *y*-Achse gegenüber der oberen linken Ecke der Bounding Box versetzt ist; ein positiver Wert für *x-o* bewirkt eine Verschiebung nach rechts, ein positiver Wert für *y-o* eine Verschiebung nach unten. Ohne dieses Argument wird das Bild mithilfe der *pos*-Angaben positioniert.

opt (optional) Die Charakteristika für die Positionierung und den Rahmen des Bildes; sie werden paarweise als *ein* Positions- und *ein* Rahmenargument angegeben.

Die *Positions*argumente sind *l* (*links*) für ein Bild auf der linken Seite des Absatzes und *r* (*rechts*) für ein Bild auf der rechten Seite.

Die Argumente für den Rahmen um das Bild sind: *d* (*dash*) für eine gestrichelte Rahmenlinie, *f* (*frame*) für durchgezogene Linien, *o* (*oval*) für einen Rahmen mit abgerundeten Ecken, *s* (*shadow*) für einen schattierten

Rahmen, sowie *x* (*box*) für einen „dreidimensionalen“ Kasten. Wenn keine Option festgelegt ist, wird das Bild auf der linken Seite des Absatzes platziert.

pos (optional) Position des Bildes innerhalb seines Rahmens; wird als ein horizontales oder ein vertikales, oder paarweise als horizontales und vertikales Argument angegeben. Mögliche *horizontale* Argumente sind *l* (*links*) für die linke Seite des Rahmens und *r* (*rechts*) für die rechte Seite. Wenn keine horizontale Angabe erfolgt, wird das Bild in seinem Rahmen auf dieser Achse zentriert.

Mögliche *vertikale* Argumente sind *t* (*top*) für die Oberseite des Rahmens und *b* (*bottom*) für die Unterseite. Wenn keine vertikale Angabe erfolgt, wird das Bild in seinem Rahmen auf dieser Achse zentriert.

Wenn ein Versatz-Argument *x-o,y-o* existiert, wird das Positionsargument *pos* ignoriert.

bild Befehle, die das Bild generieren, oder z.B. eine Graphik laden. Hier kann jedes beliebige L^AT_EX-Konstrukt eingesetzt werden.

Das nächste Beispiel zeigt verschiedene Arten, wie man ein Bild in einen Absatz einbetten kann. Außerdem werden einige weitere Befehle des *picins*-Paketes vorgestellt, mit deren Hilfe sich die Positionierung und Ausgestaltung des gesetzten Bildes noch feiner steuern lässt.

Im ersten Beispiel bleiben die Standardeinstellungen von *picins* unverändert, d.h. Höhe und Breite des Inhaltes werden automatisch berechnet. In diesem Fall wird das „Bild“ auf der linken Seite des Absatzes positioniert. Der Absatz ist normal eingerückt; das lässt sich, falls gewünscht, vermeiden, indem man ihm `\noindent` voranstellt. Im zweiten Teil des Beispiels wird ein EPS-Bild (Encapsulated PostScript) geladen, um das der Text herumfließt. In diesem Falle werden die Abmessungen des Bildes aus dem *BoundingBox*-Eintrag in der EPS-Datei gelesen. Für eine klarere Darstellung wurde ein gestrichelter Rahmen hinzugefügt.

BOX

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.



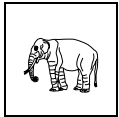
Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{picins,graphicx}
\newcommand\sample{Text für unsere Seite,
  der immer und immer wieder verwendet wird. }
\newcommand\FIG{\includegraphics
  [width=14mm]{cat}}
\parpic{\fbox{\Large\scshape Box}}
\sample\sample\par
\parpic[d]{\FIG}
\noindent\sample\sample
```

Bsp.
6-4-3

Man kann die Maße des Bildes selbst festlegen, damit L^AT_EX diese Werte für seine Formatierungsberechnungen heranzieht und nicht versucht, die Informationen aus der Quelldatei auszulesen. Wenn keine Versatz- oder Positionsargumente angegeben sind, wird der Inhalt zentriert (erstes Bild). Beim zweiten Bild ist der Inhalt um 2mm nach rechts und 14mm nach unten verschoben. Das Argument „*dr*“ erzeugt einen gestrichelten Rahmen und setzt das Bild nach rechts.

Der Befehl `\picskip{nzeilen}` veranlasst $\text{\LaTeX}$ den Absatz für *nzeilen* Zeilen mit dem vorgegebenen Einzug weiterzuführen (als würde das Bild sich um so viele Zeilen weiter nach unten erstrecken). Wenn *nzeilen* null ist, werden die nachfolgenden Zeilen nicht mehr eingezogen und ein neuer Absatz beginnt. Der horizontale Abstand zwischen Text und Bild lässt sich mit dem Befehl `\pichskip` steuern. Festlegen des Freiraums

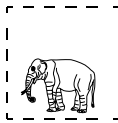


Text für unsere Seite, der immer und immer wieder verwendet wird.

Hier beweisen wir, dass sich das „Bild“ über mehr als einen Absatz erstrecken kann.

Text für unsere Seite, der immer und immer wieder verwendet wird.

Ohne die ausdrückliche Anweisung im Quellcode hätte dieser Absatz auch nur zwei verkürzte Zeilen, genau wie derjenige, der das vorige „Bild“ umfließt.

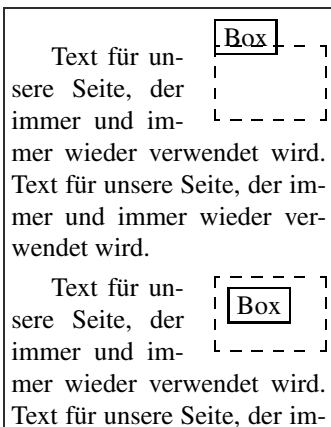


```
\usepackage{picins,graphicx}
\newcommand\FIG{\includegraphics
[width=10mm]{elephant}}

% \sample wie zuvor definiert
\parpic(15mm,15mm)[f]{\FIG}
\sample\par
Hier beweisen wir, dass sich das "Bild"
über mehr als einen Absatz erstrecken kann.
\parpic(15mm,15mm)(2mm,14mm)[dr]{\FIG}
\sample\par
\picskip{3}
Ohne die ausdrückliche Anweisung im
Quellcode hätte dieser Absatz auch nur
zwei verkürzte Zeilen, genau wie
derjenige, der das vorige "Bild"
umfließt.
```

Bsp.
6-4-4

Das Ergebnis des Versatzes von „2mm, 14mm“ im vorigen Beispiel war vielleicht etwas überraschend. Daher befasst sich das nächste Beispiel näher mit diesen Auswirkungen. Bei einem Versatz von 0mm, 0mm wird das „Bild“ so gesetzt, dass sich sein Bezugspunkt in der linken oberen Ecke des reservierten Bereiches befindet. Da die meisten $\text{\LaTeX}$ -Konstruktionen eine Box mit einem Referenzpunkt links unten an der Grundlinie erzeugen, wird das „Bild“ tatsächlich außerhalb des beabsichtigten Bereiches positioniert – also an einer ganz anderen Stelle, als ohne jeden Versatz.



mer und immer wieder verwendet wird.

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{picins}
% \sample wie zuvor definiert
\parpic(15mm,10mm)(0mm,0mm)
[dr]{\fbox{Box}}%
\sample\sample\par
\parpic(15mm,10mm)(2mm,5mm)
[dr]{\fbox{Box}}%
\sample\sample\par
\parpic(15mm,10mm)(4mm,10mm)
[dr]{\fbox{Box}}%
\sample\sample\par
\parpic(15mm,10mm)(6mm,15mm)
[dr]{\fbox{Box}}%
\sample\sample\par
```

Bsp.
6-4-5

Der Befehl `\parpic` kann innerhalb von Listenumgebungen in jeder beliebigen Verschachtelungstiefe eingesetzt werden. Damit liegt hier ein Unterschied zu anderen Paketen, die oft nur eine eingeschränkte Positionierung

von Bildern in Listen erlauben. Das nächste Beispiel zeigt eine `itemize`-Liste mit eingebetteten `\parpic`-Befehlen. Es demonstriert auch, wie sich die Liniestärke (`\linethickness`), die Länge der Striche (`\dashlength`), die Schattentiefe (`\shadowthickness`) und der 3D-Effekt (`\boxlength`) einzeln steuern lassen.

Text für unsere Seite, der immer und immer wieder verwendet wird.

- Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

- Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

- Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{picins}
% \sample wie zuvor definiert
\sample
\begin{itemize} \item
  \dashlength{2mm}
  \linethickness{1mm}
  \parpic(15mm,10mm)[dr]{BOX}
  \sample\sample
\item
  \shadowthickness{3mm}
  \linethickness{.4pt}
  \parpic(15mm,10mm)[sr]{BOX}
  \sample\sample
\item
  \boxlength{2mm}
  \parpic(15mm,10mm)[x]{BOX}
  \sample\sample
\end{itemize} \sample
```

Bsp.
6-4-6

Man kann nummerierte Legenden erzeugen, die im Abbildungsverzeichnis von $\text{\LaTeX}$ erscheinen. Da die Bilder nicht gleiten, muss man bei einer Vermischung mit gleitenden Objekten darauf achten, dass die Nummerierung nicht durcheinander gerät. Der Legendentext wird mithilfe des Befehls `\piccaption` festgelegt. Er verfügt über die gleichen Argumente wie der Standardbefehl `\caption`, wobei er die Legende aber nur für den *nächsten* `\parpic`-Befehl speichert.

Im ersten Beispiel wird der Inhalt eines Bildes in einer gerahmten, schattierten Box gesetzt, und seine Legende erscheint außerhalb des Rahmens unter dem Bild. Das entspricht der Standardposition für Bildunterschriften. Zwischen Bild und Text liegt ein Abstand von 6mm, der mit dem `\pichskip`-Befehl eingestellt wurde.

$$E = mc^2$$

Abb. 1: Einsteins Formel.

Text für unsere Seite, der immer und immer wieder verwendet wird.

wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{picins}
\newcommand\FOR{\(\displaystyle E=mc^2\)}
% \sample wie zuvor
\pichskip{6mm}
\piccaption{Einsteins Formel.}
\parpic(40mm,10mm)[s]{\FOR}
\sample\sample
```

Bsp.
6-4-7

Die Standardpositionierung der Legende kann mit der Deklaration `\piccaptionoutside` explizit festgelegt werden. Das Paket bietet außerdem die Wahl zwischen drei weiteren Positionsoptionen: `\piccaptioninside`, `\piccaptionside` und `\piccaptiontopside`. Ihre Auswirkungen werden

im nächsten Beispiel deutlich. Auch wenn `picins` den Text der Legende über einen eigenen Befehl festlegt, kann man ihre Formatierung durch Laden eines Paketes wie `caption` ändern. Als Beweis wird der Legendentext in fetter, serifenloser Schrift gesetzt.

$$E = mc^2$$

Abb. 1: Einsteins Formel.

Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\usepackage{picins}
\usepackage[labelfont={sf,bf}]
{caption}
% \sample und \FOR wie zuvor
```

Text für unsere Seite, der immer und immer wieder verwendet wird.

$$E = mc^2$$

Abb. 2: Einsteins Formel.

Abb. 2: Einsteins Formel.

```
\piccaptioninside
\piccaption{Einsteins Formel.}
\parpic(50mm,10mm)[s]{\FOR}
\sample\sample
```

Text für unsere Seite, der immer und immer wieder verwendet wird.

Abb. 3: Einsteins Formel.

$$E = mc^2$$

Abb. 3: Einsteins Formel.

```
\piccaptionside
\piccaption{Einsteins Formel.}
\parpic(30mm,10mm)[s]{\FOR}
\sample
```

Text für unsere Seite, der immer und immer wieder verwendet wird. Text für unsere Seite, der immer und immer wieder verwendet wird.

```
\piccaptiontopside
\piccaption{Einsteins Formel.}
\parpic(30mm,10mm)[sr]{\FOR}
\sample\sample
```

Bsp.
6-4-8

6.5 Gleitobjektlegenden

Um den Inhalt eines Gleitobjektes (`figure` oder `table` in Standard- $\text{\LaTeX}$) zu erläutern, setzt man normalerweise mithilfe des `\caption`-Befehls eine Legende. Nach einer Einführung in die grundlegende Syntax und der Erläuterung der Low-Level-Schnittstelle von Standard- $\text{\LaTeX}$ beschreibt dieser Abschnitt das überaus vielseitige `caption`-Paket, mit dessen Hilfe man das Layout einer Legende in vieler Hinsicht an seine eigenen Bedürfnisse anpassen kann. Wie die Beispiele zeigen, lässt es sich mit allen anderen Paketen dieses Kapitels kombinieren.

Danach werden die Pakete `subfig` und `subfloat` vorgestellt, die es ermöglichen, Gleitobjekte auf verschiedene Weise zu untergliedern. Der Abschnitt schließt mit den Paketen `sidecap` (für das Setzen von Legenden neben Gleitobjekte) und `ftpage` (zum Erzeugen ganzseitiger Gleitobjekte, deren Legende auf der gegenüberliegenden Seite steht).

```
\caption[kurztext]{legende}
```

Dieser Standard- $\text{\LaTeX}$ -Befehl ist nur innerhalb einer Gleitobjektumgebung definiert. Er erhöht den Zähler für das jeweils betreffende Gleitobjekt. Das optionale Argument `kurztext` enthält, sofern es vorhanden ist, den Text für das Abbildungs- oder Tabellenverzeichnis. Andernfalls wird das obligatorische Argument `legende` auch als Text für diese Verzeichnisse benutzt. Wenn die

Legende länger als eine Zeile ist, empfiehlt es sich, im optionalen Argument eine kurze, informative Beschreibung für das Gleitobjekt anzugeben. Andernfalls kann das Abbildungs- oder Tabellenverzeichnis so unübersichtlich werden, dass die erforderlichen Informationen nur noch schwer zu finden sind. Legendens mit mehreren Absätzen lässt L^AT_EX ohnehin nur zu, wenn das Argument *kurztext* existiert. Ansonsten erfolgt die Fehlermeldung „Runaway argument?“.

Das folgende Beispiel zeigt, wie Standard-L^AT_EX Legendens setzt. Hier empfiehlt sich ein Vergleich mit den Anpassungsmöglichkeiten der verschiedenen Pakete, die in den nächsten Abschnitten besprochen werden. Man beachte das optionale Argument des zweiten \caption-Befehls, das den Text für das Abbildungsverzeichnis vorgibt.

Abbildungen

- 1 Kurzer Legendentext 6
- 2 Kurzer Eintrag in der LOF . . . 6

1 Legende

Abbildung 1 und 2 haben Legendens.

Ein kleines Bild

Abb. 1: Kurzer Legendentext

Ein kleines Bild

Abb. 2: Langer Legendentext, der erklärt, dass dieses Bild sehr wichtig ist, obwohl es so klein ist.

```
\listoffigures
```

```
\section{Legende}
Abbildung \ref{Fig1} und \ref{Fig2}
haben Legendens.
```

```
\begin{figure}[ht]
\centerline{\fbox{\small Ein kleines Bild}}
\caption{Kurzer Legendentext}\label{Fig1}
\end{figure}
```

```
\begin{figure}[ht]
\centerline{\fbox{\small Ein kleines Bild}}
\caption[Kurzer Eintrag in der LOF]
{Langer Legendentext, der erklärt,
dass dieses Bild sehr wichtig ist, obwohl
es so klein ist.}\label{Fig2}
\end{figure}
```

Bsp.
6-5-1

Intern ruft \caption den Befehl \@makecaption{label}{legende} auf. Das Argument *label* enthält die Nummer der Legende und einen Text wie „Abbildung“; es wird intern generiert, passend zur Gleitobjektklasse. Das Argument *legende* erhält den Inhalt des obligatorischen \caption-Parameters, also den zu setzenden Text. Die Standarddefinition des Teils, der für das Setzen einer Legende verantwortlich ist, sieht ungefähr folgendermaßen aus:

```
\newcommand\@makecaption[2]{%      #1 ist z.B. "Abbildung 1"
\vspace{\abovecaptionskip}%      #2 ist der Legendentext
\sbox\@tempboxa{#1: #2}%
% Test: ist die Box breiter als eine Zeile?
\ifthenelse{\lengthtest{\wd\@tempboxa >\linewidth}}{%
{\noindent #1: #2\par}%           mehrere Zeilen
{\centering %                     einzelne Zeile
\makebox[\linewidth][c]{\usebox\@tempboxa}\par % ...
}%
\vspace{\belowcaptionskip}%
}
```

Nach einem vertikalen Anfangsabstand der Länge `\abovecaptionskip` (häufig auf 10pt voreingestellt) wird die Legende in eine temporäre Box `\@tempboxa` gesetzt, und ihre Breite wird mit der Zeilenbreite verglichen. Wenn der Text in eine Zeile passt, wird er zentriert, andernfalls wird er als Absatz mit der normalen Zeilenbreite gesetzt. Danach wird noch ein vertikaler Abstand von `\belowcaptionskip` (normalerweise auf 0pt voreingestellt) hinzugefügt, womit die Formatierung beendet ist. Die Standardklassen verwenden Low-Level-Befehle für diese Vorgänge, um den Ablauf zu beschleunigen, so dass der Code dort etwas anders aussieht.

Man kann natürlich andere Definitionen erstellen, um seine Legendens zu formatieren. Man kann sogar für jede Gleitobjektklasse verschiedene Befehle zum Erzeugen von Legendens bereitstellen. Anstelle des Befehls `\@makecaption` lassen sich die Legendens für eine `figure`-Umgebung z.B. auch mit `\@makefigcaption` formatieren.

```
\newcommand\@makefigcaption[2]{...}
\renewenvironment{figure}
    {\let\@makecaption\@makefigcaption \@float{figure}}
    {\end@float}
```

Dieser Ansatz erfordert Low-Level-Programmierung und ist nicht sehr flexibel, so dass man normalerweise besser ein Paket wie `caption` (im nächsten Abschnitt beschrieben) verwendet, das diese Arbeit übernimmt.

6.5.1 `caption` – Anpassen von Legendens

Axel Sommerfeldt entwickelte das Paket `caption`¹, um Legendens in Gleitobjektumgebungen anzupassen. Es unterstützt nicht nur die $\text{\LaTeX}$ -Standardumgebungen `figure` und `table`, sondern arbeitet auch korrekt mit dem `\rotcaption`-Befehl sowie mit den Umgebungen `sidewaysfigure` und `sidewaystable` des `rotating`-Paketes zusammen. Es funktioniert ebenso gut mit den anderen Paketen dieses Kapitels sowie den meisten anderen Paketen, die Legendens in der einen oder anderen Form produzieren (etwa `listings`, `longtable` oder `supertabular`; die Originaldokumentation enthält eine Kompatibilitätsübersicht).

Genau wie das `geometry`-Paket setzt das `caption`-Paket das erweiterte Optionskonzept (auf Basis des `keyval`-Paketes) ein, bei dem den Optionen Werte zugewiesen werden, die durch ein Gleichheitszeichen vom Optionsnamen getrennt sind. Für die meisten Optionen gibt es eine Standardeinstellung; diese wird gültig, wenn der Optionsname ohne Wert verwendet wird.

Die Anpassungsmöglichkeiten des `caption`-Paketes umfassen nahezu alle Aspekte der Formatierung und Positionierung von Legendens; sie werden im Folgenden vorgestellt. Für Anwender, denen dieser Funktionsumfang nicht ausreicht, bietet das Paket eine Schnittstelle zur Definition eigener Optionswerte (die für spezielle Formatierungen stehen). Man kann sogar eigene

¹Das `caption`-Paket ist eigentlich eine völlig überarbeitete Version von Axels `caption2`-Paket, wodurch Letzteres völlig überholt ist. Axel empfiehlt allen Anwendern von `caption2` sobald wie möglich auf `caption` umzusteigen, und ihre $\text{\LaTeX}$ -Quellen, falls erforderlich, entsprechend zu verändern.

Die generelle Form anpassen

Optionen hinzufügen. Diese Funktionalität wird beispielsweise von dem Paket `subfig` genutzt, das in Abschnitt 6.5.2 beschrieben ist.

Zunächst ein Blick auf diejenigen Optionen, die die generelle Form der Legende beeinflussen:

singlelinecheck Passt die ganze Legende (mitsamt Label) in eine einzige Zeile? Dann wähle spezielle Formatierung, typischerweise zentrieren (Schlüsselwort `true`). Durch das Schlüsselwort `false` werden diese Legenden genau wie mehrzeilige Legenden formatiert.

format Diese Option gibt die allgemeine Form der Legende vor (außer wenn sie durch die vorige Option überschrieben wird). Durch das Schlüsselwort `default` erhält man ein typisches „Standard- $\LaTeX$ “-Format, d.h. Label und Text werden in einem Block gesetzt. Wenn keine weiteren Anpassungsoptionen vorhanden sind, werden Label und Text durch einen Doppelpunkt und ein Leerzeichen getrennt und die Legende wird im Blocksatz ausgerichtet.

Durch das Schlüsselwort `hang` kann man ansonsten festlegen, dass Label (und Trennzeichen) links des Legendentextes erscheinen. Anders ausgedrückt werden alle weiteren Zeilen um die Breite des Labels eingerückt.

margin, width Normalerweise nimmt die Legende die gesamte Breite der Spalte (oder Seite) ein. Wenn man entweder für die Option `width` oder für `margin` einen bestimmten Wert festlegt, verringert sich diese Ausdehnung. In beiden Fällen wird die Legende in dem ihr verbleibenden Raum zentriert. Daher kann man in der aktuellen Version keine unterschiedlichen Werte für den linken und rechten (oder inneren und äußeren) Rand angeben.

indentation Wenn diese Option auf eine bestimmte Größe gesetzt wird, bewirkt sie einen zusätzlichen Einzug für alle Zeilen außer der ersten (z.B. zusätzlich zu einem bereits durch das Schlüsselwort `hang` verursachten Einzug).

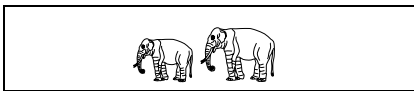


Abb. 1: Kurze Legende

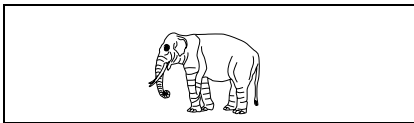


Abb. 2: Eine Legende, die mehrere Zeilen umfasst

```
\usepackage{float,graphicx}
\usepackage[format=hang,margin=10pt]{caption}
\floatstyle{boxed} \restylefloat{figure}

\begin{figure}[ht] \centering
  \includegraphics[width=8mm]{elephant}
  \includegraphics[width=10mm]{elephant}
  \caption{Kurze Legende}
\end{figure}

\begin{figure}[ht] \centering
  \includegraphics[width=15mm]{elephant}
  \caption{Eine Legende, die mehrere Zeilen umfasst}
\end{figure}
```

Bsp.
6-5-2

Anpassen der Fonts

Bei einem Blick auf das vorige Beispiel zeigt sich, dass der Abstand zwischen Box und Legende bei diesem Layout nur sehr schmal ist. Auf Seite 321 wird erklärt, wie man solche Abstände anpassen¹ kann. Zunächst folgt jedoch

¹In manchen Gleitobjektlayouts, wie z.B. „boxed“ sind sie jedoch fest eingestellt und lassen sich nicht ändern.

eine Beschreibung der Optionen, mit deren Hilfe sich die Schriftarten innerhalb der Legende ändern lassen, und die immer funktionieren.

font Diese Option legt die Fontcharakteristika für die gesamte Legende (Label und Text) fest, sofern sie nicht in Teilen von weiteren Optionen überschrieben werden. Sie kann eine durch Kommas getrennte Liste von Schlüsselwörtern für folgende Eigenschaften enthalten: Schriftfamilie (rm, sf oder tt), Schriftserie (md oder bf), Schriftform (up, it, sl oder sc) und Schriftgrad (scriptsize, footnotesize, small, normalsize, large oder Large). Enthält die Liste mehr als ein Schlüsselwort, so muss sie in geschweifte Klammern gesetzt werden, damit das innere Komma nicht als Trennzeichen zur nächsten Option fehlinterpretiert wird (siehe nächstes Beispiel).

Schlüsselwörter für das gleiche Fontattribut (z.B. Schriftform) überschreiben einander; diejenigen für unterschiedliche Attribute lassen sich jedoch wie erwartet kombinieren.

Mit dem Schlüsselwort `default` werden alle Attribute auf ihre Standardwerte zurückgesetzt.

labelfont Während die Option `font` die allgemeinen Charakteristika bestimmt, legt diese Option die (zusätzlichen) Attribute für das Label der Legende fest.

textfont Diese Option entspricht `labelfont`, gilt aber für den Text der Legende. Im nächsten Beispiel wird mit ihrer Hilfe die Schriftserie von fetter auf mittlere Stärke gesetzt.



Abb. 1: Kurze Legende

A B C D E F G H I J K L M

Tabelle 1: Eine Legende, die mehrere Zeilen umfasst

```
\usepackage{float,graphicx}
\usepackage[font={sf,bf},textfont=md]{caption}
\floatstyle{boxed} \restylefloat{table}

\begin{figure}[ht] \centering
\includegraphics[width=10mm]{Escher}
\caption{Kurze Legende}
\end{figure}

\begin{table}[ht]
\centering A B C D E F G H I J K L M
\caption{Eine Legende, die mehrere Zeilen umfasst}
\end{table}
```

Häufig ist es auch erforderlich, das Label der Legende weiter zu verändern, indem man z.B. das Standardtrennzeichen Doppelpunkt durch ein anderes ersetzt oder ganz weglässt. Genauso kann es sein, dass der Abstand zwischen Label und Text angepasst werden muss. Diese Einstellungen lassen sich über die folgenden Optionen und ihre Schlüsselwörter vornehmen:

Weiteres Anpassen des Labels

labelformat Mit dieser Option kann man ein Format für das Label auswählen. Folgende fertige Schlüsselwörter stehen zur Verfügung: `simple` (der Labeltext, z.B. „Abbildung“, und die Nummer sind nur durch ein geschütztes Leerzeichen voneinander getrennt), `parens` (die Nummer steht in Klammern) und `empty` (Label und Nummer entfallen ganz). Die Wirkung dieser Schlüsselwörter zeigt sich in zahlreichen Beispielen in diesem Kapitel. Mit der Deklaration `\DeclareCaptionLabelFormat` lassen

Bsp.
6-5-3

sich weitere Schlüsselwörter für alternative Formatierungen definieren. Sie wird auf Seite 322 erklärt.

labelsep Diese Option legt die Trennelemente zwischen dem Label und dem Legendentext fest. Vordefinierte Schlüsselwörter sind: `colon` für einen Doppelpunkt, `period` für einen Punkt, `space` für ein Leerzeichen und `newline` für einen Zeilenumbruch. Mit der Deklaration `\DeclareCaptionLabelSeparator` lassen sich weitere Schlüsselwörter für andere Formen der Separierung definieren; siehe nächstes Beispiel. Weitere Details finden sich in der Dokumentation des Paketes. Die im Beispiel verwendete `\captionsetup`-Deklaration ist auf Seite 321 beschrieben.

```
\usepackage{float,graphicx}
\floatstyle{boxed} \restylefloat{figure}
\usepackage{caption}
\DeclareCaptionLabelSeparator{period-newline}{.\newline}
\captionsetup{aboveskip=3pt,singlelinecheck=false,
              labelsep=period-newline,
              labelfont={small,bf}}
```

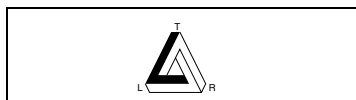


Abb. 1

```
\begin{figure}[ht] \centering
\includegraphics[width=10mm]{Escher} \caption{}
\end{figure}
\begin{figure}[ht] \centering
\includegraphics[width=10mm]{elephant}
\caption{Ein kleiner Elefant}
\end{figure}
```

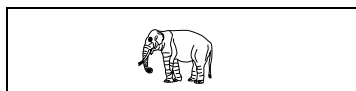


Abb. 2.

Ein kleiner Elefant

Bsp.
6-5-4

*Anpassen von
Absätzen*

Das eigentliche Format des Legendentextes innerhalb der generellen Form kann mit den beiden folgenden Optionen angepasst werden:

justification Diese Option legt den Randausgleich für den Absatz fest. Die Standardausrichtung ist Blocksatz (Schlüsselwort `justified`). Mit dem Schlüsselwort `centering` werden alle Zeilen zentriert. Die Schlüsselwörter `raggedleft` und `raggedright` erzeugen Absätze im Flattersatz an der jeweiligen Seite.

Wenn man zusätzlich das Paket `ragged2e` lädt, stehen darüber hinaus noch die Schlüsselwörter `Centering`, `RaggedLeft` und `RaggedRight` zur Verfügung, mit denen sich die Befehle des Paketes wie in Abschnitt 3.1.12 beschrieben nutzen lassen.

Es gibt noch zwei weitere besondere Arten der Ausrichtung: Das Schlüsselwort `centerfirst` zentriert die erste Zeile und setzt die restlichen im Blocksatz (wobei `\parfillskip` auf null gesetzt ist). Das Schlüsselwort `centerlast` funktioniert genau umgekehrt, d.h. die letzte Zeile wird zentriert. Beide Formen werden hin und wieder für Legenden benötigt, aber in den meisten Fällen erzeugen sie recht fragwürdige Resultate.

Mithilfe der Deklaration `\DeclareCaptionJustification` lassen sich, wie in der Dokumentation beschrieben, noch speziellere Arten der Ausrichtung definieren.

parskip Diese Option steuert den Absatzdurchschuss in einer Legende, die aus mehreren Absätzen besteht. Sie erwartet eine Länge als Wert.

Man beachte, dass Legenden mit mehreren Absätzen nur möglich sind, wenn der `\caption`-Befehl mit optionalem Argument für den *kurztext* verwendet wird.

Bild

Abb. (1) *Eine Legende über mehrere Zeilen, um den Effekt des Schlüsselwortes `centerfirst` zu zeigen.*

Bsp.
6-5-5

```
\usepackage[textfont={rm,it},labelfont={sf},
  labelformat=parens,labelsep=quad,
  justification=centerfirst,parskip=3pt]{caption}

\begin{figure}[ht] \centering
  {\fontfamily{put}\fontsize{60}{60}\bfseries Bild}
  \caption[Ein kurzer Legendentext]
    {Eine Legende über mehrere Zeilen, um den
      Effekt des Schlüsselwortes centerfirst zu zeigen.}
\end{figure}
```

Die verbleibenden Optionen beziehen sich auf die Position der Legende zum Gleitobjekt. Man beachte, dass diese Einstellungen die Legende nicht wirklich an eine bestimmte Stelle verschieben. (Man muss sie manuell verschieben oder ein Gleitobjektlayout des `float`-Paketes dazu verwenden.) Sie wirken sich lediglich auf die eingefügten Abstände aus.

*Anpassen der
Abstände um die
Legende*

aboveskip Leerraum zwischen Legende und Gleitobjekt – das heißt über („above“) der Legende, wenn diese unterhalb des Objektes steht. Er beträgt normalerweise 10pt.

belowskip Leerraum auf der Seite der Legende, die dem Gleitobjekt abgewandt ist. In den meisten Standardklassen beträgt er 0pt.

position Gibt an, dass die Legende oberhalb (Schlüsselwort `top`) oder unterhalb des Gleitobjektes liegt (Schlüsselwort `bottom`, voreingestellter Wert). Diese Option positioniert die Legende *nicht* an dieser Stelle. Das muss man immer noch selbst (oder durch ein Paket wie `float`) bewirken.

Achtung, die Optionsnamen `aboveskip` und `belowskip` sind missverständlich: Sie beschreiben *keine* reale Position, sondern werden lediglich vertauscht, wenn *angegeben* ist, dass sich die Legende oberhalb des Objektes befindet. Darin unterscheiden sie sich von den Parametern `\abovecaptionskip` und `\belowcaptionskip` des $\text{\LaTeX}$ -Standardbefehls `\caption` (siehe Seite 316), welche *tatsächlich* die reale Position der Legende zum Gleitobjekt beschreiben. Bei manchen Layouts des `float`-Paketes haben diese Optionen möglicherweise keinerlei Auswirkungen.

*Vorsicht mit den
Bedeutungen der
Optionen*

Nicht jeder kann sich vielleicht mit der zuvor beschriebenen Optionsliste anfreunden. Außerdem lassen sich damit nur die Legenden aller Gleitobjekte eines Dokumentes formatieren, egal zu welcher Klasse sie gehören. Es kann jedoch vorkommen, dass beispielsweise Legenden für Tabellen anders behandelt werden müssen als solche für Abbildungen. In solchen Fällen hilft die `\captionsetup`-Deklaration weiter.

```
\captionsetup[klasse]{options-wert-liste}
```

Mit der Deklaration `\captionsetup` lässt sich eine *options-wert-liste* festlegen, vergleichbar mit der, die beim Laden des Paketes verwendet wird. Der

Unterschied liegt darin, dass diese neue Deklaration mit dem optionalen Argument *klasse* versehen werden kann. Damit gelten alle ihre Formatierungsanweisungen ausschließlich für Legenden der angegebenen Objektklasse (z.B. *figure*) oder einer beliebigen Klasse, die mit der `\newfloat`-Deklaration des *float*-Paketes angelegt wurde.

`\DeclareCaptionStyle{name}[kurz-layout]{lang-layout}`

Die Deklaration `\DeclareCaptionStyle` bietet weitere Unterstützung. Sie verknüpft eine Options-Wert-Liste mit einem Namen, der später als Wert für die Option *style* verwendet werden kann. Das obligatorische Argument *lang-layout* enthält eine Liste von Options-Wert-Paaren zur Formatierung einer Legende, für die der Stil (Option *style*) *name* ausgewählt wurde. Im optionalen Argument *kurz-layout* können weitere Options-Wert-Paare angegeben werden, die *außerdem* ausgeführt werden, wenn die Legende als „kurz“ gilt (d.h., wenn sie in eine Zeile passt).

Wie im nächsten Beispiel gezeigt wird, kann man die Option *style* im Argument von `\captionsetup` mit weiteren Optionen kombinieren. Dort wird der Stil *default* (vordefiniert) für alle Gleitobjektklassen außer *figure* ausgewählt; seine Einstellung für *labelfont* wird jedoch überschrieben. Es ist ein Beispiel für die Möglichkeiten des Paketes – nicht für guten Geschmack.



Abb. 1. Eine Legende über mehrere Zeilen, um den Effekt des Schlüsselwortes *style* zu zeigen.

A B C D E F G H I J

Tabelle 1: Eine Legende über mehrere Zeilen, um den Effekt des Schlüsselwortes *style* zu zeigen.

```
\usepackage{caption,graphicx}
\DeclareCaptionStyle{italic}
  {labelfont={sf,bf},textfont={rm,it},indentation=18pt,
   labelsep=period,justification=raggedright}
\captionsetup{style=default,labelfont={sf,bf}}
\captionsetup[figure]{style=italic}

\begin{figure}[ht]
  \centering \includegraphics{cat}
  \caption{Eine Legende über mehrere Zeilen, um den
    Effekt des Schlüsselwortes style zu zeigen.}
\end{figure}

\begin{table}[ht]
  \centering \fbox{A B C D E F G H I J}
  \caption{Eine Legende über mehrere Zeilen, um den
    Effekt des Schlüsselwortes style zu zeigen.}
\end{table}
```

 Bsp.
6-5-6

`\DeclareCaptionLabelFormat{name}{code}`

Diese Deklaration definiert oder verändert das Schlüsselwort *name* der Option *labelformat* dahingehend, dass es ihm den *code* zum Formatieren des Labels zuweist. Das Argument *code* verfügt dabei wiederum über zwei formale Parameter: #1 (eine Zeichenfolge wie „Abbildung“) und #2 (die Gleitobjektnummer). Die folgende Definition für das Schlüsselwort *parens* würde beispielsweise das gesamte Label in Klammern setzen:

`\DeclareCaptionLabelFormat{parens}{( #1 \nobreakspace #2 )}`

Dieser Ansatz würde zwar in allen bisherigen Beispielen problemlos funktionieren, trotzdem birgt er ein potentielles Risiko: Wenn #1 aus irgendeinem Grunde leer ist (z.B. weil man `\figurename` so geändert hat, dass kein Text erzeugt wird), setzt die obige Definition ein Leerzeichen vor die Nummer. Für solche Fälle hält das `caption`-Paket den Befehl `\bothIfFirst` bereit.

```
\bothIfFirst{eins}{zwei}
\bothIfSecond{eins}{zwei}
```

Der Befehl `\bothIfFirst` prüft, ob Argument *eins* leer ist; wenn nicht, werden sowohl Argument *eins* als auch Argument *zwei* gesetzt. Andernfalls wird keines von beiden gesetzt. Mit diesem Befehl kann die vorige Deklaration folgendermaßen verbessert werden:

```
\DeclareCaptionLabelFormat{parens}
      {(\bothIfFirst{#1}{\nobreakspace}#2)}
```

Vielleicht sollen die Label für die Legenden so aussehen, wie in diesem zweiten Beispiel: „(4) Abbildung“. Dazu könnte man ein neues Format namens `parensfirst` aufsetzen und es dann der Option `labelformat` zuweisen:

```
\DeclareCaptionLabelFormat{parensfirst}
      {(#2)\bothIfSecond{\nobreakspace}{#1}}
\captionsetup{labelformat=parensfirst}
```

Auf die gleiche Weise kann man mithilfe der Deklaration `\DeclareCaptionLabelSeparator` neue Schlüsselwörter für die Option `labelsep` generieren.

```
\DeclareCaptionLabelSeparator{name}{code}
```

Diese Deklaration weist dem Schlüsselwort *name* den Inhalt *code* zu. Das neue Schlüsselwort kann dann als Wert für die Option `labelsep` verwendet werden. Möchte man z.B. zwischen Label und Text einen Abstand von einem Geviert einsetzen, der sich etwas dehnen lässt, dann hilft folgende Deklaration weiter:

```
\DeclareCaptionLabelSeparator{widespace}
      {\hspace{1em plus .3em}}
```

Dieser Abstand kann dann mittels `labelsep=widespace` im Argument von `\captionsetup` oder `\DeclareCaptionStyle` eingesetzt werden.

Zusätzlich zu einzelnen Anpassungen des Labelformats lassen sich mithilfe von `\DeclareCaptionFormat` auch eigene generelle Legendenformate oder mit `\DeclareCaptionJustification` besondere Arten der Ausrichtung vorgeben. Bei diesen Deklarationen handelt es sich um speziellere Erweiterungen mit etwas aufwendigerer Programmierung, so dass hier kein Beispiel angeführt wird. Nähere Informationen bietet die Dokumentation des Paketes.

*Neue
Legendenformate und
-ausrichtungen*

Externe Konfigurationsdateien

Derartige Deklarationen können in der Dokumentenpräambel verwendet werden. Wenn man immer wieder mit den gleichen Einstellungen arbeitet, kann man sie auch in einer Konfigurationsdatei speichern (z.B. `mycaption.cfg`) und diese Konfiguration dann folgendermaßen laden:

```
\usepackage[config=mycaption]{caption}
```

Auch wenn sich die Option `config` mit weiteren Optionen kombinieren lässt, ist es wahrscheinlich übersichtlicher, weitere Änderungen in der Präambel mit `\captionsetup` zu deklarieren.

Legenden, die sich über mehrere Gleitobjekte erstrecken

Manchmal sind Abbildungen oder Tabellen so groß, dass sie nicht auf eine einzelne Seite passen. Für Tabellen lässt sich dieses Problem möglicherweise mit den Paketen `longtable` oder `supertabular` lösen. Für mehrseitige Abbildungen gibt es jedoch noch kein Paket, das sie automatisch über mehrere Seiten verteilt.

Bisher bot das Paket `captcont` von Steven Cochran eine allgemeine Lösung für dieses Problem, indem es ermöglichte, die Nummer der Legende über mehrere Gleitobjekte hinweg beizubehalten. Nun ist diese Funktion auch im `caption`-Paket bereits integriert. Wenn die aktuelle Nummer der Legende beibehalten werden soll, muss vor dem `\caption`-Befehl der Befehl `\ContinuedFloat` erfolgen.

Wenn die fortlaufende Legende nicht im „...-verzeichnis“ erscheinen soll, muss man den `\caption`-Befehl mit leerem optionalen Argument (siehe Beispiel 6-5-13 auf Seite 331) oder den Befehl `\caption*` verwenden, der den Verzeichniseintrag und die Nummerierung unterdrückt.

Abbildungen		
1	Groß . . .	6
1	Groß (Forts.) .	7
Eine Abbildung am Fuß der Seite, die am		
Abbildung		
Abb. 1: Groß		
6		
Abbildung		
Abb. 1: Groß (Forts.)		
Kopf der nächsten Seite weitergeht. Man beachte, dass beide Teile im Abbildungsverzeichnis erschei-		
7		

```
\usepackage{caption}
\listoffigures \medskip
\begin{figure}[!b]
\centering \fbox{Abbildung}
\caption{Groß}
\end{figure}
Eine Abbildung am Fuß der Seite,
die am Kopf der nächsten Seite
weitergeht.
\begin{figure}[!t]
\ContinuedFloat
\centering
\fbox{\rule[-.5cm]{0pt}{1.5cm}%
Abbildung}
\caption{Groß (Forts.)}
\end{figure}
Man beachte, dass beide Teile im
Abbildungsverzeichnis erscheinen.
```

Bsp.
6-5-7

Wie aus den verschiedenen Beispielen ersichtlich, arbeitet das `caption`-Paket problemlos mit den anderen Paketen in diesem Kapitel zusammen. Dabei ist jedoch zu beachten, dass es in einigen Fällen *nach* den Paketen geladen werden muss, deren Legenden man verändern möchte.

6.5.2 subfig – Gleitobjekte weiter strukturieren

Mit dem Paket `subfig` von Steven Cochran kann man Verweise auf kleine „untergeordnete“ Abbildungen und Tabellen setzen, und diese Objekte leichter in einer einzigen Gleitobjektumgebung positionieren, sowie mit Legende und Label versehen. Falls gewünscht, kann man im jeweiligen Verzeichnis (z.B. Abbildungsverzeichnis) untergeordnete Legendes für diese Unterobjekte aufführen. Außerdem ist eine globale Legende möglich.

Das Paket basiert auf dem `caption`-Paket aus dem vorigen Abschnitt und nutzt dessen gesamten Funktionsumfang zum Anpassen des Legendenlayouts.¹ Der zentrale Befehl zum Festlegen eines Unterobjektes in einem Gleitobjekt ist `\subfloat`.

`\subfloat[verz-eintrag][legende]{objekt}`

Das obligatorische Argument *objekt* enthält den Inhalt des Unterobjektes, das optionale Argument *legende* enthält seine Legende, und das optionale Argument *verz-eintrag* legt, soweit erforderlich, eine alternative Legende für den Verzeichniseintrag fest. Ohne die optionalen Argumente wird keine Legende (und kein Label) erzeugt. Mit einem leeren *legende*-Argument kann man ein rein (alpha-)numerisches Label erzeugen.

Wenn das Argument *verz-eintrag* leer ist, erfolgt für dieses Objekt kein Eintrag im entsprechenden Verzeichnis. Diese Sonderfunktion ist nur von Bedeutung, wenn untergeordnete Legendes überhaupt in Verzeichnissen erscheinen sollen. Auf Seite 330 befinden sich weitere Informationen zu diesem Verfahren.

Das erste Beispiel zeigt eine Abbildung mit zwei `\subfloat`-Objekten. Um auf diese Unterobjekte zu verweisen, müssen den `\subfloat`-Befehlen Label zugeordnet werden. (Dabei ist zu beachten, dass die `\label`-Befehle *innerhalb* des obligatorischen *objekt*-Argumentes stehen müssen.) Außerdem wird nach dem `\caption`-Befehl ein `\label`-Befehl für die äußere figure-Umgebung gesetzt, so dass man außerhalb der Umgebung einzeln auf jedes der Objekte verweisen kann.

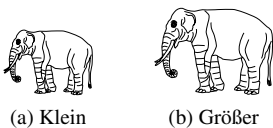


Abb. 1: Zwei Elefanten

```
\usepackage{subfig} \usepackage{graphicx}
\begin{figure} \centering
\subfloat[Klein]
{\includegraphics[width=12mm]{elephant}\label{sf1}}
\quad
\subfloat[Größer]
{\includegraphics[width=16mm]{elephant}\label{sf2}}
\caption{Zwei Elefanten}\label{elefanten}
\end{figure}
Unterabbildung-\ref{sf1} in Abbildung-\ref{elefanten}
ist kleiner als Unterabbildung-\ref{sf2}.
```

Bsp.
6-5-8

Unterabbildung 1a in Abbildung 1
ist kleiner als Unterabbildung 1b.

¹Das Paket war in einer früheren Version unter dem Namen `subfigure` bekannt. In seinen Anpassungsmöglichkeiten ähnelte es in vieler Hinsicht dem Paket `caption2` von Axel Sommerfeldt. Es gab jedoch einige wichtige Unterschiede. Als `caption2` weiterentwickelt wurde, überredete einer der Autoren dieses Buches Steven, die neue Version seines Paketes auf dem `caption`-Paket aufzusetzen. Die Ergebnisse sind in diesem Abschnitt beschrieben.

Da das `subfig`-Paket auf dem `caption`-Paket basiert, lassen sich die Layouts der Legenden von untergeordneten Gleitobjekten mit den Optionen des letzteren verändern. Wenn es nicht bereits geladen ist, lädt `subfig` das `caption`-Paket *ohne* Optionen. Daher muss `caption` entweder (wie im folgenden Beispiel) zuerst geladen werden, oder man muss es nach dem Laden von `subfig` mithilfe einer oder mehrerer `\captionsetup`-Deklarationen anpassen.

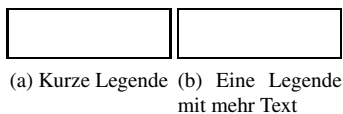


Abb. 1: Standard-Unterlegenden

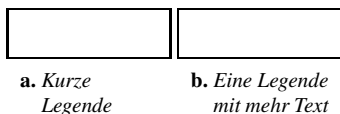


Abb. 2: Angepasste Unterlegenden

```
\usepackage[font=sf]{caption}
\usepackage{subfig}
\newcommand\LCap{Eine Legende mit mehr Text}
\newcommand\FIG{\fbox{\parbox{.4\textwidth}{\strut}}}

\begin{figure}[ht] \centering
\subfloat[Kurze Legende]{\FIG} \subfloat[\LCap]{\FIG}
\caption{Standard-Unterlegenden}
\end{figure}

\captionsetup[subfloat]{format=hang,textfont=it,
margin=5pt,labelfont={rm,bf},labelformat=simple,
labelsep=period,justification=raggedright}

\begin{figure}[ht] \centering
\subfloat[Kurze Legende]{\FIG} \subfloat[\LCap]{\FIG}
\caption{Angepasste Unterlegenden}
\end{figure}
```

Bsp.
6-5-9

Standardeinstellungen des `subfig`-Paketes

Wie man sieht, lassen sich die Optionen zur Anpassung des Layouts für die Legenden zu verschiedenen Zeitpunkten einstellen. Beim Laden des `subfig`-Paketes sind einige Voreinstellungen bereits abgeschlossen. So fällt auf, dass sich die Einstellung der Schrift mit `font=sf` beim Laden des `caption`-Paketes nicht auf die untergeordneten Legenden auswirkt, da die Schrift für die untergeordneten Legenden mit `font=footnotesize` bereits festgelegt wurde. Eine weitere Voreinstellung, auf die man schließen kann, ist die Verwendung des Schlüsselwortes `parens` für die `labelformat`-Option. Aber die meisten anderen Änderungen des Legendenlayouts für Gleitobjekte werden an die untergeordneten Objekte vererbt.

Alle untergeordneten Legenden anpassen

Diese Voreinstellungen lassen sich überschreiben, indem man beim Laden des `subfig`-Paketes beliebige Optionen des `caption`-Paketes verwendet; alternativ kann man diese auch später (wie im Beispiel gezeigt) mithilfe der Deklaration `\captionsetup` der Gleitobjektklasse „`subfloat`“ zuweisen. Dadurch werden alle nachfolgenden untergeordneten Legenden verändert, bis eine weitere Deklaration die neuen Einstellungen überschreibt.

Untergeordnete Legenden pro Klasse anpassen

Und wenn man nur die untergeordneten Gleitobjekte einer bestimmten `<klasse>` (z.B. `figure`) anpassen will, muss man in der `\captionsetup`-Deklaration `sub<klasse>` anstelle von `subfloat` angeben.

Abstände um Unterobjekte

Mithilfe des `subfig`-Paketes lassen sich noch einige weitere Anpassungen über Optionen vornehmen, die nicht im `caption`-Paket enthalten sind; diese Optionen erwarten Längenangaben als Wert. Sie legen die Abstände um ein untergeordnetes Gleitobjekt fest. Wenn sich die Legende entsprechend der Voreinstellung (das heißt `position=bottom`) unterhalb des Objektes befindet, erhält man das in Abbildung 6.1 auf der gegenüberliegenden Seite dargestellte Layout.

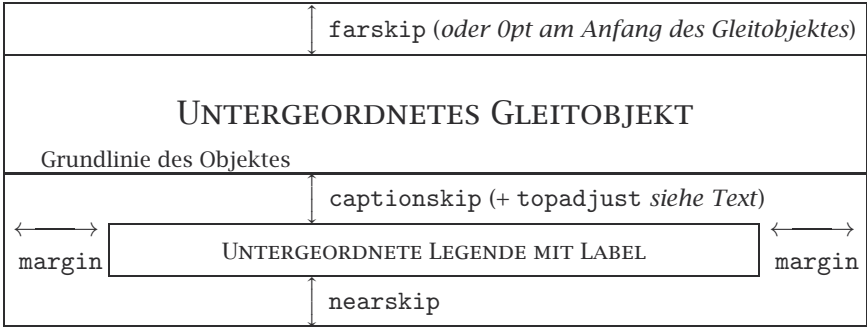


Abbildung 6.1: Layout der Abstände des subfig-Paketes

farskip Legt den Freiraum auf der Seite des untergeordneten Objektes fest, die der globalen Legende abgewandt ist (d.h. oberhalb, wenn sich die globale Legende am Fuß des Gleitobjektes befindet). Dieser Abstand wird ignoriert, wenn er das gesamte Gleitobjekt einleitet. Seine Voreinstellung ist 10pt.

nearskip Legt den Freiraum auf der Seite des Unterobjektes fest, die der globalen Legende zugewandt ist, um das Unterobjekt und seine Legende von dem umgebenden Material abzusetzen. Er beträgt normalerweise 0pt.

captionskip Legt den vertikalen Abstand zwischen dem Unterobjekt und seiner Legende fest (voreingestellt auf 4pt). Wenn keine Legende existiert, wird dieser Abstand nicht gesetzt.

topadjust Wenn sich die untergeordnete Legende oberhalb des Unterobjektes befindet (d.h. Abbildung 6.1 wird durch `position=top` auf den Kopf gestellt), dann wird dieser Abstand zu der Entfernung `captionskip` zwischen der globalen Legende und dem Unterobjekt hinzuaddiert. Dieser Wert wird ignoriert bei `position=bottom` auf Ebene des Unterobjektes.

Die Breite der Legende entspricht der des untergeordneten Gleitobjektes, auf beiden Seiten abzüglich des Wertes der Option `margin` des `caption`-Paketes.

Wenn sich die Legende über dem Unterobjekt befindet (also durch `position=top` für das Unterobjekt), dann wird `captionskip` um `topadjust` erhöht, um für diesen Fall den Abstand zwischen der Legende und dem Objekt auszugleichen. Es ist auch zu beachten, dass sich `farskip` und `nearskip` an der Position der globalen Legende orientieren. Wenn diese Legende oben steht (durch `position=top` auf Ebene des Gesamtobjektes), tauschen `farskip` und `nearskip` ihre Plätze.

Intern verwendet `\subfloat` einen Zähler, um die Unterobjekte durchnummerieren und das Label für ihre Legenden zu erzeugen. Der Zähler heißt `sub<klasse>`, wobei sich *klasse* auf die Art des aktuellen Unterobjektes bezieht (der Zähler für Unterabbildungen heißt z.B. `subfigure`). Seine Darstellung wird durch `\thesub<klasse>` definiert und ist auf `\alph{sub<klasse>}` voreingestellt. Diese Zähler werden für jedes Unterobjekt erhöht, egal ob eine Legende gesetzt wurde oder nicht.

Label für untergeordnete Legenden

Das nächste Beispiel zeigt ein etwas komplexeres Layout, das mehrere der besprochenen Optionen einsetzt. Es enthält drei Untertabellen, wovon zwei nebeneinander über einer dritten erscheinen. Durch die Einstellung der Optionen stehen die Legenden in kleiner, schräg gestellter Schrift über ihren Tabellen. Einzeilige Legenden erscheinen linksbündig; mehrzeilige sind mit Flattersatz an der rechten Seite und hängendem Einzugsatz formatiert. Um Einblick in weitere Anpassungsmöglichkeiten zu geben, wurde der Befehl `\thesubtable` (welcher die „Nummer“ für untergeordnete table-Objekte erzeugt) so umdefiniert, dass er für die Untertabellen eine zweistufige Legendenummerierung setzt. Jeder der `\subfloat`-Befehle, sowie die äußere table-Umgebung, sind mit jeweils einem `\label`-Befehl versehen. Dadurch lassen sich die verschiedenen Elemente einzeln ansprechen.

Tabelle 1: Drei Untertabellen

(1.1) Eins

(1.2) Zwei

Tabelle 1

Tabelle 2

(1.3) Dritte Tabelle mit viel
längerer Legende

Tabelle 3

Tabelle 1 enthält die Untertabellen (1.1) bis (1.3). Aber jetzt nicht 11.3 verwenden (siehe Text).

```
\usepackage{subfig}
\captionsetup[table]{position=top,aboveskip=5pt}
\captionsetup[subtable]{singlelinecheck=false,
                        format=hang,font={sl,small},
                        justification=raggedright}
\renewcommand\thesubtable{\thetable.\arabic{subtable}}
\newcommand\TAB[2]
    {\fbox{\parbox{\#2\textwidth}{Tabelle #1}}}

\begin{table}
\caption{Drei Untertabellen}\label{tbl}
\subfloat[Eins]{\TAB{1}{.4}\label{tbl1}}\hfill
\subfloat[Zwei]{\TAB{2}{.4}\label{tbl2}}\hfill
\subfloat[Dritte Tabelle mit viel längerer Legende]
    {\TAB{3}{.8}\label{tbl3}}
\end{table}

Tabelle \ref{tbl} enthält die Untertabellen
\subref{tbl1} bis \subref{tbl3}. Aber jetzt nicht
\ref{tbl3} verwenden (siehe Text).
```

Bsp.
6-5-10

Die Verweise auf die einzelnen Untertabellen im vorigen Beispiel wurden mithilfe des Befehls `\subref` erzeugt, der sie entsprechend der Einstellung von `listofformat` formatiert (siehe Seite 330). Dadurch lässt sich das in der letzten Zeile des Beispiels sichtbare Problem des `\ref`-Befehls umgehen, welches sich durch den umdefinierten `\thetable`-Befehl ergibt. Da `\ref` die Tabellennummer „1“ mit der Untertabellennummer (z.B. „1.3“) kombiniert, wird von ihm folglich die fehlerhafte Nummer „11.3“ erzeugt.

Die Sternform des Befehls, `\subref*`, gibt nur die reine Nummer des Unterobjektes aus (z.B. den Wert von `\thesubtable`), die sich für die Konstruktion komplexerer Verweise, wie z.B. „Abbildung 1(a-c)“, eignet.

Unterobjekte ohne
Legende

Manchmal möchte man Unterobjekte mit einem Label ohne zusätzlichen Legendentext versehen. Das ist beispielsweise bei einer Bilder- oder Fotoserie üblich: Die Gesamtlegende erklärt die einzelnen Unterobjekte. Das lässt sich leicht erreichen, indem man den `\subfloat`-Befehl mit einem leeren optionalen Argument verwendet; dadurch erhält man ein Unterobjekt mit Label. Das nächste Beispiel zeigt ein solches Layout.

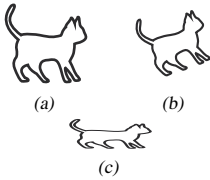


Abb. 1: Eine Gruppe von Katzen: (a) die erste Katze, (b) eine kletternde Katze, und (c) eine gestreckte.

Bsp.
6-5-11

```
\usepackage{graphicx}
\usepackage[font={scriptsize,sl},captionskip=3pt]{subfig}
\newcommand\FIG[1]{\includegraphics[#1]{cat}}

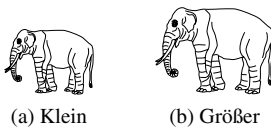
\begin{figure}
\centering
\subfloat[]{\FIG{width=3pc}\label{k1}}
\quad
\subfloat[]{\FIG{angle=20,width=3pc}\label{k2}}
\quad
\subfloat[]{\FIG{height=1pc,width=3pc}\label{k3}}
\caption[Eine Gruppe von Katzen]{Eine Gruppe von Katzen:
\subref{k1} die erste Katze, \subref{k2} eine
kletternde Katze, und \subref{k3} eine gestreckte.}
\end{figure}
```

Man kann einzelne Gleitobjekte bei Bedarf auch noch weiter anpassen, etwa wenn sie außergewöhnlich geformte Unterobjekte oder zuviel Weißraum enthalten. In Beispiel 6-5-8 von Seite 325 könnte man etwa die Gesamtlegende durch die folgende Zeile *am Anfang* des Gleitobjektes näher an die Legenden der Unterobjekte heranbringen:

[Manuelle
Feineinstellungen](#)

```
\captionsetup[subfloat]{nearskip=-3pt}
```

Diese Deklaration bezieht sich dann nur auf das aktuelle Gleitobjekt und entfernt einen Teil des Abstandes `aboveskip` über der Gesamtlegende.



(a) Klein (b) Größer

Abb. 1: Zwei Elefanten

Bsp.
6-5-12

```
\usepackage{subfig} \usepackage{graphicx}

\begin{figure}
\centering
\captionsetup[subfloat]{nearskip=-3pt}
\subfloat[Klein]
{\includegraphics[width=12mm]{elephant}\label{sf1}}
\quad
\subfloat[Größer]
{\includegraphics[width=16mm]{elephant}\label{sf2}}
\caption[Zwei Elefanten]\label{elefanten}
\end{figure}
```

Bisher wurden nur Unterobjekte in den Umgebungen `figure` und `table` besprochen. Wenn man mit eigenen Gleitobjektklassen arbeitet, möchte man diese vielleicht auch weiter untergliedern. Das gelingt mithilfe der `\newsubfloat`-Deklaration.

```
\newsubfloat[options-wert-liste]{klasse}
```

Eine Bedingung für den Einsatz von `\newsubfloat` ist, dass die Umgebung für die entsprechende Gleitobjektklasse bereits existieren muss – zum Beispiel als Umgebung, die mit dem Befehl `\newfloat` des `float`-Paketes deklariert wurde. Dann richtet `\newsubfloat` den Befehl `\subfloat` so ein, dass

er in der entsprechenden Gleitumgebung verwendet werden kann (z.B. durch Deklarieren des Zählers `\sub<float-ty>` für die Label). In dem optionalen Argument *options-wert-liste* kann man Layoutoptionen festlegen, die nur für diese besondere Klasse von untergeordneten Gleitobjekten gelten.

Verzeichniseinträge
erzeugen

Die Legenden der Unterobjekte werden automatisch in die externe Datei eingetragen, welche die entsprechende „...verzeichnis“-Liste enthält. Diese Dateien haben die Erweiterungen `.lof` (für Abbildungen), `.lot` für (Tabellen) beziehungsweise die Erweiterung, die als drittes Argument von `\newfloat` angegeben wurde.

Die untergeordneten Legenden erscheinen jedoch nicht in diesen Verzeichnissen, da normalerweise nur die Legenden der obersten Ebene gesetzt werden. Um dieses Verhalten zu ändern, muss man den Wert des Zählers *erwdepth* auf 2 setzen (wobei *erw* die Erweiterung der entsprechenden „...verzeichnis“-Datei ist). Durch `\setcounter{lofdepth}{2}` werden beispielsweise die Legenden von Unterabbildungen ausgegeben, mit der entsprechenden Einstellung für *lotdepth* die von Untertabellen.

Wie in Abschnitt 2.3.2 erläutert, lässt sich das Layout der Verzeichniseinträge anpassen, indem man `\l@subfigure`, `\l@subtable` und ähnliche Befehle ändert; die entsprechenden Befehlsnamen bestehen immer aus der Objektklasse und dem Präfix `\l@sub`. Das Paket `subfig` verfügt jedoch bereits über drei Optionen zum Formatieren der Verzeichniseinträge, die wahrscheinlich in den meisten Fällen genügend Spielraum bieten.

listofindent Der Einzug der untergeordneten Legende im entsprechenden Verzeichnis. Er ist auf `3.8em` voreingestellt.

listofnumwidth Die Breite für das Label im Verzeichnis. Sie ist auf `2.5em` voreingestellt.

listofformat Das Format des Labels für den Untereintrag im Verzeichnis. Als Schlüsselwerte sind `empty`, `simple`, `parens`, `subsimple` sowie `subparens` (die Voreinstellung) verfügbar. Mit der Deklaration `\DeclareCaptionListOfFormat` lassen sich weitere Formatoptionen festlegen; das genaue Verfahren ist in der Paketdokumentation ausführlich erläutert.

Das formatierte Ergebnis wird auch von dem Befehl `\subref` weiterverwendet, so dass sich eine Veränderung dieses Optionswertes auch auf die Verweise auswirkt, die mit diesem Befehl erzeugt werden.

Das nächste Beispiel zeigt, in welcher Form die Untereinträge im Verzeichnis erscheinen. Der Zähler *lofdepth* wird so eingestellt, dass diese ausgegeben werden, und die Option *listofindent* erhält den Wert `5em`, um sie etwas einzurücken. Dabei wird ein fortgesetztes Gleitobjekt verwendet, um zu demonstrieren, dass auch die untergeordnete Nummerierung weiterläuft. Um den „...verzeichnis“-Eintrag für das fortgesetzte Gleitobjekt zu unterdrücken, erhält der `\caption`-Befehl ein leeres optionales Argument – diese Sonderfunktion ist im `caption`-Paket für eben diesen Zweck vorgesehen. Stattdessen könnte man auch mit `\caption*` sowohl die Legendennummer als auch den Eintrag im Verzeichnis unterdrücken.

Abbildungen

1	Drei Abbildungen . .	1
1(a)	Eins	1
1(b)	Zwei	1
1(c)	Drei	2

Abbildung I	Abbildung II
(a) Eins	(b) Zwei

Bsp.
6-5-13

Abb. 1: Drei Abbildungen

```
\usepackage[nearskip=-3pt,captionskip=5pt]{subfig}
\captionsetup[subfloat]{listofindent=5em,
listofformat=parens}
\setcounter{lofdepth}{2}
\listoffigures \medskip
\begin{figure}[!ht]
\centering
\subfloat[Eins]{\fbox{Abbildung I}} \quad
\subfloat[Zwei]{\fbox{Abbildung II}}
\caption{Drei Abbildungen}
\end{figure}
\pagebreak % <-- zur Darstellung
\begin{figure}[!ht]
\centering \ContinuedFloat
\subfloat[Drei]{\fbox{Abbildung III}}
\caption[] {Drei Abbildungen (Forts.)}
\end{figure}
```

Genau wie das caption-Paket unterstützt auch subfig mit der Option *Externe Konfigurationsdateien* config externe Konfigurationsdateien, in denen bevorzugte Einstellungen gespeichert werden können. Das Beispiel

```
\usepackage[config=xcaption]{subfig}
```

lädt die Datei xcaption.cfg. Auch wenn sich die Option config mit weiteren Optionen kombinieren lässt, ist es wahrscheinlich übersichtlicher, weitere Änderungen in der Präambel mit \captionsetup zu deklarieren.

6.5.3 subfloat – Serien von Gleitobjekten

Das Paket subfloat von Harald Harders kann eine untergeordnete Nummerierung für Tabellen oder Abbildungen erzeugen (analog zu der subequations-Umgebung des amsmath-Paketes). Während das Paket subfig untergeordnete Elemente innerhalb eines Gleitobjektes nummeriert, ermöglicht das subfloat-Paket eine Unternummerierung der Hauptlegenden der Gleitobjekte.

Abbildungen (bzw. Tabellen), die als Serie nummeriert werden sollen, sollten sich in einer subfigures-Umgebung (bzw. subtables-Umgebung) befinden. Alternativ können sie auch zwischen die Befehle \subfiguresbegin und \subfiguresend (bzw. \subtablesbegin und \subtablesend) gesetzt werden. Während die Umgebungen den generellen Regeln zur Verschachtelung von Umgebungen folgen, können die Befehle überall verwendet werden. Diese größere Flexibilität kann sich unter besonderen Umständen als hilfreich erweisen, z.B. wenn sich Serien von Abbildungen mit Serien von Tabellen überschneiden.

Das folgende Beispiel zeigt drei Abbildungen. Die ersten beiden befinden sich in einer subfigures-Umgebung und sind daher als Serie nummeriert („1a“ und „1b“). Beide Label werden von den L^AT_EX-Befehlen \listoffigures und \ref korrekt gehandhabt.

Abbildungen

1a	Die Erste	1
1b	Die Zweite	1
2	Die Dritte	2

Abbildung I

Abb. 1a: Die Erste

Abbildung II

Abb. 1b: Die Zweite

Die Abbildungen 1a und

1b sind in diesem Beispiel als Serie nummeriert; Abbildung 2 nicht.

Abbildung III

Abb. 2: Die Dritte

```
\usepackage{subfloat}
\listoffigures \bigskip
\begin{subfigures}
\begin{figure}[!ht] \centering
\fbbox{Abbildung I}
\caption{Die Erste}\label{FI}
\end{figure}
\begin{figure}[!ht] \centering
\fbbox{Abbildung II}
\caption{Die Zweite}\label{FII}
\end{figure}
\end{subfigures}
Die Abbildungen \ref{FI} und
\ref{FII} sind in diesem Beispiel
als Serie nummeriert;
Abbildung~\ref{FIII} nicht.
\begin{figure}[!ht] \centering
\fbbox{Abbildung III}
\caption{Die Dritte}\label{FIII}
\end{figure}
```

Bsp.
6-5-14

Wie im vorigen Beispiel kombiniert das Standardlabel eine arabische Zahl für die Hauptabbildung mit Kleinbuchstaben für die einzelnen Serienelemente. Das Label kann durch Umdefinieren des Befehls `\thesubfloatfigure` angepasst werden. Innerhalb dieser Definition lassen sich die Nummer der Hauptabbildung mit dem Befehl `\themainfigure`¹ und die Nummer der Unterabbildung mit dem Zähler `subfloatfigure` erzeugen. Also kann man mit der folgenden Definition Abbildungsserien als „2.1“, „2.2“, usw. nummerieren:

```
\renewcommand\thesubfloatfigure
{\themainfigure.\arabic{subfloatfigure}}
```

Für Tabellen bestehen die gleichen Möglichkeiten durch `\thesubfloattable` und `\themaintable`, sowie den Zähler `subfloattable`.

Für einen automatischen Verweis auf die Gesamtzahl der Serienelemente mit der gleichen Abbildungsnummer enthält das Paket die Option `countmax`. Durch diese Option werden alle Gleitobjekte einer `subfigures`- bzw. `subtables`-Umgebung gezählt und die Anzahl wird an den Zähler `subfloatfiguremax` (bzw. `subfloattablemax`) weitergereicht. Mit der folgenden Definition:

```
\renewcommand\thesubfloatfigure{\themainfigure
(\arabic{subfloatfigure}/\arabic{subfloatfiguremax})}
```

könnte man beispielsweise Label wie „2(1/3)“, „2(2/3)“ und „2(3/3)“ erzeugen, wenn die zweite Abbildungsserie aus drei Elementen besteht. Diese Zählung stützt sich intern auf den `\label`- und `\ref`-Mechanismus und benötigt deshalb zwei Durchläufe – damit ist sie sowohl zeit- als auch ressourcenaufwendig. Daher erfolgt in der Standardeinstellung keine Zählung.

¹Aus technischen Gründen kann man den Befehl `\thefigure` nicht für Unterabbildungen einer Serie verwenden. Aus diesem Grunde existiert sein „Alias“ `\themainfigure`.

6.5.4 sidecap – Legenden seitlich setzen

In ihrem Paket `sidecap` stellen Hubert Gäßlein und Rolf Niepraschk zwei neue Umgebungen vor: `SCfigure` und `SCtable`. Sie entsprechen den $\LaTeX$ -Umgebungen `figure` und `table`, setzen die Legenden dabei jedoch seitlich des Gleitobjektes in einer `minipage`-Umgebung mit einstellbarer Breite.

Das Paket unterstützt eine Reihe von Optionen zum Anpassen von Legendenformat und -position.

`outercaption/innercaption` Die Legende erscheint jeweils dem äußeren (Voreinstellung) oder dem inneren Rand einer Seite zugewandt, d.h. sie wechselt ihre Position je nach linker oder rechter Seite.

`leftcaption/rightcaption` Die Legende ist immer dem linken bzw. rechten Rand der Seite zugewandt.

`wide` Legende oder Gleitobjekt können sich, falls erforderlich, bis in den Rand hinein erstrecken.

`margincaption` Die Legende wird in den Rand gesetzt, wobei für den eigentlichen Inhalt des Gleitobjektes die gesamte Kolumnenbreite zur Verfügung steht. Bei dieser Option kann außerdem die Position des Gleitobjektes im Verhältnis zu den Rändern der Druckfahne mit folgenden Optionen festgelegt werden: `innerbody`, `outerbody`, `centerbody`, `leftbody` oder `rightbody`.

`raggedright/raggedleft/ragged` Es erfolgt kein Randausgleich für den Text der Legende. Wenn nur wenig Platz verfügbar ist, liefert diese Option häufig bessere Ergebnisse. Bei der Option `ragged` wechselt der nicht ausgeglichene Rand je nach linker und rechter Seite, so dass sie sich am besten mit `innercaption`, `outercaption` oder `margincaption` kombinieren lässt. Wenn das Paket `ragged2e` von Martin Schröder verfügbar ist, wird es mit eingesetzt.

Wenn man das `sidecap`-Paket mit dem `caption`-Paket kombiniert, lässt sich die Ausrichtung sowohl mithilfe der oben aufgeführten Optionen als auch über die Option `justification` des `caption`-Paketes vornehmen. Lediglich die Option `ragged` ist einmalig, da das `caption`-Paket keine Möglichkeit bietet, die Ausrichtung je nach Seite zu variieren.

```
\begin{SCfigure}[rel-breite][position] <L-R material> \end{SCfigure}
\begin{SCtable}[rel-breite][position] <L-R material> \end{SCtable}
```

Die Umgebungen `SCfigure` und `SCtable` (sowie ihre Sternformen, die sich über zwei Spalten erstrecken) verfügen über zwei *optionale* Argumente. Das Argument *rel-breite* legt die Breite der Legende im Verhältnis zu derjenigen der Tabelle oder Abbildung fest (voreingestellt auf 1.0). Ein hoher Wert (z.B. 20) reserviert die größtmögliche auf der Seite vorhandene Breite. Das zweite Argument *position* entspricht dem $\LaTeX$ -Standardargument zum Positionieren von Gleitobjekten (z.B. `[htb]`). Im Gegensatz zu $\LaTeX$ -Standardgleitobjekten wird das Objekt hier als horizontales Material angesehen (damit es gemessen werden kann). Um an dieser Stelle vertikales Material zu erhalten, verwendet man eine `minipage`-Umgebung innerhalb des Gleitobjektes.

Das erste Beispiel zeigt eine Tabelle und eine Abbildung mit seitlich gesetzten Legenden. Die Tabelle wurde mit den Standardeinstellungen erzeugt, so dass Legende und Tabelle die gleiche Breite haben. Die Abbildung wurde mit einer dreimal so breiten Legende gesetzt. Mit den Voreinstellungen hätte sich die Legende über zwei Zeilen erstreckt, obwohl genügend Platz vorhanden ist. Bis auf die Ausrichtung wurde das Layout der Legende über die Optionen des caption-Paketes eingestellt.

AAA	BBB	Tabelle 1. Eine kleine Tabelle mit relativ langer Legende
CCC	DDD	
EEE	FFF	

Bild I

Abb. 1. Kleines Bild

Absatztext, der zeigt, wie die Gleitobjekte horizontal in der Druckfahne ausgerichtet sind.

```
\usepackage[ragged]{sidecap}
\usepackage[labelfont={sf,bf},textfont=it,
  labelsep=period]{caption}
Absatztext, der zeigt, wie die Gleitobjekte
horizontal in der Druckfahne ausgerichtet sind.
\begin{SCTable} \caption{Eine kleine Tabelle mit
  relativ langer Legende}
\begin{tabular}{|ll|} AAA & BBB \\
  CCC & DDD \\ EEE & FFF \end{tabular}
\end{SCTable}
\begin{SCfigure}[3] \caption{Kleines Bild}
\framebox[.2\linewidth][c]{Bild I}
\end{SCfigure}
```

Bsp.
6-5-15

Ändern der Standardeinstellungen

Zusätzlich zu den Optionen verfügt das Paket über verschiedene Parameter zum Verändern der Formatierung. Die Größe des Abstandes zwischen Objekt und Legende lässt sich durch Undefinieren des Befehls `\sidecaptionsep` (mit `\renewcommand`) ändern. Als Voreinstellung wird der aktuelle Wert des Parameters `\marginparsep` verwendet. Anstatt das Verhältnis zwischen der Breite der Legende und des Gleitobjektes immer wieder über ein optionales Argument der Umgebungen festzulegen, kann man auch `\sidecaptionrelwidth` undefinieren, um die entsprechende Voreinstellung anzupassen. Bei Tabellen ist die Legende bündig nach oben, bei Abbildungen bündig nach unten ausgerichtet. Diese Vorgabe lässt sich mit einer Deklaration wie der folgenden ändern: `\sidecaptionvpos{table}{b}`; dabei sollte das zweite Argument `t`, `c` oder `b` sein.

Das nächste Beispiel setzt alle drei Anpassungsmöglichkeiten ein und die Gleitobjekte dürfen sich in den Rand hinein erstrecken (Option `wide`). Durch den für `\sidecaptionrelwidth` gewählten Wert sind sie sogar gezwungen, allen verfügbaren Platz zu nutzen.

AAA	BBB	Tabelle 1: Eine Tabelle mit relativ langer Legende
CCC	DDD	
EEE	FFF	

Ein Absatztext, der zeigt, wie das Gleitobjekt horizontal in der Druckfahne ausgerichtet ist.

```
\usepackage[wide]{sidecap}
\renewcommand\sidecaptionsep{15pt}
\renewcommand\sidecaptionrelwidth{20}
\sidecaptionvpos{table}{c}
Ein Absatztext, der zeigt, wie das Gleitobjekt
horizontal in der Druckfahne ausgerichtet ist.
\begin{SCTable} \caption{Eine Tabelle mit
  relativ langer Legende}
\begin{tabular}{|ll|} AAA & BBB \\
  CCC & DDD \\ EEE & FFF \end{tabular}
\end{SCTable}
```

Bsp.
6-5-16

Es ist nicht einfach für das Paket, Legende und Gleitobjekt, also Objekte mit beliebigem Inhalt, visuell korrekt zueinander auszurichten. In den meisten Fällen, wenn das Gleitobjekt z.B. aus einer `tabular`-Umgebung besteht, sind die Ergebnisse zufrieden stellend. Wenn es allerdings fortlaufenden Text enthält, z.B. als Teil einer `minipage`-Umgebung, muss man die Ausrichtung möglicherweise verbessern, etwa mithilfe eines `\strut`-Befehls, der eine Linie ohne horizontale Ausdehnung erzeugt. Dieses Verfahren wird im nächsten Beispiel gezeigt. Der zweite `\strut`-Befehl in der letzten Zeile ist bei einer nach oben bündigen Legende eigentlich nicht erforderlich, sondern nur bei einer nach unten bündigen (also bei `\sidecaptionvpos{table}{b}`).

Das Beispiel zeigt auch die Wirkung der Option `ragged`: Die Legende wird mit Flattersatz links gesetzt, wenn sie im linken Rand erscheint.

		<pre> \usepackage[margincaption,ragged]{sidecap} % \sample wie zuvor definiert \begin{SCTable} \caption{Eine nicht bündige Legende} \begin{minipage}{\linewidth} \sample Und noch eine Zeile für das Beispiel. \end{minipage}\end{SCTable} \begin{SCTable} \caption{Die manuell korrigierte Legende} \begin{minipage}{\linewidth} \strut \sample Und noch eine Zeile. \unskip\strut \end{minipage}\end{SCTable} </pre>
Tabelle 1: Eine nicht bündige Legende	Text für unsere Seite, der immer und immer wieder verwendet wird. Und noch eine Zeile für das Beispiel.	
Tabelle 2: Die manuell korrigierte Legende	Text für unsere Seite, der immer und immer wieder verwendet wird. Und noch eine Zeile.	

Bsp.
6-5-17

6.5.5 fltpage – Legenden auf einer eigenen Seite

Bei großen Abbildungen oder Tabellen verbleibt auf der Seite manchmal nicht mehr genug Platz für die Legende. Das Paket `fltpage` von Sebastian Gross begegnet diesem Problem durch die Umgebungen `FPfigure` und `FPtable`. Sie gleichen den Umgebungen `figure` bzw. `table`, setzen jedoch die Legende für eine ganzseitige Abbildung bzw. Tabelle bei zweiseitigem Druck (`twoside`) auf die gegenüberliegende und bei einseitigem Druck (`oneside`) auf die vorhergehende oder nachfolgende Seite.

Das Verhalten des Paketes lässt sich über eine Reihe von Optionen steuern, welche die Position der Legende zum Gleitobjekt festlegen (Optionen in Klammern sind Alias-Bezeichnungen):

closeFloats Die ganzseitigen Gleitobjekte werden an der *nächstmöglichen* Position ausgegeben. Im `twoside`-Modus wird die Legende am Fuß der gegenüberliegenden Seite platziert; im `oneside`-Modus erscheint sie immer auf der vorangehenden Seite.

rightFloats (CaptionBefore) Das Gleitobjekt erscheint immer auf einer rechten Seite und die Legende auf der Seite davor.

leftFloats (CaptionAfterwards) Das Gleitobjekt erscheint immer auf einer linken Seite und die Legende auf der Seite danach.

Die „isolierte“ Legende zu einem ganzseitigen Gleitobjekt wird durch eine horizontale Linie vom übrigen Text abgesetzt. Diese Linie lässt sich mit der Option `noSeparatorLine` unterdrücken. Um die Verbindung zwischen Legende und Gleitobjekt noch deutlicher herauszustellen, fügt das Paket bei Angabe der Option `varioref` Hinweise wie „Tabelle xx. (auf der gegenüberliegenden Seite)“ hinzu. In diesem Fall werden die entsprechenden Texte mithilfe des `varioref`-Paketes in der Sprache des Dokumentes erzeugt.¹

Das nächste Beispiel zeigt eine einfache Zusammenstellung der Prinzipien, auf denen das `fltpage`-Paket basiert. Im Beispiel wird eine künstliche einseitige Tabelle erzeugt. Dazu wird ein Rahmen mit einer unsichtbaren Linie (der Stärke null) in eine Box gesetzt, deren Abmessungen minimal kleiner sind als die Maße der Seite.² Die Legende wird am Fuß der gegenüberliegenden Seite positioniert. Durch Laden des Paketes `varioref` und Einsatz der Option `varioref` wird der Text „(auf der nächsten Seite)“ vom `fltpage`-Paket automatisch eingefügt.

Abbildungen

1 Sehr groß . . . 6

1 Ganzseitige Float

Abbildung 1 ist ein ganzseitiges Gleitobjekt, mit Legende auf einer anderen Seite.

Abb. 1 (auf der nächsten Seite): Legende für ein ganzseitiges Gleitobjekt, für die auf der Seite kein Platz mehr war.

6

Ganzseitige Abbildung

7

```
\usepackage[twoside,varioref,
  german,closeFloats]{fltpage}

\listoffigures
\section{Ganzseitige Float}
Abbildung~\ref{FP} ist ein
ganzseitiges Gleitobjekt,
mit Legende auf einer
anderen Seite.
\begin{FPfigure}
  \setlength\fbboxsep{0pt}
  \framebox[.97\linewidth][c]
    {\rule[-3cm]{0pt}
      {.97\textheight}%
      Ganzseitige Abbildung}
\caption[Sehr groß]
{Legende für ein
ganzseitiges Gleitobjekt,
für die auf der Seite kein
Platz mehr war.}\label{FP}
\end{FPfigure}
```

Bsp.
6-5-18

Nur eingeschränkt
verwendbar

Leider wird dieses Paket nicht mehr weiterentwickelt. Daher lässt es sich nicht mit anderen Gleitobjektklassen als `figure` und `table` verwenden (wie z.B. mit solchen, die mit dem `float`-Paket definiert wurden). Außerdem kann es zu Problemen kommen, wenn Gleitobjekte im Quelldokument zu nahe beieinander stehen (der Inhalt des zweiten könnte den des ersten überschreiben). Trotzdem liefert es, wenn man es mit Sorgfalt verwendet, eine Lösung für den schwierigen Umgang mit großen Gleitobjekten, wie sie zur Zeit von keinem anderen Paket geboten wird.

¹Wenn das Layout der Legende durch das Paket `caption` angepasst wird, kann es sein, dass diese Funktion fehlschlägt.

²Dieser Schritt ist erforderlich, um zu verhindern, dass die `\framebox`-Linien durch ihre Stärke überlaufende Boxen erzeugen. Der Abstand `\fbboxsep` zwischen Rahmen und Inhalt wird ebenfalls auf null Punkte gesetzt.

KAPITEL 7

Zeichensätze und Kodierungen

7.1 Einleitung	337
7.2 Fontcharakteristika – Zeichensatzmerkmale	342
7.3 Zeichensätze im Text	348
7.4 Fonts in mathematischen Formeln	359
7.5 Die Standardfontunterstützung in $\text{\LaTeX}$	364
7.6 PSNFSS – PostScript Fonts in $\text{\LaTeX}$	381
7.7 Eine Sammlung von Fontpaketen	393
7.8 Die $\text{\LaTeX}$ -Welt der Symbole	411
7.9 Die Low-Level-Schnittstelle	423
7.10 Einbinden neuer Zeichensätze	430
7.11 Kodierungsmodelle in $\text{\LaTeX}$	452
7.12 Dokumente aus dem letzten Jahrtausend	475

7.1 Einleitung

Ein Großteil der Arbeit von $(\text{\La})\text{\TeX}$ als Satzsystem besteht darin, aus dem Quelldokument die Lage der Buchstaben auf der Ausgabeseite zu errechnen. Dabei weiß das System jedoch nur relativ wenig über diese Zeichen. Sie werden hauptsächlich als „black boxes“ betrachtet, die eine bestimmte Höhe über und eine Tiefe unter der Grundlinie sowie eine bestimmte Breite aufweisen. Diese Informationen werden für jeden Zeichensatz separat in einer externen Datei gespeichert, die als $\text{\TeX}$ -Fontmetrik- oder `.tfm`-Datei bezeichnet wird.

Die zu diesen `.tfm`-Dateien gehörenden Buchstabenformen treten erst in Erscheinung, nachdem $(\text{\La})\text{\TeX}$ eine `.dvi`-Datei erstellt hat. Die Daten über die Position der Buchstaben in der `.dvi`-Datei werden über ein Treiberprogramm mit den Informationen über die Buchstabenformen aus einer `.pk`-Datei oder einer Umrissbeschreibung (z.B. PostScript) zusammengeführt. Dieser Treiber erzeugt dann die Darstellung auf dem Ausgabemedium. In der Regel wird für

jedes Ausgabemedium ein separater Treiber benötigt, z.B. für den Bildschirm, für einen Laserdrucker mit geringer Auflösung, sowie für andere Geräte. Bei $\text{\TeX}$ -Versionen wie $\text{\pdfTeX}$ oder $\text{\VTeX}$, die den Arbeitsschritt $\text{\.dvi}$ -Erzeugung auslassen und direkt PDF- oder PostScript-Material ausgeben, stellt sich die Situation etwas anders dar (jedoch sehr ähnlich, soweit es $\text{\LaTeX}$ betrifft). In diesen Fällen werden die Zeichenformen dann „hinzugefügt“, wenn das zugrunde liegende Formatierungsprogramm das endgültige Ausgabeformat erzeugt. Der Treiber ist hier also integriert, aber das grundlegende Konzept ist das Gleiche.

7.1.1 Die Geschichte des $\text{\LaTeX}$ -Fontauswahlverfahrens (New Font Selection Scheme – NFSS)

Als $\text{\TeX}$ 1979 entwickelt wurde, richtete Donald Knuth nur etwa ein Dutzend Zeichensätze (Fonts) für das Programm ein: die „Almost Computer Modern“ Fonts. Da dies die einzigen zur Verfügung stehenden Zeichensätze waren, verwendete man einen direkten Zugriffsmechanismus: mithilfe einiger weniger Befehle konnte man von einem auf den anderen externen Zeichensatz umschalten.

Auch fünf Jahre später, als $\text{\LaTeX}$ erstmals veröffentlicht wurde, hatte sich an dieser Situation nicht viel geändert. Die zusammen mit $(\text{\LaTeX})$ verteilten Zeichensätze hießen nun lediglich nicht mehr „Almost Computer Modern“ sondern „Computer Modern“, wobei letztere nur eine leichte Verbesserung der ersten Fonts darstellten. Entsprechend funktionierte die Zeichensatzauswahl in $\text{\LaTeX}$ nach dem gleichen Schema wie im $\text{\TeX}$ -Basisprogramm; es waren jedoch einige Größenbefehle hinzugefügt worden, mit denen die Buchstaben in 10 festgelegten Größen ausgegeben werden konnten.

Die Zeichensatzauswahl in $\text{\LaTeX}$ basierte also nicht auf einem generischen Konzept. Wurde z.B. ein Gliederungsbefehl (durch den Befehl $\text{\bf}$ in seiner Definition) so angelegt, dass er eine fettere Schrift erzeugte, dann erzeugte z.B. die Anweisung $\text{\sf}$ (für einen serifenlosen Font) in einer derartigen Überschrift keine fett gedruckte, serifenlose Schrift, sondern vielmehr eine normale serifenlose Schrift (das Attribut für Fettdruck wurde ignoriert). Ebenso produzierte $\text{\bf}$ innerhalb von hervorgehobenen Texten nicht die gewünschte fettgedruckte, kursive Schrift, sondern lediglich eine normale, fettgedruckte Antiqua (Serifenschrift, englisch Roman).

$\text{\LaTeX}$ reagierte so, weil alle Befehle für den Fontwechsel, wie etwa $\text{\bf}$, jeweils auf einen festen Zeichensatz zugriffen. Anstatt also nur ein Attribut des gerade geladenen Fonts zu ändern, wurde einfach der gesamte Font ausgetauscht. $\text{\LaTeX}$ erweiterte den Basismechanismus von $\text{\TeX}$ zwar durch die Bereitstellung von Größenbefehlen; das damals zugrunde liegende Konzept hatte jedoch den großen Fehler, dass die Referenztabellen in $\text{\LaTeX}$ fest verknüpft waren, was einen Wechsel zu anderen Zeichensätzen schwierig, wenn nicht sogar unmöglich machte.

Inzwischen kamen die ersten Laserdrucker zu erschwinglichen Preisen auf den Markt und mit ihnen viele Schriftfamilien, wie z.B. die PostScript-Fonts und viele andere. Auch die Anzahl an Zeichensätzen im METAFONT-Quellformat (die für jeden $(\text{\LaTeX})$ -Benutzer frei verfügbar sind) stieg stark

an. Leider gab es jedoch kein einfaches und einheitliches Verfahren, diese neuen Zeichensätze in $\text{\LaTeX}$ zu integrieren. In nahezu allen Fällen implizierte Schriftsatz mit $\text{\LaTeX}$ auch immer die Verwendung der Computer Modern Fonts. Mit dem Befehl `\newfont` konnten zwar einzelne Zeichensätze hinzu geladen werden, diese wurden aber nicht im eigentlichen Sinne integriert: Zum einen reagierten sie nicht auf die $\text{\LaTeX}$ -Größenbefehle, zum anderen war es außerordentlich kompliziert ein ganzes Dokument mit einem solchen Zeichensatz zu setzen.

Es gab ein paar Ansätze, andere Fonts in $\text{\LaTeX}$ zu integrieren. Sie beschränkten sich jedoch auf den Austausch der fest installierten Fonttabellen. Damit war die neue $\text{\LaTeX}$ -Variante genauso unflexibel wie die ursprüngliche Version; es wurde lediglich ein anderer Zeichensatz erzwungen.

Diese unbefriedigende Situation wurde schließlich 1989 mit Erscheinen des von Frank Mittelbach und Rainer Schöpf entwickelten „New Font Selection Scheme“ (NFSS) [124, 126] behoben. Sein Bekanntheitsgrad stieg enorm, nachdem es erfolgreich in $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\LaTeX}$ eingesetzt worden war (siehe Kapitel 8). NFSS basiert auf einem generischen Konzept und ermöglicht die unabhängige Veränderung einzelner Zeichensatzattribute und die einfache Integration weiterer Schriftfamilien in ein $\text{\LaTeX}$ -System. Das Konzept basiert auf fünf Attributen, die unabhängig voneinander eingestellt werden können, um auf verschiedene Fonts, Fontcharakteristika oder Fontfamilien zuzugreifen. Für die Umsetzung dieses Konzeptes wurden einige $\text{\LaTeX}$ -Befehle umdefiniert und andere hinzugefügt.

Einige Zeit später programmierte Mark Purtil ein Prototyp für skalierbare Zeichensätze. Auf dieser Basis entwarf und implementierte Frank Mittelbach NFSS2 und integrierte dabei unter anderem Entwicklungen von Sebastian Rahtz (über PostScript-Fonts) und einigen anderen Personen. Diese Version wurde 1994, als die aktuelle $\text{\LaTeX}$ -Version ($\text{\LaTeX} 2_{\epsilon}$) heraus kam, zum Standard-Fontauswahlverfahren.

Dieses System ist nun weltweit seit über zehn Jahren im Gebrauch und hat sich als stabil und nützlich erwiesen, auch wenn von einigen Seiten Erweiterungen als wünschenswert angesehen werden. Das $\text{\LaTeX}$ -Projektteam würde solche experimentellen Erweiterungen in Form externer Pakete begrüßen. Diese könnten dann zu einem späteren Zeitpunkt in einen Nachfolger des jetzigen Systems integriert werden.

7.1.2 Eingabe- und Ausgabekodierungen

Die größere Auswahl an Fonts machte im Nebeneffekt unter anderem deutlich, dass zwei miteinander verwandte Bereiche in $\text{\TeX}$ nicht mehr fest miteinander verknüpft werden sollten: die Eingabe- und die Ausgabekodierung (d.h. die Fontkodierung).

Beim Drücken einer Taste auf einer Tastatur wird (normalerweise) eine 8-Bit-Zahl erzeugt, die für ein bestimmtes Zeichen steht. Die Eingabekodierung beschreibt dabei, welche Zahl welchem Zeichen entspricht. Bei verschiedenen nationalen Tastaturen oder Betriebssystemen können diese Entsprechungen stark voneinander abweichen. Auf der deutschen Tastatur, die einer der Autoren für diesen Text verwendete, erzeugt die Taste „ä“ z.B. unter Linux oder Windows die 8-Bit-Zahl „228“, unter MS-DOS jedoch die Zahl „132“.

Wird ein Dokument in einer Datei gespeichert, dann verbleiben nur die 8-Bit-Zahlen als Information über die einzelnen Zeichen; die Eingabekodierung selbst wird nicht explizit gespeichert. Wenn man also eine Datei in eine andere Umgebung überträgt, wie z.B. aus den USA nach Großbritannien, dann kann es sein, dass beispielsweise die Dollarzeichen im Dokument plötzlich als Pfund-Zeichen interpretiert werden, wenn man die Datei mit einem Programm (Editor) betrachtet, das von der falschen Eingabekodierung ausgeht.

Als Abhilfe für Probleme mit der Eingabekodierung entwickelte das L^AT_EX-Projektteam in den Jahren 1994–1995 das `inputenc`-Paket. Mit seiner Hilfe können Anwender die Eingabekodierung für Dokumente oder Teile von Dokumenten explizit deklarieren. Dadurch lassen sich diese sicher von einer L^AT_EX-Installation auf eine andere übertragen und man erhält identische Ausdrücke.¹

Das Konzept der
Eingabekodierung

Das Paket `inputenc` interpretiert die 8-Bit-Zahlen in der Datei (welche für die Zeichen stehen) und setzt diese in „interne L^AT_EX-Entsprechungen“ um, die alle Zeichen, die sich in L^AT_EX darstellen lassen, eindeutig (wenn auch in einem Ad-hoc-Verfahren) adressieren. Zur weiteren Verarbeitung, wie beispielsweise zum Schreiben in eine Hilfsdatei, verwendet L^AT_EX ausschließlich diese interne Darstellung, um jegliche Fehlinterpretation zu vermeiden.

Zu irgendeinem Zeitpunkt muss L^AT_EX diese internen Entsprechungen jedoch wieder mit den Glyphen (d.h. den tatsächlichen Zeichenformen in bestimmten Fonts) verknüpfen, so dass erneut eine Zuordnung stattfindet. T_EX-Zeichensätze verfügen über maximal 256 Glyphen. Diese werden nicht über Namen sondern über 8-Bit-Zahlen adressiert, welche die Lage der Zeichen in der Zeichensatztafel beschreiben (d.h. ein großer eindeutiger Namensraum wird auf mehrere kleinere abgebildet). Und es ist sicher keine große Überraschung, dass diese Zeichenpositionen wiederum sehr variieren können.

Wenn man also die Bedeutung des Dollarzeichens korrekt aus der externen Datei an L^AT_EX übergeben hat, kann man immer noch ein falsches Zeichen auf dem Papier erhalten, sobald man für den Ausdruck einen Font wählt, der an der Position (Slot), an der man eigentlich das Dollarzeichen vermutet, ein unerwartetes Zeichen enthält.² NFSS muss entweder dafür sorgen, dass die internen Zeichendarstellungen von L^AT_EX für die Ausgabe korrekt umgesetzt werden, oder, falls das aus irgendeinem Grund nicht möglich ist, dass der Anwender eine ordentliche Fehlermeldung erhält.

Zusammengesetzte
Zeichen mit Akzent
verhindern
Silbentrennung

Wenn Zeichensätze anstelle von Basis- und Akzentzeichen (die T_EX erst zu Akzentbuchstaben zusammensetzen muss), bereits fertig mit Akzenten versehene Zeichen enthalten, dann sollte man diese Zeichen bevorzugt verwenden, da sie normalerweise besser aussehen. Auch technische Gründe sprechen dafür: Der T_EX-Basisbefehl `\accent` unterdrückt die Silbentrennung. Ein solcher Mangel kann noch hingenommen werden, wenn Akzentbuchstaben, wie im Englischen, nur selten vorkommen. Wenn aber zum Beispiel in einem

¹Es gibt noch andere Lösungsansätze für dieses Problem. Von manchen Seiten wird für den Einsatz von Übersetzungstabellen geworben, die fest in T_EX verankert sind. Das funktioniert jedoch nur, solange alle Personen, die Dokumente untereinander austauschen, die gleichen festen Tabellen verwenden.

²Das Beispiel des \$-Zeichens, das sich in ein £-Zeichen verwandelt, ist dabei kein Produkt der Phantasie: einige der Originalzeichensätze in T_EX haben diesen merkwürdigen Aufbau und Knuth [87, S. 339] empfiehlt sogar, das Pfund-Zeichen mit `{\it\$}` zu erzeugen.

französischen Text alle Wörter mit Akzenten nicht getrennt werden, entwickelt sich das Umbrechen von Zeilen schnell zu einem Alptraum.

Um den verschiedenen Möglichkeiten gerecht zu werden, muss ein Befehl wie `\'e` (die interne $\text{\LaTeX}$ -Entsprechung für den Akzentbuchstaben $\acute{e}$) manchmal eine Reihe komplexer Abläufe initiieren, die auch den Basisbefehl `\accent` einschließen. In anderen Fällen informiert er den Absatzalgorithmus lediglich darüber, dass hier die Glyphen aus einem bestimmten Slot des aktuellen Fonts eingefügt werden soll.

$\text{\LaTeX}$ bewältigt diese Aufgaben mit dem Konzept der Fontkodierungen auf zwei Arten: Entweder werden die $\text{\LaTeX}$ -internen Zeichendarstellungen den entsprechenden Glyphenpositionen zugeordnet, oder die erforderlichen Zeichen werden erst noch aus den Glyphen des Fonts zusammengesetzt, der für den Ausdruck verwendet wird. Obwohl das Konzept der Fontkodierung in NFSS2 bereits vollständig umgesetzt war, dauerte es noch einige Jahre, bis mit der aktuellen Version Ruhe einkehrte (die internen Abläufe wurden in dem Maße, in dem die Entwickler zunehmend Einblick in die entsprechende Problematik erlangten, immer wieder umprogrammiert).

*Das Konzept der
Fontkodierung*

* * *

Die folgenden Abschnitte beschreiben die Version 2 von NFSS, die Ende 1992 fertig gestellt wurde und seit 1994 ein fester Bestandteil von Standard- $\text{\LaTeX}$ ist. Soweit es die Benutzerschnittstelle betrifft, soll es auch in $\text{\LaTeX}$ 3 integriert werden.

Hier werden zunächst die Konzepte der Fontcharakteristika im Allgemeinen erläutert und die wichtigsten Attribute vorgestellt, die $\text{\LaTeX}$ für den orthogonalen Zeichensatzwechsel benutzt. Danach folgt eine Beschreibung der High-Level-Schnittstelle – d.h. der Befehle, die der Benutzer normalerweise verwenden sollte. Dazu gehören die Befehle für den Text (Abschnitt 7.3), besondere Funktionen für mathematische Formeln (Abschnitt 7.4), sowie ein Überblick über die grundlegenden Support- und Erweiterungspakete für NFSS – und zwar diejenigen, die mit $\text{\LaTeX}$ verteilt werden (Abschnitt 7.5). Auch die Pakete und Befehle, die sich mit den zuvor erwähnten Kodierungsaufgaben befassen, werden abgedeckt.

Einer der größten Vorteile, die das $\text{\LaTeX}$ -Fontauswahlverfahren bietet, ist die Leichtigkeit, mit der neue Zeichensätze für den Text eingebunden werden können. Neben den Computer Modern Schriftfamilien, die standardmäßig verwendet werden, lassen sich durch Angabe des entsprechenden Paketes in der Präambel einfach weitere Schriftfamilien hinzufügen. Voraussetzung für die erfolgreiche Verarbeitung und für das Ausdrucken ist natürlich, dass die zugehörigen Fontdateien (z.B. die `.tfm`-, `.pk`-, Type1- oder TrueType-Dateien) auf dem System installiert sind. Die nächsten drei Abschnitte behandeln größere und kleinere Fontpakete. Abschnitt 7.6 befasst sich mit der Standard-PostScript-Unterstützung für $\text{\LaTeX}$, PSNFSS, die zu den erforderlichen Paketen jeder $\text{\LaTeX}$ -Distribution gehört.

Danach folgt eine Sammlung anderer interessanter Pakete zum Anpassen der Grundschriften eines Dokumentes (Abschnitt 7.7) sowie eine Einführung in die $\text{\LaTeX}$ -Welt der Sonderzeichen (Abschnitt 7.8). Alle beschriebenen Pakete sind kostenlos und die meisten (wenn nicht sogar alle) sind Teil einer

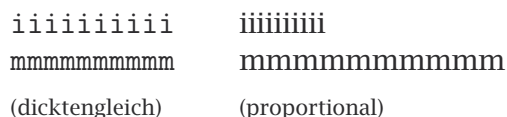


Abbildung 7.1: Wichtige Fontcharakteristika

modernen Distribution von L^AT_EX. Es werden auch einige Hinweise auf die Unterstützung kommerzieller Zeichensätze gegeben.

Der letzte Teil dieses Kapitels beschäftigt sich mit Low-Level-Schnittstellen, die für die Definition komplexer neuer Befehle nützlich sind, und besonders dann wichtig werden, wenn man L^AT_EX neue Zeichensätze zur Verfügung stellen will. Dazu gehören Low-Level-Befehle zum Ändern individueller Fontattribute (Abschnitt 7.9), sowie Befehle zum Einrichten neuer Fonts in L^AT_EX (Abschnitt 7.10). Außerdem werden die L^AT_EX-Kodierungsmodelle für Texte und mathematische Formeln besprochen (Abschnitt 7.11). Das Kapitel schließt mit einem Abschnitt, der sich mit der Kompatibilität zu sehr alten L^AT_EX-Dokumenten befasst.

7.2 Fontcharakteristika – Zeichensatzmerkmale

Zeichensätze können nach verschiedenen Kriterien in sich überschneidende Klassen eingeteilt werden. Bei der Entscheidung, welche Schriftfamilie in einem bestimmten Kontext verwendet werden soll, kann es häufig sehr hilfreich sein, diese Merkmale oder Charakteristika zu kennen. Als weiterführende Literatur zu diesem Thema empfehlen sich zum Beispiel die Bücher [28, 41, 118] oder der Artikel [54].

7.2.1 Dicktengleiche und Proportionalschriften

Zeichensätze können entweder aus Zeichen mit unveränderlicher (dicktengleich) oder zueinander proportionaler Breite bestehen. In dicktengleichen Schriften nimmt jedes Zeichen immer den gleichen horizontalen Raum ein, egal welche Form es hat. Im Gegensatz dazu hängt die Breite eines Buchstabens in einer Proportionalschrift von seiner Form ab. Abbildung 7.1 zeigt, dass ein „i“ einer dicktengleichen Schrift genauso viel Raum einnimmt wie ein „m“, während es in einer Proportionalschrift wesentlich schmaler ist. Folglich passen bei Verwendung von Proportionalschriften (auch typographische Schriften genannt) in der Regel mehr Wörter auf eine Seite und sie sind leichter lesbar als dicktengleiche Schriften. Die unterschiedlichen Buchstabenzwischenräume bei dicktengleichen Fonts erschweren es dem Auge nämlich, Wortgrenzen zu erkennen, wodurch der Text schwieriger zu entziffern ist.

Aber auch dicktengleiche Schriften haben ihre Berechtigung. Im geeigneten Zusammenhang können sie durchaus die Qualität eines Dokumentes erhöhen. In Tabellen oder Computerlisten, wo es auf eine genaue Ausrichtung der Daten ankommt, ist eine dicktengleiche Schrift die erste Wahl. In Computerhandbüchern ist es üblich, Programmcode in diesen Schriften darzustellen, um ihn deutlich vom übrigen Text abzuheben.



Abbildung 7.2: Buchstaben mit und ohne Serifen im Vergleich

Doch dicktengleiche Schriften werden nicht nur zur Auszeichnung verwendet. Man kann sie auch als Grundschrift für den gesamten Text nehmen. Dann sieht der Text aus, als wäre er von Hand auf einer Schreibmaschine geschrieben. In manchen Situationen ist dieses Aussehen mit Absätzen ohne Randausgleich passender als die professionell erscheinende Proportional-schrift. Man sollte jedoch beachten, dass dicktengleiche Schriften im Blocksatz eher kläglich wirken. (In Abschnitt 3.1.11 wird erklärt, wie sich der Blocksatz ein- und ausschalten lässt.)

7.2.2 Schriften mit und ohne Serifen

Eine weitere nützliche Einteilung ist diejenige in Serifenschriften und serifenlose Schriften. Serifen sind die kleinen Quer- und Haarstriche am Ende eines Schriftstrichs (vgl. Abbildung 7.2). Ihren Ursprung haben Serifen in römischen Inschriften, die in Stein gemeißelt wurden. Daher werden sie häufig als Antiqua oder „Roman“- (römische) Schrift bezeichnet.

Traditionell wurde diese Schrift für längere Texte benutzt, da es hieß, dass sie damit leichter zu lesen seien. Lange Zeit dachte man, dass die Serifen den Augen mehr Anhaltspunkte zur Identifizierung der Buchstaben lieferten. Dies trifft sicherlich zu, wenn man nur Teile der Buchstaben sieht. Neuere Studien [151] ergaben jedoch, dass die Lesegeschwindigkeit durch Abwesenheit der Serifen nicht beeinträchtigt wird, wenn der gesamte Text sichtbar ist.

7.2.3 Schriftfamilien und ihre Attribute

Neben der groben Klassifikation von Schriften nach Serifen und Dichte gibt es eine Einteilung der Schriften in Schriftfamilien. Mitglieder einer Familie folgen denselben Designmaßstäben und unterscheiden sich lediglich in Schriftgrad (Größe der Zeichen), Schriftstärke (Stärke der Linien), Schriftbreite (durchschnittliche Breite der Zeichen) und Schriftform (Gestaltung der Buchstaben).

Schriftformen

Bei der Klassifizierung eines Mitglieds einer Schriftfamilie ist seine Form ein sehr wichtiges Merkmal. Es ist häufig eine Frage der persönlichen Vorliebe, ob eine Reihe von Zeichensätzen mit verschiedenen Formen eine oder mehrere Schriftfamilien bilden sollte. Donald Knuth bezeichnete beispielsweise seine Sammlung von 31 Computer Modern Fonts als eine Familie [91]. Im traditionellen Sinne bilden sie eigentlich eher eine aus vielen einzelnen zusammengesetzte Metafamilie.¹

¹Mithilfe des Designtools METAFONT lassen sich aus derselben Quelldatei die unterschiedlichsten Schriften erstellen. Es verwundert also nicht, dass 1989 auf Basis der Quelldateien für die Computer Modern Fonts eine weitere Schriftfamilie geschaffen wurde [96]. Diese Schriftfamilie, Concrete Roman, entstand einfach durch Verändern einiger METAFONT-Parameter in den Quelldateien. Da das Endergebnis jedoch so unterschiedlich war, beschloss Knuth, dieser Schriftfamilie einen anderen Namen zu geben.

A	B	C	a	b	c	x	y	z
A	B	C	a	b	c	x	y	z
<i>A</i>	<i>B</i>	<i>C</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>x</i>	<i>y</i>	<i>z</i>

Abbildung 7.3: Aufrechte und kursive Zeichen im Vergleich

Es gibt zwar keine einheitlichen Konventionen für die Bezeichnung von Schriftformen, aber das ist unwichtig, solange man sich in L^AT_EX an ein bestimmtes Schema hält.

Die aufrechte Form

Nahezu jede Fontfamilie enthält eine so genannte „aufrechte“ Variante.¹ Innerhalb der Schriftfamilie, die hauptsächlich in diesem Buch verwendet wird (Lucida Bright), ist der z.B. vorliegende Text in einer aufrechten Schriftform gesetzt.

Die kursive Schriftform

Eine weitere wichtige Schriftform, die es in fast allen Schriftfamilien gibt, ist die *Kursivschrift* (englisch: italic). Sie ist nach rechts geneigt und unterscheidet sich zudem auch in ihrer Buchstabenform von ihrem aufrechten Gegenstück (vgl. Abbildung 7.3). In der ersten Zeile dieser Abbildung erscheinen Buchstaben der Computer Modern Serifenfamilie in aufrechter Form. In der dritten Zeile sind dieselben Buchstaben kursiv dargestellt. Zum besseren Vergleich stehen in der zweiten Zeile die Kursivbuchstaben ohne die übliche Neigung, d.h. die Buchstaben wurden künstlich aufgerichtet.

Geneigte oder oblique Form

Serifenlose Schriftfamilien enthalten häufig keine echte Kursivform, sondern nur eine geneigte, die sich aber ansonsten in nichts von ihrem aufrechten Gegenstück unterscheidet. Diese Form wird häufig auch als „slanted“, „sloped“ oder „oblique“ bezeichnet.

Kapitälchen

Vorgetäuschte Kapitälchen

Eine weitere gebräuchliche Variante sind die „Kapitälchen“, bei denen alle Kleinbuchstaben als Großbuchstaben mit reduzierter Höhe dargestellt werden, wie in Abbildung 7.4 auf der gegenüberliegenden Seite zu sehen. Wenn diese Form in einer bestimmten Familie nicht vorhanden ist, verwenden Typographen manchmal Großbuchstaben eines kleineren Schriftgrades.² Mit diesem Verfahren erzielt man jedoch nicht dieselbe Qualität wie mit eigens gestalteten Kapitälchen. Die echten Kapitälchen haben eine andere Schriftbreite und -stärke als die Großbuchstaben des gleichen Zeichensatzes, die lediglich auf die gewünschte Höhe der Kapitälchen reduziert wurden (in Abbildung 7.4 ist deutlich zu erkennen, dass die Striche der „falschen“ Kapitälchen viel zu dünn sind).

Die Frage, ob „Kapitälchen“ als eine Schriftform oder als eigenständiges Attribut anzusehen sind, ist noch in der Diskussion. Bei letzterer Interpretation verfügen Fonts über ein „Buchstabenart“-Attribut, das entweder gemischte Schreibung (der Normalfall), reine Großschreibung (Versalien), Kapi-

¹Im Englischen wird die aufrechte Schriftform manchmal auch als „Roman“ bezeichnet. Dies liegt daran, dass bis vor kurzem die Grundschrift von Dokumenten fast immer eine Serifenschrift war. Viele Leute hielten „Roman“ deshalb für das Gegenteil von kursiv. Dementsprechend wird der Begriff in einigen Büchern als Bezeichnung für eine aufrechte Schriftform und nicht für eine Serifenschrift verwendet.

²Als Faustregel gilt hier, Großbuchstaben aus einem Zeichensatz zu verwenden, der etwa 10% größer ist als die x-Höhe (Höhe eines x) des ursprünglichen Zeichensatzes. Abschnitt 7.10.3 auf Seite 440 beschäftigt sich ausführlicher mit der Frage, wie man die x-Höhe eines T_EX-Zeichensatzes bestimmen kann.



Abbildung 7.4: Echte und falsche Kapitälchen im Vergleich



Abbildung 7.5: Outline- und schattierte Schriftformen

tälchen oder reine Kleinschreibung bedeuten kann. Für bestimmte Fontfamilien wäre dies sicher die bessere Lösung, aber zurzeit unterstützt die $\text{\LaTeX}$ -Fontauswahl nur vier Attributachsen, in denen Kapitälchen als eine Ausprägung des Schriftform-Attributs geführt werden.¹

Es gibt noch einige weitere, allerdings weniger wichtige Schriftformen. Einige Familien enthalten Schriftschnitte, bei denen das Innere der Buchstaben besonders gestaltet ist. Die wichtigste unter ihnen ist wohl die „Umriss“- („Outline“-) Form, bei der das Innere der Buchstaben leer ist. Zu Darstellungszwecken gibt es in einigen Schriftfamilien auch so genannte „schattierte“ Formen, d.h. die Buchstaben erscheinen dreidimensional. Abbildung 7.5 zeigt einige Beispiele dieser Formen und Tabelle 7.26 auf Seite 426 zeigt die internen NFSS-Namen.

Durch Setzen einiger METAFONT-Parameter auf ganz bestimmte Werte, ließen sich besondere Varianten der Computer Modern Metafamilie erzeugen. So gibt es z.B. eine aufrechte Kursivschrift, d.h. eine Schriftform, in der die Buchstaben aufrecht stehen, sonst aber die typische Form der Kursivbuchstaben besitzen (siehe die zweite Zeile in Abbildung 7.3 auf der vorherigen Seite). Diese Form wurde hauptsächlich entwickelt, um die Leistungsfähigkeit von METAFONT zu demonstrieren. Vielleicht entdeckt aber auch der ein oder andere Anwender seine Vorliebe für solch ungewöhnliche Schriftformen.

Schriftbreite und Schriftstärke

Die verschiedenen Schriftformen einer Familie können sich in ihrer Schriftstärke voneinander unterscheiden. Dieses Merkmal bezeichnet die Stärke der Striche, aus denen die einzelnen Buchstabenformen bestehen. Auch hier gibt es keine einheitlichen Namenskonventionen, aber die Einteilung in geeignete Klassen ist recht einfach. Einige Schrifthersteller bezeichnen die im normalen Text verwendeten Schriftstärken als „book“ (normal, Buch), andere als „medium“ (normal). Dünne Striche werden „leicht“ oder „mager“ genannt, während dickere meist als „fett“ bezeichnet werden. Bei großen Schriftfamilien ist häufig eine detaillierte Einteilung notwendig, so dass die Bandbreite hier

¹In einigen Fällen sind Kapitälchen sogar als eigenständige Fontfamilie angelegt, um Kombinationen wie kursive Kapitälchen zu ermöglichen.

manchmal von „ultraleicht“ über „extraleicht“, „leicht“, „mager“, usw. bis hin zu „ultrafett“ geht. Es kann aber auch vorkommen, dass in einer Schriftfamilie nur sehr wenige unterschiedliche Schriftstärken existieren. In der Schriftfamilie Computer Modern Roman gibt es beispielsweise nur zwei Schriftstärken: „normal“ und „fett“.

Ein weiteres wichtiges Fontattribut ist die „Breite“, d.h. der Platz, den ein gestreckter oder gestauchter Buchstabe im Vergleich zu seiner normal (medium) breiten Form einnimmt. In der Computer Modern Roman Familie gibt es eine fette Schrift in „normaler Breite“ und in „weiter Breite“. Schmale Fonts eignen sich sehr gut für Titel und Überschriften, bei denen normal breite Schriften in hohen Schriftgrößen zuviel Raum einnehmen würden. Einige Satzsysteme sind sogar in der Lage, die Buchstaben automatisch zu stauchen, damit sie in einen vorgegebenen Freiraum hinein passen, z.B. um eine Zeile in einer Überschrift genau auszufüllen. Über diese Fähigkeit verfügt (L^A)T_EX nicht; die Ergebnisse sind allerdings auch oft in ästhetischer Hinsicht recht fragwürdig.

Die NFSS-Konventionen zur Benennung von Schriftstärken bzw. Schriftbreiten findet man in Tabelle 7.25 auf Seite 425.

Schriftgrade

Die Schriftgröße (auch Schriftgrad oder Schriftkegel genannt) wird im anglo-amerikanischen Raum traditionell in Pica-Punkten (pt) angegeben. 72.27 Punkte entsprechen einem Zoll (Inch).¹ Die Schriftgröße ist kein absolutes Maß irgendeines speziellen Merkmals. Sie stellt vielmehr einen Orientierungswert des Schriftentwicklers für den Benutzer dar. Bei einer 10pt-Schrift sind die Buchstaben beispielsweise in der Regel kleiner als 10pt, und nur Zeichen wie etwa Klammern weisen ungefähr diese Höhe auf.

Zwei gleich hohe Schriften müssen nicht notwendigerweise immer zueinander passen, denn das endgültige Erscheinungsbild einer Schrift hängt von vielen verschiedenen Faktoren ab, wie z.B. der Höhe der Kleinbuchstaben (der x-Höhe), der Strichstärke, der Tiefe der Unterlängen (dem Teil der Buchstaben unterhalb der Grundlinie wie z.B. im Buchstaben q), usw.

Im (L^A)T_EX-Bereich sind Zeichensätze häufig in Schriftgraden vorhanden, die durch Multiplikation der Basisgröße mit 1.2 entstehen, also in geometrischer Progression [87, S. 17]. Dieses Konzept wurde gewählt, weil dadurch die Erstellung einer vergrößerten Masterkopie, die später wieder photographisch verkleinert werden kann, erleichtert wird. Dadurch lässt sich die Auflösung des endgültigen Mediums erheblich verbessern. Wenn beispielsweise eine Broschüre im DIN A5 Format erstellt werden soll, könnte man sie mit einem Vergrößerungsfaktor von $1.44 \approx \sqrt{2}$ auf A4 Papier ausdrucken. Bei einer photographischen Verkleinerung eines Ausdrucks mit einer herkömmlichen Laserdrucker-Auflösung von 600dpi (dots per inch = Punkte pro Zoll) hätte der endgültige Ausdruck eine Auflösung von 864dpi. Damit ist er qualitativ hochwertiger als das Produkt, das man normalerweise mit einem solchen Laserdrucker erzielen kann.

¹PostScript verwendet ein etwas anderes Mess-System, in dem 72 Punkte ein Zoll ergeben. Diese Einheiten, die manchmal auch „big points“ (große Punkte) genannt werden, sind in T_EX als bp verfügbar.

Eine 10pt-Schrift sieht anders aus als eine vergrößerte
5pt-Schrift

Abbildung 7.6: Skalierte und entworfene Schriften (Computer Modern)

Dieses geometrische Vergrößerungsschema, das normalerweise bei den mit dem Programm METAFONT erstellten (L^A)T_EX-Zeichensätzen verwendet wird, ist im professionellen Umfeld jedoch eher unüblich. Gewöhnlich werden die Punktgrößen 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 24, 30 und 36 verwendet. Allerdings sind nicht alle Zeichensätze in allen Größen verfügbar; und manchmal werden auch zusätzliche Schriftgrade angeboten, z.B. für sehr große Überschriften oder sehr kleine Hoch- und Tiefstellungen. Der Bedarf an festen Größen hat seinen Ursprung in der verwendeten Technik. Um in einer bestimmten Größe drucken zu können, mussten die entsprechenden gegossenen Lettern verfügbar sein. In der heutigen digitalisierten Welt liegen die Fonts normalerweise als Vektorgraphiken vor und können damit nach Bedarf skaliert werden. Daher werden viele kommerzielle Fontfamilien heutzutage nur noch in einer einzigen Entwurfsgröße ausgeliefert.

Der Einsatz vergrößerter und verkleinerter Fonts anstelle von Zeichensätzen, die für eine bestimmte Größe entworfen wurden, liefert häufig nur ein mäßiges bis unbefriedigendes Ergebnis. Für das menschliche Auge verändern sich Zeichensätze nämlich nicht linear. Von Hand entworfene Zeichensätze mit großem Schriftgrad sind in der Regel schmaler als vergrößerte Zeichensätze derselben Familie. Auch wenn es innerhalb eines kleinen Bereiches durchaus akzeptabel sein kann, Schriften zu vergrößern oder zu verkleinern, sollte man doch möglichst immer Zeichensätze verwenden, die eigens für die gewünschte Größe entworfen wurden. Der Unterschied zwischen skalierten und für eine Größe entworfenen Zeichensätzen ist in Abbildung 7.6 dargestellt; die Abweichungen sind zugegebenermaßen nicht immer so deutlich zu erkennen.

7.2.4 Fontkodierungen

Wie in der Einleitung zu diesem Kapitel erwähnt, adressiert T_EX die Zeichen eines Fonts über 8-Bit-Zahlen. Eine solche Zuordnung wird Fontkodierung genannt. Von L^AT_EX aus gesehen, sind zwei Zeichensätze der gleichen Fontkodierung austauschbar, da sie bei der gleichen Eingabe im Ausdruck die „gleichen“ Zeichen erzeugen. Um zu demonstrieren, was passiert, wenn man einen Zeichensatz mit einer Kodierung verwendet, die nicht der des Eingabemediums entspricht, ist hier noch einmal der erste Satz dieses Abschnitts (im Zapf Dingbats Font) abgedruckt:



Das Ergebnis ist ein interessantes Muster, aber nicht wirklich das, was man in einem normalen Dokument sehen möchte.

Wenn man Fonts nach ihrer Fontkodierung einteilt, kann man andere Charakteristika, wie ihre Familie oder Serie ändern, ohne dass man Gefahr läuft, kryptische Ausdrücke zu erhalten.

OT1-Kodierung Die ursprünglich mit T_EX herausgegebenen Zeichensätze bestanden aus jeweils nur 128 Zeichen, so dass sie keinen Platz für eigenständige Akzentbuchstaben hatten. Stattdessen müssen derartige Zeichen mithilfe des T_EX-Basisbefehls `\accent` oder ähnlicher Verfahren zusammengesetzt werden. Das führt dazu, dass Wörter mit diakritischen Zeichen von L^AT_EX nicht automatisch getrennt werden können. Auch das automatische Unterschneiden (englisch: Kerning; die Korrektur des Zwischenraums zwischen bestimmten Zeichen eines Fonts) funktioniert bei ihnen nicht. Die Kodierung dieser Zeichensätze wird als OT1 bezeichnet. Auch wenn OT1 weiterhin die voreingestellte Kodierung für L^AT_EX ist, sollte man sie besser nicht für andere Sprachen als Englisch oder Latein verwenden.

T1-Kodierung Als Alternative definierte die T_EX-Anwendergemeinschaft eine Kodierung namens T1 mit 256 Zeichen, die es T_EX erlaubt, über 30 Sprachen, die auf der Lateinischen Ausgangsschrift basieren, korrekt (mit Silbentrennung und Kerning) zu setzen (Abschnitt 7.5.1 auf Seite 365 enthält hierzu nähere Details). Daher wird dringend empfohlen, diese Kodierung zu verwenden. Heutzutage sind fast alle Fontfamilien, die sich mit L^AT_EX einsetzen lassen, in dieser Kodierung erhältlich; einige liegen sogar *nur* in der T1-Kodierung vor. Wenn man nach dem `\documentclass`-Befehl `\usepackage[T1]{fontenc}` angibt, wird T1 zur Standardkodierung. Das Paket `fontenc` wird in Abschnitt 7.5.3 ausführlicher behandelt. Nähere Informationen zu Fontkodierungen finden sich auf Seite 427 und in Abschnitt 7.11 auf Seite 452.

7.3 Zeichensätze im Text

Beim Schreiben eines Textes in L^AT_EX wird der richtige Zeichensatz in der Regel automatisch durch die (logischen) Markup-Befehle bestimmt, die zur Strukturierung des Dokumentes eingesetzt werden. Die Fontattribute für Abschnittsüberschriften, wie etwa große, fettgedruckte Zeichen, werden durch die Dokumentenklasse definiert und immer dann aktiviert, wenn etwa ein `\section`-Befehl verwendet wird, so dass der Anwender die Attribute nur im Ausnahmefall selbst verändern muss.

Manchmal ist es jedoch notwendig, die Attribute auf direktem Wege festzulegen. Dies ist z.B. der Fall, wenn man alle Fontattribute gleichzeitig ändern will, um etwa für die Grundschrift eine vollkommen andere Schriftfamilie auszuwählen. Diese Änderung kann häufig schon einfach durch Angabe eines geeigneten Paketes vorgenommen werden (Abschnitte 7.6 und 7.7 enthalten Beschreibungen solcher Pakete).

Ein weiterer Grund für explizite Attribute kann die Kennzeichnung bestimmter Passagen sein, z.B. zur deutlicheren Hervorhebung von Beispielen, Abkürzungen oder Firmennamen. In diesem Buch werden beispielsweise Paketnamen in einer serifenlosen Schrift dargestellt. Für diese Formatierung kann man einfach den Namen in den Befehl `\textsf{. . .}` einsetzen; es ist jedoch eleganter, dazu einen neuen Befehl (wie etwa `\LPack`) zu definieren, um dem Quelldokument zusätzliche Informationen mitzugeben. Durch das De-

finieren einzelner Befehle für logisch unterschiedliche Elemente eines Dokumentes (auch wenn diese momentan nicht speziell formatiert werden) lassen sich nachträgliche Änderungen wesentlich einfacher einheitlich durchführen.

Und manchmal möchte man vielleicht auch einfach eine Entscheidung der Dokumentenklasse außer Kraft setzen. Eine Tabelle soll z.B. kleiner gesetzt werden, damit sie noch auf eine Seite passt. Das hat durchaus seine Berechtigung, denn ein Text lässt sich nur bis zu einem gewissen Grad automatisch formatieren. Für die endgültige Version sind daher häufig noch manuelle Formatierungen – wie etwa das Einfügen von Seitenumbrüchen – erforderlich. Durch explizite Formatierungen wird ein Dokument bei späterer Nachbearbeitung aber auch fehleranfälliger und schwieriger zu handhaben. Deshalb sollte man direkte Formatierungsbefehle für den Zeichensatzwechsel, genau wie alle anderen visuellen Formatierungsanweisungen, so selten wie möglich verwenden.

7.3.1 Standardfontbefehle in L^AT_EX

Die Schrift, die man für den überwiegenden Teil eines Dokumentes verwendet, wird „Grundschrift“, „Basisschrift“ oder „Brottschrift“ genannt. Sie wird zu Beginn des Textes und an einigen anderen Stellen, wie z.B. in Fußnoten und Abbildungen, automatisch ausgewählt. Bei einigen logischen Elementen wie etwa in Abschnittsüberschriften, wird je nach Dokumentenklasse die Schriftart oder -größe automatisch geändert. Diese Änderungen laufen im Hintergrund ab. Die Aufgabe des Autors beschränkt sich dabei lediglich darauf, das Dokument mit dem richtigen logischen Markup zu versehen. Wenn jedoch Teile des Textes durch Auswahl geeigneter Schriften manuell hervorgehoben werden sollen, dann können dazu die im Folgenden beschriebenen Befehle verwendet werden.

Die meisten Fontwechselbefehle gibt es in zwei Formen: als Befehl mit einem Argument wie z.B. `\textbf{...}` und als Deklaration wie z.B. `\bfseries`. Deklarationen verfügen nicht über Argumente. Sie weisen L^AT_EX vielmehr an, die momentanen Einstellungen zu ändern und diese ab sofort (bis zum Ende der aktuellen Umgebung oder bis zu einer schließenden Klammer) beizubehalten. Deshalb sollte man keinen Befehl wie `\bfseries{...}` schreiben, weil dadurch der gesamte nachfolgende Text bis zum Ende der Umgebung fettgedruckt wird.

Um einzelne Wörter oder kurze Passagen durch eine andere Schrift hervorzuheben, sollte man die Fontbefehle mit Argument verwenden. Deklarationen sind eher für die Definition neuer Umgebungen oder Befehle geeignet. Wie das folgende Beispiel zeigt, kann man für längere Passagen auch die Umgebungsvariante der Deklaration (Deklarationsname ohne vorhergehenden Backslash) benutzen:

Einige Wörter in diesem Satz sind **fett gedruckt**.

Der Fettdruck geht hier weiter, der Rest des Absatzes wird aber normal gesetzt.

Einige Wörter in diesem Satz sind
`\begin{bfseries}`fett gedruckt.

Der Fettdruck geht hier weiter,`\end{bfseries}`
der Rest des Absatzes wird aber normal
gesetzt.

Fontbefehle mit einem Argument erlauben jedoch keine Absatzumbrüche in ihren Argumenten. In Abschnitt 7.3.3 auf Seite 355 werden die Vor- und Nachteile der Befehle und Deklarationen gegeneinander abgewägt.

Die Grundschrift

Die Grundschrift erhält man durch Eingabe des Befehls `\textnormal` bzw. der Deklaration `\normalfont`. Beide Formen werden eigentlich nur in Befehls- oder Umgebungsdefinitionen verwendet, bei denen ungeachtet der äußeren Bedingungen immer derselbe Zeichensatz verwendet werden soll. Der Befehl, mit dem die Befehlsnamen in diesem Buch generiert werden, sieht zum Beispiel ungefähr so aus:

```
\newcommand\Lcs[1]{%
  {\normalfont\ttfamily\textbackslash#1}%
  \index{#1@{\normalfont\ttfamily\textbackslash#1}}}
```

Ohne `\normalfont` würden Befehlsnamen unter Umständen wie *\dieser hier* gesetzt werden.

Standardschriftfamilien

L^AT_EX verfügt standardmäßig über drei Schriftfamilien, die durch kurze Befehlssequenzen ausgewählt werden können: eine Serifenschrift für Texte, die man mit dem Befehl `\textrm` erhält, eine serifenlose Schrift, die mit `\textsf` aufgerufen wird, und eine Schreibmaschinenschrift (Typewriter, normalerweise dicktengleich), auf die man mit `\texttt` zugreifen kann. Die entsprechenden Deklarationen lauten `\rmfamily`, `\sffamily`, bzw. `\ttfamily`.

Die tatsächlichen Schriftfamilien, die man mit diesen Befehlen auswählt, sind von der Dokumentenklasse abhängig. Sie können aber durch Pakete oder in der Präambel geändert werden (siehe Abschnitt 7.3.5). Bei der Installation sind Computer Modern Roman für die Serifenschrift, Computer Modern Sans für die serifenlose Schrift und Computer Modern Typewriter für die Schreibmaschinenschrift vorgegeben. Verwendet man andere Einstellungen, dann sollte man diese Standardfamilien so definieren, dass sie gemeinsam ein harmonisch abgerundetes Bild ergeben. Außerdem müssen die externen Fonts in einer für das Ausgabegerät geeigneten Auflösung verfügbar sein.

In diesem Buch dient Lucida Bright als Serifenschrift, Lucida Sans als serifenlose Schrift und European Modern Typewriter als Schreibmaschinenschrift. Für diese Einstellung wurde lediglich¹ das Paket `lucidabr` geladen und `\ttdefault` zu `emtt` umdefiniert; Abschnitt 7.3.5 enthält weitere Details zum Ändern der Standardtextfonts.

In den meisten Dokumentenklassen ist die Serifenschrift, die man mit `\textrm` erhält, gleichzeitig die Grundschrift, so dass der Befehl `\textrm` nicht sehr häufig verwendet wird. Wenn ein Buchdesigner jedoch eine serifenlose Schrift als Grundschrift deklariert hat, stellt `\textrm` die alternative Serifenschrift dar.

¹Etwas näher an der Wahrheit: Für die zweite Ausgabe dieses Buches wurden die Lucida Fonts etwas verkleinert und der European Modern Typewriter Font etwas vergrößert, um die x-Höhe beider Familien besser aneinander anzupassen. Dies geschah mithilfe eigens entworfener `\DeclareFontShape`-Deklarationen.

Standardschriftserien

Ein weiteres veränderliches Zeichensatzattribut ist die *Schriftserie*. In $\text{\LaTeX}$ setzt sich die Schriftserie aus zwei Attributen zusammen: der Schriftbreite und der Schriftstärke. $\text{\LaTeX}$ verfügt über zwei Befehle zum Ändern der Serien: `\textmd` und `\textbf`. Die entsprechenden Deklarationen sind `\mdseries` bzw. `\bfseries`. Mit dem ersten Befehl wird ein Zeichensatz mittlerer Schriftbreite und -stärke ausgewählt, während der zweite auf eine fettere Schriftserie umschaltet. Die jeweils gültigen Werte hängen von der Dokumentenklasse, ihren Optionen oder den eingesetzten Paketen ab. Im Falle der Computer Modern Schriftfamilien schaltet `\textbf` auf die breitfette Variante des gerade geladenen Zeichensatzes um und `\textmd` wieder zurück auf die mittlere Schriftstärke und -breite derselben Schrift.

Wenn man die Attribute der Schriftserien genauer einstellen möchte, definiert man dazu am besten High-Level-Befehle, und zwar mithilfe der Low-Level-Deklaration `\fontseries`, die in Abschnitt 7.9.1 beschrieben wird. Einige Pakete, die große Schriftfamilien für $\text{\LaTeX}$ bereitstellen, verfügen über solche zusätzlichen Befehle.

Standardschriftformen

Ein drittes Fontattribut, das unabhängig von den anderen geändert werden kann, ist die *Form* der gerade verwendeten Schrift. Für die meisten Dokumente ist eine aufrechte Form voreingestellt. Diese erhält man, falls erforderlich, mit dem Befehl `\textup` oder der Deklaration `\upshape`.

Die wahrscheinlich wichtigsten Befehle zum Ändern der Schriftform sind `\textit` und `\textsc` für einen Wechsel auf *Kursivschrift* bzw. auf KAPITÄLCHEN. Die entsprechenden Deklarationen sind `\itshape` und `\scshape`.

Eine Alternative zu `\textit` ist der Befehl `\textsl` (mit der zugehörigen Deklaration `\slshape`), der auf eine geneigte Form umschaltet. Häufig enthält eine Schriftfamilie nur entweder die kursiven oder die geneigten Schnitte. Computer Modern Roman verfügt jedoch über beide.

An der Stelle, an der von einer geneigten auf eine aufrechte Schriftform umgeschaltet wird, geraten die Zeichen meist zu dicht aneinander, besonders wenn der letzte geneigte Buchstabe eine Oberlänge hat. Der zusätzliche Weißraum, der an diesem Übergang eingefügt werden muss, um ein optisch ausgewogenes Schriftbild zu erhalten, wird als „Kursivkorrektur“ bezeichnet. Die Breite dieses Leerraums hängt von der einzelnen Zeichenform ab und wird in der `.tfm`-Datei gespeichert. Die Fontbefehle mit Argument führen die Kursivkorrektur automatisch durch; bei Deklarationen muss man sie dagegen mit `\/` von Hand vornehmen. Bei einer aufrechten Schrift beträgt die Kursivkorrektur der Buchstaben normalerweise null oder nimmt einen sehr kleinen Wert an. Es gibt jedoch einige Ausnahmen: Um z.B. in Computer Modern ein fettes „f“ in einfache Anführungsstriche zu setzen, sollte man `{\bfseries f/}` oder `\textbf{f/}` eingeben, da man ansonsten in einigen Fonts ‘f’ d.h. sich berührende Glyphen erhält.

Bei geneigten oder kursiven Schriftformen hat die Kursivkorrektur normalerweise einen positiven Wert, wobei der aktuelle Wert von der Form des jeweiligen Zeichens abhängt. Das nächste Beispiel zeigt wie man Deklarationen richtig einsetzt, um eine Schrift in ihre geneigte Form zu ändern.

Beim Wechsel von einer *kursiven* oder *geneigten* Form zu einer aufrechten sollte eine *Kursivkorrektur erfolgen*, außer wenn das nächste Zeichen ein kleines Interpunktionszeichen (Punkt oder Komma) ist.

`\raggedright`

Beim Wechsel von einer `{\itshape kursiven\}` oder `{\slshape geneigten\}` Form zu einer aufrechten sollte eine `{\itshape Kursivkorrektur erfolgen}`, außer wenn das nächste Zeichen ein kleines Interpunktionszeichen (Punkt oder Komma) ist.

Bsp.
7-3-2

Verwendet man stattdessen den entsprechenden Befehl mit Argument, wird die Kursivkorrektur automatisch hinzugefügt. Dieses Thema wird in Abschnitt 7.3.3 ausführlicher besprochen.

Kapitälchen werden manchmal in Überschriften oder für Namen benutzt. Für Namen kann man beispielsweise den folgenden Befehl `\name` definieren:

```
\newcommand\name[1]{\textsc{#1}}
```

Oder mithilfe zweier Deklarationen:

```
\newcommand\name[1]{\normalfont\scshape #1}
```

Die erste Definition schaltet einfach auf die gewünschte Form um, während die zweite Definition zunächst alle Fontattribute auf ihre Standardeinstellungen zurücksetzt. Welche Definition besser geeignet ist, hängt von den verfügbaren Fonts und von der Art des Dokumentes ab. In Computer Modern enthalten nur die Roman- und Typewriter-Familien Kapitälchen. Hier wäre die zweite Möglichkeit zu empfehlen, um auch in einem `\sffamily`-Kontext Kapitälchen (wenn auch mit Serifen) zu erzeugen. Im Falle der ersten Definition würde $\text{\LaTeX}$ versuchen, auf eine Kapitälchenform mittlerer Schriftstärke innerhalb der Schriftfamilie Computer Modern Sans umzuschalten. Da es diesen Schriftschnitt nicht gibt, würde $\text{\LaTeX}$ nach einem Ersatz suchen, indem es zunächst das Attribut für die Schriftform auf den voreingestellten Wert zurücksetzt. Dadurch würden jedoch keine Kapitälchen erzeugt. (Abschnitt 7.9.3 enthält nähere Informationen zur Fontersetzung.)

Auch für die Definition eines Akronymbefehls kann sich die `\scshape`-Deklaration als nützlich erweisen:

```
\newcommand\acro[1]{\scshape\MakeLowercase{#1}}
```

Diese Definition enthält den $\text{\LaTeX}$ -Befehl `\MakeLowercase`, der alle Buchstaben seines Argumentes in Kleinbuchstaben umsetzt (im Gegensatz zum $\text{\TeX}$ -Basisbefehl `\lowercase` ändert er auch Buchstaben, die durch Befehle erzeugt werden, wie etwa `\OE`). Dadurch enthält das Argument von `\acro` nur noch Kleinbuchstaben, die dann in Kapitälchen gesetzt werden.

Ein weiterer Fontbefehl, der in $\text{\LaTeX}$ eine Sonderstellung einnimmt, ist der Befehl `\emph`, mit dem man Hervorhebungen im normalen Text kennzeichnet. Die entsprechende Deklaration ist `\em`. Traditionell werden hervorgehobene Wörter im Text kursiv gesetzt. Wenn innerhalb eines kursiven Textes etwas hervorgehoben werden soll, wird normalerweise wieder zur aufrechten Form zurückgeschaltet. Diese Konvention wird vom `\emph`-Befehl unterstützt: Er schaltet auf die `\itshape`-Form um, wenn die aktuelle Schriftlage aufrecht ist, und auf `\upshape`, wenn der Text bereits geneigt gesetzt ist (also `\itshape`

<code>\tiny</code>	Text	<code>\normalsize</code>	Text		
<code>\scriptsize</code>	Text	<code>\large</code>	Text	<code>\huge</code>	Text
<code>\footnotesize</code>	Text	<code>\Large</code>	Text		Text
<code>\small</code>	Text	<code>\LARGE</code>	Text	<code>\Huge</code>	Text

Die von den Befehlen erzeugten Schriftgrade entsprechen denen, die in diesem Buch verwendet wurden. Sie sind abhängig von der benutzten Dokumentenklasse.

Tabelle 7.1: Standardbefehle für Schriftgrade

oder `\slshape` entsprechend). Somit muss sich der Benutzer nicht um die Schriftlage des Textes kümmern, wenn er den Befehl `\emph` oder die Deklaration `\em` verwendet.

Allerdings muss man darauf achten, dass Kursivkorrekturen an beiden Enden des hervorgehobenen Textes nötig sein können. Bei dem Befehl `\emph` wird die Kursivkorrektur automatisch durchgeführt, so dass er normalerweise der Deklaration vorzuziehen ist.

Bsp.
7-3-3

```
{\em Allerdings muss man darauf achten, dass
Kursivkorrekturen an\ { \em beiden Enden\} des
hervorgehobenen Textes nötig sein können}.
Bei dem Befehl \verb=\emph= wird die Kursivkorrektur
\emph{automatisch} durchgeführt, so dass er
normalerweise der Deklaration vorzuziehen ist.
```

Die aufrechte Form eignet sich nicht immer für Hervorhebungen innerhalb von bereits hervorgehobenem Text. Als gängige typographische Empfehlung gilt daher, Kapitälchen für die zweite, die „innere“ Ebene der Hervorhebungen zu verwenden. Standard- $\text{\LaTeX}$ unterstützt dieses Verfahren nicht direkt. Es lässt sich jedoch mit dem Befehl `\emminnershape` des Paketes `fixltx2e` einrichten.

Allerdings muss man darauf achten, dass Kursivkorrekturen an BEIDEN EN-DEN des hervorgehobenen Textes NÖTIG sein können.

Bsp.
7-3-4

```
\usepackage{fixltx2e}
\renewcommand\emminnershape{\scshape}
{\em Allerdings muss man darauf achten, dass
Kursivkorrekturen an\ { \em beiden Enden\} des
hervorgehobenen Textes \emph{nötig} sein können}.
```

Hervorzuhebenden Text zu unterstreichen, gilt im Verlagswesen als schlechter Stil. Diese Form wird nur benutzt, wenn das Ausgabegerät keine andere Möglichkeit bietet, den Text hervorzuheben – z.B. wenn man eine Schreibmaschine verwendet. Die Abschnitte 3.1.6 und 3.1.7 stellen Pakete vor, die mit `\em` Unterstreichungen erzeugen.

Standardschriftgrade

$\text{\LaTeX}$ verfügt über zehn Deklarationen zum Ändern des Schriftgrades (siehe Tabelle 7.1). Da Größenänderungen normalerweise nur in Befehlsdefinitionen vorgenommen werden, gibt es zu diesen zehn Deklarationen keine entsprechenden Befehle mit einem Argument. Die Deklarationsnamen wurden von $\text{\LaTeX}$ 2.09 übernommen, ihre Funktion hat sich in der heutigen $\text{\LaTeX}$ -Version jedoch leicht geändert. In $\text{\LaTeX}$ 2 ϵ wird durch Größenbefehle lediglich die Größe des aktuellen Schriftschnitts verändert, während alle anderen Attribute

unverändert bleiben. In $\text{\LaTeX}$ 2.09 bewirkte dieser Befehl gleichzeitig einen Wechsel zur Grundschrift.

Der mit diesen Befehlen ausgewählte Schriftgrad wird durch die Einstellungen in der Dokumentenklasse und die gegebenenfalls zugehörigen Optionen (wie z.B. 11pt) bestimmt. Die Anweisung `\normalsize` bezieht sich dabei in der Regel auf den Schriftgrad der Grundschrift. Die Befehle zur Änderung der Schriftgröße haben eine feste Reihenfolge, angefangen mit `\tiny` als kleinstem und `\Huge` als größtem Schriftgrad. Manchmal können sich mehrere Befehle auf die gleiche reale Schriftgröße beziehen: wenn z.B. für `\normalsize` ein großer Grad gewählt wird, liefert `\Huge` möglicherweise das gleiche Ergebnis wie `\huge`. Die Reihenfolge wird jedenfalls immer beachtet.

Die Größenbefehle für den Haupttext (d.h. `\normalsize`, `\small` und `\footnotesize`) wirken sich normalerweise auch auf die Abstände um Listen und abgesetzte Formeln aus. Um dieses Verhalten zu ändern, sollte man ihre Definition daher nicht einfach durch einen `\fontsize`-Befehl ersetzen, sondern stattdessen die Originaldefinitionen abwandeln, die in der Datei `classes.dtx` dokumentiert sind. Weitere Details zu diesem Thema findet man auch auf Seite 152.

Leider gibt es derzeit in $\text{\LaTeX}$ keine relativen Größenbefehle, d.h. Befehle, mit denen man den Schriftgrad z.B. um 2pt vergrößern kann. Dieses Problem wird teilweise durch das Paket `relsize` behoben, das in Abschnitt 3.1.4 auf Seite 90 beschrieben ist.

7.3.2 Kombinieren von Standardfontbefehlen

Wie bereits gezeigt wurde, lassen sich Fontbefehle und Deklarationen für den Zeichensatzwechsel miteinander kombinieren. Dadurch wird ein Schriftschnitt ausgewählt, welcher zu der Kombination der entsprechenden Attribute passt, z.B.:

Man kann einen Text **in einer großen, fetten, serifenlosen Schrift** setzen, dabei bleibt der Zeilenvorschub jedoch unverändert! $\text{\LaTeX}$ verwendet den Wert, *der am Ende* des Absatzes gilt!

Man kann einen Text
`{\sffamily\bfseries\large`
 in einer großen, fetten, serifenlosen
`Schrift}` setzen, dabei bleibt der
 Zeilenvorschub jedoch unverändert! `\`
`\LaTeX{}` verwendet den Wert,
`\emph{der am Ende}` des Absatzes gilt!

Bsp.
7-3-5

Hinter den Kulissen passiert dabei folgendes: Der Befehl `\sffamily` wechselt zur serifenlosen Standardschriftfamilie, `\bfseries` zu ihrer fettgedruckten Serie, und `\large` wählt dann einen größeren Schriftgrad, wobei alle anderen Attribute unverändert bleiben. (Der Zeilenvorschub wird nicht verändert, da der Befehl `\large` am Ende des Absatzes bereits nicht mehr gültig ist.) Für alle genannten Zwischenschriften werden Fontmetrikdateien (`.tfm`-Dateien) geladen, auch wenn sie nie verwendet werden. Im vorigen Beispiel wird so nach dem Befehl `\sffamily` die Schrift „sans serif medium 10pt“ geladen, nach `\bfseries` „sans serif bold extended 10pt“ und nach `\large` „sans serif bold extended 14pt“, die schließlich auch verwendet wird. Auf diese Weise zwingen High-Level-Fontbefehle $\text{\LaTeX}$ unnötigerweise dazu Zeichensätze zu laden, die gar nicht zum Einsatz kommen. Normalerweise ist das

<i>Befehl</i>	<i>Entspricht</i>	<i>Aktion</i>
<code>\textrm{...}</code>	<code>{\rmfamily...}</code>	Text in Serifenschrift setzen
<code>\textsf{...}</code>	<code>{\sffamily...}</code>	Text in serifenloser Schrift setzen
<code>\texttt{...}</code>	<code>{\ttfamily...}</code>	Text in Schreibmaschinen-Schrift setzen
<code>\textmd{...}</code>	<code>{\mdseries...}</code>	Text in normaler Schriftstärke setzen
<code>\textbf{...}</code>	<code>{\bfseries...}</code>	Text in fett gedruckter Schrift setzen
<code>\textup{...}</code>	<code>{\upshape...}</code>	Text in aufrechter Schrift setzen
<code>\textit{...}</code>	<code>{\itshape...}</code>	Text in <i>Kursivschrift</i> setzen
<code>\textsl{...}</code>	<code>{\slshape...}</code>	Text in <i>geneigter</i> Schrift setzen
<code>\textsc{...}</code>	<code>{\scshape...}</code>	Text in KAPITÄLCHEN setzen
<code>\emph{...}</code>	<code>{\em...}</code>	Text <i>hervorheben</i>
<code>\textnormal{...}</code>	<code>{\normalfont...}</code>	Text in Grundschrift setzen.

Tabelle 7.2: Standard-Fontwechselbefehle und Deklarationen

nicht weiter tragisch. Der einzige Nachteil ist eine leichte Verlängerung der Verarbeitungsdauer, die entsteht, wenn solch eine Kombination das erste Mal benutzt wird. Wenn jedoch viele verschiedene Kombinationen dieser Art im Dokument vorkommen, wäre zu überlegen, ob man diese nicht besser durch explizite Low-Level-Deklarationen erzeugt (siehe Abschnitt 7.9).

7.3.3 Fontbefehle und Deklarationen im Vergleich

In den vorhergehenden Abschnitten wurden bereits einige Beispiele für Fontbefehle mit Argument gezeigt, die Zeichensatzattribute ändern. Diese Fontwechselbefehle beginnen alle mit `\text...` (mit Ausnahme des Befehls `\emph`). Damit wird betont, dass sie für die Verwendung im normalen Text bestimmt sind. Außerdem sind die Namen so einprägsamer. Gegenüber den Deklarationsformen haben sie den Vorteil, mit anderen $\text{\LaTeX}$ -Strukturen verträglich zu sein. Sie dienen dazu, kurze Passagen in einer bestimmten Schriftfamilie, -serie oder -form zu setzen. Tabelle 7.2 zeigt, wie sich diese Befehle auswirken.

Ein weiterer Vorteil besteht darin, dass gegebenenfalls notwendige Kursivkorrekturen zu beiden Seiten des Argumentes automatisch durchgeführt werden. Bei einem Fontwechsel muss man sich darum also keine Gedanken mehr machen.

Es gibt nur sehr wenige Situationen, in denen dieser zusätzliche Leerraum unangebracht ist. Die meisten Typographen empfehlen z.B., keine Kursivkorrektur vorzunehmen, wenn auf den Zeichensatzwechsel ein kleines Interpunktionszeichen, wie z.B. ein Komma, folgt. Da die Menge der erforderlichen Korrekturen zum Teil auch eine Frage des persönlichen Geschmacks ist, kann jeder Benutzer selbst definieren, in welchen Situationen die Kursivkorrektur unterdrückt werden soll. Zu diesem Zweck muss man lediglich die Zeichen,

vor denen keine Kursivkorrektur mehr erfolgen soll, in die Liste `\nocorrlist` eintragen.¹ Die Standarddefinition für diesen Befehl lautet folgendermaßen:

```
\newcommand{\nocorrlist}{,.}
```

Es empfiehlt sich, die am häufigsten verwendeten Zeichen an den Anfang der Liste zu stellen, da so die Verarbeitung etwas schneller wird.

Zusätzlich zur globalen Anpassung kann die Kursivkorrektur auch an einzelnen Stellen unterdrückt werden, und zwar mithilfe des Befehls `\nocorr`. Er muss links oder rechts im Argument eines `\text{...}`-Befehls eingefügt werden, je nachdem, auf welcher Seite man die Kursivkorrektur unterbinden will.

Verwendet man die High-Level-Fontbefehle in $\text{\LaTeX}$, so wird die Kursivkorrektur automatisch korrekt durchgeführt. Man muss $\text{\LaTeX}$ nur selten durch Hinzufügen des Befehls `\nocorr` unterstützen.

```
\emph{Verwendet man die
High-Level-Fontbefehle in \LaTeX{}, so
wird die Kursivkorrektur automatisch
\emph{korrekt} durchgeführt}. Man muss
\LaTeX{} nur \emph{selten} durch
Hinzufügen des Befehls \verb=\nocorr=
unterstützen.
```

Bsp.
7-3-6

Im Gegensatz dazu eignen sich die Deklarationsformen eher dazu, eigene Befehle oder Umgebungen zu definieren.

- Diese Umgebung erzeugt eine fettgedruckte Aufzählung.
- Sie wurde mithilfe der $\text{\LaTeX}$ -Umgebung `itemize` und NFSS-Deklarationen definiert.

```
\newenvironment{bfitemize}{\begin{itemize}%
\normalfont\bfseries\raggedright}{\end{itemize}}
\begin{bfitemize}
\item Diese Umgebung erzeugt eine fettgedruckte
Aufzählung.
\item Sie wurde mithilfe der \LaTeX{}-Umgebung
\texttt{itemize} und NFSS-Deklarationen definiert.
\end{bfitemize}
```

Bsp.
7-3-7

7.3.4 Zugriff auf alle Zeichen eines Fonts

Manche Zeichen lassen sich nicht direkt über die Tastatur eingeben, auch wenn sie im gewählten Zeichensatz enthalten sind. Deshalb erfolgt der Zugriff auf viele nützliche Zeichen mithilfe von Befehlen wie `\ss` oder `\AE`, die dann die Zeichen „ß“ und „Æ“ erzeugen. Andere werden durch eine bestimmte Zeichenfolge generiert (dies ist eine Fonteigenschaft), wie z.B. `ffi`, das als „ffi“ gesetzt wird, oder `---`, das in den $\text{\TeX}$ -Standardfonts als „—“ erscheint.

Durch den Befehl `\symbol` kann man außerdem jedes in einem Font vorhandene Zeichen ansteuern, indem man die Nummer des Zeichens im aktuellen Kodierschema als Dezimal-, Oktal- (eingeleitet durch `'`) oder Hexadezimalzahl (eingeleitet durch `"`) eingibt. Allerdings muss man dann selbst darauf

¹Bei Verwendung von Paketen, die den `\catcode` eines Zeichens in `\nocorrlist` nachträglich ändern, wird das Zeichen von der Unterdrückungsfunktion nicht mehr erkannt. In diesem Fall sollte die Liste neu deklariert werden.

<i>Parameter</i>	<i>Voreinstellung</i>	<i>Beschreibung</i>
<code>\encodingdefault</code>	OT1	Kodierschema der „Grundschrift“
<code>\familydefault</code>	<code>\rmdefault</code>	Familie der „Grundschrift“
<code>\seriesdefault</code>	m	Serie der „Grundschrift“
<code>\shapedefault</code>	n	Form der „Grundschrift“
<code>\rmdefault</code>	cmr	Durch <code>\rmfamily</code> und <code>\textrm</code> ausgewählte Familie
<code>\sfdefault</code>	cmss	Durch <code>\sffamily</code> und <code>\textsf</code> ausgewählte Familie
<code>\ttdefault</code>	cmtt	Durch <code>\ttfamily</code> und <code>\texttt</code> ausgewählte Familie
<code>\bfdefault</code>	bx	Durch <code>\bfseries</code> und <code>\textbf</code> ausgewählte Serie
<code>\mddefault</code>	m	Durch <code>\mdseries</code> und <code>\textmd</code> ausgewählte Serie
<code>\itdefault</code>	it	Durch <code>\itshape</code> und <code>\textit</code> ausgewählte Form
<code>\sldefault</code>	sl	Durch <code>\slshape</code> und <code>\textsl</code> ausgewählte Form
<code>\scdefault</code>	sc	Durch <code>\scshape</code> und <code>\textsc</code> ausgewählte Form
<code>\updefault</code>	n	Durch <code>\upshape</code> und <code>\textup</code> ausgewählte Form

Tabelle 7.3: Voreingestellte Fontattribute

achten, dass die angewählte Fontposition wirklich eine Glyphe enthält.

Die Cork-Fontkodierung (T1) enthält unter anderem die Zeichen P, § und ¶, auf die man mit dem Befehl `\symbol` zugreifen kann. Sie fehlen aber in der alten T_EX-Kodierung (OT1):

Bsp.
7-3-8

!■■■!

`\fontencoding{T1}\selectfont`

Die Cork-Fontkodierung (`\texttt{T1}`) enthält unter anderem die Zeichen `\symbol{"DE}`, `\symbol{'237}` und `\symbol{32}`, auf die man mit dem Befehl `\verb=\symbol=` zugreifen kann. `\fontencoding{OT1}\selectfont` Sie fehlen aber in der alten T_EX-Kodierung (`\texttt{OT1}`): `!\symbol{32}!\symbol{"DE}!\symbol{'237}!`

Mit dem Programm `nfssfont.tex` (siehe Abschnitt 7.5.7 auf Seite 380) kann man sich die entsprechenden Codenummern zu den Zeichen eines Fonts ausgeben lassen.

7.3.5 Ändern der voreingestellten Textfonts

L^AT_EX verfügt über eine Reihe von internen Parametern, mit denen sich das allgemeine Erscheinungsbild eines Dokumentes leichter verändern lässt. Sie beeinflussen das Verhalten der High-Level-Fontwechselbefehle, die in den vorhergehenden Abschnitten vorgestellt wurden. Tabelle 7.3 enthält eine Liste dieser Parameter. Mit dem Befehl `\renewcommand` kann man die Werte für diese Parameter in Paketdateien oder der Präambel ändern. Geeignete Werte lassen sich aus den Zeichensatztabellen in diesem Kapitel ansehen.

Durch die folgende Definition in der Präambel:

`\renewcommand\familydefault{cmss}`

würde beispielsweise der gesamte Text des Dokumentes in Computer Modern Sans gesetzt, da diese Definition die Schriftfamilie der von L^AT_EX verwendeten

Grundschrift verändert. Genauer gesagt wird die Grundschrift durch die Werte von `\encodingdefault`, `\familydefault`, `\seriesdefault` und `\shapedefault` bestimmt. Daher müssen diese Befehle so definiert sein, dass sie in Kombination einem Schriftschnitt entsprechen, der auch in den internen $\text{\LaTeX}$ -Tabellen vorhanden ist.

Voreingestellte
Kodierung nicht
optimal



Der voreingestellte Wert von `\encodingdefault` ist zurzeit OT1. Das bedeutet, $\text{\LaTeX}$ geht davon aus, dass die meisten Fonts die $\text{\TeX}$ -Originalkodierung benutzen. Eigentlich dient diese Einstellung nur der Kompatibilität: Meistens ist es besser, die T1-Kodierung zu verwenden, da sie viele zusätzliche Zeichen enthält, über die OT1 nicht verfügt, und auch bei Wörtern mit Akzentbuchstaben eine korrekte Silbentrennung erlaubt (siehe Abschnitt 7.5.1). Einige moderne Fonts sind nur noch in T1 erhältlich, d.h. sie unterstützen OT1 gar nicht.

Man sollte auch beachten, dass sich nicht jede Fontkodierung als Standard für ein Dokument eignet. Als Voraussetzung muss sie einige der sichtbaren ASCII-Zeichen an ihren Standardpositionen enthalten; nähere Details hierzu sind in Abschnitt 7.11 auf Seite 452 behandelt. Der Befehl `\encodingdefault` lässt sich durch Laden des Paketes `fontenc` mit einer oder mehreren Optionen ändern; siehe Abschnitt 7.5.3. Weitere Informationen zu Fontkodierungen findet man in Abschnitt 7.9.1.

Falsche
Voreinstellung
für fett kann Probleme
verursachen



Als weitere Möglichkeit kann man den Schriftserienbefehl `\bfdefault` so definieren, dass er den Wert `b` erzeugt. Dadurch verwendet `\bfseries` dann eine **fette** und keine **breitfette** Schrift, was der Voreinstellung für Computer Modern entspricht. Diese Einstellung birgt jedoch einige Risiken, da z.B. bei Computer Modern nur die Serifenschriften fette Schriftschnitte mittlerer Schriftbreite haben. Die Computer Modern Sans Fonts verfügen nicht über fette sondern nur über breitfette Schriftschnitte. Ohne weitere Korrekturen würde die Anforderung etwa einer fettgedruckten serifenlosen Schrift (d.h. `\sffamily\bfseries`) dazu führen, dass $\text{\LaTeX}$ nach einem Ersatzfont sucht und schließlich einen Font mittlerer Schriftstärke auswählt. (Wie in Abschnitt 7.10.3 erläutert, lässt sich dieses Ergebnis vermeiden, indem man die breitfette Form der serifenlosen Familie als Ersatz für die fette Variante mittlerer Breite definiert.)

Ein Beispiel, in dem einige Standardeinstellungen verändert sind, findet sich in Abschnitt 7.10.8 auf Seite 451, der sich mit dem manuellen Einrichten von PostScript-Fonts befasst.

Durch seine Voreinstellung nimmt der Befehl `\familydefault` bei Änderungen von `\rmdefault` ebenfalls den neuen Wert an, sofern er nicht anders definiert wird. Umgekehrt haben Änderungen an `\familydefault` jedoch keine Auswirkungen auf `\rmdefault`.

7.3.6 $\text{\LaTeX}$ 2.09-Fontbefehle

Die zweibuchstabigen Fontbefehle aus $\text{\LaTeX}$ 2.09, wie z.B. `\bf`, sind in $\text{\LaTeX}$ 2 ϵ nicht mehr direkt definiert. Wenn überhaupt, tauchen sie stattdessen in den $\text{\LaTeX}$ 2 ϵ -Klassendateien auf. Aus Kompatibilitätsgründen gibt es in den Standardklassen Definitionen für diese Befehle, die ihr Verhalten in $\text{\LaTeX}$ 2.09 nachahmen. Je nach persönlichem Geschmack kann man sie in einem Paket

oder der Präambel durchaus umdefinieren; grundlegende Fontbefehle wie etwa `\bfseries` sollten hingegen nicht auf diesem Wege verändert werden.

Da die alten $\text{\LaTeX}$ 2.09-Fontbefehle jetzt von Dokumentenklassen oder Anwendern frei definiert werden können, sind sie nicht mehr länger im Code für $\text{\LaTeX}$ 2_ε enthalten. Stattdessen werden alle internen Verweise auf Fonts mithilfe von High- oder Low-Level-Befehlen des Fontauswahlschemas von $\text{\LaTeX}$ gesetzt. Diese Konventionen sollten von Paket- und Klassenentwicklern beachtet werden, um ein durchgängig einheitliches Verhalten zu gewährleisten.

 **\bf**
und Verwandte
nicht mehr verwenden

7.4 Fonts in mathematischen Formeln

Im Unterschied zu normalem Text ist ein automatischer Zeichensatzwechsel in mathematischen Formeln im Allgemeinen unerwünscht. Für Mathematiker vermittelt jeder Schriftschnitt eine ganz bestimmte Information. So können fettgedruckte, aufrecht stehende Buchstaben beispielsweise der Darstellung von Vektoren dienen. Wenn die Zeichen sich aufgrund äußerer Bedingungen änderten, würde das Ergebnis verfälscht. Aus diesem Grund werden Zeichensätze in Formeln anders gehandhabt als im normalen Text.

Zeichen in mathematischen Formeln können grob in zwei Klassen eingeteilt werden: in Sonderzeichen und in alphanumerische Zeichen (Buchstaben und Ziffern). Intern unterscheidet $(\text{\La})\text{\TeX}$ zwischen acht verschiedenen Arten von mathematischen Zeichen (um für angemessene Zwischenräume sorgen zu können), aber für den Anwender reicht normalerweise eine Unterteilung in zwei Klassen aus.

Einige Sonderzeichen, wie etwa $=$ können direkt über die Tastatur eingegeben werden. Die meisten müssen jedoch über eine Befehlssequenz erzeugt werden – so steht `\leq` z.B. für $\leq$. Die alphanumerischen Zeichen, welche die zweite große Gruppe der mathematischen Zeichen bilden, werden normalerweise direkt über die Tastatur eingegeben.

In Standard- $\text{\LaTeX}$ sind über 200 Sonderzeichen vordefiniert, mit deren Hilfe man nahezu jede gewünschte Formel setzen kann. Diese Sonderzeichen sind über mehrere Zeichensätze verstreut, aber sie werden so angesteuert, dass man ihre interne Darstellung nicht kennen muss. Wenn nötig, können auf ähnliche Weise weitere Sonderzeichen-Fonts bereitgestellt werden; siehe Abschnitt 7.10.7.

Der wichtigste Unterschied zwischen den Sonderzeichen und den alphanumerischen Zeichen besteht darin, dass die Sonderzeichen innerhalb einer Formel graphisch immer auf exakt dieselbe Weise dargestellt werden, während der Benutzer das Erscheinungsbild der alphanumerischen Zeichen durchaus ändern kann. Die Befehle, mit deren Hilfe die Darstellung alphanumerischer Zeichen in einer Formel verändert werden kann, werden im folgenden „Alphabetbefehle“ genannt und die von ihnen erzeugten Zeichensätze „Mathematikalphabete“. Die Alphabetbefehle sind von den Fontbefehlen außerhalb der Formeln unabhängig, so dass sich eine Formel auch dann nicht verändert, wenn sie z.B. innerhalb einer Theoremmgebung steht, in der Texte standardmäßig kursiv gesetzt werden. Dieses Verhalten ist sehr wichtig, da die Form der Zeichen in mathematischen Formeln von Bedeutung ist und sich auch dann nicht ändern darf, wenn die Formel an eine andere Stelle im Dokument verschoben wird.

Benutzer, die mit der alten Methode der Zeichensatzauswahl vertraut sind, werden vielleicht überrascht sein, dass Befehle wie `\bfseries` nicht in Formeln verwendet werden können. Dies ist der Preis für die größere Flexibilität, welche die Wahl der Zeichensatzattribute im normalen Text bietet – eine Flexibilität, die in Formeln unerwünscht ist. Zum Ändern der Form einiger Zeichen in komplexeren Formeln braucht man deshalb einen anderen Mechanismus: die Alphabetbefehle.

7.4.1 Besondere Alphabetbefehle für Formeln

Für Mathematiker reichen ein Alphabet und eine noch so große Anzahl von Sonderzeichen nicht aus, um ihre Gedanken zu Papier zu bringen. Sie tendieren dazu, alle verfügbaren Zeichensätze zu nutzen, um besondere Konzepte zu verdeutlichen. Neben der Verwendung anderer Alphabete, wie beispielsweise des griechischen, die hauptsächlich als Sonderzeichen benutzt werden – `\alpha`, `\beta`, usw. – findet man serifenlose Buchstaben für Matrizen, fett gedruckte Kleinbuchstaben zur Darstellung von Vektoren, sowie die verschiedenen Frakturschriften zur Kennzeichnung von Gruppen, Idealen oder Körpern. Einige Mathematiker benutzen zur Darstellung von Mengen auch kalligraphische Zeichen. Es gibt unendlich viele Konventionen – und vor allem in jedem Fachgebiet andere. Daher bietet $\text{\LaTeX}$ die Möglichkeit, neue Alphabetbefehle zu definieren und diese einer gewünschten Schriftformgruppe zuzuordnen, anstatt sich auf eine feste, unveränderliche Reihe von Befehlen verlassen zu müssen. Diese Alphabetbefehle setzen alle alphanumerischen Zeichen, die ihnen als Argument übergeben werden, in Fonts der entsprechenden Schriftformgruppe. (Sonderzeichen lassen sich nicht auf diese Weise ändern.) Wie noch in Abschnitt 7.4.3 zu sehen sein wird, können die Befehle in verschiedenen Formeln unterschiedliche Schriften verwenden; innerhalb einer Formel erzeugen sie jedoch ungeachtet ihrer Umgebung immer das gleiche Format.

Vordefinierte Alphabetbefehle

Anwender können ganz nach Bedarf eigene Alphabetbefehle definieren, einige sind jedoch auch schon in $\text{\LaTeX}$ vordefiniert. Tabelle 7.4 auf der nächsten Seite enthält eine Liste dieser Befehle. Wie man an den letzten Zeilen sehen kann, entstammen die Buchstaben für Formeln normalerweise dem mathematischen Alphabet `\mathnormal`. Die Buchstaben, die mit `\mathit` erzeugt werden, haben dagegen kleinere Zwischenräume, und können deshalb besonders gut für ausführliche Variablenamen verwendet werden, die in einigen Disziplinen gebräuchlich sind.

Die Alphabetbefehle in $\text{\LaTeX}$ haben ein Argument, das normalerweise aus einem einzelnen Buchstaben oder einem Wort besteht, das in einer speziellen Schrift gesetzt werden soll.

Deshalb berechnet sich G als

$$G = \mathcal{A} + \sum_{i=1}^n \mathcal{B}_i \quad (1)$$

```
Deshalb berechnet sich  $\mathsf{G}$  als
\begin{equation}
\mathsf{G} = \mathcal{A} +
\sum_{i=1}^n \mathcal{B}_{_i}
\end{equation}
```

Bsp.
7-4-1

Befehl		Beispiel
<code>\mathcal</code>	<code>\$\mathcal{A}=a\$</code>	$\mathcal{A} = a$
<code>\mathrm</code>	<code>\$\mathrm{max}_i\$</code>	$\max_i$
<code>\mathbf</code>	<code>\$\sum x = \mathbf{v}\$</code>	$\sum x = \mathbf{v}$
<code>\mathsf</code>	<code>\$\mathsf{G}_1^2\$</code>	G_1^2
<code>\mathtt</code>	<code>\$\mathtt{W}(a)\$</code>	$W(a)$
<code>\mathnormal</code>	<code>\$\mathnormal{abc}=abc\$</code>	$abc = abc$
<code>\mathit</code>	<code>\$\differ\neq\mathit{differ}\$</code>	$differ \neq differ$

Tabelle 7.4: Vordefinierte Alphabetbefehle in $\mathrm{\LaTeX}$

Die Verwendung von Fontbefehlen in Formeln ist somit anders als in $\mathrm{\LaTeX}$ 2.09, wo ein Zeichensatzwechsel mit Befehlen wie `\rm` ausgelöst wurde (`.\{\rm A\}.`). Die alte Syntax der wichtigsten zweibuchstabigen Fontwechselbefehle wie `\rm`, `\sf`, `\bf`, `\it` und `\tt` wird in den Standarddokumententklassen weiterhin unterstützt. Für die übrigen kann die Syntax durch das `oldfont`-Paket erzwungen werden; siehe Abschnitt 7.12.1. In neuen $\mathrm{\LaTeX}$ -Dokumenten sollte man jedoch von ihrer Verwendung absehen.

Wie bereits erwähnt, besteht ein weiterer Unterschied zwischen dem alten Fontauswahlverfahren in $\mathrm{\LaTeX}$ 2.09 und NFSS darin, dass Fontdeklarationen für normalen Text nicht mehr in Formeln verwendet werden dürfen, weil sie nur einige Merkmale der aktuellen Schrift verändern, anstatt auf einen bestimmten Zeichensatz umzuschalten. Wenn man also in einer Formel `\bfseries..` anstatt `\mathbf{..}` schreibt, gibt $\mathrm{\LaTeX}$ eine Fehlermeldung aus.

Die Namen der Alphabetbefehle sind so gewählt, dass sie eher aussagekräftig als einfach sind: Sie beginnen alle mit `\math`. Bei häufigerer Benutzung ist es deshalb empfehlenswert, in der Präambel einige Abkürzungen für diese Befehle zu definieren, wie z.B.:

```
\newcommand\mrm{\mathrm}
```

Man mag sich fragen, welches das Standard-Mathematikalphabet ist, d.h. aus welchem Zeichensatz die alphanumerischen Zeichen ausgewählt werden, wenn kein expliziter Alphabetbefehl verwendet wird, wie etwa in der Formel $x = 12345$. Tatsächlich gibt es kein spezielles Standard-Mathematikalphabet. $\mathrm{\LaTeX}$ kann so eingerichtet werden, dass Zeichen aus verschiedenen Zeichensätzen eingesetzt werden, solange der Benutzer nicht explizit einen speziellen Zeichensatz ausgewählt hat. Das ist, wie das folgende Beispiel zeigt, normalerweise der Fall:

Kein Standard-Mathematikalphabet

$x = 12345$	(1)	<code>\begin{eqnarray}</code>		
$x = 12345$	(2)	<code>x</code>	<code>&=&</code>	<code>12345</code>
$x = 12345$	(3)	<code>\mathrm{x}</code>	<code>&=&</code>	<code>\mathrm{12345}</code>
		<code>\mathnormal{x}</code>	<code>&=&</code>	<code>\mathnormal{12345}</code>
		<code>\end{eqnarray}</code>		

Bsp.
7-4-2

Wie oben zu sehen, ändert `\mathrm` keine Ziffern und `\mathnormal` keine Buchstaben. Wenn nicht anders angegeben, stammen die Ziffern aus dem Ma-

thematikalphabet, das mit dem Befehl `\mathrm` aufgerufen wird, und die Buchstaben aus jenem, das mit `\mathnormal` verknüpft ist.¹ Dieses Verhalten lässt sich mit dem Befehl `\DeclareMathSymbol` steuern, der in Abschnitt 7.10.7 erläutert wird.

Definieren neuer Alphabetbefehle

Neue Alphabetbefehle werden mit dem Befehl `\DeclareMathAlphabet` definiert. Um z.B. eine geneigte, serifenlose Schrift als Mathematikalphabet zugänglich zu machen, muss man zuerst einen neuen Befehlsnamen, wie etwa `\msfsl`, festlegen, mit dem das neue Alphabet ausgewählt werden soll. Danach sucht man sich aus den Zeichensatztabellen, die in diesem Kapitel aufgeführt sind (beginnend auf Seite 365), die passende Gruppe von Schriftschnitten aus, um sie dem neuen Alphabetbefehl zuzuweisen. Unter anderem zeigt sich dort, dass die Familie Computer Modern Sans eine Serie mittlerer Breite und Stärke mit aufrechten und geneigten Schriftformen besitzt. Wenn man etwa die geneigte Form dieser Familie als Alphabetbefehl verwenden will, richtet man diesen mithilfe der `\DeclareMathAlphabet`-Deklaration in $\text{\LaTeX}$ ein.

```
\DeclareMathAlphabet{befehl}{kodierung}{familie}{serie}{form}
```

Neben dem Alphabetbefehl verfügt diese Deklaration über vier weitere Argumente: das Kodierschema, die Familie, die Serie und die Form des zu verwendenden Fonts. Der hier definierte Alphabetbefehl schaltet immer auf eine geneigte Computer Modern Sans mittlerer Breite und Stärke um.

```
\DeclareMathAlphabet{\msfsl}{OT1}{cmss}{m}{sl}
```

Dies zeigt sich in der Formel

```
\begin{equation}
  \sum \msfsl{A}_{i} = a \tan \beta
\end{equation}
```

Dies zeigt sich in der Formel

$$\sum A_i = a \tan \beta \quad (1)$$

Bsp.
7-4-3

Man kann einen bereits existierenden Alphabetbefehl auch in einer Paketdatei oder der Präambel des Dokumentes umdefinieren. Die Deklaration

```
\DeclareMathAlphabet{\mathsf}{OT1}{pag}{m}{n}
```

überschreibt z.B. die Voreinstellung für den Befehl `\mathsf`. Dieser Befehl schaltet nun in Formeln auf Adobe Avant Garde um. Man sollte allerdings eine Kleinigkeit beachten: Wenn das betreffende Mathematikalphabet Teil eines Symbolfonts ist, der bereits aus einem anderen Grund von $\text{\LaTeX}$ geladen wurde (z.B. durch `\mathcal`), verwendet man hier besser den Befehl `\DeclareSymbolFontAlphabet`. So werden die doch begrenzten $\text{\TeX}$ -Ressourcen für Formeln besser genutzt; Näheres hierzu auf Seite 447.

¹Es ist zwar seltsam, aber der Mathematikfont, der dem `\mathnormal`-Alphabet entspricht, enthält eigentlich Mediävalziffern. Als die Computer Modern Fonts entwickelt wurden, war Speicherplatz noch rar, so dass Donald Knuth eine Reihe „nichtmathematischer“ Zeichen in diese Fonts quetschte, die eigentlich nur im Text verwendet werden.

7.4.2 Textfontbefehle in Formeln

Wie bereits erwähnt, können Deklarationen für normalen Text, wie zum Beispiel `\rmfamily`, nicht in Formeln verwendet werden. Fontwechselbefehle mit Argument, wie z.B. `\textrm`, lassen sich hingegen sowohl in normalen Texten als auch in Formeln einsetzen. Diese Befehle können z.B. benutzt werden, um die Formelumgebung zeitweise zu verlassen und Text einzufügen, der logisch gesehen zu dem Text vor und nach der Formel gehört. Die für diesen Zwischentext eingesetzte Schrift hängt von den Umgebungsbedingungen ab, d.h. sie ergibt sich aus den aktuellen Werten für die Kodierung, Schriftfamilie, Schriftserie und Schriftform, wie man im folgenden Beispiel sehen kann.

Das Ergebnis lautet

Bsp.
7-4-4

$x = 10$ **und folglich** $y = 12$

```
\sffamily Das Ergebnis lautet
\[ x = 10 \texttt{bf{ und folglich }} y = 12 \]
```

Wie zu erkennen, wurde die serifenlose Familie beibehalten und die Schriftserie auf fett gesetzt. Hier ist vielleicht der Befehl `\text` aus dem Paket `amstext` zweckmäßiger. Er übernimmt die aktuellen Werte für Kodierung, Schriftfamilie, -serie und -form unverändert (siehe Abschnitt 8.6.1).

7.4.3 Formellayouts

$\text{\LaTeX}$ ermöglicht dem Benutzer nicht nur, Teile von Formeln durch Alphabetbefehle zu verändern, sondern auch das Erscheinungsbild einer Formel als Ganzes. Jede Formel wird in einem bestimmten „Formellayout“ gesetzt, das sich außerhalb des Formelmodus durch den Befehl `\mathversion` auswählen lässt und das Erscheinungsbild der nachfolgenden Formeln bestimmt.

$\text{\LaTeX}$ kennt die beiden Formellayouts „normal“ und „bold“. Darüber hinaus lassen sich weitere durch spezielle Pakete hinzufügen. So richtet zum Beispiel das Paket `mathtime` (für die kommerziellen MathTime-Fonts) ein Layout mit dem Namen „heavy“ ein, mit dessen Hilfe Formeln in den ultrafetten Sonderzeichen der MathTime-Fonts gesetzt werden.

Wie der Name andeutet, ist `\mathversion{normal}` die Voreinstellung. Das Formellayout „bold“ hingegen erzeugt fettere Alphabetzeichen und Symbole, obwohl große Operatoren wie `\sum` normalerweise nicht geändert werden. Im nächsten Beispiel wird dieselbe Formel zuerst in normalem, dann in fettem Layout dargestellt.¹

Bsp.
7-4-5

$$\sum_{j=1}^z j = \frac{z(z+1)}{2} \quad (1)$$

$$\sum_{j=1}^z j = \frac{z(z+1)}{2} \quad (2)$$

```
\begin{equation}
\sum_{j=1}^z j = \frac{z(z+1)}{2}
\end{equation}
\mathversion{bold}
\begin{equation}
\sum_{j=1}^z j = \frac{z(z+1)}{2}
\end{equation}
```

Der Befehl `\mathversion` mag in einigen Situationen wie z.B. Überschriften sehr nützlich sein, jedoch bleibt zu beachten, dass er das Layout (und

¹Aus historischen Gründen gibt es in $\text{\LaTeX}$ zwei weitere Befehle, mit denen man auf eines der Standardlayouts schalten kann: `\boldmath` und `\unboldmath`.

damit unter Umständen auch die Bedeutung) der gesamten Formel ändert. Wenn man nur einen Teil der Zeichen fetter setzen will, sollte man nicht die `\mathversion` ändern. Stattdessen eignet sich hier der Alphabetbefehl `\mathbf` für die einzelnen Zeichen und/oder der Befehl `\bm` des `bm`-Paketes (siehe Abschnitt 8.8.2).

Wenn das Formellayout mit dem `\mathversion`-Befehl geändert wird, prüft $\text{\LaTeX}$ in seinen internen Tabellen, wo sich die verschiedenen Sonderzeichen für das neue Layout befinden. Gegebenenfalls werden auch einige oder alle Alphabetbefehle verändert und mit anderen Schriftformen verknüpft.

Was geschieht aber nun mit den selbst definierten Alphabetbefehlen, wie etwa `\msfsl` aus Beispiel 7-4-3? Sofern sie lediglich mithilfe von `\DeclareMathAlphabet` definiert wurden, bleiben sie in allen Formellayouts unverändert.

Wenn ein Alphabetbefehl in einem bestimmten Layout eine andere Schrift erzeugen soll, muss dies $\text{\LaTeX}$ über den Befehl `\SetMathAlphabet` mitgeteilt werden. Der Alphabetbefehl `\mathsf` wird z.B. per Voreinstellung folgendermaßen definiert:

```
\DeclareMathAlphabet{\mathsf}{OT1}{cmss}{m}{n}
\SetMathAlphabet{\mathsf}{bold}{OT1}{cmss}{bx}{n}
```

In der ersten Zeile wird festgelegt, dass in allen Layouts Computer Modern Sans normal die Standardschrift für `\mathsf` ist. Die zweite Zeile besagt, dass im Layout „bold“ stattdessen die Schrift Computer Modern Sans breitfett verwendet wird.

```
\SetMathAlphabet{befehl}{layout}{kodierung}{familie}{serie}{form}
```

Wie das vorige Beispiel zeigt, besitzt `\SetMathAlphabet` sechs Argumente: Das erste repräsentiert den Namen des Alphabetbefehls, das zweite den Namen des Formellayouts, für das diese Einstellung generiert wird; die übrigen vier bezeichnen die Kodierung, sowie die Schriftfamilie, -serie und -form, die dem Layout zugeordnet werden.

Ein bestehender Alphabetbefehl kann, wie bereits erwähnt, mithilfe von `\DeclareMathAlphabet` umdefiniert werden. In diesem Fall werden alle vorherigen `\SetMathAlphabet`-Deklarationen für diesen Befehl aus den internen $\text{\LaTeX}$ -Tabellen entfernt. Damit erzeugt er in allen Formellayouts die gleichen Resultate, sofern keine neuen `\SetMathAlphabet`-Deklarationen hinzugefügt werden.

7.5 Die Standardfontunterstützung in $\text{\LaTeX}$

Dieser Abschnitt wird durch eine kurze Einführung in die Standardtextfonts eingeleitet, die zu $\text{\LaTeX}$ gehören: Computer Modern und European Computer Modern. Danach werden die Standardpakete für die Ein- und Ausgabekodierung besprochen. Der Abschnitt schließt mit der Beschreibung eines Paketes zum Überwachen der Fontverarbeitung in $\text{\LaTeX}$ und eines weiteren Paketes zur Ausgabe von Zeichentabellen (das in den späteren Teilen dieses Kapitels intensiv genutzt wurde).

Familie	Serie	Form(en)	Schriftbild
<i>Computer Modern Roman (T1, OT1, TS1)</i>			
cmr	m	n, it, sl, sc, ui	COMPUTER ROMAN KAPITÄLCHEN
cmr	bx	n, it, sl	<i>Comp. Mod. Roman breitfett kursiv</i>
cmr	b	n	Computer Modern Roman fett aufrecht
<i>Computer Modern Sans (T1, OT1, TS1)</i>			
cmss	m	n, sl	<i>Computer Modern Sans geneigt</i>
cmss	bx	n	Computer Modern Sans breitfett
cmss	sbc	n	Computer Modern Sans halbfett schmal
<i>Computer Modern Typewriter (T1, OT1, TS1)</i>			
cmtt	m	n, it, sl, sc	<i>Computer Modern Typewriter kursiv</i>
cmvtt	m	n, it	Proportional Computer Modern Typewriter
<i>Computer Modern Fibonacci (T1, OT1)</i>			
cmfib	m	n	Computer Modern Fibonacci
<i>Computer Modern Funny Roman (T1, OT1)</i>			
cmfr	m	n, it	Computer Modern Funny Roman
<i>Computer Modern Dunhill (T1, OT1)</i>			
cmdh	m	n	Computer Modern Dunhill

Tabelle 7.5: Klassifizierung der Computer Modern Fontfamilien

7.5.1 Computer Modern – Die L^AT_EX-Standardfonts

Neben T_EX entwickelte Donald Knuth eine Fontfamilie namens Computer Modern; siehe Tabelle 7.5. Bis in die frühen 1990er Jahre standen für T_EX – und damit für L^AT_EX – im Grunde nur diese Zeichensätze zur Verfügung. Jeder dieser Zeichensätze enthält 128 Zeichen (T_EX arbeitete ursprünglich mit einer 7-Bit-Kodierung), so dass kein Platz für Akzentbuchstaben als eigenständige Zeichen blieb. Entsprechend muss bei diesen Fonts jeder Akzentbuchstabe mithilfe des T_EX-Basisbefehls `\accent` konstruiert werden, was wiederum bedeutet, dass Wörter mit Akzentbuchstaben nicht automatisch getrennt werden können. Während diese Einschränkung bei englischen Dokumenten mit wenigen Fremdwörtern noch hinzunehmen ist, stellt sie für andere Sprachen ein großes Hindernis dar.

*Die Original
T_EX-Fontkodierung*

Es überrascht kaum, dass diese Mängel für T_EX-Anwender in Europa von großer Bedeutung waren. So wurde T_EX schließlich 1989 mit interner und externer 8-Bit-Unterstützung neu aufgesetzt. Auf der T_EX-Anwender-Konferenz in Cork (1990) wurde eine 8-Bit-Standardkodierung für Textfonts (T1) entwickelt, die viele diakritische Zeichen enthält (siehe Tabelle 7.32 auf Seite 462) und mit der Dokumente in über 30 Sprachen, die auf dem lateinischen Alphabet basieren, gesetzt werden können. An der Universität

*T1 auch bekannt als
„Cork“-Kodierung*

Bochum wurden (unter der Leitung von Norbert Schwarz) die Computer Modern Schriftfamilien neu implementiert. Außerdem wurden zusätzliche Zeichen entworfen, so dass die entstandenen Zeichensätze völlig mit dem Kodierschema übereinstimmten. Die erste Version dieser Fonts wurde unter dem Namen „DC Fonts“ veröffentlicht. Mittlerweile hat Jörg Knappen sie in ihre endgültige Form gebracht und sie sind nun als „European Computer Modern Fonts“, kurz „EC Fonts“ im Umlauf.¹

EC Fonts

Sowohl die Computer Modern als auch die EC Fonts werden als $\text{\LaTeX}$ -Standard betrachtet und müssen bei jeder Installation verfügbar sein. Obwohl sie ursprünglich mit METAFONT entwickelt wurden, gibt es heutzutage auch frei verfügbare Type 1 PostScript-Versionen. Die Computer Modern-Versionen wurden von Blue Sky Research entwickelt; Y&Y fügten die $\text{\LaTeX}$, AMS und Euler Fonts hinzu. Die METAFONT-Quellen der EC Fonts wurden kürzlich von Vladimir Volovich zu Type 1 PostScript konvertiert. Seine Version ist unter dem Namen CM-Super-Fontpaket bekannt und enthält neben den EC Fonts außerdem EC Concrete, EC Bright und LH (Cyrillic Computer Modern). CM-Super unterstützt neben der T1-Kodierung auch die $\text{\LaTeX}$ -Standardkodierungen TS1, T2A, T2B, T2C und X2. Die CM-Super Fonts wurden in einem automatischen Verfahren in das Type 1-Format konvertiert, und auch wenn der Konvertierungsalgorithmus sehr ausgereift war, kann man nicht die gleiche Qualität erwarten, wie bei einer manuellen Konvertierung.

PostScript

Type 1-Versionen

CM-Super Fonts

Da die PostScript-Fonts die gleiche Fontmetrik haben wie ihre METAFONT-Entsprechungen, benötigen sie in einem $\text{\LaTeX}$ -Dokument kein gesondertes Paket zu ihrer Unterstützung. Nach ihrer Installation werden sie automatisch von dem Treiber verwendet, der die .dvi-Ausgabe nach PostScript konvertiert (z.B. dvips). Die .fd-Standarddateien für Computer Modern stellen nur eindeutig definierte Schriftgrade bereit, damit nicht allzu viele Bitmap-Fonts generiert werden müssen. Beim Einsatz von PostScript können Zwischengrößen (über `\fontsize`) jedoch auch ohne diesen Nebeneffekt verwendet werden. Das Paket `fix-cm` bedient sich dieser Funktion.

Obwohl die EC Fonts ursprünglich als passgenaue Erweiterung (und Ersatz) für die Computer Modern 7-Bit-Zeichensätze gedacht waren, wurden letztendlich nicht alle Zeichen beibehalten. Das deutsche ß wurde z.B. neu entworfen – diese Entscheidung des Fontdesigners traf nicht überall auf Zustimmung.

Computer Modern scharfes s: ß
EC Modern scharfes s: ſ

```
\fontencoding{OT1}\fontfamily{cmr}\selectfont
Computer Modern scharfes s: \ss \par
\fontencoding{T1}\fontfamily{cmr}\selectfont
EC Modern scharfes s: \ss
```

Bsp.
7-5-1

Durch die CM-Super Fonts stellt dies kein Problem mehr dar: Wer das CM-Zeichen dem EC-Zeichen vorzieht, kann einfach in der Datei `cm-super-t1.enc` `germandbls` gegen `germandbls.alt` austauschen.²

¹Nicht zu verwechseln mit den European Modern Fonts™, einer Reihe hochwertiger kommerzieller Fonts von Y&Y, die auf dem Computer Modern-Design basieren, jedoch eine etwas andere Metrik aufweisen [67].

²Noch besser ist es, wenn man die geänderte Kodierungsdatei umbenennt und dann die entsprechenden Verweise in der (dvips) Zuordnungsdatei auf den neuen Namen anpasst.

Das ist jedoch nicht der einzige Unterschied zwischen den ursprünglichen Computer Modern und den neuen EC Fonts. Bei den EC Fonts gibt es viel mehr individuelle Entwurfsgrößen für größere Schriftgrade (während die CM Fonts einfach linear skaliert wurden) und in dieser Hinsicht ist sehr deutlich zu erkennen, dass es sich tatsächlich um zwei unterschiedliche Schriftfamilien handelt.¹ Durch das folgende Beispiel fallen diese Unterschiede wohl am deutlichsten ins Auge.

Der Fuchs springt
schnell über den Zaun!

```
\fontencoding{OT1}\sffamily\bfseries
\Huge      Der Fuchs springt \par
\normalsize schnell über den Zaun!
```

Bsp.
7-5-2

Der Fuchs springt
schnell über den Zaun!

```
\fontencoding{T1}\sffamily\bfseries
\Huge      Der Fuchs springt \par
\normalsize schnell über den Zaun!
```

Das Thema ist unproblematisch, wenn man das EC-Design mag und durchgängig T1 verwendet. Andernfalls gibt es verschiedene Lösungsansätze. Eine Herangehensweise besteht darin, einen anderen Satz von Fontdefinitionen zu nutzen, die nicht auf *alle* verschiedenen EC-Fontdesigns zugreifen und den traditionellen CM Fonts ähnlicher sind, jedoch von besserer typographischer Qualität. Das Paket `fix-cm` von Walter Schmidt, das mit L^AT_EX verteilt wird, stellt eine solche Lösung zur Verfügung. Dieses Paket muss unmittelbar nach der Deklaration der Dokumentenklasse geladen werden, oder mithilfe von `\RequirePackage` sogar noch vorher, denn es wirkt sich nur auf Fonts aus, die L^AT_EX noch nicht geladen hat – und die Dokumentenklasse könnte Fonts laden.

Der Fuchs springt
schnell über den Zaun!

```
\usepackage{fix-cm}
\fontencoding{OT1}\sffamily\bfseries
\Huge      Der Fuchs springt \par
\normalsize schnell über den Zaun!
```

Bsp.
7-5-3

Der Fuchs springt
schnell über den Zaun!

```
\fontencoding{T1}\sffamily\bfseries
\Huge      Der Fuchs springt \par
\normalsize schnell über den Zaun!
```

Man kann als weitere Lösung auch die Almost European Fonts (von Lars Engbrechtsen) oder die EZ Fonts (von Robert Fuster) verwenden. Bei beiden handelt es sich um virtuelle Fontsätze, die auf den Computer Modern Fonts beruhen. Mit Ausnahme einer kleinen Auswahl von Zeichen, die nicht in CM enthalten sind, entsprechen sie der T1-Kodierung.

Dieser Ansatz hat jedoch einige Nachteile. Die Lösungen unterstützen z.B. nicht die begleitenden Symbolfonts, so dass die zusätzlichen Sonderzeichen, die das Paket `textcomp` bereitstellt, gar nicht verwendet werden können.

¹Der Fehler, der während der Entwicklung von NFSS gemacht wurde, bestand darin, NFSS vorzugaukeln, dass es sich um dieselbe Familie handelt (z.B. `cmr`, `cmss`), die lediglich in einer anderen Kodierung vorliegt. Diese Fehlentscheidung kann jedoch leider nicht ohne große Schwierigkeiten hinsichtlich der Rückwärtskompatibilität behoben werden.

Suchprobleme in
.pdf-Dokumenten



Was noch wichtiger ist: Da die virtuellen Fonts zusammengesetzte Zeichen benutzen, lassen sich die resultierenden .pdf-Dateien nicht nach Wörtern mit diakritischen Zeichen durchsuchen. An den entsprechenden Stellen in der Datei steht dann nämlich anstatt eines Akzentbuchstabens (aus einem Zeichen) eine komplizierte Bauanleitung. Anders ausgedrückt, täuscht diese Lösung L^AT_EX vor, dass es sich um einzelne Zeichen handelt (um Silbentrennung und Kerning zu ermöglichen). Diese Information geht jedoch auf dem Weg in die Ausgabedatei wieder verloren, so dass keine reibungslose Nachbearbeitung erfolgen kann.

Soweit es die ausgewählten Fonts betrifft, ergibt das ae-Paket jedoch das gleiche Ergebnis wie fix-cm.

```
\usepackage{ae}
```

```
\fontencoding{T1}\sffamily\bfseries  
\Huge Der Fuchs springt \par  
\normalsize schnell über den Zaun!
```

Bsp.
7-5-4

Der Fuchs springt schnell über den Zaun!

Latin Modern in
Aussicht

Im Jahre 2002 gründeten und finanzierten drei europäische T_EX User-groups (DANTE, GUTenberg und NTG) ein Projekt, das alle Varianten der Computer Modern Roman-Schriften in eine einzige Latin Modern-Schriftfamilie integrieren soll. Das Projekt wird von Bogusław Jackowski und Janusz Nowacki durchgeführt und die erste offizielle Version der Latin Modern Fonts wurde auf dem DANTE-Treffen im Jahre 2003 vorgestellt.

Bei den **Latin Modern** Schriften handelt es sich um sorgsam ausgearbeitete PostScript Type 1-Fonts nach dem Design der *Computer Modern* Familien von Knuth. Sie enthalten alle Zeichen, die man zum Setzen der europäischen Sprachen mit lateinischer Schrift benötigt. Zur Zeit unterstützen sie die Kodierungen T1 und TS1. In einem späteren Stadium wird sich das Projekt auch mit Zeichen für indianische Sprachen, Vietnamesisch sowie für die Transliteration befassen. Außerdem sind 8-Bit-Kodierungen für Mathematikfonts geplant (auf Grundlage früherer Arbeiten von CLASEN/VIETH und ZIEGLER [40,175]).

```
\usepackage{lmodern} \usepackage[T1]{fontenc}
```

Bei den \textbf{Latin Modern} Schriften handelt es sich um sorgsam ausgearbeitete PostScript Type\,1-Fonts nach dem Design der \emph{Computer Modern} Familien von Knuth. Sie enthalten alle Zeichen, die man zum Setzen der europäischen Sprachen mit lateinischer Schrift benötigt. Zur Zeit unterstützen sie die Kodierungen \texttt{T1} und \texttt{TS1}. In einem späteren Stadium wird sich das Projekt auch mit Zeichen für indianische Sprachen, Vietnamesisch sowie für die Transliteration befassen. Außerdem sind 8-Bit-Kodierungen für Mathematikfonts geplant (auf Grundlage früherer Arbeiten von \textsc{Clasen/Vieth} und \textsc{Ziegler}~[40,175]).

Bsp.
7-5-5

Als dieses Buch geschrieben wurde, befanden sich die Fonts noch im Stadium der Feinabstimmung. So wurden beispielsweise noch weitere Kerning-Paare und sprachspezifische Ligaturen hinzugefügt. Man kann davon ausgehen, dass die Latin Modern Fonts in einer zukünftigen Version die L^AT_EX-Standardschriften sein werden; vorerst lassen sie sich über das Paket lmodern mit der T1-Kodierung nutzen.

7.5.2 inputenc – Auswählen der Eingabekodierung

Wenn der Computer die Eingabe von Akzentbuchstaben erlaubt, sei es über einzelne Tasten oder ein anderes Eingabeverfahren (z.B. in dem man für à erst ‘ und dann a eingibt), und sie auch im Editor richtig anzeigt...

Quand ils furent revenus un peu à eux, ils marchèrent
vers Lisbonne ; il leur restait quelque argent, avec
lequel ils espéraient se sauver de la faim après
avoir échappé à la tempête (Voltaire)

...dann würde man diesen Text in L^AT_EX idealerweise sofort verwenden, anstatt noch \‘a, \^e, usw. eingeben zu müssen.

Während letzteres Verfahren bei Sprachen wie Französisch oder Deutsch noch möglich ist, benötigen Russisch und Griechisch direkte Eingabemöglichkeiten, da (fast) jeder ihrer Buchstaben L^AT_EX-intern durch einen Befehlsnamen abgebildet wird. Die Russische Definition für \reftextafter enthält z.B. den folgenden Text (der „auf der nächsten Seite“ bedeutet):

```
\cyrn\cyra\ \cyr\cyr1\cyre\cyrd\cyr\cyr2\cyrshch\cyre\cyrshrt
\ \cyr\cyr3\cyr4\cyra\cyrn\cyr1\cyr2\cyr3\cyr4
```

Es ist völlig klar, dass niemand auf Dauer mit solchen Eingaben arbeiten möchte. Der Vorteil liegt jedoch darin, dass dieser Text universell portierbar ist, d.h. er wird auf allen L^AT_EX-Installationen korrekt interpretiert. Andererseits ist die Eingabe von

на следующей странице

auf einer geeigneten Tastatur in jedem Falle vorzuziehen, sofern L^AT_EX diese Angaben verstehen kann. Das Problem ist, dass in einer Datei nicht die Zeichen, die wir sehen, gespeichert werden, sondern Oktette (acht Bits), die diesen Zeichen entsprechen. Unter anderen Umständen (bei einer anderen Kodierung) können die gleichen Oktette für andere Zeichen stehen.

Wie bestimmt L^AT_EX, welche Interpretation die Richtige ist? Solange man auf dem gleichen Computer bleibt und alle Programme die Oktette in den Dateien (beim Lesen und Schreiben) auf die gleiche Weise interpretieren, gibt es normalerweise keine Probleme. In solchen Situationen kann es sinnvoll sein, einen automatischen Übersetzungsmechanismus zu aktivieren, der in einigen neueren T_EX-Versionen enthalten ist. Wenn jedoch irgendeine Datei von diesem System auf einen anderen Computer übertragen wird, ist es sehr wahrscheinlich, dass die Verarbeitung misslingt, oder noch schlimmer, scheinbar gelingt, aber falsche Ergebnisse mit falschen Zeichen erzeugt.

Für diese Fälle wurde das Paket inputenc entwickelt. Sein Hauptzweck besteht darin, L^AT_EX mitzuteilen, welche Kodierung in einem Dokument oder Teil eines Dokumentes verwendet wurde. Dazu lädt man das Paket mit dem Namen der Kodierung als Option. Zum Beispiel:

```
\usepackage[cp1252]{inputenc} % Windows 1252 (Western Europe) Codepage
```

Von diesem Augenblick an weiß jede $\text{\LaTeX}$ -Installation, wie die Oktette im verbleibenden Dokument zu interpretieren sind,¹ ungeachtet der Kodierungen, die in anderen Fällen auf diesem Computer verwendet werden.

Hierzu als typisches Beispiel ein kurzer Text, der mit der in Russland verbreiteten `koi8-r`-Kodierung geschrieben wurde. Rechts sieht man den Text, wie er auf einem Computer mit Latin 1-Kodierung erscheint (z.B. in Deutschland). Links zeigt sich, dass $\text{\LaTeX}$ den Text trotzdem richtig interpretieren konnte, da die entsprechende Eingabekodierung angegeben ist.

Русский язык (Die russische Sprache)	<pre>\usepackage[russian]{babel} \usepackage[koi8-r]{inputenc}</pre>	òõóöëë ñűë (Die russische Sprache)

 Bsp.
7-5-6

Weiter unten folgt eine Liste der derzeit von `inputenc` unterstützten Kodierungen. Die Schnittstelle ist gut dokumentiert und es lassen sich leicht neue Kodierungen hinzufügen. Wenn eine benötigte Kodierung hier nicht aufgeführt ist, lohnt es sich daher in die Dokumentation des `inputenc`-Paketes zu schauen², um zu sehen, ob sie bereits hinzugefügt wurde. Man kann auch das Internet nach `inputenc`-Kodierungsdateien von anderen Autoren durchsuchen. Kodierungen für kyrillische Sprachen werden z.B. mit anderen Paketen zur Unterstützung kyrillischer Fonts verteilt.

Der ISO-Standard 8859 [72] definiert eine Reihe wichtiger Single-Byte-Kodierungen, von denen `inputenc` alle unterstützt, die mit dem Lateinischen Alphabet zu tun haben. Für die Betriebssysteme MS-DOS und Windows wurden von IBM und Microsoft eine Reihe von Single-Byte-Kodierungen definiert, von denen zurzeit ein Teil unterstützt wird. Außerdem sind einige von anderen Computerherstellern definierte Kodierungen verfügbar. Die etwas unsystematische (aber stetig wachsende) Auswahl resultiert hauptsächlich aus den Beiträgen der $\text{\LaTeX}$ -Anwendergemeinschaft.

latin1 Die ISO 8859-1 Kodierung (auch bekannt als Latin 1). Sie kann die meisten westlichen Sprachen darstellen, einschließlich Albanisch, Dänisch, Deutsch, Englisch, Färöisch, Finnisch, Französisch, Galizisch, Isländisch, Irisch, Italienisch, Katalanisch, Niederländisch, Norwegisch, Portugiesisch, Spanisch und Schwedisch.

latin2 Die ISO Latin 2 Kodierung (ISO 8859-2) unterstützt die slawischen Sprachen Mitteleuropas, die das lateinische Alphabet verwenden. Sie kann für die folgenden Sprachen eingesetzt werden: Deutsch, Kroatisch, Polnisch, Rumänisch, Slowakisch, Slowenisch, Tschechisch und Ungarisch

latin3 Dieser Zeichensatz (ISO 8859-3) wird für Esperanto, Galizisch, Maltesisch und Türkisch benutzt.

latin4 Die ISO Latin 4 Kodierung (ISO 8859-4) kann für Sprachen wie Estnisch, Lettisch und Litauisch genutzt werden.

¹Das stimmt nur, wenn $\text{\TeX}$ nicht so eingestellt wurde, dass es beim Lesen einer Datei einige fest zugeordnete Umwandlungen vornimmt. Wie in der Einleitung zu diesem Kapitel erwähnt, wurden viele $\text{\TeX}$ -Umgebungen dahingehend erweitert, solche Transformationen zu unterstützen. Wenn diese jedoch aktiviert werden, können Dokumente in verschiedenen Sprachen nicht mehr parallel verarbeitet werden.

²Die Datei `inputenc.dtx` mit $\text{\LaTeX}$ verarbeiten.

`latin5` Die ISO Latin 5 Kodierung (ISO 8859-9) ist eng verwandt mit Latin 1, tauscht jedoch die selten benötigten isländischen Buchstaben aus Latin 1 gegen türkische aus.

`latin9` Latin 9 (oder ISO 8859-15) ist eine weitere kleine Abwandlung von Latin 1, bei der das Euro-Zeichen und ein paar weitere Zeichen, wie die œ-Ligatur, ergänzt sind, die für die französische und finnische Sprache fehlten. Sie findet immer weitere Verbreitung als Ersatz für Latin 1.

`cp437` Codepage IBM 437 (MS-DOS Latin, aber mit vielen graphischen Zeichen zum Zeichnen von Kästen).

`cp437de` Codepage IBM 437 mit "ß" (scharfes s) anstelle von β (griechisches Beta) wie auf deutschen Tastaturen üblich.

`cp850` Codepage IBM 850 (MS-DOS mehrsprachig ≈ `latin1`).

`cp852` Codepage IBM 852 (MS-DOS mehrsprachig ≈ `latin2`).

`cp858` Codepage IBM 858 (IBM 850 ergänzt um Euro-Zeichen).

`cp865` Codepage IBM 865 (MS-DOS Norwegen).

`cp1250` Codepage Windows 1250 (Mittel- und Osteuropa).

`cp1252` Codepage Windows 1252 (Westeuropa).

`cp1257` Codepage Windows 1257 (Baltisch).

`ansinew` Windows 3.1 ANSI-Kodierung; Synonym für `cp1252`.

`decmulti` DEC Multinational Character Set-Kodierung.

`applemac` Macintosh (Standard)-Kodierung.

`macce` Codepage Macintosh Mitteleuropa.

`next` Next Computertastatur-Kodierung.

`utf8` Unterstützung für die Unicode UTF-8-Kodierung.

Die meisten T_EX-Installationen sind so voreingestellt, dass sie 8-Bit-Zeichen akzeptieren. Ohne weitere Anpassungen, wie etwa durch `inputenc`, können jedoch unerwartete Ergebnisse auftreten: Zeichen können verschwinden, oder man erhält eben das Zeichen, das sich im aktuellen Zeichensatz an der Position befindet, welche dem Oktett entspricht, und das vielleicht das erwünschte Zeichen ist – oder auch nicht. Dieses Verhalten war lange Zeit völlig normal, so dass es in L^AT_EX 2_ε nicht geändert wurde, da sich einige Anwender darauf verlassen. Um solche Fehler unterbinden zu können, bietet `inputenc` die Option `ascii`, die jedes Zeichen außerhalb des Bereiches 32–126 verbietet.

`\inputencoding{kodierung}`

Ursprünglich diene das Paket `inputenc` dem Zweck, die Kodierung für ein ganzes Dokument zu beschreiben – daher auch der Gebrauch von Optionen in der Präambel. Mithilfe des Befehls `\inputencoding` kann man die Kodierung jedoch auch mitten im Dokument ändern. Das Argument dieses Befehls ist der Name der Kodierung. Seine Verarbeitung erfordert relativ viele Ressourcen, da jedes Mal normalerweise mehr als 120 Zeichen neu zugeordnet

werden. Es sind jedoch durchaus einige Anwendungsbeispiele bekannt, in denen die Kodierung innerhalb eines Absatzes mehrfach geändert wird und die anscheinend trotzdem vernünftig funktionieren.

UTF-8-Unterstützung

Als `inputenc` geschrieben wurde, war $\text{\LaTeX}$ zumeist auf Computern installiert, die mit den in diesem Abschnitt beschriebenen Single-Byte-Kodierungen arbeiteten. In letzter Zeit verbreitet sich jedoch immer mehr eine andere Kodierung, da die Systeme zunehmend Unicode unterstützen: UTF-8. Diese Kodierung mit variablen Längen repräsentiert Unicode-Zeichen durch ein bis vier Oktette. Vor kurzem machten einige Linux-Distributionen UTF-8 zu ihrer Standardkodierung und verblüfften ihre $\text{\LaTeX}$ -Anwender damit, dass Dateien, die über die normale Tastatur geschrieben worden waren, von $\text{\LaTeX}$ nicht mehr akzeptiert wurden. Aus diesem Grund wurde das `inputenc`-Paket daraufhin um die Option `utf8` zur UTF-8-Unterstützung erweitert. Technisch gesehen handelt es sich dabei nicht um eine vollständige UTF-8-Einbindung. Es werden nur Zeichen zugeordnet, für die Entsprechungen in den $\text{\LaTeX}$ -Standardfonts existieren (d.h. hauptsächlich lateinische und kyrillische Zeichensätze); alle anderen Zeichen verursachen eine geeignete Fehlermeldung. Außerdem werden keine zusammengesetzten Unicode-Zeichen unterstützt – das sollte jedoch in der Praxis keine Schwierigkeit darstellen.

Umlaute in UTF-8: äöü
In Latin 1 interpretiert als:
Ã¸Ã¼

```
\usepackage[utf8]{inputenc}
\usepackage{textcomp} % für die Latin-Interpretation
Umlaute in UTF-8:      ^^c3^^a4^^c3^^b6^^c3^^bc
\par\inputencoding{latin1}% umschalten zu Latin 1
In Latin 1 interpretiert als: ^^c3^^a4^^c3^^b6^^c3^^bc
```

Bsp.
7-5-7

In UTF-8 werden ASCII-Zeichen durch sich selbst dargestellt und die meisten Lateinischen Zeichen belegen zwei Bytes. Im Quelltext des Beispiels werden die Zwei-Byte-Darstellungen der deutschen Umlaute in UTF-8 als hexadezimale $\text{\TeX}$ -Zeichen dargestellt, d.h. jedes Oktett wird durch `^^` eingeleitet. In einem nicht UTF-8-fähigen Editor erscheinen sie wahrscheinlich in ähnlicher Form, wie bei ihrer Interpretation als Latin 1-Zeichen.

Die UTF-8-Unterstützung durch `inputenc` beschränkt sich momentan¹ auf den Teilbereich von Unicode, der direkt durch die Zuordnungsoptionen von `inputenc` unterstützt wird, wie sie auf Seite 370 beschrieben sind. Ein Paket mit einer vollständigeren UTF-8-Unterstützung (auch für chinesische, koreanische und japanische Zeichen), das entsprechend natürlich auch komplizierter einzurichten ist, findet sich mit dem Paket `ucs` von Dominique Unruh. Wem `inputenc` nicht reicht, der könnte es einmal damit versuchen.

7.5.3 fontenc – Auswählen von Fontkodierungen

Um eine Textfontkodierung in $\text{\LaTeX}$ verwenden zu können, muss diese zunächst in der Dokumentenklasse, einem Paket oder der Dokumentenpräambel geladen werden. Genauer gesagt müssen die Definitionen geladen werden, mit deren Hilfe der Zugriff auf die Zeichen in den Zeichensätzen einer bestimmten Kodierung erfolgt. Normalerweise bedient man sich dazu des `fontenc`-Paketes,

¹Es handelt sich hier eher um ein Ressourcenproblem als um ein technisches und kann sich daher bald ändern.

das eine durch Kommas getrennte Liste von Fontkodierungen als Option erhält. Die letzte der aufgeführten Kodierungen wird automatisch zur voreingestellten Kodierung für das Dokument. Beim Laden kyrillischer Kodierungen wird außerdem die Liste der Befehle, auf die sich `\MakeUppercase` und `\MakeLowercase` auswirken, automatisch erweitert. Das Beispiel

```
\usepackage[T2A,T1]{fontenc}
```

lädt alle erforderlichen Definitionen für die kyrillische T2A- und die T1 (Cork)-Kodierung, wobei letztere zur Voreinstellung für das Dokument wird.

Im Gegensatz zu anderen Paketen kann man `fontenc` mehrfach laden und dem `\usepackage`-Befehl dabei jedes Mal verschiedene Werte im optionalen Argument übergeben. Das ist notwendig, damit zum einen eine Dokumentenklasse einen bestimmten Satz an Kodierungen laden kann, dem Anwender andererseits aber trotzdem die Möglichkeit bleibt, in der Präambel noch weitere zu laden. Der wiederholte Einsatz von `fontenc` hat keine Nebenwirkung (außer vielleicht eine Veränderung der Standardfontkodierung für das Dokument).

 `fontenc` darf
mehrfach geladen
werden

Wenn man für ein Dokument Pakete zur Sprachunterstützung verwendet (z.B. die des `babel`-Systems), dann werden die erforderlichen Fontkodierungen häufig schon von diesen Paketen geladen.

7.5.4 textcomp – Zusätzliche Textsymbole

Als in Cork die T1-Fontkodierung definiert wurde, beschloss man, viele Standardtextsymbole, wie etwa †, auszulassen und stattdessen so viele zusammengesetzte Zeichen wie möglich zu integrieren. Der Gedanke dahinter war, dass Zeichen, die in Wörtern vorkommen und dadurch der Silbentrennung unterliegen, im gleichen Font existieren müssen, während man sonstige Symbole ohne großen Aufwand aus zusätzlichen Fonts heranziehen kann. Diese Sonderzeichen wurden daher in einer begleitenden Kodierung zusammengefasst.

Die erste Version dieser Kodierung (TS1) wurde 1995 von Jörg Knappen [80, 81] entwickelt. Sebastian Rahtz lieferte mit dem Paket `textcomp` eine passende L^AT_EX-Schnittstelle dazu.

Leider orientierte sich die TS1-Kodierung, genau wie T1, an der Zeichenverfügbarkeit in der T_EX-Welt und ließ dabei außer Acht, dass die Mehrzahl der kommerziellen Fonts mit anderen Sätzen von Zeichen arbeiten. Daher ist diese Kodierung nur für sehr wenige Schriftfamilien, darunter EC und CM Bright, vollständig implementiert. Es existieren zwar Umsetzungen dieser Kodierung für die meisten PostScript-Fonts, wobei jedoch die Hälfte der Zeichen fehlt und somit nur quadratische Tintenkleckse erzeugt.¹ In Tabelle 7.6 auf den Seiten 374–375 sind die Symbole dargestellt, die mit `textcomp` verfügbar sind, sowie die Befehle, mit denen man auf sie zugreifen kann. Befehle in blauer Farbe zeigen an, dass die entsprechenden Glyphen in PostScript-Fonts sehr wahrscheinlich nicht zur Verfügung stehen.

¹Auch die T1-Kodierung hat dieses Problem im Zusammenhang mit PostScript-Fonts. Hier fehlen in den meisten Zeichensätzen jedoch glücklicherweise nur fünf (zudem selten gebrauchte) Zeichen (siehe Beispiel 7-9-2 auf Seite 428).

Akzentzeichen

Á	<code>\capitalacute_A</code>	Ä	<code>\capitalbreve_A</code>	Å	<code>\capitalcaron_A</code>
Â	<code>\capitalcedilla_A</code>	Â	<code>\capitalcircumflex_A</code>	Ä	<code>\capitaldieresis_A</code>
Ã	<code>\capitaldotaccent_A</code>	À	<code>\capitalgrave_A</code>	Ă	<code>\capitalhungarumlaut_A</code>
Ä	<code>\capitalmacron_A</code>	Â	<code>\capitalnewtie_A</code>	Ů	<code>\capitalogonek_U</code>
Å	<code>\capitalring_A</code>	Ö	<code>\capitaltie_OO</code>	Ã	<code>\capitaltilde_A</code>
ô	<code>\newtie_o</code>	Ⓐ	<code>\textcircled_A</code>	ö	<code>\t_oö</code>

Ziffern (hochgestellte, Brüche, gemeine)

¹	<code>\textonesuperior</code>	²	<code>\texttwosuperior</code>	³	<code>\textthreesuperior</code>
¼	<code>\textonequarter</code>	½	<code>\textonehalf</code>	¾	<code>\textthreequarters</code>
₀	<code>\textzerooldstyle</code>	₁	<code>\textoneoldstyle</code>	₂	<code>\texttwooldstyle</code>
₃	<code>\textthreeoldstyle</code>	₄	<code>\textfouroldstyle</code>	₅	<code>\textfiveoldstyle</code>
₆	<code>\textsixoldstyle</code>	₇	<code>\textsevenoldstyle</code>	₈	<code>\texteightoldstyle</code>
₉	<code>\textnineoldstyle</code>				

Paarweise Symbole

<	<code>\textlangle</code>	>	<code>\textrangle</code>	⌈	<code>\textlbrackdbl</code>
⌋	<code>\textrbrackdbl</code>	↑	<code>\textuparrow</code>	↓	<code>\textdownarrow</code>
←	<code>\textleftarrow</code>	→	<code>\textrightarrow</code>	{	<code>\textlquill</code>
}	<code>\textrquill</code>				

Währungs- und Wirtschaftssymbole

฿	<code>\textbaht</code>	¢	<code>\textcent</code>	¢	<code>\textcentoldstyle</code>
₯	<code>\textcolonmonetary</code>	¤	<code>\textcurrency</code>	\$	<code>\textdollar</code>
\$	<code>\textdollaroldstyle</code>	₫	<code>\textdong</code>	€	<code>\texteuro</code>
f	<code>\textflorin</code>	₲	<code>\textguarani</code>	₺	<code>\textlira</code>
₦	<code>\textnaira</code>	₱	<code>\textpeso</code>	£	<code>\textsterling</code>
₩	<code>\textwon</code>	¥	<code>\textyen</code>		
Ⓟ	<code>\textcircledP</code>	©	<code>\textcopyleft</code>	©	<code>\textcopyright</code>
%	<code>\textdiscount</code>	€	<code>\textestimated</code>	‰	<code>\textpertenthousand</code>
‰	<code>\textperthousand</code>	※	<code>\textreferencemark</code>	®	<code>\textregistered</code>
₪	<code>\textservicemark</code>	™	<code>\texttrademark</code>		

Blau kennzeichnet Symbole, die in den meisten PostScript-Fonts nicht verfügbar sind.

Tabelle 7.6: Durch das Paket `textcomp` bereitgestellte Befehle

*Zeichenteilmengen der
TS1-Kodierung*

Um diesen Problemen entgegenzuwirken, ist dem heutigen `textcomp`-Paket von den meisten Fontfamilien bekannt, inwieweit diese der TS1-Kodierung entsprechen. Außerdem verfügt es über eine Reihe von Optionen mit denen sich der Vorrat an neuen Befehlen für unbekannte Fontfamilien einschränken lässt.

Bei unbekannten Fontfamilien stehen mit der Option `safe` im Wesentlichen nur die Befehle des ISO-Adobe-Zeichensatzes zur Verfügung (ohne `\textcurrency`, aber zusätzlich mit einem künstlich erzeugten `\texteuro`). Die Option `euro` ersetzt das vorgetäuschte Euro-Zeichen durch eine echte Glyphe. Wenn dieses Symbol im Zeichensatz aber nicht verfügbar ist, erzeugt `\texteuro` dann einen hässlichen Tintenklecks.

Die Paketoption `full` aktiviert hingegen auch für solche Zeichensätze, die `textcomp` unbekannt sind, alle Befehle. Im Einzelnen funktionieren da-

Fußnotenzeichen					
*	<code>\textasteriskcentered</code>		<code>\textbardbl</code>		<code>\textbrokenbar</code>
•	<code>\textbullet</code>	†	<code>\textdagger</code>	‡	<code>\textdaggerdbl</code>
◦	<code>\textopenbullet</code>	¶	<code>\textparagraph</code>	·	<code>\textperiodcentered</code>
¶	<code>\textpilcrow</code>	§	<code>\textsection</code>		
Wissenschaftliche Symbole					
°C	<code>\textcelsius</code>	°	<code>\textdegree</code>	÷	<code>\textdiv</code>
¬	<code>\textlnot</code>	Ω	<code>\textmho</code>	—	<code>\textminus</code>
μ	<code>\textmu</code>	Ω	<code>\textohm</code>	♂	<code>\textordfeminine</code>
♂	<code>\textordmasculine</code>	±	<code>\textpm</code>	√	<code>\textsurd</code>
×	<code>\texttimes</code>				
Verschiedene					
”	<code>\textacutedbl</code>	´	<code>\textasciicute</code>	˘	<code>\textasciibreve</code>
˘	<code>\textasciicaron</code>	¨	<code>\textasciidieresis</code>	`	<code>\textasciigrave</code>
—	<code>\textasciimacron</code>	○	<code>\textbigcircle</code>	␣	<code>\textblank</code>
★	<code>\textborn</code>	=	<code>\textdblhyphen</code>	≡	<code>\textdblhyphenchar</code>
†	<code>\textdied</code>	o o	<code>\textdivorced</code>	/	<code>\textfractionsolidus</code>
”	<code>\textgravedbl</code>	?	<code>\textinterrobang</code>	↓	<code>\textinterrobangdown</code>
♣	<code>\textleaf</code>	∞	<code>\textmarried</code>	♪	<code>\textmusicalnote</code>
Nº	<code>\textnumero</code>	'	<code>\textquotesingle</code>	,	<code>\textquotestraightbase</code>
„	<code>\textquotestraightdblbase</code>	R	<code>\textrecipe</code>	—	<code>\textthreequartersemdash</code>
~	<code>\texttildelow</code>	—	<code>\texttwelveudash</code>		

Blau kennzeichnet Symbole, die in den meisten PostScript-Fonts nicht verfügbar sind.

Tabelle 7.6: Durch das Paket `textcomp` bereitgestellte Befehle (Forts.)

durch die eigentlich gültigen L^AT_EX-Befehle `\textcircled` und `\t` nicht mehr, sobald man für das Dokument einen Zeichensatz auswählt, der in seiner TS1-Kodierung nicht über die erforderlichen Symbole verfügt. Darum lässt die Option `almostfull` diese beiden Befehle unangetastet, um ein Ergebnis wie das folgende zu vermeiden.

Bsp. 7-5-8	<code>\usepackage[force,full]{textcomp}</code>	
	CM Fonts:  	CM Fonts: <code>\textcircled{x}</code> \quad <code>\t oo</code> \par Times Fonts: Times Fonts:   <code>\fontfamily{ptm}\selectfont\textcircled{x}</code> \quad <code>\t oo</code> \par

Da der Font Times Roman dem Paket `textcomp` bekannt ist, erzeugt die Option `full` immer noch ein korrektes Ergebnis; für die Tintenkleckse im vorigen Beispiel musste außerdem die Option `force` ergänzt werden. Diese Option zwingt `textcomp` dazu, alle fontspezifischen Informationen zu ignorieren, und immer den Zeichenvorrat zu verwenden, den die zusätzliche Option fordert.¹

Mit `textcomp` (mit oder ohne einschränkende Optionen), erhält man zahlreiche neue Befehle für den Zugriff auf die neuen Symbole. Außerdem werden

¹Man sollte diese Option tunlichst meiden, da sie ohne jegliche Fehlermeldung falsche Ausgaben erzeugen kann.

eine Reihe von Symbolen, die L^AT_EX traditionell aus Mathematikfonts bezogen hat (wie etwa `\textbullet` oder `\textdagger`), jetzt aus den begleitenden Zeichensätzen geladen; folglich wechseln sie hin und wieder ihre Form, wenn die Fontattribute (Familie, Serie, Form) geändert werden.

`\usepackage[safe]{textcomp}`
`\textdagger\textparagraph\textbullet{} vs. \`
`\fontfamily{ptm}\selectfont\textdagger\textparagraph\textbullet`

Bsp.
7-5-9

Obwohl diese Lösung meistens das passende Ergebnis erzeugt, kann sie zu unerwarteten Änderungen führen. So verwendet die `itemize`-Umgebung z.B. in der Voreinstellung `\textbullet`, um Elemente der ersten Ebene zu markieren. Bevorzugt man das etwas größere Aufzählungszeichen, so muss man die geänderten Voreinstellungen wieder auf die richtige Mathematikfont-Kodierung zurücksetzen. Dabei handelt es sich normalerweise um die Kodierung OMS¹; vgl. Beispiel 7-5-9.

`\usepackage[safe]{textcomp}`
`\DeclareTextSymbolDefault{\textbullet}{OMS}`
`\textbullet{} nun wie \fontfamily{ptm}\selectfont\textbullet`

Bsp.
7-5-10

In diesem Fall wäre es aber sicher vernünftiger, die Definition für `\labelitemi` (siehe Abschnitt 3.3.1) z.B. folgendermaßen anzupassen:

`\renewcommand\labelitemi{\normalfont\UseTextSymbol{OMS}{\textbullet}}`

Diakritische Zeichen über Großbuchstaben sind in einigen Fontdesigns manchmal flacher geneigt als über den entsprechenden Kleinbuchstaben. Auch die EC Fonts folgen dieser Tradition. Dort sind beispielsweise die Gravis-Akzente über `ò` und `Ö` unterschiedlich (anders als bei dem Font Lucida, der in diesem Buch als Grundschrift dient). Das wird immer dann problematisch, wenn man einen ungewöhnlichen Buchstaben benötigt, der in der T1-Kodierung nicht als einzelnes Zeichen verfügbar ist, sondern erst aus einem diakritischen Zeichen und dem Grundbuchstaben zusammengesetzt werden muss. In diesem Fall wird das gleiche Akzentzeichen für Groß- und Kleinbuchstaben benutzt, was zu deutlichen Unterschieden führen kann (vgl. das `Ẋ` im nächsten Beispiel). Zur Lösung dieses Problems erzeugen die `\capital...`-Befehle aus Tabelle 7.6 auf Seite 374 eigens passende diakritische Zeichen für Großbuchstaben.

`òẊ ÒẊ ÔẊ` `\usepackage[T1]{fontenc} \usepackage[safe]{textcomp}`
`\Huge \'o\'x \'O\'X \capitalgrave O\capitalgrave X`

Bsp.
7-5-11

L^AT_EX verfügt über den Befehl `\textcompwordmark`, ein unsichtbares Zeichen ohne horizontale Ausdehnung, mit dessen Hilfe man z.B. unerwünschte Ligaturen (auf Kosten der automatischen Silbentrennung) aufbrechen kann. Das `textcomp`-Paket stellt dieses Symbol mit einer Höhe von `1ex` bereit, so dass es als Argument eines Akzentbefehls zwischen zwei Buchstaben gesetzt

¹Die richtige Kodierung findet sich in der Deklaration für die Voreinstellung in der Datei `latex.ltx`.

werden kann. Im nächsten Beispiel wird mit diesem Befehl die deutsche Abkürzung für -burg erzeugt. Außerdem enthält das `textcomp`-Paket zwei zusätzliche Zeichen für zusammengesetzte Wörter: `\textascendercompwordmark` und `\textcapitalcompwordmark`, mit der Höhe der Oberlängen bzw. der Großbuchstaben des jeweiligen Fonts.

Bsp.
7-5-12

b̃g (klappt nicht)
b̃g BG

```
\usepackage[T1]{fontenc} \usepackage[safe]{textcomp}
b\u{g} (klappt nicht) \par
b\u\textcompwordmark g \quad B\u\textcapitalcompwordmark G
```

Das obige Beispiel funktioniert nur mit T1-kodierten Fonts (für `\textcapitalcompwordmark` ist außerdem `textcomp` erforderlich). In der Standarddefinition verwendet L^AT_EX für `\textcompwordmark` kein echtes Zeichen ohne horizontale Ausdehnung, sondern in Ermangelung dessen einen Leerraum der Breite null.

Da das \$-Zeichen sowohl in den OT1- als auch in den T1-Kodierungen vorkommt, gibt es keinen Grund, seine Definition zu entfernen und L^AT_EX die TS1-Version aufzuzwingen, wenn man mit dieser Kodierung arbeitet. Wenn man jedoch immer automatisch die Variante des Dollarzeichens \$ (`\textdollaroldstyle`) verwenden möchte, muss man die Deklarationen in den anderen Kodierungen entfernen, damit L^AT_EX automatisch auf TS1 umschaltet.

```
\DeclareTextCommandDefault{\textdollar} % neuen Standard
{\UseTextSymbol{TS1}\textdollaroldstyle} % einrichten
\UndeclareTextCommand{\textdollar}{OT1} % keine Defs aus
\UndeclareTextCommand{\textdollar}{T1} % OT1/T1 verwenden
```

Das Umdeklarieren funktioniert natürlich nur, wenn die Fonts des Dokumentes das gewünschte Zeichen in der TS1-Kodierung enthalten. Für dieses Buch hätte man damit keinen Erfolg, da die hier verwendete Grundschrift Lucida Bright nur über den eingeschränkten ISO-Adobe Zeichensatz verfügt. Wenn man sich nun fragt, wo das \$-Zeichen und ähnliche Symbole in diesem Buch herkommen, so liegt die Antwort auf der Hand: aus den EC Fonts.

Aber was kann man tun, wenn man z.B. `\textborn` verwenden möchte, dies aber nicht von der aktuellen Fontfamilie unterstützt wird? Zum einen kann man die Voreinstellung des `textcomp`-Paketes mithilfe von `\DeclareTextCommandDefault` überschreiben. Dahinter steckt die Idee, per Voreinstellung auf eine Fontfamilie umzuschalten, die das gewünschte Symbol enthält (z.B. `cmr`, wenn die Grundschrift über Serifen verfügt und `cmss` für eine serifenlose Schrift), und danach mithilfe des Befehls `\UseTextSymbol` das Symbol aus der TS1-Kodierung dieser Familie zu nehmen.¹

Bsp.
7-5-13

Burkhard und Holger
*8.11.1997

```
\usepackage[safe]{textcomp}
\DeclareTextCommandDefault{\textborn}
{{\fontfamily{cmr}\selectfont\UseTextSymbol{TS1}{\textborn}}}
\fontfamily{ptm}\selectfont
Burkhard und Holger \textborn 8.11.1997
```

¹Bei abstrakteren Symbolen erhält man mit diesem Ansatz in der Regel recht akzeptable Resultate; bei Akzenten kann die Qualität der Ergebnisse sehr schwanken.

Dieses Verfahren eignet sich für alle Symbole des `textcomp`-Paketes. Die Definition für Akzente sieht ähnlich aus. Dieses Mal erhält der Befehl für die Voreinstellung jedoch ein Argument und in der Definition kommt `\UseTextAccent` zum Einsatz. Zum Beispiel:

```
\DeclareTextCommandDefault{\newtie}[1]
  {{\fontfamily{cmr}\selectfont\UseTextAccent{TS1}{\newtie}{#1}}}
```

Im Falle von Symbolen (aber nicht bei Akzenten) versucht `textcomp` eigentlich, das Problem fehlender Zeichen zu lösen, indem es lokal auf eine in `\textcompstbsubstdefault` gespeicherte Fontfamilie (normalerweise Computer Modern Roman) umschaltet und – mit einer passenden Fehlermeldung – das Zeichen aus dieser Familie setzt. Durch die Option `warn` erhält man hier nur Warnungen anstelle von Fehlern. Solche Ersetzungen sind natürlich unschön, insbesondere im Falle von „Textsymbolen“, wenn der aktuelle Font optisch nicht zum Ersetzungsfont passt. Im nächsten Beispiel ist der Ersatzfont Computer Modern Sans. Dabei ist darauf zu achten, dass die gewählte Familie die TS1-Kodierung auch wirklich uneingeschränkt unterstützt; andernfalls erzeugt die geänderte Definition endlose Fehlermeldungen!

```
\usepackage[warn]{textcomp}
\renewcommand\textcompstbsubstdefault{cmss}
\fontfamily{phv}\selectfont Helvetica mit \textnumero,
\textohm, \textcopyright, \textpildcrow. Nicht perfekt,
aber besser als gar nichts.
```

Helvetica mit №, Ω, ©,
¶. Nicht perfekt, aber besser
als gar nichts.

Bsp.
7-5-14

Entsprechend ihrer Spezifikation enthält die TS1-Kodierung Mediävalziffern (auch Minuskel-, Charakter- oder gemeine Ziffern genannt), sowie die Satzzeichen Punkt und Komma. So lassen sich Datumsangaben und andere (positive) Zahlen in diesen Ziffern setzen, indem man einfach auf die TS1-Fontkodierung umschaltet. Leider enthalten die meisten PostScript-Fonts (außer in den kommerziellen „Expertensätzen“) keine solchen Ziffern, so dass diese Methode nur mit wenigen Schriftfamilien richtig funktioniert.¹

```
\usepackage[warn,safe]{textcomp}
\newcommand\born[1]{\textborn
  {\fontencoding{TS1}\selectfont #1}}
\raggedright
\fontfamily{phv}\selectfont Arno \born{29.11.1984},
\fontfamily{ccr}\selectfont
Burkhard und Holger \born{8.11.1997}
```

Arno *■■■■■■■■■,
Burkhard und Holger
*8.11.1997

Bsp.
7-5-15

Das `textcomp`-Paket löst dieses Problem, indem es den Befehl `\oldstylenums` so undefiniert, dass er automatisch die Mediävalziffern der TS1-Kodierung verwendet, wenn sie im aktuellen Font enthalten sind. Andernfalls gibt er eine Warnung aus und erzeugt stattdessen Versalziffern (auch Majuskelziffern genannt).

¹Wenn die Symbole, wie im Beispiel, durch Umschalten auf TS1 direkt angesteuert werden, bleiben einschränkende Optionen (wie etwa `safe`) ohne Auswirkung.

Bsp.
7-5-16

Arno *29.11.1984,
Burkhard und Holger
*8.11.1997

```
\usepackage[warn]{textcomp}
\newcommand\born[1]{\textborn\oldstylenums{#1}}
\raggedright
\fontfamily{phv}\selectfont Arno \born{29.11.1984},
\fontfamily{ccr}\selectfont
Burkhard und Holger \born{8.11.1997}
```

Wer Fonts besitzt, die `textcomp` nicht kennt (oder aus irgendeinem Grund bezüglich ihres Umfangs falsch einschätzt), kann dem Paket mithilfe der Konfigurationsdatei `textcomp.cfg` die erforderlichen Informationen über die betreffende Schriftfamilie vermitteln. Die kommerziellen Lucida Blackletter Fonts enthielten z.B. ursprünglich nur die grundlegenden ISO-Adobe-Symbole, so dass `textcomp` hier auf Nummer sicher geht und auch nur diese Symbole zulässt. Mittlerweile enthalten die Fonts jedoch auch das `\textohm`-Zeichen. Wenn man nach Laden des Paketes (oder in der Konfigurationsdatei) den Befehl `\DeclareEncodingSubset` verwendet, kann man das Zeichen nun auch aus dieser Fontfamilie setzen.

Bsp.
7-5-17

Nun ist Ω verfügbar,
aber ■ misslingt ohne
Warnung.

```
\usepackage[T1]{fontenc} \usepackage{textcomp} \raggedright
\DeclareEncodingSubset{TS1}{hlcf}{3}
\fontfamily{hlcf}\selectfont Nun ist \textohm{} verfügbar,
aber \texteuro{} misslingt ohne Warnung.
```

Die Dokumentation in der Datei `ltoutenc.dtx` der Standard-L^AT_EX-Distribution enthält weitere Einzelheiten zu `\DeclareEncodingSubset` und den Nummern (im dritten Argument), die die verfügbaren Untermengen der Zeichen definieren.

7.5.5 exscale – Skalieren großer Operatoren

Normalerweise wird der Font für große mathematische Symbole nur in einer Größe verwendet. Gewöhnlich reicht dies aus, da dieser Zeichensatz die meisten Zeichen schon in mehreren Größen enthält und L^AT_EX über spezielle Funktionen verfügt, welche die geeignete Größe auswählen. Wenn in einem Dokument jedoch viele Formeln in großer Schrift benötigt werden – z.B. in Überschriften – kann es vorkommen, dass die ausgewählten Zeichen zu klein sind. In diesem Fall kann man auf das Paket `exscale` zurückgreifen, das passende Erweiterungsfonts für die jeweiligen Schriftgrade bereitstellt. Dieses Paket funktioniert allerdings nur in Dokumenten, die Computer Modern Mathematikfonts verwenden. Andere Pakete, die mit alternativen Einstellungen für Mathematikfonts arbeiten, bieten diese Funktionalität jedoch häufig als Paketoption.

7.5.6 tracefnt – Überwachen der Fontauswahl

Das Paket `tracefnt` dient dazu, Probleme bei der Zeichensatzauswahl aufzuspüren. Es unterstützt eine Reihe von Optionen, mit denen sich Art und Umfang der Meldungen, die von NFSS auf den Bildschirm und in der Protokolldatei ausgegeben werden, nach Bedarf einstellen lassen.

errorshow Diese Option unterdrückt alle Meldungen und Warnungen auf dem Bildschirm; sie werden nur in der Protokolldatei aufgezeichnet. Echte Fehler werden natürlich nach wie vor angezeigt. Da Warnungen über Fontersetzungen darauf hinweisen können, dass das Ergebnis fehlerhaft ist, sollte man die Protokolldatei aufmerksam durchsehen, bevor man ein wichtiges Dokument ausdruckt.

warningshow Bei Angabe dieser Option werden Warnungen und Fehlermeldungen auf dem Bildschirm ausgegeben. Es werden genauso viele Informationen ausgegeben wie in $\text{\LaTeX 2}\epsilon$, wenn das Paket `tracefnt` nicht geladen ist.

infoshow Dies ist die Standardoption für das Paket `tracefnt`. Zusätzliche Informationen, die normalerweise nur in der Protokolldatei aufgezeichnet werden, erscheinen damit auch auf dem Bildschirm.

debugshow Diese Option erzeugt zusätzliche Meldungen über Änderungen des Textfonts und seine Wiederherstellung am Ende einer Gruppe in Klammern oder einer Umgebung. Diese Option ist mit Bedacht zu verwenden, denn sie kann sehr große Protokolldateien erzeugen, die viel Speicherplatz benötigen.

Neben diesen Optionen zur „Standardüberwachung“¹ unterstützt das Paket `tracefnt` noch die folgenden Optionen:

pausing Alle Warnungen werden wie Fehler behandelt, um die Fehlersuche in wichtigen Dokumenten zu vereinfachen.

loading Diese Option zeigt an, welche externen Fonts geladen werden. Wenn allerdings das Format oder die Dokumentenklasse bereits einige Fonts geladen hat, werden diese von der Option nicht angezeigt.

7.5.7 `nfssfont.tex` – Anzeigen von Glyphentabellen

Die $\text{\LaTeX}$ -Distribution enthält eine Datei namens `nfssfont.tex`, mit deren Hilfe sich neue Fonts testen, Zeichensatztabellen mit allen Zeichen anzeigen und ähnliche Operationen in Bezug auf Zeichensätze durchführen lassen. Diese Datei ist eine überarbeitete Version des ursprünglich von Donald Knuth geschriebenen Programms `testfont.tex`. Wird dieses Programm von $\text{\LaTeX}$ abgearbeitet, so fordert es den Anwender zunächst auf, den Namen des zu testenden Zeichensatzes einzugeben. Hier kann man, falls bekannt, entweder den Namen des externen Fonts ohne Erweiterung eingeben – wie etwa `cmr10` (Computer Modern Roman 10pt) – oder man gibt eine leere Antwort. Im letzteren Fall wird man aufgefordert, eine NFSS-Fontspezifikation einzugeben, d.h. die Namen der Kodierung (voreingestellt auf `T1`), der Schriftfamilie (voreingestellt auf `cmr`), der Schriftserie (voreingestellt auf `m`) und der Schriftform (voreingestellt auf `n`) sowie den Schriftgrad (voreingestellt auf 10pt). Daraufhin lädt das Paket den entsprechenden externen Zeichensatz.

Als nächstes folgt die Aufforderung, einen Befehl einzugeben. Der wichtigste Befehl ist wohl `\table`, der eine Zeichensatztabelle wie jene auf Seite 446 erzeugt. Auch der Befehl `\text`, der ein längeres Textbeispiel erstellt,

¹Paketprogrammierer, die eine Überwachung ihrer Pakete unterstützen wollen, sollten wenn möglich von diesen vier Standardnamen Gebrauch machen.

ist von Interesse. Mit `\init` schaltet man auf einen neuen Zeichensatz um; `\bye` oder `\stop` beenden den Test; und mit dem Befehl `\help` erhält man Informationen über die weiteren möglichen Tests (die derzeit noch weitgehend auf die OT1-Kodierung zugeschnitten sind).

Mit etwas Aufwand kann man das Programm auch ohne Abfragen verwenden, sofern die vorhandene L^AT_EX-Installation die Umlenkung von Eingaben unterstützt. Wenn z.B. die Datei `nfssfont.in` aus den folgenden Zeilen besteht:

```
cmr10
\table \newpage \init

T1
cmss
bx
n
10
\text \bye
```

würde der Programmaufruf `latex nfssfont < nfssfont.in` (auf UNIX-Rechnern) den Inhalt dieser Datei lesen und zunächst eine Zeichensatztablette für den Font `cmr10` und dann ein Textbeispiel für `T1/cmss/bx/n/10` erzeugen.

Dabei sind zwei Dinge zu beachten: Zum einen gibt das Programm `nfssfont.tex` implizit einen `\init`-Befehl aus, so dass die erste Eingabezeile entweder einen Fontnamen enthalten oder völlig leer bleiben muss (in letzterem Falle um anzuzeigen, dass eine NFSS-Klassifikation folgt). Zum anderen müssen die Eingaben für `\init` in einzelnen Zeilen ohne zusätzliche Daten stehen, da das Zeilenende gleichzeitig das Ende der Antwort auf eine im interaktiven Modus gestellte Frage ist (wie etwa: “Font encoding [T1]: \encoding=”). Auch Kommentare sind nicht möglich, da sie das Zeilenende verbergen würden.

7.6 PSNFSS – PostScript Fonts in L^AT_EX

Das PSNFSS-System, das ursprünglich von Sebastian Rahtz entwickelt wurde, besteht aus einer Reihe von Paketen, welche die Verwendung von PostScript Fonts im L^AT_EX-Fontauswahlschema erleichtern. Es deckt die 35 „elementaren“ PostScript-Fonts ab (über die alle Level 2 PostScript-Drucker und der ghostscript-Interpreter verfügen), sowie die freien Charter und Utopia Fonts.¹ Die aktuelle PSNFSS-Version wird von Walter Schmidt gepflegt und gehört in die Gruppe der als „erforderlich“ eingestuften Hilfsdateien, die bei keiner L^AT_EX-Installation fehlen sollten.

Im alltäglichen Gebrauch benötigt man sicherlich nur eines (oder einige) der Pakete, die in Tabelle 7.7 auf der nächsten Seite aufgeführt sind, um die

¹Wenn die Utopia Fonts bei einer T_EX-Version fehlen, kann man sie vom CTAN-Verzeichnis `fonts/utopia` herunterladen. In der T_EX-Dokumentation ist erklärt, wie sie installiert werden.

Paket	Serifenschrift	Serifenlose Schrift	Schreibmaschinenschrift	Formeln
(none)	CM Roman	CM Sans Serif	CM Typewriter	CM Math
mathptmx	Times			Times + Symbol
mathpazo	Palatino			Palatino + Pazo
charter	Charter			
utopia*	Utopia			
chancery	Zapf Chancery			
helvet		Helvetica		
avant		Avant Garde		
courier			Courier	
bookman	Bookman	Avant Garde	Courier	
newcent	New Century Schoolbook	Avant Garde	Courier	

Veraltete Pakete

times	Times	Helvetica	Courier
palatino	Palatino	Helvetica	Courier
mathptm	Times		Times + Symbol + CM
mathppl	Palatino		Palatino + Symbol + Euler

*Ein alternatives Paket, das Formeln unterstützt, ist fourier (beschrieben in Abschnitt 7.7.7).

Tabelle 7.7: Von PSNFSS-Paketen verwendete Fonts

voreingestellten Schriften mit Serifen, ohne Serifen und/oder die Schreibmaschinenschrift zu ändern. Beim Betrachten der Tabelle fällt auf, dass nur zwei Pakete versuchen, neue Fonts für Formeln einzurichten, und dass die ersten acht Pakete jeweils nur Fonts in einer der drei Textkategorien verändern. Um also Times als Serifenschrift, Helvetica als serifenlose Schrift und Courier als Schreibmaschinenschrift zu verwenden, müsste man die Pakete `mathptmx`, `helvet` und `courier` laden. Warum wird dann das Paket `times`, das alle diese Änderungen auf einmal vornimmt, als veraltet angesehen?

Helvetica passend zu anderen Fonts skalieren

Einer der Gründe besteht darin, dass eine in ihrer nominalen Größe geladene Helvetica im Zusammenspiel mit Times und Courier einfach zu groß ist. In einem Layout, in dem Helvetica z.B. nur für Überschriften eingesetzt wird, ist dies nicht weiter von Belang. Wenn die Fonts jedoch im Fließtext gemischt werden (was in $\text{\LaTeX}$ durch Befehle wie `\textsf` ja recht einfach ist), dann erzeugt ein Paket wie `times` eher fragwürdige Resultate. Das Paket `helvet` hingegen bietet durch die Option `scaled` eine Verkleinerung der Fonts auf 95% der angeforderten Größe. Diese Option ist eigentlich ein Schlüssel-Wert-Paar, mit dem eine noch genauere Steuerung möglich ist – `scaled=0.92` lädt die Fonts mit 92% ihrer Originalgröße.

Es gibt jedoch einen Fall, in dem sich trotzdem das `times`-Paket empfiehlt: Wenn man die Einstellungen für die Mathematikfonts nicht ändern, oder einen Satz anderer Fonts für Formeln verwenden möchte. Dabei kann man das `helvet`-Paket immer noch nachträglich laden, um eine Größenanpassung vorzunehmen.

Die PSNFSS-Sammlung verwendet durchgängig die Nomenklatur von Karl Berry [19]; die Klassifizierung und die externen Fontbezeichnungen sind in Tabelle 7.8 auf der nächsten Seite aufgeführt. Mithilfe dieser Tabelle kann man auch ohne Laden eines Paketes leicht auf einzelne Fonts zugreifen, z.B. über den Befehl `\usefont` (siehe Beispiel 7-6-1 weiter unten). Da sich diese Fonts sehr einfach auf jede beliebige Größe skalieren lassen, bieten sich hier attraktive Möglichkeiten für Überschriften oder Titelseiten, denn diese Methode erleichtert den Einsatz von anderen Schriftgraden, als sie mit den L^AT_EX-Standardbefehlen erzeugt werden.

[Direkter Zugriff auf Fonts](#)

Utopia- Bold

```
\centering
\fontsize{16mm}{18mm}% Größe wählen
\usefont{T1}{put}{b}{n}% Font wählen
```

Utopia-Bold

Bsp.
7-6-1

Die PSNFSS-Sammlung enthält nur zwei Pakete, welche die Einstellungen für Formeln ändern: `mathptmx` wählt zu Times Roman passende Mathematik-fonts aus; es ist in Abschnitt 7.6.2 auf Seite 387 beschrieben. Und `mathpazo` stellt zu Palatino passende Mathematikfonts ein (siehe Abschnitt 7.6.3 auf Seite 389). Die Pakete `mathptm` und `mathple` sind ihre Vorläufer, die nur noch verteilt werden, um Kompatibilität bei älteren Dokumenten zu erreichen. Abgesehen von der PSNFSS-Sammlung gibt es noch ein paar andere Pakete, die Fonteinstellungen für Formeln ändern (meist in Zusammenhang mit kommerziellen Schriften). Einige der kostenlosen Pakete sind in Abschnitt 7.7 auf Seite 393 beschrieben, darunter auch eines, das Utopia für Text und Formeln verwendet. Abschnitt 8.8.3 enthält eine ganze Reihe von Beispielseiten mit verschiedenen Text- und Mathematikfonts.

Die meisten Dokumentenklassen, die für Computer Modern entworfen wurden, verwenden einen Zeilenvorschub (`\baselineskip`) von 10pt/12pt. Für viele PostScript-Fonts ist dieser Abstand zu gering, da sie über eine größere x-Höhe verfügen. Da dies jedoch eine Frage des Layouts ist, die auch durch die gewählte Zeilenlänge sowie weitere Faktoren beeinflusst wird, passen die Pakete der PSNFSS-Sammlung den Zeilenvorschub nicht selbständig an. Man kann ihn jedoch für die jeweilige Dokumentenklasse in der Präambel mit der Deklaration `\linespread{faktor}` um den angegebenen *faktor* ändern. So würde ein Wert von `\linespread{1.033}` den Zeilenvorschub beispielsweise von 12pt auf annähernd 12,4pt vergrößern. Um optimale Resultate zu erzielen, benötigt man allerdings eine Dokumentenklasse, die eigens für die ausgewählten Fonts entworfen wurde; in Ermangelung einer solchen Klasse kann man alternativ die Befehle `\normalsize`, `\footnotesize` usw. anpassen (nähere Angaben hierzu auf Seite 354). Außerdem ist zu beachten, dass ein geänderter Zeilenvorschub zu einer beträchtlichen Anzahl von „Underfull \vbox“-Warnungen führen kann, sobald die Höhe des Satzspiegels `\textheight` für die Anzahl an Zeilen keine ganze Zahl mehr ergibt (Näheres hierzu auf Seite 974).

[Den Zeilenvorschub anpassen](#)

Familie	Serie	Form(en)	Externe PostScript Fontnamen und Beispiele
Times (OT1, T1, TS1)			
ptm ptm	m b, (bx)	n, sl, it, sc n, sl, it, sc	Times-Roman (ptmr), Times-Italic (ptmri) Times-Bold (ptmb), Times-BoldItalic (ptmbi)
Palatino (OT1, T1, TS1)			
ppl ppl	m b, (bx)	n, sl, it, sc n, sl, it, sc	Palatino-Roman (pplr), Palatino-Italic (pplri) Palatino-Bold (pplb), Palatino-BoldItalic (pplbi)
New Century Schoolbook (OT1, T1, TS1)			
pnc pnc	m b, (bx)	n, sl, it, sc n, sl, it, sc	NewCenturySchlbk-Roman (pnocr), NewCenturySchlbk-Italic (pncri) NewCenturySchlbk-Bold (pnacb), NewCenturySchlbk-BoldItalic (pnacbi)
Bookman (OT1, T1, TS1)			
pbk pbk	m b, (bx)	n, sl, it, sc n, sl, it, sc	Bookman-Light (pbkl), Bookman-LightItalic (pbkli) Bookman-Demi (pbkd), Bookman-DemiItalic (pbkdi)
Helvetica (OT1, T1, TS1)			
phv phv phv phv	m b, (bx) mc bc	n, sl, sc n, sl, sc n, sl, sc n, sl, sc	Helvetica (phvr), Helvetica-Oblique (phvro) Helvetica-Bold (phvb), Helvetica-BoldOblique (phvbo) Helvetica-Narrow (phvrrn), Helvetica-Narrow-Oblique (phvron) Helvetica-Narrow-Bold (phvbrn), Helvetica-Narrow-BoldOblique (phvbon)
Avant Garde (OT1, T1, TS1)			
pag pag	m b, (bx)	n, sl, sc n, sl, sc	AvantGarde-Book (pagk), AvantGarde-BookOblique (pagko) AvantGarde-Demi (pagd), AvantGarde-DemiOblique (pagdo)
Courier (OT1, T1, TS1)			
pcr pcr	m b, (bx)	n, sl, sc n, sl, sc	Courier (pcrr), CourierOblique (pcro) Courier-Bold (pcrb), Courier-BoldOblique (pcrbo)
Zapf Chancery (OT1, T1, TS1)			
pzc	m	it	ZapfChancery-MediumItalic (pzcmi)
Utopia (OT1, T1, TS1)			
put put	m b, (bx)	n, sl, it, sc n, sl, it, sc	Utopia-Regular (putr), Utopia-Italic (putri) Utopia-Bold (putb), Utopia-BoldItalic (putbi)
Charter (OT1, T1, TS1)			
bch bch	m b, (bx)	n, sl, it, sc n, sl, it, sc	CharterBT-Roman (bchr), CharterBT-Italic (bchri) CharterBT-Bold (bchb), CharterBT-BoldItalic (bchbi)
Symbol und Zapf Dingbats (U)			
psy pzd	m m	n n	Symbol (psyr): Σψœβολ Zapf Dingbats (pzdr): ⌘Ⓜ☒♣▪■※⦿▼▲

Tabelle 7.8: Klassifizierung der Fontfamilien der PSNFSS-Distribution

L^AT_EX ist so voreingestellt, dass es eine Serifenschrift (Roman bzw. Antiqua) als Grundschrift auswählt. Pakete wie `helvet` oder `avant` ändern zwar (über `\sfdefault`) die voreingestellte serifenlose Schrift, nicht aber die Voreinstellung für die Grundschrift. Zur Änderung der Grundschrift muss man die Zeile

*Serifenlose Schrift als
Grundschrift*

```
\renewcommand\familydefault{\sfdefault}
```

in der Präambel einfügen.

Neben der Unterstützung für die gängigen PostScript-Textfonts enthält die PSNFSS-Sammlung das interessante Paket `pifont`. Es richtet eine Reihe von Befehlen für die so genannten Pi Fonts ein (d.h. spezielle Symbolfonts wie Zapf Dingbats und Symbol). Es wird in Abschnitt 7.6.4 auf Seite 390 beschrieben.

7.6.1 Schriftproben von PSNFSS-unterstützten Fonts

Dieser Abschnitt enthält Textbeispiele für die Fonts, die von der PSNFSS-Sammlung unterstützt werden. Für die Beispiele wurden mithilfe von `\fontsize` zunächst Schriftgrad und Zeilenvorschub explizit festgelegt, und danach wurde der jeweilige Font mit dem Befehl `\usefont` ausgewählt. Die erste Schriftprobe wurde z.B. mit der folgenden Anweisung erzeugt: `\fontsize{9}{13}\usefont{T1}{pag}{m}{n}`.

Avant Garde Gothic wurde von Herb Lubalin und Tom Carnase entworfen. Als Grundlage diente ihnen das unverwechselbare Logo des Magazins *Avant Garde*. Es handelt sich um einen geometrischen, serifenlosen Schrifttyp, dessen Grundformen aus Linien und Kreisen bestehen. Er eignet sich sehr gut für Überschriften und kurze Texte, benötigt jedoch einen großzügigen Durchschuss. Ed Benguiat entwarf eine (kommerzielle) schmalere Version, die auch bei längeren Texten gut zu lesen ist.

*ITC Avant Garde
Gothic
9pt/13pt (pag)*

For the price of £45, almost anything can be found floating
in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345–67890
JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often
naïve vis-à-vis the dæmonic phoenix's official rôle in fluffy soufflés?

Bookman wurde ursprünglich im Jahre 1860 von Alexander Phemister für die Zeichensatzhersteller Miller & Richard in Schottland entworfen (von Bitstream käuflich zu erwerben). Das ITC-Remake von Ed Benguiat verfügt über eine größere x-Höhe und relativ geringe Strichstärkenunterschiede und eignet sich sehr gut als Grundschrift und für Überschriften.

*ITC Bookman
10pt/12pt (pbk)*

For the price of £45, almost anything can be found
floating in fields. ¡THE DAZED BROWN FOX QUICKLY GA-
VE 12345–67890 JUMPS! — ¿But aren't Kafka's Schloß
and Æsop's Œuvres often naïve vis-à-vis the dæmonic
phoenix's official rôle in fluffy soufflés?

Bitstream Charter
10pt/12.4pt (*bch*)

Bitstream Charter stammt von Matthew Carter und wurde für Ausgabege-
räte mit geringer Auflösung entwickelt; daher enthält die Schrift rechteckige
Serifen und verzichtet auf den übermäßigen Gebrauch von Kurven und Schrä-
gen. Sie ist vielseitig verwendbar, unter anderem für Bücher und Benutzerdo-
kumentationen.

For the price of £45, almost anything can be found floating in
fields. ;THE DAZED BROWN FOX QUICKLY GAVE 12345–67890
JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often
naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy souf-
flés?

Courier
10pt/12pt (*pcr*)

Bei Courier handelt es sich um eine weite, dicktengleiche Schrift dünner
Strichstärke. Sie wurde von Howard Kettler von IBM entworfen und später von
Adrian Frutiger überarbeitet. Heutzutage wird sie häufig mit Times Roman
kombiniert, um auffällige Kontraste zu erzielen. Diese Kombination ist sicher-
lich unter anderem so beliebt, da sie mit jedem PostScript-Gerät ausgegeben
werden kann. Abschnitt 7.7.4 zeigt Alternativen hierzu.

For the price of £45, almost anything can be
found floating in fields. ;THE DAZED BROWN FOX
QUICKLY GAVE 12345–67890 JUMPS! -- ¿But aren't
Kafka's Schloß and Æsop's Œuvres often naïve
vis-à-vis the dæmonic phœnix's official rôle in
fluffy soufflés?

Helvetica
10pt/13pt (*phv*)

Helvetica wurde ursprünglich von Max Miedinger für den Schweizer
Schriftenhersteller Haas entworfen, daher der Name. Sie wurde später vom
Schriftenhersteller Stempel erweitert und von Mergenthaler Linotype in den
USA noch einmal verbessert. Es heißt, dass Helvetica die beliebteste Schrift
aller Zeiten sei.

For the price of £45, almost anything can be found floating
in fields. ;THE DAZED BROWN FOX QUICKLY GAVE 12345–
67890 JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œu-
vres often naïve vis-à-vis the dæmonic phœnix's official rôle in
fluffy soufflés?

*New Century
Schoolbook*
10pt/12.5pt (*pnc*)

Die Schrift New Century Schoolbook wurde Anfang des 20. Jahrhunderts
von Morris Benton für die American Type Founders gestaltet. Sie wurde im
Auftrag eines Verlages erstellt, der auf der Suche nach einer besonders gut
lesbaren Schrift für Grundschulbücher war.

For the price of £45, almost anything can be found floa-
ting in fields. ;THE DAZED BROWN FOX QUICKLY GA-
VE 12345–67890 JUMPS! — ¿But aren't Kafka's Schloß and
Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's of-
ficial rôle in fluffy soufflés?

Palatino von Hermann Zapf ist heute eine der am häufigsten verwendeten Schriften. Man spürt förmlich den Pinselstrich, mit dem sie geschaffen wurde, was ihr große Eleganz verleiht. Sie wurde zwar ursprünglich für Überschriften entwickelt, doch aufgrund ihrer Lesbarkeit gewann Palatino bald auch als Grundschrift an Beliebtheit.

Palatino
10pt/11.5pt (*ppl*)

For the price of £45, almost anything can be found floating in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345–67890 JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?

Times Roman ist die Linotype-Version der Monotype-Schrift Times New Roman, die ursprünglich unter Anleitung von Stanley Morison für die englische Zeitung *London Times* gestaltet wurde. Der Adobe Font, den viele PostScript-Geräte verwenden, bedient sich der Linotype-Variante für den Schriftgrad 12pt.

Times Roman
10pt/12pt (*ptm*)

For the price of £45, almost anything can be found floating in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345–67890 JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?

Utopia von Robert Slimbach vereint die Spannung in der Vertikalen und die deutlichen Kontraste in der Strichstärke, wie sie für Transitional-Schriften des 18. Jahrhunderts üblich waren, mit modernen Neuerungen in der Form und Linienführung.

Utopia
10pt/12.5pt (*put*)

For the price of £45, almost anything can be found floating in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345–67890 JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?

Zapf Chancery ist eine zeitgenössische Kursive, die jenen Kanzleischriften nachempfunden ist, die während der Renaissance in Italien von den päpstlichen Schreibern verwendet wurden. Da sie sehr gut zu lesen ist, eignet sie sich hervorragend für kurze Texte, sowie für Einladungen oder Urkunden.

ITC Zapf Chancery
10pt/12pt (*pzc*)

For the price of £45, almost anything can be found floating in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345–67890 JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?

7.6.2 mathptmx – Times Roman in Formeln und Text

Das Paket `mathptmx` legt Times als Grundschrift fest und richtet geeignete Mathematikfonts für die entsprechenden Dokumente ein. Es greift ausschließlich

auf kostenlose Type 1-Fonts zurück und ist daher in mancher Hinsicht kommerziellen Lösungen unterlegen, deren Schriften speziell für diesen Zweck gestaltet wurden. Dafür hat es den Vorteil, dass es (zumindest potentiell) auf jeder $\text{\TeX}$ -Installation verfügbar ist.¹

An dem Paket haben Alan Jeffrey, Sebastian Rahtz und Ulrik Vieth mitgewirkt. Es basiert auf früheren Arbeiten von Alan Jeffrey [76], im Einzelnen auf dem Paket `mathptm` (dem Vorläufer von `mathptmx`) sowie insbesondere auf dem `fontinst`-System [58, S. 393–404], das $\text{\TeX}$ erstmalig den generellen Zugriff auf PostScript-Fonts ermöglichte.

Technisch gesehen verwendet das `mathptmx`-Paket eine Sammlung virtueller Zeichensätze, welche die von $\text{\TeX}$ benötigten Mathematikfonts aus verschiedenen Quellen heranziehen, nämlich: Times Roman, Times Italic, Symbol, verschiedenen Computer Modern Fonts (hauptsächlich für Begrenzungssymbole, große Operatoren, Pfeile und Ähnliches), sowie aus Formal Script von Ralph Smith (RSFS). Die RSFS Fonts eignen sich besser für ein Schreibschrift- oder kalligraphisches Alphabet als Zapf Chancery, die in `mathptm` für diesen Zweck eingesetzt wird.

Ein Beispiel für eine trigonometrische Funktion:

$$\sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}}$$

Die Schreibschrift sieht so aus: $\mathcal{ABC}$.

`\usepackage{mathptmx}`

Ein Beispiel für eine trigonometrische Funktion:

`[ \sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}} ]`

Die Schreibschrift sieht so aus: $\mathcal{ABC}$.

Bsp.
7-6-2

Das Paket hat einiges mit dem Paket `mathpazo` gemeinsam. Zum einen setzt es geneigte griechische Großbuchstaben anstelle von aufrechten, wenn es mit der Option `slantedGreek` geladen wird. Dabei setzen die beiden Zusatzbefehle `\upDelta` und `\upOmega` jedoch in jedem Fall ein aufrechtes Δ bzw. Ω . Zum anderen stellt es automatisch die Funktionen des `exscale`-Paketes zur Verfügung: Dadurch werden große Operatoren und Begrenzungssymbole passend zur aktuellen Schriftgröße skaliert.

Leider deaktiviert das Paket andererseits `\boldmath`, und zwar ganz einfach, weil es keine fette Version des Adobe Symbol Fonts gibt. Man kann zwar mit `\mathbf` auf ein fett gedrucktes Mathematikalphabet zugreifen, aber eben nur auf aufrechte lateinische Buchstaben und Ziffern. Ein eher fragwürdiges Ergebnis erzielt man auch, wenn man mithilfe des Paketes `bm` einzelne Symbole fetter gestaltet, da der Befehl `\bm` bestenfalls eine „Arme-Leute-Version“ einer fetten Schrift erzeugt, indem er die Symbole mit leichtem Versatz übereinander druckt.

Passende fette
Schriften fehlen



Fettdruck ist nicht einfach: $\alpha \neq A$ und sieht bestenfalls fragwürdig aus: $A \neq A = \alpha - \gamma$.

`\usepackage{mathptmx, bm}`

Fettdruck ist nicht einfach: `\boldmath $\alpha \neq A$` und sieht bestenfalls fragwürdig aus: `$A \neq \mathbf{A} = \bm{\alpha} - \bm{\gamma}`.

Bsp.
7-6-3

Hinzu kommt das (kleine) potentielle Problem, dass die Befehle `\jmath`,

¹Die $\text{\TeX}$ -Installation muss dazu virtuelle Fonts unterstützen, was auf nahezu alle Distributionen zutrifft.

`\coprod` und `\amalg` nicht zur Verfügung stehen. Wenn einer dieser Punkte sich zu einem echten Problem entwickelt, sollte man als Alternative die TX Fonts (Abschnitt 7.7.5) und die kommerziellen Lösungen MathTime (Professional) von Michael Spivak bzw. TM-Math von MicroPress in Betracht ziehen.

7.6.3 mathpazo – Palatino in Formeln und Text

Für Adobe Palatino in Kombination mit passenden Mathematikfonts gab es ursprünglich das Paket `mathpple` von Walter Schmidt, das auf früheren Arbeiten von Aloysius Helminck basierte. Es beruhte auf dem gleichen Ansatz wie `mathptm`, d.h. es arbeitete ebenfalls mit virtuellen Fonts, die in diesem Fall Symbole aus den Schriften Palatino, Symbol, Euler und CM Math kombinierten. Da diese Schriften nur teilweise zu Palatino passten, entwickelte Diego Puga einen Satz von Type 1-Fonts (Pazo Math), um die Fehler in der `mathpple`-Lösung zu beheben. Die Pazo Math Fonts enthalten Zeichen, die es in Palatino nicht gibt, und die, wenn sie ersatzweise aus Computer Modern oder aus dem Symbol Font herangezogen werden, kombiniert mit Palatino seltsam aussehen. Darunter befinden sich eine Reihe von mathematischen Zeichen, das griechische Alphabet in Großbuchstaben (aufrecht und geneigt), ein Blackboard Bold-Alphabet (für Mengenzeichen) sowie einige weitere Glyphen (wie z.B. das Euro-Zeichen) in normalen und fetten Stärken und in aufrechter und geneigter Lage.

Auf diese Fonts kann man mit dem `mathpazo`-Paket zugreifen, das von Diego Puga und Walter Schmidt für die PSNFSS-Sammlung entwickelt wurde. Das Paket legt Palatino als Grundschrift fest und stellt für Formeln virtuelle Fonts zur Verfügung, die auf Palatino Italic, die Math Pazo Fonts und (für die verbleibenden Symbole) auf die CM Fonts zugreifen.

Ein Beispiel für eine trigonometrische Funktion:

$$\sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}}$$

Bsp.
7-6-4

Die Schreibschrift sieht so aus: $\mathcal{ABC}$.

`\usepackage{mathpazo}`

Ein Beispiel für eine trigonometrische Funktion:
`\[ \sin \frac{\alpha}{2} =`
`\pm \sqrt{\frac{1-\cos\alpha}{2}} \]`

Die Schreibschrift sieht so aus: `\mathcal{ABC}`.

Dieses Paket ist dem `mathptmx`-Paket sehr ähnlich. Es unterstützt insbesondere die Option `slantedGreek` um griechische Großbuchstaben geneigt zu setzen anstatt aufrecht (Voreinstellung). Dabei setzen die beiden Zusatzbefehle `\upDelta` und `\upOmega` jedoch in jedem Fall ein aufrechtes Δ bzw. Ω . Außerdem stellt es die Funktionen des `exscale`-Paketes zur Verfügung.

Anders als das `mathptmx`-Paket, das mit dem Adobe Symbol Font arbeitet, für den es keine fett gedruckte Variante gibt, bietet das `mathpazo`-Paket jedoch vollen Zugriff auf Symbole in fetter Schriftstärke.

Bsp.
7-6-5

Fettdruck ist einfach: $\alpha \neq A$ und
 passt gut: $A \neq \mathbf{A} = \alpha - \gamma$.

`\usepackage{mathpazo,bm}`

Fettdruck ist einfach: `\{\boldmath$\alpha`
`\neq A$}` und passt gut:
`$A \neq \mathbf{A} = \bm{\alpha} - \bm{\gamma}.`

Wie bereits zuvor erwähnt, enthält der Math Pazo Font ein Blackboard Bold-Alphabet (für Mengenzeichen), das sich mithilfe des Alphabetbefehls `\mathbb` nutzen lässt. Dieser Zeichensatz besteht aus den lateinischen Großbuchstaben und der Ziffer „1“. Aber Achtung: Alle anderen Ziffern werden stillschweigend ignoriert!

ABCDEF GHIJK 0123 `\usepackage{mathpazo}`
 `$\mathbb{ABCDEFGHIJK}$ $\mathbb{0123}$`

Bsp.
7-6-6

Wenn `\mathbb` ein anderes Alphabet von einem anderen Paket nutzen soll, unterdrückt man am Besten den entsprechenden Pazo Math Zeichensatz direkt beim Laden des Paketes durch die Option `noBBppl`.

Kommerzielle Palatino
Fonts

Das Paket verfügt außerdem über zwei zusätzliche Möglichkeiten, um kommerzielle Palatino Fonts¹ als Grundschrift einzurichten: Die Option `sc` wählt anstelle der einfachen Palatino eine Palatino in echten Kapitälchen aus (Schriftfamilie `pp1x`) und die Option `osf` stellt auf eine Palatino in Kapitälchen mit Mediävalziffern (Schriftfamilie `pp1j`) um.

7.6.4 pifont – Zugriff auf Pi- und Symbolfonts

Zeichensätze mit einer Sammlung von Sonderzeichen, die in Textfonts normalerweise nicht vorkommen, werden Pi-Fonts genannt. Ein solcher Zeichensatz, der PostScript-Font Zapf Dingbats, kann mithilfe des `pifont`-Paketes genutzt werden, das ursprünglich von Sebastian Rahtz geschrieben wurde und nun integrierter Bestandteil des PSNFSS ist.

Zugriff auf Zapf
Dingbats-Glyphen

Tabelle 7.9 auf der gegenüberliegenden Seite zeigt die Zeichen der PostScript Zapf Dingbats, auf die man direkt zugreifen kann. Mit dem `\ding`-Befehl kann man ein bestimmtes Zeichen auswählen. Das Argument des Befehls ist dabei eine ganze Zahl in dezimaler, oktaler oder hexadezimaler Notation, die das Zeichen entsprechend der Tabelle kennzeichnet. So ergibt `\ding{'46}` beispielsweise ☺.

Die Umgebung `dinglist` ist eine Abwandlung der `itemize`-Liste. Ihr Argument ist die Nummer des Zeichens, das vor jedem Punkt der Auflistung stehen soll.

- Erster Punkt. `\usepackage{pifont}`
- Zweiter Punkt der Liste. `\begin{dinglist}{'344}`
- Ein letzter Punkt. `\item Erster Punkt. \item Zweiter Punkt`
`der Liste. \item Ein letzter Punkt.`
`\end{dinglist}`

Bsp.
7-6-7

Mit der Umgebung `dingautolist` kann man aus einer Folge von Zapf Dingbats-Zeichen eine nummerierte Liste erstellen. Hier enthält das Argument die Nummer des ersten Zeichens der Folge. Diese Zahl wird für die weiteren Punkte der Aufzählung jeweils um eins erhöht. Dadurch werden Startpositionen wie '254, '266, '300 und '312 (in Oktalzahlen) der Tabelle 7.9 be-

¹Diese Fonts sind käuflich zu erwerben und gehören *nicht* zu den 35 Basisfonts.

	‘0	‘1	‘2	‘3	‘4	‘5	‘6	‘7	
‘04x									“2x
‘05x									
‘06x									“3x
‘07x									
‘10x									“4x
‘11x									
‘12x									“5x
‘13x									
‘14x									“6x
‘15x									
‘16x									“7x
‘17x				‘	‘	“	“		
‘24x									“Ax
‘25x					①	②	③	④	
‘26x	⑤	⑥	⑦	⑧	⑨	⑩	⑪	⑫	“Bx
‘27x	⑬	⑭	⑮	⑯	⑰	⑱	⑲	⑳	
‘30x	①	②	③	④	⑤	⑥	⑦	⑧	“Cx
‘31x	⑨	⑩	⑪	⑫	⑬	⑭	⑮	⑯	
‘32x	⑰	⑱	⑲	⑳	→	→	↔	↕	“Dx
‘33x									
‘34x									“Ex
‘35x									
‘36x									“Fx
‘37x									
	“8	“9	“A	“B	“C	“D	“E	“F	

Tabelle 7.9: Glyphen des PostScript-Fonts Zapf Dingbats

sonders interessant, da an diesen Positionen verschieden gestaltete, in Kreise gesetzte Zahlenreihen (1-10) beginnen.

① Der erste Punkt der Liste.

① Ein Unterpunkt.

② Ein weiterer Unterpunkt.

② Der zweite Punkt verweist auf ①.

Bsp.
7-6-8

Verweise auf Listenpunkte zeigen keine Verschachtelung: ① aber ①.

```
\usepackage{pifont}
```

```
\begin{dingautolist}{‘300}
```

```
\item Der erste Punkt der Liste.\label{lst:a}
```

```
\begin{dingautolist}{‘312}
```

```
\item Ein Unterpunkt.\label{lst:b}
```

```
\item Ein weiterer Unterpunkt.
```

```
\end{dingautolist}
```

```
\item Der zweite Punkt verweist auf \ref{lst:b}.
```

```
\end{dingautolist}
```

Verweise auf Listenpunkte zeigen keine

Verschachtelung: \ref{lst:a} aber \ref{lst:b}.

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!	∀	#	∃	%	&	ə	"2x
'05x	(	)	*	+	,	–	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	<	=	>	?	
'10x	≅	A	B	X	Δ	E	Φ	Γ	"4x
'11x	H	I	∅	K	Λ	M	N	O	
'12x	Π	Θ	P	Σ	T	Υ	ς	Ω	"5x
'13x	Ξ	Ψ	Z	[	∴	]	⊥	–	
'14x	—	α	β	χ	δ	ε	φ	γ	"6x
'15x	η	ι	φ	κ	λ	∞	ν	ο	
'16x	π	θ	ρ	σ	τ	υ	ϖ	ω	"7x
'17x	ξ	ψ	ζ	{		}	~		
'24x		Υ	'	≤	√	∞	⊗	♣	"Ax
'25x	♦	♥	♠	↔	←	↑	→	↓	
'26x	Υ	±	"	≥	×	∞	∂	∞	"Bx
'27x	√	≠	≡	≈	⊃		—	⌋	
'30x	⌘	∩	ℝ	∅	⊗	⊕	∅	∩	"Cx
'31x	∪	⊃	⊇	⊄	⊂	⊆	∈	∉	
'32x	∠	∇	®	©	™	Π	√	.	"Dx
'33x	ℝ	∧	∨	↔	⇐	↑↑	⇒	↓↓	
'34x	◊	⟨	®	©	™	Σ	∫		"Ex
'35x	⌊	⌈		⌊	⌈	{	⌊		
'36x		⟩	⌋	⌋		⌋	⌋		"Fx
'37x	⌋	⌋		⌋	⌋	{	⌋		
	"8	"9	"A	"B	"C	"D	"E	"F	

Tabelle 7.10: Glyphen des PostScript-Fonts Symbol

Mit dem Befehl `\dingline` kann man sogar eine ganze Zeile (mit einem Einzug von 0.5 Zoll auf der linken und rechten Seite) mit einem beliebigen Zeichen füllen, wobei das Argument wiederum das gewünschte Zeichen angibt. Um nur Teile einer Zeile zu füllen gibt es den Befehle `\dingfill`. Er verhält sich ähnlich wie der $\text{\LaTeX}$ -Befehl `\dotfill`, nur dass er anstelle von Punkten das angegebene Zeichen einsetzt.

```
\usepackage{pifont}
```

```
\dingline{35} \par\medskip
\noindent\dingfill{233} text text
\dingfill{235} text text \dingfill{236}
```

Bsp.
7-6-9

Abgesehen von der direkten Unterstützung für Zapf Dingbats verfügt das `pifont`-Paket über einen generellen Mechanismus, mit dem sich jeder Pi-Font verarbeiten lässt, welcher der NFSS-Klassifizierung $U/familie/m/n$ entspricht – wie z.B. der Symbol-Font mit dem Schriftfamilien-Namen `psy`.

Der Zugriff auf einzelne Zeichen eines solchen Pi-Fonts erfolgt über den Befehl `\Pisymbol`, der als erstes Argument den Namen der *familie* und als zweites die Position der Glyphe im Font erhält. Mithilfe dieses Befehls hat man direkten Zugriff auf die Zeichen des Symbolfonts aus Tabelle 7.10 auf der vorherigen Seite. Der Befehl `\Pisymbol{psy}{210}` ergibt z.B. ®. Eigentlich ist auch der bereits besprochene Befehl `\ding` einfach nur eine Abkürzung für `\Pisymbol` mit `psd` als erstem Argument.

*Zugriff auf einzelne
Glyphen in einem
Pi-Font*

Wer nur griechische Buchstaben benötigt, kann mit dem Befehl `\Pifont` auf die entsprechenden Zeichen aus Tabelle 7.10 zugreifen. Das ist natürlich kein Ersatz für eine ordentlich gestaltete griechische Schrift, wohl aber eine Lösung für Notfälle – z.B. um den Text über dem Eingang zu Platons Akademie zu setzen, der bedeutet: „Zutritt nur für Geometer“:

Bsp.
7-6-10

```
\usepackage{pifont}
\MHAEIS AΓΕΩΜΕΤΡΗΤΟΣ ΕΙΣΙΤΩ. {\Pifont{psy} MHDEIS\ AGEWMETRHTOS\ EISITW}.
```

Außerdem lassen sich mithilfe der Umgebung `Pilist` einfache Aufzählungen oder mit `Piautolist` nummerierte Listen erstellen:

Bsp.
7-6-11

```
\usepackage{pifont}
⇒ Erster Punkt. \begin{Pilist}{psy}{'336}
⇒ Der Zweite. \item Erster Punkt. \item Der Zweite.
\end{Pilist}
★ Erster Punkt. \begin{Piautolist}{psd}{'115}
★ Der Zweite. \item Erster Punkt. \item Der Zweite.
\end{Piautolist}
```

Auch die Befehle `\dingline` und `\dingfill` sind nur Abkürzungen der allgemeineren Befehle `\Piline` und `\Pifill`, wie im nächsten Beispiel zu sehen. Die letzte Zeile des Beispiels weist seltsame Lücken auf. Sie rühren daher, dass `\Piline` und `\Pifill` ihre Symbole in einem unsichtbaren Gitter anordnen, so dass Symbole in verschiedenen Zeilen bündig untereinander erscheinen.

Bsp.
7-6-12

```
\usepackage{pifont}
⇒ ⇒ ⇒ text ⇔ ⇔ text ⇐ ⇐ ⇐
\Piline{psd}{36} \par\medskip
\noindent\Pifill{psy}{222} text
\Pifill{psy}{219}text\Pifill{psy}{220}
```

7.7 Eine Sammlung von Fontpaketen

Bisher wurden nur Fontpakete besprochen, die fest zu L^AT_EX gehören, also Pakete, die entweder direkt mit verteilt werden, oder, wie PSNFSS, Teil der „erforderlichen“ Erweiterungen sind. Es gibt natürlich noch eine Vielzahl von Paketen zu diesem Thema. Heutzutage sind sie fast alle Bestandteil einer L^AT_EX-Distribution. Wenn sie auf dem lokalen System fehlen, kann man sie von einem Webarchiv oder einer T_EX-Organisation erhalten (siehe Anhang C).

Familie	Serie	Form(en)	Schriftbild
<i>Concrete Roman (T1, TS1, OT1)</i>			
ccr	m	n, it, sl, sc	Concrete Roman normal
ccr	c	sl	Concrete Roman schmal geneigt (nur OT1 und 9pt)
<i>Concrete Math (OML)</i>			
ccm	m	it	Concrete Math. α Ω
<i>Concrete Math (OMS)</i>			
ccy	m, c	n	$\mathcal{Q} \setminus \nabla \sqcup \mathcal{M} \dashv \neg \otimes \otimes$

Tabelle 7.11: Klassifizierung der Concrete Fontfamilien

Die Pakete in diesem Abschnitt ändern die Fonts für die Grundschrift (und manchmal auch diejenigen für Formeln). Wie die Überschrift schon andeutet, handelt es sich lediglich um eine Auswahl des verfügbaren Angebotes. Weitere Verweise finden sich in den Online-Paketkatalogen [48, 171] oder in einer der FAQs über L^AT_EX [47, 143].

7.7.1 eco – Mediävalziffern in Computer Modern

Die Mathematikfonts der Original Computer Modern Fonts enthalten einen Satz von Mediävalziffern (z.B. 1982) – nicht etwa, weil es sich um mathematische Zeichen handelt, sondern weil Donald Knuth versuchte, den begrenzten Platz in den Zeichensätzen so gut wie möglich auszunutzen, so dass er die entsprechenden Glyphen an freien Stellen in den Mathematikfonts unterbrachte. Da es bei den EC Fonts nur um die Überarbeitung der Fontkodierung für Texte ging, wurde diese Unregelmäßigkeit in den Mathematikfonts leider übernommen.¹ Die Gestalter der Textbegleitkodierung (TS1) nahmen zwar auch Mediävalziffern in den Zeichenvorrat auf, jedoch bisher ohne größeren praktischen Nutzen, da zu viele Schriftfamilien nur eine Untermenge der TS1-Kodierung umsetzen (vgl. Abschnitt 7.5.4, Seite 378 für nähere Informationen).

In Mathematikfonts verborgene Mediävalziffern lassen sich auf einfache Weise über den L^AT_EX-Befehl `\oldstylenums` abrufen, der sowohl im Text als auch in Formeln verwendet werden kann. Das Argument des Befehls enthält die Zahl, die in Mediävalziffern gesetzt werden soll. Verwendet man den Befehl im Text, so werden auch Leerzeichen im Argument berücksichtigt; es sollte jedoch keine weiteren Zeichen außer Ziffern enthalten, wenn man überraschende Ergebnisse vermeiden will. Ein Problem mit der voreingestellten Definition dieses Befehls ist, dass er immer Mediävalziffern in Computer Modern Roman erzeugt, egal welche Fonts man sonst gerade verwendet. Darum enthält das `textcomp`-Paket eine abgeänderte Variante, welche die Mediävalziffern in der aktuellen Schrift abrufen, sofern sie in der entsprechenden Schriftfamilie vorhanden sind (nähere Details in Abschnitt 7.5.4).

¹Justin Ziegler entwickelte zusammen mit dem L^AT_EX3-Projektteam eine rationellere Fontkodierung für Mathematikfonts mit 256 Zeichen [175]. Seine theoretischen Ansätze wurden jedoch leider bisher nur in Form eines Prototyps mit virtuellen Fonts umgesetzt [40].

Dieser Ansatz mag ausreichen, wenn man normalerweise mit Versalziffern arbeitet und nur hin und wieder Mediävalziffern benötigt. Wenn in einem Dokument jedoch alle Zahlen im Text als Mediävalziffern gesetzt werden sollen, kann man dem Autor nicht zumuten, jede einzelne Zahl explizit über einen Befehl zu formatieren. In diesem Falle benötigt man Textfonts, die an den Standardpositionen Mediäval- anstelle von Versalziffern enthalten.

Die EC Fonts enthalten beide Ziffernarten (wenn auch nicht ganz an den gewohnten Stellen), so dass es nur eine Frage der Zeit war, dass jemand eine Reihe virtueller Fonts entwickelte, welche als Standardziffern im Text Mediävalziffern einsetzen. Die eco Fonts von Sebastian Kirsch sind entsprechend umkodiert und über das Paket eco verfügbar. Dieses Paket wirkt sich jedoch nur auf Zahlen im Text aus, so dass man mathematische Ziffern unbedingt weiterhin mit den richtigen Befehlen versehen muss. Andernfalls ähnelt das Ergebnis dem nächsten Beispiel.

1996 entwickelte Sebastian Fonts, die im Text Mediäval- und in Formeln Versalziffern verwenden. Man schreibe daher niemals „der Wert kann 1 oder -1 sein“, da beide Zahlen Versalziffern sein sollten. Auch im Text sind Versalziffern möglich: 1996.

Bsp.
7-7-1

`\usepackage{eco}`

1996 entwickelte Sebastian Fonts, die im Text Mediäval- und in Formeln Versalziffern verwenden. Man schreibe daher niemals "‘der Wert kann 1 oder \$-1\$ sein'", da beide Zahlen Versalziffern sein sollten. Auch im Text sind Versalziffern möglich: `\newstylenums{1996}`.

7.7.2 ccfonts, concmath – Die Concrete Fonts

Für den Text seines Buches *Concrete Mathematics* [61] gestaltete Donald Knuth eine neue Schrift [96], die mit den Euler Mathematikfonts von Hermann Zapf [174] harmonisieren sollte. Diese Schriftfamilie namens Concrete Roman wurde mit geänderten Parametereinstellungen aus den METAFONT-Quellen für die Computer Modern-Schriften generiert.

Ausgehend von den Arbeiten an den EC Fonts war es relativ einfach Concrete Roman-Zeichensätze in den Kodierungen T1 und TS1 zu erstellen (ursprünglich von Frank Mittelbach, stammt die aktuelle Version von Walter Schmidt). Die Zeichensätze dieser Familie sind in Tabelle 7.11 auf der gegenüberliegenden Seite aufgeführt. Nach der von Knuth [96] skizzierten Konstruktionsmethode entwickelte Ulrik Vieth einen Satz von begleitenden Concrete Mathematikfonts, die auch den gesamten Bereich der AMS-Symbole abdecken (wie sie durch die Pakete amssymb und amsfonts zur Verfügung gestellt werden).

Das erste Paket, das den Zugriff auf diese Schriftfamilie ermöglichte, war beton (von Frank Jensen). Das folgende Beispiel zeigt die Kombination von einem Text in Concrete mit den Euler Mathematikfonts (siehe auch Abschnitt 7.7.10 auf Seite 407):

Concrete Roman passt gut zu Euler Math, wie man hier sehen kann:

Bsp.
7-7-2

$$\sum_{0 \leq k < n} k = \frac{n(n-1)}{2}$$

`\usepackage{beton,euler}`

Concrete Roman passt gut zu Euler Math, wie man hier sehen kann:

`\[ \sum_{0 \leq k < n} k = \frac{n(n-1)}{2} \]`

Eine neuere Entwicklung, die ebenfalls Concrete Fonts in Formeln erlaubt und die T1- und TS1-Kodierungen unterstützt, ist das Paket `ccfonts` (von Walter Schmidt). Beide Pakete achten auf kleine aber wesentliche typographische Details, wie etwa eine leichte Erhöhung des Wertes von `\baselineskip` (vgl. die Erläuterungen weiter unten). Da die Concrete-Schriften keine fettgedruckte Serie enthalten, bietet das Paket `ccfonts` die Option `boldsans`, mit deren Hilfe man die halbfette Schriftserie der Computer Modern Sans-Schriften als Ersatz verwenden kann. Ohne weitere Anpassungen werden durch diese Option die Überschriften in den Standardklassen in der entsprechenden Schriftserie gesetzt.

1 Testen von Überschriften

Ein Beispiel für eine trigonometrische Funktion:

$$\sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}}$$

Die Schreibschrift sieht so aus: *ABC*.

Aus `textcomp`: \$ € * ω † ...

```
\usepackage[boldsans]{ccfonts}
```

```
\usepackage[full]{textcomp}
```

```
\section{Testen von Überschriften}
```

Ein Beispiel für eine trigonometrische Funktion:

```
\[ \sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}} \]
```

Die Schreibschrift sieht so aus:

```
$\mathcal{ABC}$.\\
```

Aus `textcomp`: `\textdollaroldstyle` `\texteuro`

```
\textborn\ \textmarried\ \textdied\ \ldots
```

Bsp.
7-7-3

Da die Schriftstärke der Concrete Schriften wesentlich größer ist als z.B. die der Computer Modern, empfiehlt es sich, sie mit einem größeren Durchschuss zu setzen, als er in den meisten Dokumentenklassen vorgesehen ist. Darum erhöht das Paket den Zeilenvorschub automatisch auf 10/13 und ähnliche Verhältnisse für andere Grundschriftgrößen. Wenn diese Anpassung aus irgendeinem Grunde nicht erwünscht ist, lässt sie sich mit der Option `standard-baselineskips` aufheben.

Die Funktionalität des `exscale`-Paketes wird mit der Paketoption `exscale` zur Verfügung gestellt; Einzelheiten hierzu finden sich in Abschnitt 7.5.5 auf Seite 379. Das `exscale`-Paket selbst kann nicht genutzt werden, da es nur mit den Computer Modern Mathematikfonts funktioniert.

Wird das Paket `amssymb` oder `amsfonts` geladen, dann stellt das `ccfonts`-Paket automatisch auf die Concrete-Varianten der AMS-Symbolfonts um.

Und schließlich bietet das Paket mit der Option `slantedGreek` die Möglichkeit, griechische Großbuchstaben geneigt zu setzen anstatt aufrecht (Voreinstellung). Dabei setzen die beiden Zusatzbefehle `\upDelta` und `\upOmega` jedoch in jedem Fall ein aufrechtes Δ bzw. Ω .

7.7.3 cmbright – Die Computer Modern Bright Fonts

Eine weitere Schriftfamilie, die nach den METAFONT-Quellen der CM Fonts gestaltet wurde, bilden die Computer Modern Bright (CM Bright) Fonts von Walter Schmidt, die in Tabelle 7.12 auf der nächsten Seite aufgeführt sind. Diese Familie serifenloser Schriften wurde als gut lesbare Grundschrift gestaltet. Sie enthält passende Schreibmaschinen- und Mathematikfonts, einschließlich der AMS-Symbole.

Familie	Serie	Form(en)	Schriftbild
CM Bright (OT1, T1, TS1)			
cmbr	m	n, sl	CM Bright Medium
cmbr	sb	n, sl	<i>CM Bright halbfett geneigt</i>
cmbr	bx	n	CM Bright breitfett
CM Typewriter Light (OT1, T1, TS1)			
cmtl	m	n, sl	Typewriter Light normal
CM Bright Math (OML)			
cmbrm	m, b	it	<i>Bright Math. α Ω</i>
CM Bright Math (OMS)			
cmbrs	m	n	$\mathcal{B}\nabla\}$ $\langle\sqcup\mathcal{M}\dashv\sqcup\langle\neg\otimes\otimes$

Tabelle 7.12: Klassifizierung der Computer Modern Bright Fontfamilien

Durch Laden des cmbright-Paketes in der Präambel werden diese Familien im gesamten Dokument verwendet. Für eine korrekte Silbentrennung in nicht englischen Sprachen empfiehlt es sich, dieses Paket, wie im nächsten Beispiel gezeigt, mit fontenc zu kombinieren. Alle CM Bright Zeichensätze unterstützen vollständig die T1- und die TS1-Kodierung.

1 Ein CM Bright-Dokument

Die Familie CM Bright enthält Schreibmaschi-nenschriften und passende Fonts für Formeln, z.B.:

$$\sum_{0 \leq k < n} k = \frac{n(n-1)}{2}$$

```
\usepackage[T1]{fontenc}
\usepackage{cmbright}
\section{Ein CM Bright-Dokument}
Die Familie CM Bright enthält
\texttt{Schreib\~maschi\~nenschriften} und
passende Fonts für Formeln, z.B.:
\[ \sum_{0 \leq k < n} k = \frac{n(n-1)}{2} \]
```

Bsp.
7-7-4

Zur Anpassung an die serifenlosen Fonts verwendet das Paket per Voreinstellung einen etwas größeren Durchschuss als die Standardklassen. Diese Einstellung lässt sich durch die Paketoption standard-baselineskips wieder aufheben. Auch in anderer Hinsicht funktioniert das Paket ähnlich wie andere Arbeiten von Walter Schmidt: Die Option slantedGreek erzeugt geneigte griechische Großbuchstaben, wobei \upDelta und \upOmega ein aufrechtes Δ- bzw. Ω-Zeichen setzen. Wird das Paket amssymb oder amsfonts geladen, dann stellt das cmbright-Paket automatisch auf die CM Bright-Varianten der AMS-Symbolfonts um.

Die METAFONT-Umsetzungen der Fonts können kostenlos von den CTAN-Archiven heruntergeladen werden; Versionen im Type1-Format kann man bei MicroPress erwerben. Kürzlich hat Harald Harders eine kostenlose Type1-Version (allerdings ohne manuelles Hinting) unter dem Namen hfbright veröffentlicht. Wie in Abschnitt 7.5.1 erwähnt, decken die frei erhältlichen CM-Super Type1-Fonts in Teilen auch die CM Bright Fonts ab.

Kodierung	Familie	Serie	Form(en)	PostScript Fontnamen und Beispiele
T1, TS1	ul9	m	n, sl	<i>LuxiMono</i> , <i>LuxiMono-Oblique</i>
T1, TS1	ul9	b	n, sl	LuxiMono-Bold , <i>LuxiMono-BoldOblique</i>

Tabelle 7.13: Klassifizierung der LuxiMono Fontfamilien

7.7.4 luximono – Ein vielseitiger Schreibmaschinenfont

Die Auswahl an dicktengleichen Schriften (auch Schreibmaschinen- oder Typewriter-Schriften genannt) für Programm-Listings und andere Anwendungsbereiche ist nicht sehr groß. Zwar verfügen die Computer Modern Schriften über eine geeignete Typewriter-Familie (cmtt), wenn man die Grundschrift jedoch ersetzt, stehen nicht mehr allzu viele Schreibmaschinenfonts zur Auswahl. Adobe Courier hat eine große Laufweite und ist schon allein aus diesem Grund häufig keine gute Wahl. Eine andere Möglichkeit kann darin bestehen, cmtt beizubehalten, doch diese Schrift passt vielleicht nicht gut zu der neuen Grundschrift des Dokumentes.

Vor kurzem wurde mit der Version 4.2 von XFree86, der freien Variante des X-Window-Systems, eine neue, frei verteilbare, dicktengleiche Schriftfamilie namens LuxiMono veröffentlicht. Dieser Type1 kodierte PostScript Font enthält fette, geneigte und geneigtefette Schriftschnitte (vgl. Tabelle 7.13). In dieser Hinsicht unterscheidet er sich von anderen dicktengleichen Fonts, die häufig nur als normale Schriftserie, und seltener in kursiven oder geneigten Lagen angeboten werden.

Diese Fonts wurden von Kris Holmes und Charles Bigelow (Bigelow and Holmes, Inc.) gestaltet und von der URW++ Design and Development GmbH mit Hinting- und Kerning-Tabellen versehen. Sie lassen sich mithilfe des luximono-Paketes von Walter Schmidt in L^AT_EX integrieren.

Das nächste Beispiel vergleicht LuxiMono (mit der Option scaled auf 85% verkleinert), Computer Modern Typewriter und Adobe Courier miteinander. Dabei hat LuxiMono immer noch die größte x-Höhe (\fontdimen5), und gleichzeitig die geringste Breite. Courier liegt mit ihrer großen Laufweite am anderen Ende des Spektrums und CM Typewriter befindet sich locker irgendwo zwischen den beiden Extremen.

The dazed brown fox quickly gave
12345-67890 jumps! x-height=4.50502pt
(LuxiMono)

The dazed brown fox quickly gave
12345-67890 jumps! x-height=4.3045pt
(CM Typewriter)

The dazed brown fox quickly gave
12345-67890 jumps!
x-height=4.25989pt (Adobe
Courier)

```
\usepackage[T1]{fontenc}
\usepackage[scaled=0.85]{luximono}
\newcommand\allletters{The dazed brown
fox quickly gave 12345--67890 jumps!
x-height=\the\fontdimen5\font\ }

\raggedright
\texttt{\allletters (LuxiMono)}
\par \renewcommand\ttdefault{cmtt}
\texttt{\allletters (CM Typewriter)}
\par \renewcommand\ttdefault{pcr}
\texttt{\allletters (Adobe Courier)}
```

Bsp.
7-7-5

Wenn die Option scaled keinen Wert erhält, werden die Fonts auf 87% skaliert, so dass ihre Laufweite ungefähr der von Computer Modern Typewri-

ter entspricht. Um exakt die gleiche Laufweite zu erhalten, sollte für 10pt-Fonts 0.87478 verwendet werden, während für 11pt-Dokumente 0.86124 der richtige Wert wäre. Das liegt daran, dass LuxiMono linear skaliert wird, während die Computer Modern Fonts eigens in verschiedenen Größen gestaltet sind. Ohne Skalierung verfügt LuxiMono über die gleiche Laufweite wie Adobe Courier.

Bsp.
7-7-6

Dieser Font enthält ein €-Symbol.	<code>\usepackage[T1]{fontenc}\usepackage[scaled]{luximono}</code>
Dieser Font enthält ein €-Symbol.	<code>\usepackage[euro]{textcomp} \raggedright</code>
	<code>\texttt{Dieser Font enthält ein \texteuro{}-Symbol.}</code>
	<code>\par \renewcommand\ttdefault{cmtt}</code>
	<code>\texttt{Dieser Font enthält ein \texteuro{}-Symbol.}</code>

Man beachte, dass die LuxiMono-Fonts nur in der T1-Kodierung unterstützt werden (siehe die Verwendung von `fontenc` in den Beispielen). Sie verfügen über den Zeichenvorrat der `textcomp`-Symbole, die typischerweise zu PostScript-Fonts gehören – also diejenigen, die beim Laden von `textcomp` mit der Option `safe` deklariert werden. Da jedoch auch das Euro-Zeichen vorhanden ist, lädt man das Paket am besten mit der Option `euro`.¹

7.7.5 txfonts – Alternative Unterstützung für Times Roman

Durch das Paket `mathptmx` unterstützt die PSNFSS-Sammlung Times Roman sowohl als Grundschrift für Text als auch für Formeln, wobei für mathematische Zeichen hauptsächlich die Schriften Times Italic und Adobe Symbol verwendet werden (siehe Abschnitt 7.6.2). Im Jahr 2000 veröffentlichte Young Ryu seinen eigenen Satz virtueller Fonts, zusammen mit begleitenden Type 1-Fonts, für die Darstellung von Formeln in Dokumenten, die Times Roman als Grundschrift verwenden.

Die zusätzlichen Zeichensätze enthalten Glyphen, die normalerweise in PostScript-Fonts fehlen – z.B. alle textcomp-Symbole, alle Symbole der AMS-Fonts (siehe Kapitel 8) sowie weitere Zeichen. Dadurch sind diese Fonts vollständiger als ihre Entsprechungen im L^AT_EX Standard-PSNFSS-Paket.

Die Fonts stehen nach Laden des Paketes `txfonts` in der Präambel zur Verfügung. Wenn das Paket geladen wird, richtet es Times Roman als Grund-
schrift und Adobe Helvetica (skaliert auf 95%) als serifenlose Schrift ein. Als
Schreibmaschinenschrift wird eine dicktengleiche Schrift des Paketentwick-
lers eingesetzt.

Man vergleiche das nächste Beispiel mit Beispiel 7-6-2 von Seite 388. Die zusätzliche Zeile am Ende zeigt einige Symbole aus `textcomp`, die in `mathptmx` nicht verfügbar sind.

Ein Beispiel für eine trigonometrische Funktion:

$$\sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}}$$

Bsp.
7-7-7

Schreibschrift: *ABC*.
Aus textcomp: \$ € ★ ω † ...

```
\usepackage{txfonts}\usepackage[full]{textcomp}
Ein Beispiel für eine trigonometrische Funktion:
\[ \sin \frac{\alpha}{2} =
      \pm \sqrt{\frac{1-\cos \alpha}{2}} \]
Schreibschrift:  $\mathcal{ABC}$ .\\
Aus textcomp: \textdollaroldstyle \texteuro\
\textborn \textmarried \textdied \ldots
```

¹Um das Euro-Zeichen in den verschiedenen T_EX-Fonts sehen zu können, muss die neueste Version der Datei `8x.enc` installiert sein.

Kodierung	Familie	Serie	Form(en)	Schriftbild
<i>TX Roman</i>				
OT1, T1, TS1, LY1	txr	m	n, it, sl, sc	TX Roman normal
OT1, T1, TS1, LY1	txr	bx, (b)	n, it, sl, sc	TX Roman fett kursiv
<i>TX Sans</i>				
OT1, T1, TS1, LY1	txss	m	n, (it), sl, sc	TX Sans normal
OT1, T1, TS1, LY1	txss	bx, (b)	n, (it), sl, sc	TX Sans fett geneigt
<i>TX Typewriter</i>				
OT1, T1, TS1, LY1	txtt	m	n, (it), sl, sc	TX Typewriter normal
OT1, T1, TS1, LY1	txtt	bx, (b)	n, (it), sl, sc	TX TYPEWRITER FETT KAPITÄLCHEN
<i>TX Math</i>				
OML	txmi	m, bx	it	<i>TX Math.</i> $\alpha \Omega$
OMS	txms	m, bx	n	$\mathcal{X} \lfloor \nabla \rfloor \rangle \nwarrow \{ \} \sqcup \sqcup \mathcal{M} \dashv \sqcup \neg \otimes \otimes$
U	txsya, txsyb	m, bx	n	$\cong \checkmark \Delta \sim \nabla \lambda \equiv \Leftarrow \Rrightarrow$

Tabelle 7.14: Klassifizierung der TX Fontfamilien

Die TX Fonts (siehe Tabelle 7.14) unterstützen die Textfontkodierungen OT1, T1, TS1 und LY1. Die OT1-Kodierung wurde jedoch nicht ganz originalgetreu umgesetzt: Einige ihrer Mängel wurden (fälschlicherweise) umgangen (z.B. die Tatsache, dass in „echten“ OT1-Fonts nur entweder das \$- oder das f-Zeichen verfügbar ist). Diese Korrekturen führen dazu, dass die neuen Definitionen nicht zusammen mit anderen OT1-kodierten Fonts funktionieren. Da OT1 immer noch die unter L^AT_EX voreingestellte Kodierung ist, kann diese Änderung Schwierigkeiten machen.¹

Das folgende Beispiel veranschaulicht diese Problematik. In der OT1-kodierten Computer Modern sind plötzlich alle Zeichen falsch: das \$-Zeichen wird zu einem f-Zeichen und alle anderen werden einfach fallengelassen. Mit T1 gibt es jedoch keine Probleme, so dass man txfonts immer mit `\usepackage[T1]{fontenc}` kombinieren sollte.²

```
\usepackage{txfonts}
\fontencoding{OT1}\selectfont      % LaTeX-Voreinstellung
\L.\l.\textdollar.\textsterling.\r{A}.\r{a}\hfill (txfont)

\fontfamily{cmtt}\itshape          % CM Typewriter kursiv
\L.\l.\textdollar.\textsterling.\r{A}.\r{a}\hfill
                                                    (nur Fehler)
```

L.I.\$.\$.\$\AA.\aa (txfont)

..f... (nur Fehler)

L.l.\$.\$.\$\AA.\aa (okay)

```
\fontencoding{T1}\selectfont      % ... in T1
\L.\l.\textdollar.\textsterling.\r{A}.\r{a}\hfill (okay)
```

Bsp.
7-7-8

¹Genauer gesagt, führen diese Fonts eine neue Kodierung ein, die OT1 ähnelt aber nicht identisch ist – und nennen diese fälschlicherweise OT1.

²Wie bereits in Abschnitt 7.3.5 besprochen, ist T1 ohnehin generell vorzuziehen.

Kodierung	Familie	Serie	Form(en)	Schriftbild
PX Roman				
OT1, T1, TS1, LY1	pxr	m	n, it, sl, sc	PX Roman normal
OT1, T1, TS1, LY1	pxr	bx, (b)	n, it, sl, sc	<i>PX Roman fett kursiv</i>
PX Math				
OML	pxmi	m, bx	it	PX Math. $\alpha \Omega$
OMS	pxms	m, bx	n	$\mathcal{P}\mathcal{X} \lfloor \nabla \rfloor \sqcup \swarrow \{ \} \sqcup \sqcup \mathcal{M} \dashv \sqcup \langle \neg \otimes \otimes$
U	pxsya, pxsyb	m, bx	n	$\mp \surd \Delta \neg \nabla \lambda \neq \leq \rightleftharpoons$

Tabelle 7.15: Klassifizierung der PX Fontfamilien

Hinzu kommt das schwerwiegendere Problem, dass in der derzeitigen Ausgabe der Fonts die Vor- und Nachbreiten der Glyphen für Formeln extrem klein sind, so dass sich hoch- oder tiefgestellte Zeichen sogar berühren.

Ein problematisches Beispiel:

```
\usepackage{amsmath,txfonts}
```

Ein problematisches Beispiel:

```
\[ t[u_1, \dots, u_n] = \sum_{k=1}^n \binom{n-1}{k-1} (1-t)^{n-k} t^{k-1} u_k \]
```

Bsp.
7-7-9

$$t[u_1, \dots, u_n] = \sum_{k=1}^n \binom{n-1}{k-1} (1-t)^{n-k} t^{k-1} u_k$$

Möglicherweise werden diese Probleme bei einer zukünftigen Version der Fonts behoben. Zum Vergleich das vorhergehende Beispiel mit mathptmx:

Ein problematisches Beispiel:

```
\usepackage{amsmath,mathptmx}
```

Ein problematisches Beispiel:

```
\[ t[u_1, \dots, u_n] = \sum_{k=1}^n \binom{n-1}{k-1} (1-t)^{n-k} t^{k-1} u_k \]
```

Bsp.
7-7-10

$$t[u_1, \dots, u_n] = \sum_{k=1}^n \binom{n-1}{k-1} (1-t)^{n-k} t^{k-1} u_k$$

Zusammenfassend lässt sich sagen, dass die TX Schriftfamilien noch einige Mängel beim Formelsatz aufweisen, dabei jedoch über eine Vielzahl von mathematischen und Textzeichen verfügen, einschließlich aller Symbole der AMS-Mathematikfonts und aller textcomp-Symbole. Wenn man besonders viele Symbole in Type1-Fonts benötigt, wie z.B. zum Erstellen von PDF-Dokumenten, dann stellen diese Schriften eine interessante Alternative dar.

7.7.6 pxfonts – Alternative Unterstützung für Palatino

Young Ryu entwickelte auch einen Satz virtueller Fonts mit begleitenden Type1-Fonts für die Darstellung von Formeln in Dokumenten, die Adobe Palatino als Grundschrift verwenden. Die PX Fonts (siehe Tabelle 7.15) werden durch Laden des Paketes pxfonts eingerichtet. Für serifenlose und Schreibmaschinenschriften greift das Paket auf Fonts aus dem txfonts-Paket zurück (Helvetica verkleinert und TX Typewriter), so dass beide Fontsätze installiert sein müssen.

Familie	Serie	Form(en)	Beispiele
<i>Utopia (T1, TS1)</i>			
futs	m, b, (bx)	n, sl, it, (sc)	Utopia-Regular Utopia-BoldItalic
<i>Fourier, Buchstaben in Formeln (FML)</i>			
futm, futmi	m	it	$\Delta\Theta\Lambda$ $\alpha\beta\gamma$ <i>abcdef</i> $\Delta\Theta\Lambda$ $\alpha\beta\gamma$
<i>Fourier, mathematische Symbole (FMS)</i>			
futm	m	n	$\neg\int\mathbb{C}\parallel$ <i>ABCEDEFGHIJKLMNOP</i>

Tabelle 7.16: Klassifizierung der Fourier-GUTenberg Fontfamilien

Das nächste Beispiel entspricht Beispiel 7-7-7 auf Seite 399, dieses Mal jedoch unter Verwendung des Paketes `pxfonts`.

Ein Beispiel für eine trigonometri-
sche Funktion:

$$\sin\frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos\alpha}{2}}$$

Die Schreibschrift sieht so aus: *ABC*.
Aus `textcomp`: \$€★☉†...

```
\usepackage{pxfonts}\usepackage[full]{textcomp}
```

Ein Beispiel für eine trigonometrische Funktion:

```
\[ \sin \frac{\alpha}{2} =
```

$\pm \sqrt{\frac{1 - \cos\alpha}{2}}$ `\]`

Die Schreibschrift sieht so aus:

`$_{\mathcal{ABC}}$.``\]`

Aus `textcomp`: `\textdollaroldstyle` `\texteuro``\`
`\textborn` `\textmarried` `\textdied` `\ldots`

Bsp.

7-7-11

Da die PX Fonts genauso aufgebaut sind wie die TX Fonts, treten hier die gleichen OT1-Probleme auf wie in Beispiel 7-7-8 auf Seite 400. Formeln werden ebenfalls sehr eng gesetzt, aber die Auswirkungen sind nicht so deutlich, wie bei den TX Fonts. Zum Vergleich hier auch noch einmal Beispiel 7-7-9 auf der vorherigen Seite:

Ein problematisches Beispiel:

$$t[u_1, \dots, u_n] = \sum_{k=1}^n \binom{n-1}{k-1} (1-t)^{n-k} t^{k-1} u_k$$

```
\usepackage{amsmath,pxfonts}
```

Ein problematisches Beispiel:

```
\[ t[u_1, \dots, u_n] = \sum_{k=1}^n
```

$\binom{n-1}{k-1} (1-t)^{n-k} t^{k-1} u_k$ `\]`

Bsp.

7-7-12

7.7.7 Die Fourier-GUTenberg Fonts

Adobe stiftete dem X-Consortium vier Fonts aus der Utopia-Familie (Utopia Regular, Utopia Italic, Utopia Bold und Utopia BoldItalic). Auch wenn es sich dabei nicht um Freeware handelt, sind diese Fonts doch kostenlos und werden grundlegend durch die PSNFSS-Sammlung unterstützt (siehe Abschnitt 7.6).

Die Fourier-GUTenberg-Sammlung von Michel Bovani ist eine Satzungem-
bung, die auf der Utopia-Schrift basiert, wobei der Zeichenvorrat jedoch auf ei-
ne vollständige T1-Kodierung aufgestockt wurde (OT1 wird *nicht* unterstützt).
Sie wurde außerdem um eine geeignete Auswahl an mathematischen Symbo-
len, griechischen Buchstaben in geneigter und aufrechter Form sowie ein pas-
sendes kalligraphisches und ein Blackboard Bold-Alphabet ergänzt. So lassen

sich mit ihrer Hilfe ganze Dokumente erstellen, ohne dass man auf andere Schriften zurückgreifen muss. Tabelle 7.16 auf der vorherigen Seite zeigt die Fontkodierung, Abbildung 8.5 auf Seite 531 veranschaulicht die Sammlung auf einer eigenen Beispielseite.

Ein Beispiel für eine trigonometrische Funktion:

$$\sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}}$$

Bsp.
7-7-13

Die Alphabete sind $\mathcal{ABC}$ und $\mathbb{ABC}$.

```
\usepackage{fourier}
```

Ein Beispiel für eine trigonometrische Funktion:

$$\sin \frac{\alpha}{2} = \pm \sqrt{\frac{1 - \cos \alpha}{2}}$$

Die Alphabete sind $\mathcal{ABC}$ und $\mathbf{ABC}$.

Das fourier-Paket unterstützt über die Option `upright` den Formelsatz „à la française“ – mit griechischen Buchstaben und lateinischen Großbuchstaben in aufrechter Lage. Man vergleiche das nächste Beispiel mit der Darstellung in Beispiel 8-4-1 auf Seite 502.

Bsp.
7-7-14

$$0 \leftarrow \underset{\zeta}{F \times \Delta(n-1)} \xrightarrow{\partial_0 \alpha(b)} E^{\partial_0 b}$$

```
\usepackage{amsmath}\usepackage[upright]{fourier}
```

$$\begin{aligned} & \left[0 \right] \xleftarrow{\left[\zeta \right]} F \times \Delta \left(n - 1 \right) \\ & \xrightarrow{\left\{ \partial_0 \right\} \alpha \left(b \right)} E^{\left\{ \partial_0 b \right\}} \end{aligned}$$

Wenn man, wie im vorigen Beispiel, zusätzliche Unterstützung für Formeln aus dem `amsmath`-Paket benötigt, sollte man dieses vorher laden, damit bestimmte Einstellungen für die Formatierung von Formeln in Utopia nicht überschrieben werden. Aus dem gleichen Grunde sollte auch das Paket `amssymb` immer zuerst geladen werden, wobei `fourier` bereits eine Reihe von Symbolen enthält, die sonst nur über `amssymb` verfügbar sind. Das Paket enthält sogar einen kleinen Zeichenvorrat, den man sonst nirgendwo findet (z.B. bestimmte Integralzeichen, einige Begrenzungssymbole sowie weitere Sonderzeichen). Einige von ihnen sind im nächsten Beispiel aufgeführt.

Bsp.
7-7-15

$$\llbracket \llbracket \times \# \times \rrbracket \oint \oint \oint f \overline{xxxxxx}$$

```
\usepackage{fourier}
```

```
\setlength\delimitershortfall{-2pt} % Begrenzungszeichen
                                   % vergrößern
```

$$\left[\left(\begin{array}{c} \text{\xswordsup} \\ \text{\nparallelslant} \\ \text{\xswordsdown} \end{array} \right) \right]$$

Wenn man den Befehlsnamen das Präfix `other` voranstellt, kann man im gleichen Dokument aufrechte und geneigte griechische Buchstaben verwenden. Zum Beispiel:

Bsp.
7-7-16

$$\Omega_{\beta} \neq \Omega_{\beta}$$

```
\usepackage[upright]{fourier}
```

$$\left[\Omega_{\beta} \neg \Omega_{\beta} \right]$$

Ohne die Option `upright` (oder mit der voreingestellten Option `sloped`), werden die Buchstaben entsprechend der Konventionen für den Formelsatz

Familie	Serie	Form(en)	PostScript Fontnamen und Beispiele
<i>URW Antiqua Condensed</i> (OT1, T1, TS1)			
uaq	(m), mc	n, sl, (it), sc	URWAntiquaT-RegularCondensed
<i>URW Grotesk Bold</i> (OT1, T1, TS1)			
ugq	b, (bx), (m)	n, sl, (it), sc	URWGroteskT-Bold

Tabelle 7.17: Klassifizierung der URW Antiqua und Grotesk Fonts

geneigt dargestellt, also große griechische Buchstaben in aufrechter und alle anderen in geneigter Lage. In diesem Fall wirken die `\other...`-Befehle entsprechend gegenläufig.

$$\Omega_{\beta} \neq \Omega_{\beta}$$

```
\usepackage[sloped]{fourier}
\[ \Omega_{\beta} \neq \otherOmega_{\otherbeta} ]
```

Bsp.
7-7-17

Zurzeit bietet `fourier` keine Unterstützung für den Befehl `\boldmath`. Eine Kombination mit dem Paket `bm` führt daher meistens zu einer „Arme-Leute-Version“ des Fettdrucks: siehe Abschnitt 8.8.2.

Unterstützung für den
kommerziellen
Expertensatz

Als Ergänzung für die kostenlosen Schriften bietet Adobe einen Expertensatz zum Kauf, der Mediävalziffern, echte Kapitälchen und eine halbfette sowie eine extrafette Schriftserie enthält. Das `fourier`-Paket unterstützt diese Schriften mithilfe einiger zusätzlicher Optionen: Die Option `expert` stellt zur Auswahl der zusätzlichen Fontserien die Befehle `\textsb` (halbfett) und `\textblack` (extrafett) bereit, und ermöglicht den Zugriff auf die echten Kapitälchen durch `\textsc`. Die Option `oldstyle` unterstützt die gleichen Befehle und stellt zusätzlich die Ziffern im Text auf Mediävalziffern um (in diesem Fall kann man mit `\lining` auf Versalziffern zugreifen). Und die Option `fulloldstyle` verwendet, zusätzlich zum Funktionsumfang von `oldstyle`, Mediävalziffern auch in Formeln.

7.7.8 Die URW Antiqua und Grotesk Fonts

URW Antiqua
Condensed
10pt/12pt (uaq)

Das deutsche Unternehmen URW hat zwei kostenlose PostScript-Fonts zur Verfügung gestellt: URW Antiqua Condensed und URW Grotesk Bold. Sie können in $\text{\LaTeX}$ als virtuelle Fonts und `.fd`-Dateien genutzt werden. Der Zugriff erfolgt entsprechend der Klassifizierung in Tabelle 7.17. Das folgende Beispiel wurde mit dem Befehl `\fontfamily{uaq}\selectfont` gesetzt:

For the price of £45, almost anything can be found floating
in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345-67890
JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often naï-
ve vis-à-vis the dæmonic phœnix's official rôle in fluffy soufflés?

URW Grotesk Bold
10pt/12pt (ugq)

Wie der Name schon sagt, ist die URW Grotesk Bold nur in der Serie fett erhältlich (wobei $\text{\LaTeX}$ zwar den Befehl für die normale Serie unterstützt, damit aber auch nur den fett gedruckten Font aufruft). Sie eignet sich ohnedies

nicht für Fließtexte, wohl aber für Überschriften und andere abgesetzte Texte.

For the price of £45, almost anything can be found floating in fields. ¡THE DAZED BROWN FOX QUICKLY GAVE 12345-67890 JUMPS! — ¿But aren't Kafka's Schloß and Æsop's Œuvres often naïve vis-à-vis the dæmonic phoenix's official rôle in fluffy soufflés?

7.7.9 yfonts – Altdeutsche Schriften

Es gibt eine Sammlung schöner Fonts, mit denen sich Dokumente in Gotisch, Schwabacher und Fraktur setzen lassen. Sie sind traditionellen Schriften nachempfunden und wurden von Yannis Haralambous [64] mit METAFONT¹ gestaltet. Mittlerweile sind diese Fonts auch in Type1-Versionen erhältlich. Durch Laden des Paketes yfonts von Walter Schmidt kann man die Schriften nutzen. Das Paket definiert intern einige lokale Kodierungen, die an die besonderen Eigenschaften der Fonts angepasst sind, und integriert sie vollständig in das L^AT_EX-Fontmanagement.

Die Deklarationen \gothfamily, \swabfamily und \frakfamily wechseln jeweils zu Gotisch, Schwabacher bzw. Fraktur. Wenn man ein ganzes Dokument in einer dieser Schriften setzen möchte, sollte der entsprechende Befehl direkt *nach* \begin{document} stehen. Da die Fonts keiner Standardkodierung folgen, lassen sich die Voreinstellungen des Dokumentes (z.B. \familydefault) nicht umdefinieren. Zusätzlich zu den Deklarationsformen stehen passende \text...-Befehle zum Formatieren kurzer Textstücke zur Verfügung.

Das Paket enthält die Schriften **Gotisch**, auch **Textur** genannt, **Schwabacher** und **Fraktur**, die auch als „gebrochene Schriften“ bekannt sind.

```
\usepackage{yfonts}\usepackage[document]{ragged2e}
```

Das Paket enthält die Schriften \textgoth{Gotisch, auch Textur genannt}, \textswab{Schwabacher} und \textfrak{Fraktur}, die auch als \textfrak{‘ge\~bro\~che\~ne Schriften’} bekannt sind.

Bsp.
7-7-18

Die Schriften sind in den üblichen L^AT_EX-Größen ab 10pt verfügbar, so dass die Befehle zum Verändern des Schriftgrades (wie etwa \normalsize und größer) funktionieren. Es gibt jedoch keine weiteren Fontserien oder -formen, so dass Befehle wie \emph, \textit oder \textbf außer einer Warnung nichts bewirken. Gemäß der historischen Tradition kann man Schwabacher für Hervorhebungen in Frakturtexten verwenden.

Akzente werden im normalen L^AT_EX-Verfahren erzeugt (z.B. \"a für ä). Um die Eingabe zu erleichtern, enthalten die Fonts außerdem Ligaturen, die Umlaute darstellen (z.B. "a). In Fraktur und Schwabacher gibt es zusätzlich alternative Umlaute, die sich über *a und ähnliche Ligaturen aufrufen lassen. Wenn das yfonts-Paket mit der Option varumlaut geladen wird, erzeugt \"a automatisch die alternativen Zeichen.

¹Das Kompilieren der Fonts aus den METAFONT-Quellen führt manchmal zu Fehlermeldungen. Man erhält jedoch im Allgemeinen brauchbare Fonts, wenn man METAFONT anweist, diese zu ignorieren. Die Sammlung umfasst auch einen Font mit barocken Initialen.

Alle drei Fonts verfügen über eine Glyphen für ein „kurzes s“, das über die Ligatur `s`: aufgerufen wird, und ein „scharfes s“, auf das mit `\ss` oder mit den Ligaturen `sz` oder `"s` zugegriffen wird.

		<code>\usepackage{fonts}</code>	
		<code>\Large</code>	
		<code>\frakfamily</code> Fraktur:	<code>"a "e "u "o</code>
		<code>\hfil</code>	<code>*a *e *u *o</code>
		<code>\linebreak</code>	<code>sz \quad s viz.\ s:</code>
Fraktur: ä ë ü ö	ä ë ü ö		
ß f viz. s		<code>\swabfamily</code> Schwabacher:	<code>"a "e "u "o</code>
		<code>\hfil</code>	<code>*a *e *u *o</code>
		<code>\linebreak</code>	<code>sz \quad s viz.\ s:</code>
Schwabacher: ä ë ü ö	ä ë ü ö		
ß f viz. s		<code>\gothfamily</code> Gotisch:	<code>"a "e "u "o</code>
		<code>\hfil</code>	<code>(unavail)</code>
		<code>\linebreak</code>	<code>sz \quad s viz.\ s:</code>
Gotisch: ä ë ü ö	(unavail)		
ß l viz. s			

Bsp.
7-7-19

Der Font, der über `\gothfamily` aufgerufen wird, ist keine Kopie der Schrift, die Gutenberg für seine Bibel verwendete (und die aus insgesamt 288 Glyphen bestand). Sie folgt jedoch Gutenbergs Richtlinien für Kleinbuchstaben und verfügt über so viele Ligaturen, wie in einen 7-Bit-Font hineinpassen. Darum sind viele Standard-ASCII-Symbole nicht zu finden.

Die anderen beiden Fonts verfügen auch nur über einen Teil der sichtbaren ASCII-Zeichen. Problematisch sind dabei das Semikolon (das in Schwabacher fehlt) sowie die Zeichen `+`, `=`, `'`, `[`, `]`, `/`, `*`, `@`, `&` und `%` (die entweder fehlen bzw. falsche oder unpassende Formen erzeugen). Dieses Fehlen stellt nur selten ein Problem dar, da sie in Dokumenten, die in solchen Schriften gesetzt werden, normalerweise ohnehin nicht vorkommen. Man muss jedoch wissen, dass sie weder eine Warnung noch eine Fehlermeldung verursachen, so dass der einzige Hinweis fehlende oder falsche Zeichen im gedruckten Text sind!

		<code>\usepackage{fonts}</code>	
		<code>\newcommand\test{+ = ' [] / * \$ % & ; @</code>	
Symbole: + = ' [] / * \$ % & ; @		<code>\ttfamily</code>	<code>\test \par</code>
Fraktur Probleme: + = ' [] / * \$ % & ;		<code>\frakfamily</code> Fraktur Probleme:	<code>\test \par</code>
Schwabacher Probleme: + = ' [] / * \$		<code>\swabfamily</code> Schwabacher Probleme:	<code>\test\par</code>
%		<code>\gothfamily</code> Gotisch Probleme:	<code>\test</code>
Gotisch Probleme: ffi tt s ä ö ffi ll \$ ff ;			

Bsp.
7-7-20

Der voreingestellte Zeilenvorschub der Standardklassen ist für die alt-deutschen Schriften zu groß. Darum verfügt das Paket über den Befehl `\fraklines`, der `\baselineskip` auf einen für Fraktur oder Schwabacher geeigneten Wert setzt. Er muss nach jedem Befehl, der den Schriftgrad ändert, wiederholt werden.

Die Fontsammlung enthält außerdem, wie im nächsten Beispiel dargestellt, einen Zeichensatz mit Schmuckinitialen.

Bsp.
7-7-21

ies ist ein Blindtext, an dem sich verschiedene Dinge ablesen lassen. Der Grauwert der Schriftfläche wird sichtbar und man kann an ihm prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Bei genauem Hinsehen werden die einzelnen Buchstaben und ihre Besonderheiten erkennbar, etc

```
\usepackage[ngerman]{babel} \usepackage{color}
\usepackage[varumlaut]{yfonts}
\frakfamily\fraklines
\yinipar{\color{blue}D}ies: ist ein Blindtext, an
dem sich verschiedene Dinge ablesen lassen. Der
Grauwert der Schriftfl\"ache wird sichtbar und man
kann an ihm pr\"ufen, wie gut die Schrift zu lesen
ist und wie sie auf den Leser wirkt. Bei genauerem
Hinsehen werden die einzelnen Buchstaben und ihre
Besonderheiten erkennbar, \etc
```

Der Befehl `\yinipar` aus dem vorigen Beispiel leitet einen neuen, nicht eingerückten Absatz ein, und setzt den ersten Buchstaben als hängendes barockes Initial. Damit der Befehl funktioniert, muss der gesamte Absatz (bis einschließlich zur nächsten Leerzeile oder dem nächsten `\par`-Befehl) mit `\fraklines` formatiert werden. Andernfalls verbleibt zu viel oder zu wenig Platz für das Initial.

Alternativ kann man Initiale auch mit `\textinit` oder dem Fontwechselbefehl `\initfamily` setzen. In diesem Fall werden sie an der Grundlinie ausgerichtet. Das Beispiel zeigt auch den Befehl `\etc`, der ein einstmals beliebtes Zeichen für „etc.“ erzeugt; dieses ist nur in Fraktur verfügbar.

Die Schriftensammlung enthält noch einen zweiten Frakturfont mit etwas breiteren Glyphen bei zugleich etwas kleinerer Strichstärke. Man erhält diesen Font, wenn man `\frakdefault` wie im nächsten Beispiel undefiniert. Wenn man dieses Beispiel mit Beispiel 7-7-21 vergleicht, ist die unterschiedliche Laufweite, die zu einer überlaufenden dritten Zeile führt, deutlich zu erkennen.

Bsp.
7-7-22

ies ist ein Blindtext, an dem sich verschiedene Dinge ablesen lassen. Der Grauwert der Schriftfläche wird sichtbar und man kann an ihm prüfen, wie gut die Schrift zu lesen ist und wie sie auf den Leser wirkt. Bei genauem Hinsehen werden die einzelnen Buchstaben und ihre Besonderheiten erkennbar, etc

```
\usepackage[ngerman]{babel} \usepackage{color}
\usepackage[varumlaut]{yfonts}
\renewcommand\frakdefault{\ysmfrak}
\frakfamily\fraklines
\yinipar{\color{blue}D}ies: ist ein Blindtext, an
dem sich verschiedene Dinge ablesen lassen. Der
Grauwert der Schriftfl\"ache wird sichtbar und man
kann an ihm pr\"ufen, wie gut die Schrift zu lesen
ist und wie sie auf den Leser wirkt. Bei genauerem
Hinsehen werden die einzelnen Buchstaben und ihre
Besonderheiten erkennbar, \etc
```

7.7.10 euler, eulervm – Zugriff auf die Euler Fonts

Wie bereits zuvor erwähnt, gestaltete Hermann Zapf einen Satz wunderschöner Schriften für den Formelsatz – aufrechte Zeichen mit handschriftartiger Note – die nach dem berühmten Mathematiker Leonhard Euler [84] benannt sind. Diese Fonts können als eigenständiges (Mathematik-) Alphabet genutzt werden, oder man kann die Voreinstellungen für den Formelsatz in $\text{\LaTeX}$ generell so ändern, dass (anstelle von Computer Modern) immer die Euler Mathematikfonts verwendet werden.

Familie	Serie	Form(en)	Schriftbild
<i>Euler Roman (U)</i>			
eur	m	n	Euler Roman normal
eur	b	n	Euler Roman fett
<i>Euler Script (U)</i>			
eus	m	n	EUER SCRIPT
<i>Euler Fraktur (U)</i>			
euf	m	n	Euler Fraktur
<i>Euler Extension (U)</i>			
euex	m	n	$\sum \prod \int^{\infty} \{ \}$

Tabelle 7.18: Klassifizierung der Euler Mathematikfont-Familien

Die Euler Fonts enthalten drei Mathematikalphabete: *SCRIPT*, *Euler Fraktur*, und Euler Roman.¹ Das Schreibschrift-Alphabet lässt sich durch das Paket `eucal` nutzen, das dieses Mathematikalphabet mit dem Befehl `\mathcal` (veralteter anderer Name `\EuScript`) zur Verfügung stellt. Wird das Paket mit der Option `mathscr` geladen, dann kann man mit dem Befehl `\mathscr` auf das Mathematikalphabet zugreifen, während `\mathcal` seine ursprüngliche Definition behält.

Um Euler Fraktur in Formeln zu verwenden, bedient man sich des Paketes `eufrak`, welches das Mathematikalphabet `\mathfrak` (veralteter anderer Name `\EuFrak`) definiert. Es gibt kein eigenes Paket um gesondert auf das Euler Roman Alphabet zuzugreifen. Das nächste Beispiel zeigt die Schriften Computer Modern Calligraphic, Euler Script und Euler Fraktur Seite an Seite.

$$\mathcal{A} \neq \sum_{k < n} \mathcal{A}_k \neq \mathfrak{A}$$

```
\usepackage[mathscr]{eucal} \usepackage{eufrak}
\[\mathcal{A} \neq \sum_{k < n} \mathscr{A}_k \neq \mathfrak{A} \]
```

Bsp.
7-7-23

Die NFSS-Klassifizierung für die Schriften dieser Familien ist in Tabelle 7.18 dargestellt. Die Fonts der aktuellen Distribution von mathematischen Euler-Schriftfamilien sind nur in Kodierungen erhältlich, die sich von allen anderen Kodierungen für den Formelsatz unterscheiden. Darum wird allen die Kodierung U (unbekannt) zugewiesen.

Die ungewöhnliche Kodierung erschwert ein bloßes Ersetzen der voreingestellten CM Mathematikfonts durch die Euler Mathematikalphabete. Trotzdem geht das Paket `euler` von Frank Jensen eben diesen Weg, indem es die Einstellungen für Mathematikfonts in $\text{\LaTeX}$ fast vollständig umdeklariert. In Kombination mit dem Paket `beton`, das Concrete als voreingestellte Schriftfamilie für den Fließtext verwendet, ahmt es die in Beispiel 7-7-2 dargestellte Typographie des Buches *Concrete Mathematics* [61] von Knuth nach.

¹Keines dieser Alphabete eignet sich zum Setzen von Texten, da die Vor- und Nachbreiten der einzelnen Glyphen speziell auf den Formelsatz zugeschnitten sind.

Weitreichende Umkodierungen in Makropaketen, wie sie das `euler`-Paket vornimmt, führen häufig zu Konflikten mit anderen Paketen, die davon ausgehen, dass sich bestimmte Symbole genau an den Stellen befinden, die ihnen in den bewährten Fontkodierungen zugewiesen sind. Das Paket `eulervm` von Walter Schmidt versucht dieses Problem zu umgehen, indem es umkodierte virtuelle Fonts einsetzt, die so weit wie möglich den Standard-Mathematikkodierungen OML, OMS und OMX entsprechen.

Das Paket richtet ein `\mathnormal`-Alphabet ein, das weitgehend auf Euler Roman basiert, und ein `\mathcal`-Alphabet, das auf Euler Script beruht. Es bietet keine direkte Unterstützung für das Euler Fraktur-Alphabet; für diese Schrift muss man zusätzlich das `eufrak`-Paket laden. Die mathematischen Symbole stammen ebenfalls von den Euler Fonts, bis auf wenige Ausnahmen, die den Computer Modern Mathematikfonts entnommen sind. Wenn man das nächste Beispiel mit Beispiel 7-7-23 auf der vorherigen Seite vergleicht, stellt man fest, dass `\mathcal` sich verändert hat und dass `\sum` und die Indizes anders aussehen, da sie nun aus den Euler Fonts stammen.

Bsp.
7-7-24

$$\mathcal{A} \neq \sum_{k < n} A_k \neq \mathfrak{A} \quad \begin{array}{l} \text{\texttt{\usepackage{eulervm,eufrak}}} \\ \text{\texttt{\[\mathcal{A} \neq \sum_{k < n} A_k \neq \mathfrak{A} \]}} \end{array}$$

Durch die Option `small` lädt `eulervm` alle Euler Fonts auf 95% verkleinert, so dass sie besser zu einigen Grundschriften passen (z.B. Adobe Minion). Diese Option wirkt sich auch auf die Euler Fraktur Fonts aus, sofern sie mit `eufrak` geladen wurden, sowie auf die AMS-Symbolfonts.

Weder der Standardbefehl `\hbar` noch `\hslash` (aus dem Paket `amssymb`) können unmodifiziert mit den Euler Fonts (d.h. dem Paket `euler`) benutzt werden, da `\hslash` ein „h“ aus der Computer Modern verwendet und der Strich von `\hbar` seltsam positioniert wird.

Bsp.
7-7-25

$$\hbar \neq \hbar \quad \begin{array}{l} \text{\texttt{\usepackage{amssymb,euler}}} \\ \text{\texttt{\[\hslash \neq \hbar \]}} \end{array}$$

Dadurch wird der Nutzen des `euler`-Paketes für Themen der Physik und verwandte Bereiche eingeschränkt. Das Paket `eulervm` löst dieses Problem (teilweise), indem es die Möglichkeiten des virtuellen Fontmechanismus nutzt ([95] erläutert die Details), um eine Glyphe, bei der das „h“ mit einem korrekten Schrägstrich versehen ist, bereitzustellen. Es verfügt jedoch nur über die Version mit dem Schrägstrich (`\hslash`); verwendet man `\hbar`, so wird eine Warnung ausgegeben und trotzdem die Glyphe mit dem Schrägstrich eingesetzt.

Bsp.
7-7-26

$$\hbar \equiv \hbar \quad \begin{array}{l} \text{\texttt{\usepackage{eulervm}}} \\ \text{\texttt{\[\hslash \equiv \hbar \]}} \end{array}$$

Die Funktionalität des `exscale`-Paketes steht automatisch zur Verfügung. Abschnitt 7.5.5 auf Seite 379 enthält nähere Details hierzu.

Typische Fonteinstellungen verwenden normalerweise für Texte und Formeln die gleichen Ziffern. Die Euler Fonts enthalten aber einen Satz von markant aussehenden Ziffern, die sich deutlich von den Textziffern unterscheiden.

Gemäß der Voreinstellung werden die Zahlen der Grundschrift auch in Formeln eingesetzt. Um auf die Euler Roman-Ziffern umzuschalten, muss man sie explizit mit der Option `euler-digits` anfordern. In diesem Fall muss man sorgfältig zwischen Ziffern im Fließtext und in einem mathematischen Kontext differenzieren. So ist z.B. darauf zu achten, dass man die `$`-Zeichen nicht vergisst, wie etwa in der ersten Zeile des nächsten Beispiels.

Der Wert kann 1, 2, oder -1 sein (falsch!)	<code>\usepackage{ccfonts}</code> <code>\usepackage[euler-digits]{eulervm}</code>	
Der Wert kann 1, 2, oder -1 sein (richtig!)	Der Wert kann 1, 2, oder -1 sein (falsch!) Der Wert kann 1 , 2 , oder -1 sein (richtig!)	<code>\par</code> <code>\mathbf{1}</code> , <code>\mathbf{2}</code> , oder <code>\mathbf{-1}</code>

Bsp.
7-7-27

Normalerweise wird das mathematische Akzentzeichen `\hat` der Grundschrift entnommen. Das kann jedoch von Nachteil sein, wenn sich die Text- und Formelschriften auffallend voneinander unterscheiden. Durch Setzen der Option `euler-hat-accent` wird daher eine alternative Version aus den Euler Fonts verwendet. Im Beispiel wird diese Option durch Definieren eines alternativen Akzentbefehls namens `\varhat` nachgeahmt, um beide Varianten miteinander zu vergleichen (keine von beiden sieht wirklich perfekt aus).

$\hat{x} \neq \hat{x}$ und $\hat{\hat{x}} \neq \hat{\hat{x}}$	<code>\usepackage{palatino,eulervm,eufrak}</code> <code>\DeclareMathAccent\varhat{\mathalpha}{symbols}{222}</code> <code>\Large \$ \hat x \neq \varhat x \$</code> <code>\$ \hat \mathfrak{K} \neq \varhat \mathfrak{K} \$</code>

Bsp.
7-7-28

Am Besten lädt man das `eulervm`-Paket normalerweise erst, wenn alle anderen Textfonts bereits definiert sind, da es für die Definition der Mathematikalphabete (z.B. `\mathsf`) die Voreinstellungen des Dokumentes auswertet, die beim Laden des Paketes vorhanden sind. Im nächsten Beispiel ist die Reihenfolge der Pakete entscheidend, da auch das `ccfonts`-Paket Mathematikfonts einzurichten versucht und somit das letzte Paket gewinnt.

In dem Buch *Concrete Mathematics* [61], in dem zum ersten Mal Euler- und Concrete-Schriften gemeinsam verwendet wurden, ist zu sehen, dass geeignete $\leq$ - und $\geq$ -Zeichen auch einmal zu den Euler Mathematikfonts gehörten. Sie gingen unterwegs irgendwo verloren, obwohl man in [96] sowie in Makros, die Donald Knuth für das Buch entwickelte, immer noch Spuren von ihnen findet. Für das Paket `eulervm` brachte Walter Schmidt sie mithilfe des virtuellen Fontmechanismus wieder zurück; das nächste Beispiel bietet sich zu einem Vergleich mit Beispiel 7-7-2 auf Seite 395 an, wo das übliche $\leq$ -Zeichen abgebildet ist.

Concrete Roman passt gut zu Euler Math, wie man hier sehen kann:

$$\sum_{0 \leq k < n} k = \frac{n(n-1)}{2}$$

<code>\usepackage{ccfonts,amssymb}</code> <code>\usepackage[euler-digits]{eulervm}</code> Concrete Roman passt gut zu Euler Math, wie man hier sehen kann: <code>\[\sum_{0 \leq k < n} k = \frac{n(n-1)}{2} \]</code>

Bsp.
7-7-29

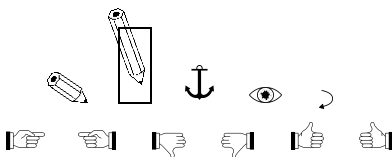
7.8 Die L^AT_EX-Welt der Symbole

Kurz nachdem T_EX und METAFONT verfügbar waren, begannen die Anwender neue Symbolfonts für das System zu entwickeln. Mit der Zeit wuchs der Vorrat an verfügbaren Zeichen auf eine beachtliche Zahl. Die *Comprehensive L^AT_EX Symbol List* von Scott Pakin [136] führt 2826 Symbole (Stand September 2003) mit ihren zugehörigen L^AT_EX-Befehlen auf. Für einige Symbole benötigt man die erforderlichen Schriften und Pakete (sie sind z.B. über einen CTAN-Host erhältlich; siehe Anhang C), die man zunächst installieren muss. Sie sind normalerweise mit Installationsanweisungen und einer allgemeinen Anleitung versehen.

Die Schriften und Pakete in diesem Abschnitt stellen nur einen Teil des verfügbaren Angebotes dar. Wenn ein Symbol hier nicht aufgeführt wird, dann sind die 70 Seiten von [136] ein wertvolles Nachschlagewerk, um zu finden, was man sucht. Zunächst werden eine Reihe von Dingbats-Fonts betrachtet, von denen einige recht ungewöhnliche Symbole enthalten. Danach folgt eine Einführung in das TIPA-System, das phonetische Symbole unterstützt. Der Abschnitt schließt damit, wie man ein einzelnes (für Europa nicht gerade uninteressantes) Symbol erhält: den Euro. Da es sich um eine relativ junge Ergänzung der Symbolwelt handelt, fehlt es in vielen Fonts, so dass man Alternativen benötigt, um es zu erzeugen. Alle Pakete und Schriften in diesem Abschnitt und in [136] sind kostenlos erhältlich.

7.8.1 dingbat – Eine Auswahl von Händen

Das dingbat-Paket von Scott Pakin ermöglicht den Zugriff auf zwei Symbolfonts von Arthur Keller (ark10.mf) und Doug Henderson (dingbat.mf). Es stellt einen Satz von Händen und einige andere Symbole zur Verfügung. Das Beispiel zeigt die meisten von ihnen. Man muss beachten, dass die `\largepencil`-Glyphen größer ist, als der Raum, den sie offiziell einnimmt (wie durch den Rahmen, der mit `\frame` um sie gesetzt wurde, angedeutet).



Bsp.
7-8-1

```
\usepackage{dingbat}
\smallpencil \quad \frame{\largepencil} \quad
\anchor \quad \eye \quad \carriagereturn\\[5pt]
\leftpointright \quad \rightpointleft \quad
\leftthumbsdown \quad \rightthumbsdown \quad
\leftthumbsup \quad \rightthumbsup
```

Diese Fonts liegen nur als METAFONT-Versionen vor, so dass sie sich nicht wirklich zum Erzeugen von PDF-Dateien (z.B. mit pdfT_EX) eignen.

7.8.2 wasysym – Waldis Symbolfont

Das wasysym-Paket von Axel Kielhorn ermöglicht den Zugriff auf die wasy-Fonts von Roland Waldi. Sie erschienen 1989 zum ersten Mal und sind mittlerweile sowohl als METAFONT- wie auch als Type1-Schriften verfügbar. Sie umfassen eine große Bandbreite von Symbolen aus verschiedenen Bereichen, darunter astronomische und astrologische Zeichen, APL, Musiknoten, Kreise sowie Polygone und Sterne (siehe Tabelle 7.19 auf der nächsten Seite).

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	△	◁	◁	▷	▷	∴	⊙	☎	"0x
'01x	✓	♠	♠	♠	♠	♠	♠	♠	
'02x	◀	▶	⚡	Ω	∪	∩	⊗	∩	"1x
'03x	┐	♀	♂	☒	⊙	∞	✂	∅	
'04x	●	☺	☺	○	☾	☾	♂	♀	"2x
'05x	<	>	^	v	☺	☹	☼	☹	
'06x	∪	☒	□	◇	☒	☒	✱	○	"3x
'07x	○	○	~	~	□	□	≤	≥	
'10x	≈	✱	✱	☆	☆	☆	▽	◐	"4x
'11x	◐	◑	◑	▲	▼				
'12x	♂	♂	♂			♂	♂	♂	"5x
'13x	♂	♂	♂	♂	♂	♂	♂	♂	
'14x	♂	♂	♂	♂	♂	♂	♂	♂	"6x
'15x	♂	♂	♂	♂	♂	♂	♂	♂	
'16x	♂	♂	♂	♂	♂	♂	♂	♂	"7x
'17x	♂	♂	♂	♂	♂	♂	♂	♂	
	"8	"9	"A	"B	"C	"D	"E	"F	

Tabelle 7.19: Glyphen des wasy Fonts

Das wasysym-Paket definiert Befehlsnamen wie `\phone` (Telefon), um auf die einzelnen Glyphen zuzugreifen. Wenn man nur wenige Zeichen aus dem Font benötigt, kann man auch alternativ die `pifont`-Schnittstelle nutzen und unter dem Namen `wasy` direkt auf die Symbole zugreifen.

☎ mit wasysym ☎ mit pifont

`\usepackage{wasysym,pifont}`

`\phone\` mit `\texttt{wasysym}`
`\Pisymbol{wasy}{7}` mit `\texttt{pifont}`

Bsp.
7-8-2

7.8.3 marvosym – Die Schnittstelle zum MarVoSym Font

Der MarVoSym Font von Martin Vogel ist ein weiterer Pi-Font, der Symbole aus vielen, zum Teil recht ungewöhnlichen Bereichen enthält, wie z.B. Wäschezeichen (falls man eigene Wäschezettel erstellen möchte ☺), astronomische und astrologische Symbole und viele andere mehr.

Das unterstützende L^AT_EX-Paket `marvosym` wurde von Thomas Henlich geschrieben, der den Font auch von TrueType zu PostScript Type 1 konvertierte. Das Paket definiert Befehlsnamen für alle Symbole, von denen einige im nächsten Beispiel aufgeführt sind. Der gesamte Satz ist in der Datei `marvodoc.pdf` aufgelistet, die der Distribution beiliegt.

	ˆ0	ˆ1	ˆ2	ˆ3	ˆ4	ˆ5	ˆ6	ˆ7	
ˆ04x		☎	==	⚠	⚠	⚠	☎	☎	ˆ2x
ˆ05x	(	)	×	+	,	-	.	/	
ˆ06x	0	1	2	3	4	5	6	7	ˆ3x
ˆ07x	8	9	→	⇒	≤	≅	≥	↔	
ˆ10x	@	○	✉	€	€	⚡	☎	—	ˆ4x
ˆ11x	☎	☎	☎	☎	☎			□	
ˆ12x	→	✂	---	✂	☎	☎	☎	☎	ˆ5x
ˆ13x	☎	☎	☎	/	/	≡	≡	/	
ˆ14x	☎	☎	☎	€	€	€	☎	—	ˆ6x
ˆ15x	☎	☎	☎	☎	↓	☎	☎	☎	
ˆ16x	→	✂	---	✂	☎	☎	☎	☎	ˆ7x
ˆ17x	☎	☎	☎	○	♂	♀	♀	☎	
ˆ20x	☎	☎	☎	☎	☎	☎	☎	☎	ˆ8x
ˆ21x	☎	☎	☎	☎	☎	☎	☎	☎	
ˆ22x	☎	●	●	■	■	●	—	□	ˆ9x
ˆ23x	☎	L	I	○	T	L	I	T	
ˆ24x	☎	β	☎	☎	€	☎	\$	☎	ˆAx
ˆ25x	☎	☎	☎	☎	☎	☎	☎	☎	
ˆ26x	☎	☎	☎		☎	☎	☎	☎	ˆBx
ˆ27x	☎	☎	☎	☎	☎	☎	☎	☎	
ˆ30x	☎	☎	☎	☎	☎	☎	☎	☎	ˆCx
ˆ31x	☎	☎	☎	☎	☎	☎	☎	☎	
ˆ32x	☎	☎	☎	☎	☎	☎	☎	☎	ˆDx
ˆ33x	☎	☎	☎	☎	☎	☎			
ˆ34x	☎	☎	☎	☎	☎	☎	☎	☎	ˆEx
ˆ35x	☎	☎	☎	☎					
ˆ36x	A	p						.	ˆFx
ˆ37x						☎	☎	☎	
	ˆ8	ˆ9	ˆA	ˆB	ˆC	ˆD	ˆE	ˆF	

Tabelle 7.20: Glyphen des MarVoSym Fonts

Bsp.
7-8-3

\usepackage{marvosym}

```

\Large \Mobilefone\ \Faxmachine\ \Fixedbearing\ \LineLoad\
\Coffeecup\ \Football\ \AtForty\ \IroningII\ \Cancer\ \Virgo\
\RewindToStart\ \ForwardToIndex\ \ComputerMouse\ \Keyboard\
\Female\ \FEMALE\ \Smiley\ \Frowny\ \Yingyang\ \Bicycle

```


Wer über eine neuere Distribution verfügt, kann auch mithilfe der Zeichentabelle 7.20 auf der vorherigen Seite über die pifont-Schnittstelle direkt auf die einzelnen Symbole zugreifen. Der Name für den Pi-Font lautet dann `mvs`. Älteren Distributionen fehlt möglicherweise die Datei `umvs.fd`, die für diese Methode benötigt wird. Sie lässt sich jedoch, wie im Folgenden gezeigt, leicht hinzufügen.

```
\begin{filecontents}{umvs.fd}
\DeclareFontFamily{U}{mvs}{}
\DeclareFontShape{U}{mvs}{m}{n}{<-> fmv8x}{}
\end{filecontents}

\usepackage{pifont}

\Huge \Pisymbol{mvs}{73} \Pisymbol{mvs}{74} \Pisymbol{mvs}{98}
\Pisymbol{mvs}{120} \Pisymbol{mvs}{121} \Pisymbol{mvs}{234}
```

Bsp.
7-8-4



7.8.4 bbding – METAFONT-Alternative zu Zapf Dingbats

Für alle, die keine PostScript Type1-Fonts verwenden können, hat Karel Horak mit METAFONT einen Zeichensatz gestaltet, der die meisten Symbole aus dem Dingbats-Font von Hermann Zapf enthält. Das Paket `bbding` von Peter Møller Neergaard dient als Schnittstelle, indem es Befehlsnamen für jedes Symbol definiert. (Die Namensgebung orientiert sich dabei an der WordPerfect-Nomenklatur für den Zapf Dingbats Font.) In der Dokumentation des Paketes sind alle Namen aufgelistet; einige von ihnen sind im nächsten Beispiel zu sehen.

```
\usepackage{bbding}

\XSolid\ \XSolidBold\ \XSolidBrush\ \Plus\ \PlusOutline\
\DavidStar\ \DavidStarSolid\ \JackStar\ \JackStarBold\
\FourStar\ \FiveFlowerPetal\ \SixFlowerOpenCenter\
\PhoneHandset\ \Peace\ \OrnamentDiamondSolid\ \ScissorLeft\
\Snowflake\ \Sparkle\ \Tape\ \TriangleDown
```

Bsp.
7-8-5



Ansonsten lassen sich die einzelnen Symbole auch mithilfe der Zeichentabelle 7.21 auf der gegenüberliegenden Seite über die pifont-Schnittstelle aufrufen. Der Font ist dann unter dem Namen `ding` verfügbar (vgl. den original Zapf-Font in Tabelle 7.9 auf Seite 391).

```
\usepackage{pifont}

\Pisymbol{ding}{13} \Pisymbol{ding}{15} \Pisymbol{ding}{8}
\Pisymbol{ding}{17} \Pisymbol{ding}{19} \Pisymbol{ding}{9}
```

Bsp.
7-8-6



7.8.5 ifsym – Uhren, Wolken, Berge und andere Symbole

Mithilfe des `ifsym`-Paketes von Ingo Klöckl kann man auf einen Satz von mit METAFONT gestalteten Symbolfonts zugreifen. Sie sind derzeit nicht als Type1-Version verfügbar. Je nach den gewählten Paketoptionen stehen verschiedene Sätze von Symbolen zur Verfügung. Hier wird nur eine kleine

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x									"0x
'01x									
'02x									
'03x									"1x
'04x									
'05x									
'06x									"2x
'07x									
'10x									
'11x									"3x
'12x									
'13x									
'14x									"4x
'15x									
'16x									
'17x									"5x
'18x									
'19x									
'20x									"6x
'21x									
'22x									
'23x									"7x
'24x									
'25x									
'26x									"8x
'27x									
'28x									
'29x									"9x
'30x									
'31x									
'32x									"Ax
'33x									
'34x									
'35x									"Bx
'36x									
'37x									
'38x									"Cx
'39x									
'40x									
'41x									"Dx
'42x									
'43x									
'44x									"Ex
'45x									
'46x									
'47x									"Fx
'48x									
'49x									
'50x									"Gx
'51x									
'52x									
'53x									"Hx
'54x									
'55x									
'56x									"Ix
'57x									
'58x									
'59x									"Jx
'60x									
'61x									
'62x									"Kx
'63x									
'64x									
'65x									"Lx
'66x									
'67x									
'68x									"Mx
'69x									
'70x									
'71x									"Nx
'72x									
'73x									
'74x									"Ox
'75x									
'76x									
'77x									"Px
'78x									
'79x									
'80x									"Qx
'81x									
'82x									
'83x									"Rx
'84x									
'85x									
'86x									"Sx
'87x									
'88x									
'89x									"Tx
'90x									
'91x									
'92x									"Ux
'93x									
'94x									
'95x									"Vx
'96x									
'97x									
'98x									"Wx
'99x									
'00x									

Tabelle 7.21: Glyphen des METAFONT-Fonts bbding

Auswahl vorgestellt. Die vollständige Dokumentation (nur in deutscher Sprache) liegt der Distribution als PostScript-Datei `ifsym.ps` bei. Alle verfügbaren Symbole sind außerdem in [136] aufgeführt.

Die Option `clock` ermöglicht den Zugriff auf sieben Uhrensymbole. Sie stellt außerdem den Befehl `\showclock` zur Verfügung, der eine analoge Uhr anzeigt, deren Zeiger die richtige Uhrzeit wiedergeben. Seine beiden Argumente legen die Stunden (0-11) und die Minuten (0-59) fest. Die Anzeige wird jeweils auf volle Einheiten von fünf Minuten abgerundet; wenn man eine höhere Stundenzahl als 11 verwendet, verschwindet das Symbol ohne Warnung. Alle Symbole sind als normale und breite Serie vorhanden.

Normal: abgerundet:

Problem:

Feste Symbole:

```

\usepackage[clock]{ifsym}
Normal: \showclock{3}{20} abgerundet:
        \textbf{\showclock{6}{17}} \\
Problem: \showclock{16}{35}          \\
Feste Symbole: \Taschenuhr{} \StopWatchStart{}
               \StopWatchEnd{} \Interval{}

```

Die Option `weather` definiert 22 Wettersymbole, von denen einige in der ersten Zeile des nächsten Beispiels dargestellt sind. Der Befehl `\Thermo` zeigt

je nach Zahlenwert seines Argumentes (0–6) ein anderes Thermometersymbol an. Für Alpinisten oder Wanderer hält die Option `alpine` 17 Symbole zur Beschreibung von Routen oder für Landkarten bereit. Die Option `misc` bietet eine Reihe gemischter Symbole, von denen einige auch in anderen Fonts zu finden sind, und die Option `geometry` ermöglicht den Zugriff auf 30 geometrische Formen, von denen einige in der vierten Zeile des Beispiels erscheinen.

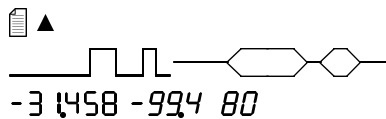


```
\usepackage[weather,alpine,misc,geometry]{ifsym}
```

```
\Sun\ \Rain\ \Snow\ \Lightning\ \SunCloud\ \Thermo{1}\Thermo{4}\\
\Summit\ \Mountain\ \Joch\ \Hut\ \Flag\ \Tent\ \Village\ \
\Cube{5}\StrokeFive\ \Radiation\ \Fire\ \Telephone\ \Letter\ \
\TriangleUp\ \RightDiamond\ \SquareShadowC\ \SpinUp\ \SpinDown
```

Bsp.
7-8-8

Mit dem Befehl `\textifsymbol` kann man über die Zeichensatzposition auf die Symbole zugreifen. Sein optionales Argument gibt den zu verwendenden Symbolfont an (Voreinstellung `ifsym`). Die Paketdokumentation enthält Zeichensatztabellen zu allen `ifsym`-Fonts. Besonders interessant ist der Befehl `\textifsym`, mit dessen Hilfe sich Impulsdigramme erzeugen lassen. Er kann auch zur Darstellung von Digitalziffern herangezogen werden (wobei `b` einen Leerraum korrekter Breite anzeigt).



```
\usepackage{ifsym}
```

```
\textifsymbol{3}\textifsymbol[ifgeo]{113}\par
\textifsym{LLL|H|L|h|l}\textifsym{MM<DD>m<d>M}\par
\textifsym{-31.458}\textit{\textifsym{-99.4b80}}
```

Bsp.
7-8-9

7.8.6 tipa – Symbole für das Internationale Phonetische Alphabet

Die TIPA-Sammlung [52] von Rei Fukui besteht aus einer Reihe von Zeichensätzen und einem zugehörigen Paket, mit deren Hilfe es möglich ist, in $\text{\LaTeX}$ phonetische Symbole zu setzen. TIPA enthält alle Symbole, einschließlich der diakritischen Zeichen, die in den Versionen des Internationalen Phonetischen Alphabets (IPA) von 1979, 1989, 1993 und 1996 festgelegt wurden. Abgesehen von den IPA-Symbolen, enthält TIPA Zeichen, die auch für andere Bereiche der Phonetik und der Linguistik von Nutzen sind, darunter die folgenden:

- Symbole der amerikanischen Phonetik, wie z.B., $\text{\textcircled{æ}}$, $\text{\textcircled{e}}$, $\text{\textcircled{ɔ}}$ und $\text{\textcircled{λ}}$;
- Symbole, die in der geschichtlichen Erforschung indogermanischer Sprachen Verwendung finden, wie z.B. $\text{\textcircled{p}}$, $\text{\textcircled{p}}$, $\text{\textcircled{h}}$, $\text{\textcircled{z}}$, $\text{\textcircled{ɛ}}$, $\text{\textcircled{ɛ}}$, und Akzente wie $\text{\textcircled{á}}$ und $\text{\textcircled{é}}$;
- Symbole zur phonetischen Darstellung ostasiatischer Sprachen, wie etwa $\text{\textcircled{ɿ}}$, $\text{\textcircled{ɿ}}$, $\text{\textcircled{ɿ}}$, $\text{\textcircled{ɿ}}$ (erfordert die Option `extra`);
- Diakritische Zeichen der *extIPA Symbols for Disordered Speech* (zur Abbildung von Sprachstörungen) sowie *VoQS (Voice Quality Symbols)* (zur Darstellung der Sprachqualität), beispielsweise, $\text{\textcircled{ñ}}$, $\text{\textcircled{f}}$ und $\text{\textcircled{m}}$ (erfordert Option `extra`).

ASCII	:	;	"		0	1	2	3	4	5	6	7	8	9
TIPA	:	˙	˘		h	i	Λ	3	u	ø	D	γ	θ	ə
ASCII	@	A	B	C	D	E	F	G	H	I	J	K	L	M
TIPA	ə	ɑ	β	ε	ð	ε	φ	γ	h	i	j	ʃ	ʌ	ŋ
ASCII	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
TIPA	ŋ	ɔ	ʔ	ʃ	r	ʃ	θ	u	v	w	x	y	z	

Tabelle 7.22: TIPA-Kürzel

Die IPA-Symbole sind in der L^AT_EX-Standardkodierung T3 kodiert, welche das `tipa`-Paket durch zusätzliche Makros unterstützt. Die Kodierung ist für die Schriftfamilien Computer Modern Roman, Sans und Typewriter (basierend auf den METAFONT-Entwürfen von Donald Knuth) verfügbar, sowie für Times Roman und Helvetica.

Genau genommen ist T3 eigentlich keine echte L^AT_EX-Textkodierung, da sie die sichtbaren ASCII-Zeichen nicht an ihren Standardpositionen enthält. Man kann jedoch den Standpunkt vertreten, dass phonetische Symbole eine eigene Sprache bilden, und für diese Sprache bietet das TIPA-System eine optimale Eingabeschnittstelle, in der Ziffern und Großbuchstaben in geeigneter Weise als Kürzel (siehe Tabelle 7.22) für die Eingabe gängiger phonetischer Symbole im Argument von `\textipa` oder der Umgebung `IPA` dienen. Alle phonetischen Symbole sind auch als Langformen verfügbar: Um ein ə zu erzeugen kann man z.B. `\textschwa` eingeben. Das nächste Beispiel zeigt das TIPA-System in einer Times- und Helvetica-Umgebung.

Bsp.
7-8-10

In linguistics, fəʊ'netɪk transcriptions are usually shown in square brackets, e.g., phonetics [fəʊ'nɛtɪks].

```
\usepackage{mathptmx,tipa}
```

```
In linguistics, f\textschwa\textupsilon
\textprimstress n\textepsilon t\i k transcriptions
are usually shown in square brackets, e.g.,
\textsf{phonetics \textipa{[f@U"nEtIks]}}.
```

TIPA verwendet `\*`, `\;`, `\:`, `\!` und `\|` als spezielle Makros, um die Eingabe phonetischer Symbole zu vereinfachen, für die es kein solches Kürzel gibt, wie sie zuvor erläutert wurden. In Standard-L^AT_EX sind diese bereits als Befehle für den Formelsatz definiert, durch das Laden des `tipa`-Paketes werden sie jedoch für Linguisten annektiert. Wenn dieser Effekt unerwünscht ist, kann man ihn mit der Option `safe` abstellen. Dann müssen die längeren Befehle verwendet werden, also z.B. `\textroundcap` anstelle von `\|c`. Die nächsten Zeilen enthalten einige komplexere Beispiele mit Ausgaben in Computer Modern Roman, Sans und Typewriter.

 *Undefinierte
Mathematikbefehle*

Bsp.
7-8-11

ŋOōō ŋOâ?ã

A) dɔg, B) kæt, C) maʊs

*kmtóm *bhrâtēr

```
\usepackage{tipa}
```

```
\begin{IPA}
```

```
\textrm{N\!o\{'\~*o}\~o \r*N\!o\^aP\~a } \par
```

```
\textsf{\!A) dOg, \!B) k\ae{}t, \!C) maUs} \par
```

```
\texttt{\!|c{k}\r\mt\!om *bhr\!at=er} \end{IPA}
```

Durch die Option `tone` stellt TIPA den Befehl `\tone` zur Verfügung, der „Betonungszeichen“ erzeugt. Der Befehl verfügt über ein Argument, das eine

Ziffernfolge zur Bezeichnung der Tonhöhen enthält, wobei 1 die niedrigste und 5 die höchste ist. Innerhalb dieses Bereiches sind beliebige Kombinationen in beliebiger Länge möglich. Dies wird in der letzten Zeile des nächsten Beispiels deutlich, das ansonsten zeigt, wie man mit dem Befehl `\tone` die vier Töne des Chinesischen darstellt.

<code>\ma</code> (Mutter)	<code>\ma</code> (Pferd)	<code>\usepackage[tone]{tipa}</code>	
<code>\ma</code> (Hanf)	<code>\ma</code> (schelten)	<code>\tone{55}ma</code> (Mutter)	<code>\tone{214}ma</code> (Pferd)
		<code>\tone{35}ma</code> (Hanf)	<code>\tone{51}ma</code> (schelten)
		<code>\tone{153325413}</code>	

Bsp.
7-8-12

Die gezeigten Beispiele vermitteln nur einen ersten Eindruck der Möglichkeiten von TIPA. Die zur TIPA-Distribution gehörige Datei `tipaman` enthält ein Handbuch, mit dessen Hilfe man sich den Funktionsumfang im Detail erschließen kann.

7.8.7 Das Euro-Zeichen (€)

Am 1. Januar 2002 wurde der Euro (€) zum offiziellen Zahlungsmittel in 12 Ländern der Europäischen Union.¹ Lange zuvor hatte die EU-Kommission bereits ein Logo für die neue europäische Währung in Auftrag gegeben. Die Kommission unterstützt nun auch den Einsatz von Zeichen, die an die jeweilige Schrift eines Dokumentes angepasst sind. In der Zwischenzeit haben viele Schriftenhersteller passend gestaltete Euro-Zeichen in ihre Fonts integriert. Es gibt jedoch immer noch viele Schriften, die nicht über ein Euro-Zeichen verfügen. Man kann z.B. nicht davon ausgehen, dass die PostScript-Standardschriften, die in den meisten Laserdruckern fest eingebaut sind, bereits Euro-Zeichen enthalten.

Der offizielle $\LaTeX$ -Befehl für das Euro-Zeichen, `\texteuro`, gehört zum `textcomp`-Paket. Viele Zeichensätze verfügen jedoch gar nicht über eine Euro-Glyphe. In diesem Fall versucht `textcomp` das Zeichen nachzubilden, indem es ein großes C mit zwei Querstrichen versieht (z.B. in Times Roman: €).

Die gängigen $\TeX$ -Schriften verfügen normalerweise über ein eigenes Euro-Symbol. Leider erscheint das Zeichen, das Jörg Knappen für die Computer Modern-Schriften (d.h. die in $\LaTeX$ voreingestellten Schriften) gestaltet hat, jedoch recht futuristisch:

Ein normaler €, ein kursiver €, ein fatter €, ein fettkursiver €. Zum Vergleich: Der serifenlose € und der Typewriter € in EC Fonts.

`\usepackage{textcomp}`

Ein normaler `\texteuro{}`, `\textit{ein kursiver \texteuro{}}`, `\textbf{ein fatter \texteuro{}}`, `\textbf{\itshape ein fettkursiver \texteuro{}}`. Zum Vergleich: Der `\textsf{serifenlose \texteuro{}}` und der `\texttt{Typewriter \texteuro{}}` in EC Fonts.

Bsp.
7-8-13

Bei den Computer Modern Bright-Familien sieht die Situation etwas besser aus. Auch wenn sie auf Basis der METAFONT-Daten der European

¹Genauer gesagt wurden an diesem Tag die entsprechenden Geldscheine und Münzen eingeführt.

Computer Modern-Fonts erzeugt wurden, wirkt ihr Euro-Zeichen angenehm, da diese Familien kaum Serifen enthalten.

Bsp.
7-8-14

Ein normaler €, *ein geneigter* €, **ein fetter** €, **ein fettgeneigter** €. Zum Vergleich: Der Typewriter € ganz in CM Bright.

```
\usepackage{cmbright,textcomp}
```

Ein normaler `\texteuro{}`, `\textsl{ein geneigter \texteuro{}}`, `\textbf{ein fetter \texteuro{}}`, `\textbf{\slshape ein fettgeneigter \texteuro{}}`. Zum Vergleich: Der `\texttt{Typewriter \texteuro{}}` ganz in CM Bright.

Aber was kann man tun, wenn die Fonts für ein Dokument nicht über das Zeichen verfügen? In diesem Fall kann man entweder gesonderte Symbolfonts einsetzen, die ein generisches Euro-Zeichen enthalten (das neutral gestaltet ist, so dass es sich mit vielen Schriften verträgt), oder Symbolfonts, die speziell für bestimmte Textfont-Familien entworfen wurden. Auf jeden Fall sollte das Symbol in verschiedenen Schriftserien und -graden verschiedener Stärken (und Weiten) verfügbar sein, so dass es beim Setzen unterschiedlichen Anforderungen gerecht werden kann (z.B. einer Überschrift, wie derjenigen für diesen Abschnitt).

eurosym – Euros für L^AT_EX

Der erste Satz von T_EX-Fonts mit einem generischen Euro-Zeichen waren wahrscheinlich die EuroSym-Fonts von Henrik Theiling. Sie sind als METAFONT-Quellen sowie als PostScript Type 1-Outlines erhältlich und enthalten ein Euro-Symbol, das den offiziellen Vorgaben entsprechend gestaltet ist. Als nette Beigabe enthalten die Fonts an der Position null ein Bild der typographischen Konstruktion. Allen, die schon immer mal wissen wollten, wie das Zeichen gestaltet sein soll, mag hiermit das folgende Beispiel Aufschluss geben:

Bsp.
7-8-15



```
\usepackage{eurosym}
```

```
\fontsize{40}{40}\usefont{U}{eurosym}{m}{n}\symbol{0}
```

Das Paket `eurosym`, mit dessen Hilfe man auf diese Fonts zugreifen kann, definiert den Befehl `\euro`. Dieser Befehl ist so voreingestellt, dass er das offizielle Symbol in einer Form erzeugt, die sich an der Schriftserie und den Formattributen der aktuellen Grundschrift orientiert. Tabelle 7.23 auf der nächsten Seite zeigt die Möglichkeiten im Überblick.

Richtige Euros

Bsp.
7-8-16

Ein normaler €, *ein geneigter* €, **ein fetter** € und *ein fettkursiver* €.

```
\usepackage{eurosym}
```

Ein normaler `\euro{}`, `\textsl{ein geneigter \euro{}}`, `\textbf{ein fetter \euro{}}` und `\textbf{\itshape ein fettkursiver \euro{}}`.

Alternativ verfügt das Paket über Befehle, mit deren Hilfe man das Euro-Zeichen aus dem „C“ des aktuellen Fonts und zwei Querbalken (die in drei Breiten vorliegen) zusammensetzen kann. Das nächste Beispiel zeigt, dass die Ergebnisse dabei von unbrauchbar bis mehr oder weniger geeignet reichen, je nach Form des verwendeten „C“ und der gewählten Balkenbreite. In jedem Fall

Der Euro des armen Mannes

Familie	Serie	Form(en)	Schriftbild
<i>EuroSym von Henrik Theiling (U)</i>			
eurosym	m	n, (it), sl, ol	Normal und Umriss: €, €
eurosym	(b), bx	n, (it), sl, ol	breitfett aufrecht und geneigt: €, €

Tabelle 7.23: Klassifizierung der EuroSym Fontfamilie

ist ein Euro-Zeichen, das eigens für eine Schrift gestaltet wurde, vorzuziehen und sollte – sofern verfügbar – auch verwendet werden.

```
\usepackage{times,eurosym}
```

€, €, € (Times)	<code>\rmfamily \geneuro, \geneuronarrow, \geneurowide</code> (Times)	<code>\</code>
€, €, € (Helvetica)	<code>\sffamily \geneuro, \geneuronarrow, \geneurowide</code> (Helvetica)	<code>\</code>
€, €, € (Courier)	<code>\ttfamily \geneuro, \geneuronarrow, \geneurowide</code> (Courier)	

Bsp.
7-8-17

Über die Paketoptionen `gen`, `gennarrow` und `genwide` kann man den `\euro`-Befehl so ändern, dass er auf `\geneuro`, `\geneuronarrow` bzw. `\geneurowide` verweist. Dabei kann das offizielle Euro-Symbol immer noch mit dem Befehl `\officialeguro` aufgerufen werden.

Und schließlich bietet das Paket den praktischen Befehl `\EUR` um Geldbeträge mit einem Euro-Zeichen zu versehen, das durch einen schmalen Leer-
raum abgesetzt ist.¹ Da entsprechend der jeweiligen Konventionen die Währungszeichen in verschiedenen Ländern unterschiedlich positioniert werden, erkennt das Paket die Optionen `left` (Voreinstellung) und `right`.

```
\usepackage[right]{eurosym}
```

Das Buch kostet 19,60€ im Handel.

Das Buch kostet `\EUR{19,60}` im Handel.

Bsp.
7-8-18

Eine weitere Möglichkeit, Währungssymbole zu formatieren, stellt das Paket `euro` dar, das auf Seite 103 beschrieben wurde.

Die Adobe Euro Fonts

Auch Adobe bietet einen Satz von Type1-Fonts mit Euro-Zeichen an. Dieser Fontsatz enthält Varianten mit und ohne Serifen (letztere sehr nah am offiziellen Logo) sowie in Schreibmaschinenschrift. Alle Varianten sind in aufrechter und kursiver Form sowie in normaler und fetter Schriftstärke verfügbar. Um diese Fonts nutzen zu können, benötigt man einen PostScript-Drucker, oder allgemeiner ausgedrückt, einen Drucker, der solche Schriften wiedergeben kann (z.B. mithilfe des Programms `ghostscript`).

Die Fonts dürfen zwar zum Drucken frei verwendet werden, Adobe lässt jedoch keine generelle Verbreitung oder Bündelung mit einer T_EX-Distribution zu. Daher müssen sie eigens von der Adobe-Website heruntergeladen werden: `ftp://ftp.adobe.com/pub/adobe/type/win/all/eurofont.exe`. Bei der Datei handelt es sich um ein selbstentpackendes Archiv für Windows. Unter Unix lässt die Datei sich mit dem Programm `unzip` entpacken.

¹Einige andere Pakete verwenden diesen Befehlsnamen für das eigentliche Euro-Zeichen – eine bedauerliche Unstimmigkeit.

Familie	Serie	Form(en)	PostScript Fontnamen und Beispiele
<i>Adobe EuroSans (U)</i>			
eurosans	m	n, it, (sl)	EuroSans-Regular (zpeurs), EuroSans-Italic (zpeuris) €, €
eurosans	b, (bx)	n, it, (sl)	EuroSans-Bold (zpeubs), EuroSans-BoldItalic (zpeubis) € , €
eurosans	mc	n, it, (sl)	EuroMono-Regular (zpeurt), EuroMono-Italic (zpeurit) €, €
eurosans	(sbc), bc	n, it, (sl)	EuroMono-Bold (zpeubt), EuroMono-BoldItalic (zpeubit) € , €

Tabelle 7.24: Klassifizierung der Adobe Euro Fontfamilien (eurosans Klassifizierung)

Nach dem Herunterladen muss man die Fonts entsprechend der Namenskonventionen von Karl Berry [19] umbenennen und, falls erforderlich, weitere Hilfsdateien für L^AT_EX installieren, wie .fd-Dateien, eine Zuordnungsdatei für dvips sowie ein Paket, das in Dokumenten den Zugriff auf die Schriften ermöglicht. Je nach T_EX-Installation (zum Beispiel der T_EX Live-CD) sind diese Dateien möglicherweise auch schon vorhanden. Ansonsten können sie von CTAN:fonts/euro heruntergeladen werden.

eurosans – Eine Art mit Adobe an Euros zu kommen

Diverse L^AT_EX-Pakete erlauben den Zugriff auf die Adobe Euro Fonts, wobei alle unterschiedliche Strategien verwenden. Wie der Name schon andeutet, kann man mit dem Paket eurosans von Walter Schmidt nur auf die EuroSans Fonts von Adobe zugreifen (siehe Tabelle 7.24). Der Grund hierfür liegt darin, dass die Varianten mit Serifen selten zu einer Grundschrift passen, während die neutraleren serifenlosen Entwürfe zu den meisten Schriften passen, außer zu Schreibmaschinenschriften. Da es sich bei den EuroMono-Schriften von Adobe eigentlich nur um schmale Versionen von EuroSans handelt, wurden sie als schmale (condensed) Schriftserie (NFSS-Klassifizierungen mc, bc und sbc) in das Paket integriert. Ihre Stärke (normal oder fett), Form (aufrecht oder geneigt) und Weite (normal oder schmal) variieren je nach den Umgebungsbedingungen im Dokument.

Ein wichtiger Aspekt dieses Paketes (der bei anderen Paketen fehlt) ist die Möglichkeit, die Fonts mit der Option `scaled` um einen bestimmten Faktor zu skalieren. Es ist so voreingestellt, dass es die Fonts auf 95% ihrer Nominalgröße reduziert. Wenn zur Anpassung an die Grundschrift ein anderer Faktor erforderlich ist, kann man diesen, wie im nächsten Beispiel, explizit festlegen.

Ein normales €-Zeichen,
ein kursiver €, ein **fetter €**
und ein **fettkursiver €**.

Ein normales €-Zeichen,
ein kursiver €, ein **fetter €**
und ein **fettkursiver €**.

```
\usepackage{lucidabr} \usepackage[scaled=0.97]{eurosans}
Ein normales \euro-Zeichen, \textit{ein kursiver \euro},
\textbf{ein fetter \euro} und
\textit{ein fettkursiver \euro}}.

\ssfamly
Ein normales \euro-Zeichen, \textit{ein kursiver \euro},
\textbf{ein fetter \euro} und
\textit{ein fettkursiver \euro}}.
```


Variationen einschränken Die Anzahl der erzeugten Varianten lässt sich durch Umdenken des \euro-Befehls reduzieren (beispielsweise, indem man bei verschiedenen Fontserien immer die gleiche Schriftform verwendet).

Ein normales €,
Zeichen, *kein kursiver* €, **ein fatter** €
und *kein fettkursiver* €.

```
\usepackage{lucladabr} \usepackage[scaled=0.97]{eurosans}
\DeclareRobustCommand{\euro}{\fontencoding{U}%
\fontfamily{eurosans}\fontshape{n}\selectfont E}}
Ein normales \euro{}-Zeichen, \textit{kein kursiver \euro},
\textbf{ein fetter \euro} und \texttt{kein fettkursiver \euro}.
```

Bsp.
7-8-20

Wenn man kein Euro-Zeichen mit Serifen benötigt, ist das eurosans-Paket normalerweise anderen Lösungen vorzuziehen, da es die vollständigste Auswahl an Schriftserien bietet und außerdem die Skalierung der Fonts unterstützt. In der Paketdokumentation ist zudem beschrieben, wie man die Fonts und – falls erforderlich – die Hilfsdateien installiert.

europs – Eine andere Art mit Adobe an Euros zu kommen

Das Paket `europs` von Jörn Clausen verfolgt einen anderen Ansatz. Es stellt den Befehl `\EUR` zur Verfügung, um auf die Adobe Euro Fonts zuzugreifen. Wie im nächsten Beispiel ersichtlich wählt es, je nach den Fontattributen des umgebenden Textes, ein anderes Symbol aus.

[illegible]

Bsp.
7-8-21

Da diese automatische Änderung nicht immer erwünscht ist (z.B. passt das Euro-Zeichen mit Serifen möglicherweise nicht zu der entsprechenden Grundschrift), bietet das Paket außerdem die Befehle `\EURtm` (mit Serifen), `\EURhv` (serifenlos), `\EURcr` (dicktengleich) – die Namen sind den drei PostScript-Schriften Times, Helvetica und Courier nachempfunden. Diese Befehle fixieren die Schriftfamilie, wobei sie jedoch weiterhin zwischen fetten und geneigten Formen variieren. Wie die letzte Zeile des vorigen Beispiels zeigt, passt jedoch keines der Zeichen wirklich gut zu diesen Schriften. Schließlich enthält das Paket noch den Befehl `\EURofc`, der das offizielle Euro-Zeichen erzeugt (d.h. eines aus der serifenlosen normalen Schrift).

marvosym – Wiedersehen mit barer Münze

Ein weiterer kostenloser PostScript-Font, der Glyphen für Euro-Zeichen enthält, ist der MarVoSym-Font, der bereits in Abschnitt 7.8.3 auf Seite 412 beschrieben wurde. Es sind drei Formen verfügbar, die an Times, Helvetica bzw. Courier angepasst sind. Da es sich um einen Pi-Font handelt, verfügt er nur

über eine Schriftstärke, so dass sein Nutzen als Quelle für das Euro-Zeichen relativ begrenzt ist. Der Font enthält zwei Glyphen im offiziellen Euro-Design mit unterschiedlichen Vor- und Nachbreiten. Um diesen Unterschied deutlicher zu machen, sind die Glyphen im nächsten Beispiel von einem Rahmen umgeben. Das Beispiel zeigt auch die anderen Währungssymbole des Paketes.

	<code>\usepackage{times,marvosym}</code>
Währungen: β , $\mathfrak{S}$, $\mathfrak{W}$, $\mathfrak{S}$, $\mathfrak{S}$, $\mathfrak{S}$	Währungen: <code>\Shilling</code> , <code>\Denarius</code> , <code>\Pfund</code> , <code>\EyesDollar</code> , <code>\EURtm</code> , <code>\</code>
Vergleiche: C €, C €, C €	Vergleiche: C <code>\EURtm</code> , <code>\textsf{C}</code> <code>\EURhv</code> , <code>\texttt{C}</code> <code>\EURcr</code> <code>\</code>
Offizielle Logos:	Offizielle Logos: <code>\</code>
€ oder €	<code>\Large</code> <code>\frame{\EUR}</code> oder <code>\frame{\EURdig}</code>

Bsp.
7-8-22

7.9 Die Low-Level-Schnittstelle

Während die High-Level-Befehle für die Verwendung in Dokumenten ausgelegt sind, werden Low-Level-Befehle hauptsächlich für die Definition neuer Befehle in Paketen oder in der Präambel verwendet; siehe auch Abschnitt 7.9.4. Zur optimalen Nutzung dieser Fontbefehle ist es sehr hilfreich, die interne Fontorganisation in $\LaTeX$ (NFSS) zu verstehen.

Eines der Ziele des NFSS ist, durch Algorithmen, die den Prinzipien des generischen Markups folgen, eine rationale Zeichensatzauswahl zu ermöglichen. Zu diesem Zweck wäre es wünschenswert, möglichst viele Zeichensatzattribute unabhängig voneinander verändern zu können. Die meisten Schriftfamilien verfügen jedoch nur über einen Bruchteil der vorstellbaren Schriftschnitte. Entsprechend würden zu viele unabhängig voneinander einstellbare Zeichensatzattribute eine zu hohe Anzahl von Kombinationen nach sich ziehen, für die keine (externen) Zeichensätze existieren. In diesem Fall würde ersatzweise ein anderer Zeichensatz geladen, der sich möglicherweise stark von dem gewünschten unterscheidet.

Intern verwaltet $\LaTeX$ fünf voneinander unabhängige Zeichensatzattribute: die jeweils aktuelle „Kodierung“, „Familie“, „Serie“, „Form“ und den aktuellen „Schriftgrad“. Das Kodierungsattribut wurde in Version 2 von NFSS eingeführt, als sich abzeichnete, dass viele Sprachen erst dann richtig unterstützt werden können, wenn die Zeichenkodierung von den anderen Zeichensatzattributen unabhängig ist.

Die Werte dieser Attribute bestimmen, welcher Zeichensatz jeweils verwendet wird. Außerdem verwaltet $\LaTeX$ eine große Anzahl von Tabellen, anhand derer die Attributkombinationen mit passenden externen Zeichensätzen verknüpft werden (das heißt mit den `.tfm`-Dateien, welche die für $(\LaTeX)$ notwendigen Informationen enthalten). Die Auswahl des Zeichensatzes in $\LaTeX$ erfolgt dann in zwei Schritten:

1. Einige der Attribute werden über die Befehle `\fontencoding`, `\fontfamily`, `\fontseries`, `\fontshape` und `\fontsize` geändert.
2. Der Zeichensatz, der dieser neuen Kombination entspricht, wird mit dem Befehl `\selectfont` geladen.

Der zweite Schritt umfasst gleich mehrere Aktionen. Zunächst prüft $\LaTeX$, ob der Font mit den gewünschten Attributen dem System bereits bekannt ist (d.h.

ob die `.tfm`-Datei schon geladen ist). In diesem Fall wird der entsprechende Zeichensatz ausgewählt. Andernfalls werden die internen Tabellen nach dem Namen des externen Zeichensatzes durchsucht, der diesen Einstellungen zugeordnet ist. Wird ein passender Zeichensatz gefunden, dann lädt $\text{\LaTeX}$ die dazugehörige `.tfm`-Datei in den Speicher und wählt danach diesen Zeichensatz aus. Wenn dieser Ablauf zu keinem Ergebnis führt, dann sucht $\text{\LaTeX}$, wie in Abschnitt 7.9.3 erklärt, nach einem Ersatzfont.

7.9.1 Setzen einzelner Zeichensatzattribute

Zu jedem Zeichensatzattribut gehört ein Befehl, mit dem sein aktueller Wert geändert werden kann. Das Argument für diesen Befehl kann praktisch jede beliebige Zeichenfolge sein, es sind jedoch nur wenige sinnvoll. Diese Werte sind kein fester Bestandteil des NFSS, sondern vielmehr Konventionen, die in den internen Tabellen festgehalten werden. In den folgenden Abschnitten werden die Namenskonventionen vorgestellt, die von Standard- $\text{\LaTeX}$ verwendet werden. Es steht jedoch jedem Anwender frei, den internen Tabellen eigene Fontdeklarationen hinzuzufügen. Beim Einbinden neuer Zeichensätze sollten die bestehenden Konventionen jedoch unbedingt beachtet werden, denn nur eine konsistente Benennung garantiert, dass in einem generisch markierten Dokument die passenden Zeichensätze ausgewählt werden.

Um über diese Schnittstelle einen bestimmten Schriftschnitt auszuwählen – z.B. Computer Modern Dunhill schmal fett kursiv 14pt – reicht es nicht aus, die Namenskonventionen zu kennen, da nicht für jede Kombination von Attributen auch ein externer Font existiert. Man könnte natürlich mit einem Befehlssatz wie dem folgenden sein Glück versuchen:

```
\fontencoding{OT1}\fontfamily{cmdh}\fontseries{bc}%
\fontshape{it}\fontsize{14}{16pt}\selectfont
```

Wie die nächsten Abschnitte noch zeigen werden, ist diese Eingabe unter dem Gesichtspunkt der Namenskonventionen völlig korrekt. Da jedoch kein externer Zeichensatz für diese Kombination existiert, müsste $\text{\LaTeX}$ ihn durch einen anderen ersetzen. Beim Ersetzungsvorgang kann so letztendlich ein Zeichensatz geladen werden, der sich von dem gewünschten erheblich unterscheidet. Man sollte deshalb in den Fonttabellen (`.fd`-Dateien) nachprüfen, ob die gewünschte Kombination existiert. Abschnitt 7.9.3 erklärt die näheren Einzelheiten des Ersetzungsvorgangs.

Auswahl der Schriftfamilie

Die Schriftfamilie wird mit dem Befehl `\fontfamily` festgelegt. Sein Argument bezeichnet die Schriftfamilie, so wie sie in den internen Tabellen deklariert ist. Die Zeichenfolge wurde beim Einrichten der Tabellen festgelegt und besteht in der Regel aus wenigen Buchstaben, wie z.B. `cmr` für die Schriftfamilie Computer Modern Roman. Die Bezeichnung für eine Schriftfamilie sollte nicht mehr als fünf Buchstaben umfassen, da sie zusammen mit bis zu drei weiteren Buchstaben als Dateiname dient, der ja in einigen Systemen nicht länger als acht Buchstaben sein darf.

Schriftstärken			Schriftbreiten			
Ultraleicht	Ultra Light	ul	Ultraschmal	Ultra Condensed	50%	uc
Extraleicht	Extra Light	el	Extraschmal	Extra Condensed	62.5%	ec
Leicht	Light	l	Schmal	Condensed	75%	c
Mager	Semi Light	sl	Halbschmal	Semi Condensed	87.5%	sc
Normal (Buch)	Medium (normal)	m	Normal	Medium	100%	m
Halbfett	Semi Bold	sb	Halbbreit	Semi Expanded	112.5%	sx
Fett	Bold	b	Breit	Expanded	125%	x
Extrafett	Extra Bold	eb	Extrabreit	Extra Expanded	150%	ex
Ultrafett	Ultra Bold	ub	Ultrabreit	Ultra Expanded	200%	ux

Die Prozentangaben sind ungefähre Werte

Tabelle 7.25: Klassifizierung von Schriftstärken und -breiten

Auswahl der Schriftserie

Das Attribut für die Auswahl der Schriftserie wird mit `\fontseries` verändert. Mit diesem Attribut wird sowohl die Schriftbreite als auch die Schriftstärke eingestellt, d.h. es ist nicht möglich, die Breite einer Schrift unabhängig von ihrer Stärke zu verändern. Diese Kombination rührt daher, dass es kaum jemals erforderlich ist, einen der Werte separat zu behandeln. Ganz im Gegenteil: In den Spezifikationen der Schriftgestalter geht eine Änderung der Schriftstärke, z.B. von normal zu fett, häufig mit einer Änderung der Weite, z.B. von normal zu breit, einher. Das ist nicht weiter überraschend, da Änderungen in der Schriftstärke das horizontale Erscheinungsbild der Zeichen so verändern, dass eine Anpassung der Schriftbreite erforderlich wird, damit ein ausgewogenes Schriftbild entsteht.

Jedes Argument des Befehls `\fontseries`, das den Namenskonventionen folgt, ist aus Abkürzungen von Schriftbreite und Schriftstärke zusammengesetzt, die so gewählt sind, dass jede Kombination eindeutig ist (siehe Tabelle 7.25). Diese Klassifizierungen werden im Argument von `\fontseries` kombiniert. Die Abkürzung `m` (für normale Stärke oder Breite) wird jedoch (wie im deutschen Sprachgebrauch) weggelassen, es sei denn der Wert `normal` wurde für beide Eigenschaften gewählt. In diesem Fall ist das Argument ein einzelnes `m`. Breitfett (`bold expanded`) wäre also beispielsweise `bx`, breit (`medium expanded`) wäre `x` und fett (`bold medium`) wäre `b`.

Auswahl der Schriftform

Die Schriftform wird mit dem Befehl `\fontshape` verändert. Die Standardformen werden durch ein- und zweibuchstabige Abkürzungen gekennzeichnet. Diese sind in Tabelle 7.26 auf der nächsten Seite aufgelistet, zusammen mit einem Beispiel, wie diese Form in Computer Modern Roman aussieht.¹

¹Die `ol`-Form wurde jedoch mithilfe von `\pcharpath`-Befehlen aus dem `pst-char`-Paket erzeugt, da Computer Modern über keine derartige Form verfügt. Solche graphischen Manipulationen werden in [58] besprochen.

Abkürzung	Beschreibung	
n	upright (or normal)	aufrecht (oder normal)
it	<i>italic</i>	kursiv
sl	<i>slanted or oblique</i>	geneigt oder oblique
sc	SMALL CAPS	Kapitälchen
ui	upright <i>italic</i>	kursiv aufrecht
ol	OUTLINE	UMRISS

Tabelle 7.26: Klassifizierung der Schriftformen

Auswahl des Schriftgrades

Der Schriftgrad wird mit dem Befehl `\fontsize{⟨größe⟩}{⟨vorschub⟩}` festgelegt. Dieser Befehl verfügt als einziger Attributbefehl über zwei Argumente: `⟨größe⟩` für die Schriftgröße und `⟨vorschub⟩` für den Zeilenvorschub (der Abstand von Grundlinie zu Grundlinie bei diesem Schriftgrad).

*Zeilenabstand,
-vorschub und
Durchschuss*

Zu Zeiten des Bleisatzes bezeichnete man die Distanz zwischen der Unterkante einer Zeile und der Oberkante einer Folgezeile als Zeilenabstand bzw. Durchschuss. Letzterer Begriff stammt aus der Bleisatztechnik: Man erzeugte einen visuell gleichmäßig erscheinenden Zeilenfall, indem man so genannte Regletten unterschiedlicher Stärke zwischen die Zeilen legte; ein Vorgang der in der Druckersprache „durchschießen“ heißt.¹ Mit der Einführung des rechnergesteuerten Fotosatzes kam der Begriff Zeilenvorschub auf, dessen Wert sich aus der Größe des Schriftgrades plus des eingestellten Zeilenabstands errechnete. Der Zeilenabstand wechselte mit dieser Technik von einer (variablen) visuellen zu einer festen Größe, die nicht mehr vom Inhalt der Zeilen bestimmt wurde.

Schriftgrad und Zeilenvorschub werden normalerweise in Punkten angegeben, wobei die Einheiten hier gemäß der Konventionen weggelassen werden. Wenn es sich bei dem Zeilenvorschub jedoch um eine elastische Größe handelt, d.h. wenn sie plus- oder minus-Anteile enthält, muss die Einheit mit angegeben werden. Eine gültige Einstellung für den Schriftgrad könnte also folgendermaßen aussehen:

```
\fontsize{14.4}{17}\selectfont
```

Doch auch wenn dieser Befehl eigentlich korrekt ist, kann es vorkommen, dass kein externer Zeichensatz in dem gewünschten Schriftgrad vorliegt. Sofern die internen Tabellen eine Größenkorrektur erlauben, versucht L^AT_EX dann eine benachbarte Schriftgröße zu finden oder gibt ansonsten eine Fehlermeldung aus. Bei skalierbaren Zeichensätzen (wie z.B. PostScript-Fonts) kann man natürlich jeden beliebigen Schriftgrad auswählen. Das Beispiel

```
\fontsize{1in}{1.2in}\selectfont Happy Birthday
```

¹Intern arbeitet T_EX genau nach diesem Prinzip: Es berechnet den benötigten Durchschuss zwischen zwei Zeilen aus deren Unter- bzw. Oberlängen und dem gewünschten Zeilenvorschub (`\baselineskip`).

erzeugt einen Geburtstagsgruß in einem Schriftgrad von einem Zoll. Doch auch mit beliebigen Schriftgraden gibt es ein Problem: Wenn $\text{\LaTeX}$ eine Formel in dieser Größe setzen muss (was ohne Wissen des Anwenders hinter den Kulissen geschehen kann), müssen alle für Formeln benötigten Fonts für die neue Größe berechnet werden. Bei einer noch nie verwendeten Größe betrifft dies normalerweise die Schriftgrade für tiefgestellte Zeichen erster und zweiter Ordnung (mindestens 12 verschiedene Fonts). Dazu müssen wahrscheinlich viele neue Zeichensätze geladen werden – was sich durch einen Blick in die Protokolldatei feststellen lässt. So kann es leicht passieren, dass man an systeminterne Grenzen stößt, wenn man in einem Dokument zu viele unterschiedliche Schriftgrade verwendet. In diesem Falle sollte man $\text{\LaTeX}$ mithilfe der Deklaration `\DeclareMathSizes` vorschreiben, welche Schriftgrade für Formeln geladen werden sollen, anstatt sich auf den internen Algorithmus zu verlassen. Abschnitt 7.10.7 enthält weitere Informationen zu diesem Thema.

Auswahl der Kodierung

Die Kodierung wird mit dem Befehl `\fontencoding` verändert, dessen Argument die interne Bezeichnung für das gewünschte Kodierschema ist. Diese Bezeichnung muss $\text{\LaTeX}$ bekannt sein, entweder als eine der vordefinierten Kodierungen, (die vom $\text{\LaTeX}$ -Kern geladen werden) oder als Kodierung, die mit dem Befehl `\DeclareFontEncoding` deklariert wurde (siehe Abschnitt 7.10.5). Eine Reihe von Bezeichnungen für Standardkodierungen sind in Tabelle 7.27 auf der nächsten Seite aufgeführt.

Das $\text{\LaTeX}$ -Fontauswahlverfahren geht von der (idealisierten) Annahme aus, dass die meisten (oder besser noch alle) Fonts für normalen Text dieselbe Kodierung haben, solange man Text der gleichen Sprache setzt. Mit anderen Worten sollte man die Kodierung nur ändern müssen, wenn man in eine andere Sprache wechselt. In diesem Fall sollten normalerweise die Unterstützungspakete für die jeweilige Sprache (z.B. die des babel-Systems) alle erforderlichen Einstellungen im Hintergrund vornehmen.

Im folgenden Beispiel wird die Kodierung manuell geändert, und zwar durch Definieren einer Umgebung `Cyr` für kyrillische Texte. Diese Umgebung ändert lokal sowohl die Fontkodierung als auch die Eingabekodierung. Das hört sich vielleicht zunächst etwas seltsam an; wenn man jedoch einen Editor oder eine Tastatur benutzt, die einfach zwischendurch die Eingabekodierung wechseln können, hat man möglicherweise seine Texte genau auf diese Art gespeichert. Damit die Sprache richtig unterstützt wird, wären natürlich noch weitere Anpassungen erforderlich, wie z.B. bei den Regeln für die Silbentrennung. Die Kodierungen werden $\text{\LaTeX}$ durch Laden des Paketes `fontenc` bekannt gemacht. `T2A` ist eine der Standardkodierungen für die kyrillische Sprache; `T1` wird zum Schluss geladen und damit zur voreingestellten Kodierung für das Dokument.

Русский язык heißt
auf Deutsch: die
russische Sprache.

```
\usepackage[T2A,T1]{fontenc}
\usepackage[koi8-r,ansinew]{inputenc}
\newenvironment{Cyr}{\inputencoding{koi8-r}%
                    \fontencoding{T2A}\selectfont}{\end{Cyr}}
\raggedright \begin{Cyr}òôóóÉÉÉ ÑÛÛ\end{Cyr}
heißt auf Deutsch: die russische Sprache.
```

Bsp.
7-9-1

Kodierung	Beschreibung	Deklariert von
T1	L ^A T _E X-Textkodierung (Lateinisch) alias „Cork“-Kodierung	L ^A T _E X
TS1	L ^A T _E X-Symbolkodierung (Lateinisch)	L ^A T _E X
T2A,B,C	L ^A T _E X-Textkodierungen (Kyrillisch)	cyrillic-Erweit.
T3	L ^A T _E X-Kodierung für phonetisches Alphabet	tipa-Paket
TS3	L ^A T _E X-Kodierung für phonetisches Alphabet (Erweiterung)	tipa-Paket
T4	L ^A T _E X-Textkodierung (afrikanische Sprachen)	—
T5	L ^A T _E X-Textkodierung (Vietnamesisch)	—
T7	L ^A T _E X-Textkodierung (reserviert für Griechisch)	—
OT1	T _E X-Text nach Definition von Donald Knuth	L ^A T _E X
OT2	T _E X-Text für kyrillische Sprachen (veraltet)	cyrillic-Erweit.
OT3	T _E X-Kodierung für phonetisches Alphabet (veraltet)	—
OT4	T _E X-Text mit Erweiterungen für Polnisch	—
OT6	T _E X-Text mit Erweiterungen für Armenisch	—
OML	T _E X-Formeltext (kursiv) von Donald Knuth	L ^A T _E X
OMS	Mathematische Symbole von Donald Knuth	L ^A T _E X
OMX	Erweiterte mathematische Symbole von Donald Knuth	L ^A T _E X
X2	Erweiterte Textkodierung (Kyrillisch)	cyrillic-Erweit.
U	Unbekannte Kodierung (für beliebigen Kram)	L ^A T _E X
L. .	Lokale Kodierung (für eigene Kodierungen)	—
LV1	Kodierung von einigen V _T E _X -Fonts	MicroPress
LY1	Alternative zur T1-Kodierung	Y&Y

Tabelle 7.27: Standardfontkodierungen unter L^AT_EX

Potentielle Probleme mit der T1-Kodierung

Leider lässt sich die T1-Kodierung für die meisten PostScript-Fonts nicht zu hundert Prozent anwenden. Die folgenden fünf Zeichen erscheinen gerne als Tintenkleckse (was auf fehlende Glyphen im Zeichensatz hindeutet). Man beachte: Promille- und Pro-Zehntausend-Zeichen werden aus dem Prozentzeichen und einer oder zwei weiteren kleinen Nullen zusammengesetzt; nur letztere Glyphe fehlt.

```

\usepackage[T1]{fontenc}
\fontfamily{cmr}\selectfont
\j{} \ng{} \NG{} \textperthousand{} \textpertenthousand \par
\fontfamily{ptm}\selectfont
\j{} \ng{} \NG{} \textperthousand{} \textpertenthousand{}

```

j ŋ Ñ ‰ ‰‰
■ ■ ■ ‰ ■

Bsp.
7-9-2

Wie bereits in Abschnitt 7.5.4 auf Seite 373 erklärt, ist die Lage bei TS1 sogar noch schlechter, da in manchen PostScript-Fonts die Hälfte der Glyphen dieser Kodierung nicht verfügbar sind.

7.9.2 Setzen mehrerer Zeichensatzattribute

Bei der Entwicklung eines Seitenlayouts (siehe Abschnitt 4.4) oder layoutbezogener Befehle möchte man häufig einen bestimmten Schriftschnitt benutzen, d.h. die Werte aller Attribute festlegen. Für diese Aufgabe stellt L^AT_EX

den Befehl `\usefont` bereit, der über vier Argumente verfügt: Kodierung, Schriftfamilie, Schriftserie und Schriftform. Er aktualisiert die Attributwerte und ruft dann den Befehl `\selectfont` auf. Wenn zusätzlich die Schriftgröße und der Zeilenvorschub festgelegt werden sollen, stellt man ihm noch einen `\fontsize`-Befehl voran, wie z.B.:

```
\fontsize{14}{16pt}\usefont{OT1}{cmdh}{bc}{it}
```

Das Ergebnis entspricht dem auf Seite 424 dargestellten, hypothetischen Beispiel.

Neben `\usefont` verfügt $\text{\LaTeX}$ über die Deklaration `\DeclareFixedFont`, mit der sich neue Befehle definieren lassen, die dann auf einen ganz bestimmten, festen Zeichensatz umschalten. Solche Befehle sind extrem schnell, da sie nicht erst in internen Tabellen nachschlagen müssen. Daher sind sie sehr nützlich für Befehlsfolgen, die zwischen festen Zeichensätzen hin- und herschalten müssen. Im `doc`-Paket (siehe Kapitel 14) könnte man z.B. mit den folgenden Definitionen Zeilennummern für den Programmcode erzeugen:

```
\DeclareFixedFont\CodelineFont{\encodingdefault}{\familydefault}
                                {\seriesdefault}{\shapedefault}{7pt}
\newcommand\theCodelineNo{\CodelineFont\arabic{CodelineNo}}
```

Wie das Beispiel zeigt, besitzt `\DeclareFixedFont` sechs Argumente: den Namen des neuen Befehls, gefolgt von den fünf Zeichensatzattributen der NFSS-Klassifizierung. Anstelle fester Werte (außer für den Schriftgrad) werden hier die Voreinstellungen für die Grundschrift zugewiesen (siehe auch Abschnitt 7.3.5). Dadurch orientiert sich der Befehl `\CodelineFont` aus dem vorigen Beispiel immer noch am Gesamtlayout des Dokumentes (über die Einstellungen für `\encodingdefault` und die weiteren Parameter). Sobald seine Definition jedoch abgeschlossen ist, bleibt seine Bedeutung fixiert, so dass spätere Änderungen an den voreingestellten Werten keinen Einfluss mehr auf ihn haben.

7.9.3 Automatische Fontersetzung

Jedes Mal, wenn ein Fontwechselbefehl nicht ausgeführt werden kann, weil $\text{\LaTeX}$ die Attributkombination nicht kennt, versucht das System einen Zeichensatz mit ähnlichen Attributen zu finden. Dabei geschieht Folgendes: Wenn die geforderte Kombination von Kodierschema, Schriftfamilie, -serie und -form nicht deklariert ist (siehe Abschnitt 7.10.3), setzt $\text{\LaTeX}$ zunächst das Attribut für die Form auf seinen voreingestellten Wert zurück, um eine bekannte Kombination zu finden. Wenn auch diese Kombination unbekannt ist, wird die Schriftserie, und schließlich, in letzter Instanz, die Schriftfamilie auf die Voreinstellung zurückgesetzt. Zuletzt wird dann die Schriftgröße im entsprechenden Eintrag in den internen Tabellen nachgeschlagen. Wenn man also z.B. `\ttfamily\bfseries\itshape` anfordert – eine Schreibmaschinenschrift in fetter Serie und kursiver Form (die es normalerweise nicht gibt) – so erhält man eine Schreibmaschinenschrift in normaler Serie und aufrechter Form, da

L^AT_EX zuerst die Form zurücksetzt und dann die Serie. Wenn man in diesem Fall aber eine Schreibmaschinenschrift in normaler Serie mit kursiver Form bevorzugt, muss man L^AT_EX diese Vorliebe über die sub-Funktion mitteilen, die auf Seite 437 erklärt ist.

Während des Ersetzungsvorgangs wird niemals die Kodierung gewechselt, da jede Veränderung an dieser Stelle eine Ausgabe falscher Zeichen nach sich ziehen könnte. Während die anderen Attribute nur das äußere Erscheinungsbild der Zeichen bestimmen, legt das Kodierschema ja fest, wie die eingegebenen Zeichen zu interpretieren sind. Es wäre schließlich katastrophal, wenn z.B. anstatt eines £-Zeichens auf einer Rechnung plötzlich ein \$-Zeichen erscheint, nur weil die Software versuchte clever zu sein.

Jedem Kodierschema muss also eine voreingestellte Schriftfamilie, -serie und -form zugeordnet sein, und L^AT_EX muss über eine Definition für eben diese Kombination von Attributen verfügen, wie in Abschnitt 7.10.5 erläutert.

7.9.4 Verwendung von Low-Level-Befehlen im Dokument

Die Low-Level-Befehle der vorhergehenden Abschnitte dienen in erster Linie der Definition von High-Level-Befehlen in Klassen- oder Paketdateien oder in der Dokumentenpräambel.

Solange für einen bestimmten Zweck ein High-Level-Befehl zur Verfügung steht, sollte man nach Möglichkeit auf die Verwendung von Low-Level-Befehlen innerhalb des eigentlichen Dokumentes verzichten. Der Grund liegt darin, dass Low-Level-Befehle präzise Anweisungen sind, auf einen bestimmten Zeichensatz umzuschalten, während High-Level-Befehle mithilfe von Paketen oder von Deklarationen in der Präambel den jeweiligen Anforderungen angepasst werden können. Angenommen, für ein Dokument wurde mit `\fontfamily{cmss}\selectfont` die Schrift Computer Modern Sans ausgewählt; wenn man dann aber später entscheidet, das gesamte Dokument mit Schriften aus der PSNFSS-Sammlung, wie z.B. Times, zu setzen, würde sich das entsprechende Paket nur auf die Teile des Dokumentes auswirken, die keine expliziten `\fontfamily`-Befehle enthalten.

7.10 Einbinden neuer Zeichensätze

7.10.1 Überblick

Wenn man neue Zeichensätze in L^AT_EX einbindet, bedeutet das im Prinzip, dass man die internen Fontauswahl-Tabellen mit all den Informationen füllt, die später benötigt werden, um einen angeforderten Font in einem Dokument mit der entsprechenden .tfm-Datei zu verknüpfen, welche die Zeicheninformationen für (L^A)T_EX enthält. Die Tabellen sind also das Verbindungsglied, durch das die Anweisung

```
\fontencoding{OT1}\fontfamily{cmdh}\fontseries{m}\fontshape{n}%
\fontsize{10}{12pt}\selectfont
```

mit der externen Datei `cmdunh10.tfm` assoziiert wird. Um neue Schriften hinzuzufügen muss man diesen Prozess umkehren. Bei jedem neuen externen

F	TT	W	[V.]	[N.]	[E]	[DD]
Herkunft	Schriftname	Stärke	Variante	Kodierung	Breite	Entwurfsgröße
z.B. p=Adobe	tm=Times	b=bold	i=italic	8t=T1	n=narrow	10=10 Punkt

Tabelle 7.28: Karl Berrys Klassifizierungsschema für Fontdateinamen

Zeichensatz muss man sich fünf Fragen stellen:

1. Wie sieht das Kodierschema des Zeichensatzes aus, d.h. welche Zeichen stehen an welcher Stelle?
2. Wie heißt die Schriftfamilie?
3. Um welche Schriftserie (Stärke und Breite) handelt es sich?
4. Welche Form hat die Schrift?
5. Wie groß ist die Schrift?

Die Antworten auf diese Fragen sind die Informationen, die man benötigt, um eigene externe Fonts entsprechend der $\text{\LaTeX}$ -Konventionen (siehe Abschnitt 7.9) zu klassifizieren. In den nächsten Abschnitten wird die Frage geklärt, wie man neue Zeichensätze in die NFSS-Tabellen einfügt, so dass sie später im Text verwendet werden können. Man braucht diese Informationen normalerweise, wenn man neue Zeichensätze benutzen möchte, z.B. um ein kleines Paket für den Zugriff auf eine neue Schriftfamilie zu schreiben. Im weiteren Verlauf werden darüber hinaus komplexere Konzepte erläutert, die dann ins Spiel kommen, wenn man z.B. für mathematische Formeln andere als die Standardzeichensätze verwenden möchte.

Um neue Fonts, die nicht für $\text{\TeX}$ entwickelt wurden, in $\text{\LaTeX}$ einzubinden, muss man möglicherweise noch einen Schritt vorher beginnen: mit dem Erzeugen von `.tfm`-Dateien und Dateien für virtuelle Fonts. Dieser Schritt wird normalerweise mithilfe des Programms `fontinst` von Alan Jeffrey durchgeführt. Es wurde von Lars Hellström weiterentwickelt und wird von ihm gepflegt. Das Programm ist in [58] und [66] beschrieben, sowie in der Programmdokumentation [74, 75].

7.10.2 Wie man Tausende von Fonts benennt

Eine Fontnomenklatur für $\text{\TeX}$ wurde von Karl Berry [18] vorgeschlagen und rief einige Diskussionen hervor [129]. Die aktuelle Version, die mittlerweile in der $\text{\TeX}$ -Welt zum Standard geworden ist, wird in [19] beschrieben. Berry versucht, die Dateinamen aller Zeichensätze durch acht alphanumerische Zeichen zu klassifizieren, wobei Groß- oder Kleinschreibung keine Rolle spielt. Diese Beschränkung auf acht Zeichen garantiert, dass auf allen Computerplattformen die gleichen Dateinamen verwendet werden können, und was noch wichtiger ist: Sie entspricht der ISO 9660-Norm für CD-ROMs. Das Namensschema ist in Tabelle 7.28 dargestellt, wobei die Abschnitte in eckigen Klammern ausgelassen werden, wenn sie voreingestellten Werten entsprechen. So wird zum Beispiel nur dann eine Entwurfsgröße angegeben, wenn der Font nicht linear skaliert ist. Tabelle 7.8 auf Seite 384 zeigt die Klassifizierung der 35 „elementaren“ PostScript-Fonts, gemäß der $\text{\LaTeX}$ -Fontschnittstelle. Für

jeden Zeichensatz ist der volle Adobe-Name aufgeführt; der entsprechende kurze (Karl Berry-) Dateiname steht in Klammern dabei (ohne den Kodierungsabschnitt). Für die Kodierungen OT1, T1 bzw. TS1 müsste man 7t, 8t bzw. 8c anhängen, um den vollständigen Dateinamen zu erhalten, also z.B. putr8t für Utopia Regular in der Kodierung T1.

Die Nomenklatur umfasst: T_EX-interne Fontbezeichnungen (wie sie in `\DeclareFontShape`-Deklarationen verwendet werden, die im nächsten Abschnitt beschrieben sind), Namen für virtuelle Fonts und deren Komponenten (z.B. besondere Umkodierungen realer Fonts) [95] sowie die Namen realer Zeichensätze. Im Falle der PostScript-Fonts unterscheiden sich die realen Namen der Zeichensätze oft von den T_EX-internen Bezeichnungen.

Ein Blick in die Unterwelt

Im letzteren Falle muss die Zuordnung der internen Fontbezeichnungen zur äußeren Welt dann geschehen, wenn das Ergebnis eines L^AT_EX-Laufes gesichtet oder gedruckt wird. Zum Beispiel verwendet der PostScript-Treiber dvips Zuordnungsdateien (mit der voreingestellten Erweiterung `.map`), die Zeilen wie die folgende enthalten:

```
putr8r Utopia-Regular "TeXBase1Encoding ReEncodeFont " <8r.enc <putr8a.pfb
```

Diese Zeile erklärt, dass man den Zeichensatz putr8r von dem externen Font putr8a.pfb erhält, indem man ihn mit einem besonderen Kodierungsvektor (in diesem Falle `8r.enc`) umkodiert. Wenn man jedoch in der Datei `t1put.fd` nachschaut – welche die `\DeclareFontShape`-Deklarationen für die Schriftfamilie Utopia mit T1-Kodierung enthält – sieht man, dass es keinen Verweis auf putr8r gibt. Stattdessen findet man Namen wie putr8t. Der Grund liegt darin, dass putr8t ein virtueller Zeichensatz ist (der mithilfe des Paketes fontinst [74, 75] erstellt wurde), der wiederum auf putr8r verweist. Letzterer Verweis ist schwer zu finden (außer durch die Namenskonventionen selbst), wenn man keinen Zugriff auf die Quellen hat, mit deren Hilfe der gerade von T_EX verwendete virtuelle Font zusammengestellt wurde. Glücklicherweise muss man sich nur selten so weit in die Tiefen eines T_EX-Systems begeben; wer es tut, findet dazu weitere Informationen in [58, Kapitel 10] oder in den zuvor aufgeführten Referenzen.

7.10.3 Deklarieren neuer Schriftfamilien und Gruppen von Schriftformen

Jede Kombination von Kodierschema und Schriftfamilie muss L^AT_EX mithilfe des Befehls `\DeclareFontFamily` bekannt gemacht werden. Dieser Befehl besitzt drei Argumente. Die ersten beiden sind Kodierung und Schriftfamilie. Der dritte ist normalerweise leer, kann jedoch besondere Optionen für das Laden eines Zeichensatzes enthalten. Nähere Erläuterungen hierzu finden sich auf Seite 438. Wenn man also eine neue Schriftfamilie – hier Computer Modern Dunhill in der alten T_EX-Kodierung – einführen will, würde man schreiben:

```
\DeclareFontFamily{OT1}{cmdh}{}
```

Eine Schriftfamilie besteht in der Regel aus vielen verschiedenen Zeichensätzen. Anstatt jedes Mitglied einer Schriftfamilie einzeln in L^AT_EX anzumelden, kombiniert man alle Fonts, die sich nur durch ihren Schriftgrad unterscheiden, und deklariert sie als eine Gruppe.

Eine solche Gruppe wird mit dem Befehl `\DeclareFontShape` in die internen $\text{\LaTeX}$ -Tabellen eingetragen. Er verfügt über sechs Argumente: Die ersten vier sind die Kodierung sowie die Schriftfamilie, -serie und -form, unter denen man später auf diese Zeichensätze zugreifen möchte. Das fünfte Argument besteht aus einer Liste von Schriftgrößen und Namen externer Zeichensätze in einem bestimmten Format, das später noch besprochen wird. Das sechste Argument ist normalerweise leer; auf Seite 438 wird erklärt, wozu es dient.

Die ersten Beispiele dienen der Einführung in die Terminologie und danach werden die Funktionen im Detail erläutert.

Beispielsweise könnte ein NFSS-Tabelleneintrag für den Computer Modern Dunhill Font normal (Schriftserie) aufrecht (Schriftform) im Kodierschema „ $\text{\TeX}$ Text“ folgendermaßen aussehen:

```
\DeclareFontShape{OT1}{cmdh}{m}{n}{ <10> cmdunh10 }{}
```

Hier wird davon ausgegangen, dass nur ein externer Zeichensatz im Schriftgrad 10pt verfügbar ist. Wenn der Font auch in 12pt vorliegt (von 10pt skaliert), würde die Deklaration folgendermaßen lauten:

```
\DeclareFontShape{OT1}{cmdh}{m}{n}{ <10> <12>cmdunh10 }{}
```

Ist der externe Font in allen Größen verfügbar, so wird die Deklaration sehr einfach. Das ist der Fall bei Type 1 PostScript-Fonts, oder wenn der Treiber über METAFONT Fonts auf Abruf erzeugen kann.

Times Roman fett (Schriftserie) aufrecht (Schriftform) in der $\text{\LaTeX}$ -Kodierung T1 würde beispielsweise folgendermaßen definiert:

```
\DeclareFontShape{T1}{ptm}{b}{n}{ <-> ptmb8t }{}
```

In diesem Beispiel wird für die Schriftgröße ein Bereich mit zwei offenen Enden deklariert (weder links noch rechts des – ist ein Wert angegeben). Dadurch wird für alle Schriftgrade dieselbe externe .tfm-Datei (ptmb8t) benutzt und auf die gewünschte Größe skaliert. Wenn für einen Font mehr als eine .tfm-Datei zur Verfügung steht, etwa `emtt10` für Fließtext und `emtt12` für Überschriften (hier handelt es sich um European Modern Typewriter), könnte die Anweisung folgendermaßen lauten:

```
\DeclareFontShape{T1}{emtt}{m}{n}{<-12> emtt10 <12-> emtt12}{}
```

In diesem Fall würde die .tfm-Datei `emtt10` für alle Schriftgrade unter 12pt benutzt und `emtt12` für die Schriftgrade 12pt oder größer.

Wie in den vorangegangenen Beispielen gezeigt, besteht das fünfte Argument des Befehls `\DeclareFontShape` aus Größenangaben, die in spitze Klammern (d.h. `<...>`) eingefasst werden, ergänzt durch die notwendigen Informationen zum Laden der einzelnen Schriftgrade (z.B. durch Bezeichnungen von Zeichensätzen). Die Informationen innerhalb der spitzen Klammern werden als „Größendaten“ und jene hinter der Klammer als „Zeichensatzdaten“ bezeichnet. Die Zeichensatzdaten sind weiter unterteilt in eine „Größenfunktion“ (häufig leer) und ihre Argumente; auf diesen Punkt wird später noch näher eingegangen. In den Argumenten des `\DeclareFontShape`-Befehls werden Leerzeichen ignoriert, damit die Einträge leichter zu lesen

sind.¹ Für den ungewöhnlichen Fall, dass ein echtes Leerzeichen erforderlich ist, steht der Befehl `\space` zur Verfügung.

Einfachgrößen und Größenbereiche

Die Größendaten, d.h. die Daten innerhalb der spitzen Klammern des fünften Argumentes von `\DeclareFontShape`, lassen sich in „Einfachgrößen“ und „Größenbereiche“ unterteilen. Unter einer Einfachgröße versteht man eine einzelne (Dezimal-)Zahl, wie `<10>` oder `<14.4>`, die im Prinzip jeden beliebigen positiven Wert annehmen kann. Da sie jedoch einen Schriftgrad bezeichnet, der in Punkten gemessen wird, bewegt sie sich im Allgemeinen zwischen 4 und 120. Ein Größenbereich besteht aus zwei Einfachgrößen, die durch einen Bindestrich voneinander getrennt sind, und gibt den Größenbereich der Schriftschnitte mit denselben Zeichensatzdaten an. Die untere Grenze (d.h. die links vom Bindestrich stehende Größe) gehört zu dem angegebenen Bereich, die obere Grenze nicht. Der Bereich `<5-10>` steht beispielsweise für Schriftgrade, die größer oder gleich 5pt und kleiner als 10pt sind. Die Zahlen vor und hinter dem Bindestrich können aber auch weggelassen werden. Die Interpretation dieser Angabe ist offensichtlich: `<->` bezeichnet einen unbegrenzten Bereich, `<-10>` steht für Schriftgrade, die kleiner sind als 10pt und `<12->` für alle Schriftgrade, die größer oder gleich 12pt sind.

Häufig haben mehrere Einfachgrößen die gleichen Zeichensatzdaten. In diesem Fall kann man alle bis auf die letzten Zeichensatzdaten weglassen, etwa:

```
\finalvskip{-\smallskipamount}
\DeclareFontShape{OT1}{panr}{m}{n}{ <5> <6> <7> <8> <9> <10>
                                     <10.95> <12> <14.4> <17.28> <20.74> <24.88> pan10 }{}
```

Dieses Beispiel deklariert den Font Pandora Roman normal als eine Gruppe, die Zeichensätze in mehreren Schriftgraden beinhaltet, welche alle durch Skalierung aus der gleichen Entwurfsgröße erzeugt werden.

Größenfunktionen

Wie bereits erwähnt, werden die Zeichensatzdaten (die Zeichenkette hinter der schließenden spitzen Klammer) in eine Größenfunktion und ihr zugehöriges Argument unterteilt. Enthält die Zeichenkette ein `*`-Zeichen, bildet der Teil links davon den Funktionsnamen und der Teil rechts davon das Argument. Ohne Stern (wie in allen bisherigen Beispielen) wird die gesamte Zeichenkette als das Argument betrachtet, und der Funktionsname ist „leer“.

Auf Basis des vom Benutzer gewünschten Schriftgrades und der Informationen im Befehl `\DeclareFontShape` erzeugen die Größenfunktionen die Spezifikation, die $\text{\LaTeX}$ benötigt, um den externen Zeichensatz zu finden und ihn in der gewünschten Größe zu laden. Sie haben auch die Aufgabe, den Benutzer über besondere Vorgänge zu informieren. Einige Funktionen unterscheiden sich zum Beispiel nur darin, ob sie eine Warnung ausgeben oder

¹Das trifft nur für Befehle zu, die direkt im Dokument verwendet werden. Wenn eine solche Deklaration innerhalb anderer Konstrukte verwendet wird (z.B. im Argument von `\AtBeginDocument`), können Leerzeichen erhalten bleiben, so dass die Einträge nicht mehr erkannt werden.

nicht. Dadurch hat der Systembetreuer die Möglichkeit, $\text{\LaTeX}$ optimal an die Bedürfnisse seiner Installation anzupassen.

Der Name einer Größenfunktion kann aus null bis mehreren Buchstaben bestehen. Einige Größenfunktionen haben zwei Argumente, wobei das erste optional und das zweite obligatorisch ist. Das optionale Argument muss in eckigen Klammern stehen. Die Spezifikation

```
<-> s * [0.9] cmfib8
```

wählt z.B. für alle möglichen Größen (der Bereich erstreckt sich hier von 0 bis ∞) die Größenfunktion `s` mit dem optionalen Argument 0.9 und dem obligatorischen Argument `cmfib8` aus.

Die Größenspezifikationen in `\DeclareFontShape` werden in der Reihenfolge geprüft, in der sie angegeben sind. Sobald eine Größenangabe zur Anforderung des Benutzers passt, wird die Größenfunktion ausgeführt. Wenn dieser Prozess einen gültigen Zeichensatz liefert, werden die weiteren Einträge nicht mehr geprüft. Andernfalls wird die Suche mit dem nächsten Eintrag fortgesetzt. Im Folgenden werden die Standard-Größenfunktionen beschrieben. Das Dokument `fontguide.tex` [109], das zur $\text{\LaTeX}$ -Distribution gehört, beschreibt, wie man im Bedarfsfalle weitere Funktionen definieren kann.

Die „leere“ Funktion Die leere Funktion wird am häufigsten benutzt und trägt deshalb den kürzesten Namen. (Jeder Tabelleneintrag belegt einen kleinen Teil des internen Speichers; die gewählte Syntax ist deshalb ein Kompromiss zwischen perfekter Benutzerschnittstelle und geringem Speicherbedarf.) Bei Einfachgrößen lädt die leere Funktion den Zeichensatz genau mit dem gewünschten Schriftgrad. Bei Größenbereichen wird der Zeichensatz genau mit dem vom Benutzer geforderten Schriftgrad geladen, sofern dieser in den angegebenen Bereich fällt. Wenn der Benutzer beispielsweise 14.4 verlangt, wird durch die Spezifikation

```
<-> panr10
```

die `.tfm`-Datei `panr10.tfm` in der Größe 14,4pt geladen. Da dieser Zeichensatz für 10pt ausgelegt ist (es handelt sich um den Pandora Roman Font in 10pt), werden alle Werte in der `.tfm`-Datei mit dem Faktor 1,44 multipliziert.

Manchmal kommt es aber auch vor, dass ein Zeichensatz in einer Größe geladen werden soll, die leicht von dem angeforderten Schriftgrad abweicht. Diese Anpassung kann z.B. dann notwendig sein, wenn die Zeichensätze einer bestimmten Schriftfamilie im Vergleich zu den anderen im Dokument verwendeten Schriften zu groß erscheinen. Zu diesem Zweck besitzt die leere Funktion ein optionales Argument, in dem ein Skalierungsfaktor angegeben werden kann, der – sofern vorhanden – mit der vom Benutzer gewünschten Größe multipliziert wird, um so die tatsächlich zu ladende Schriftgröße zu errechnen. Damit würde

```
<-> [0.95] phvr8t
```

die `.tfm`-Datei namens `phvr8t.tfm` (Helvetica in T1-Kodierung) immer mit einer Skalierung von 95% des verlangten Schriftgrades laden. Bei Verwendung des optionalen Argumentes gibt die leere Funktion eine Warnung für den

Benutzer aus, die besagt, dass der Zeichensatz nicht in der angeforderten Größe geladen wird.

Die Funktion „s“ Die s-Funktion bewirkt im Prinzip dasselbe wie die leere Funktion; sie gibt jedoch keine Warnungen aus (das s steht für „silent“). Mit dem Befehl

```
\DeclareFontShape{T1}{phv}{m}{n}{ <-> s * [0.95] phvr8t }{}
```

werden alle Bildschirmmeldungen der leeren Funktion unterdrückt. Sie werden jedoch weiterhin in der Protokolldatei aufgezeichnet, so dass man immer noch herausfinden kann, welche Zeichensätze benutzt wurden, falls etwas schief geht. Das Paket `helvet` ist auf diese Weise programmiert, wobei der Skalierungsfaktor allerdings nicht festgelegt ist, sondern der Deklaration `\DeclareFontShape` durch eine Paketoption übergeben wird.

Die Funktion „gen“ Häufig werden die Namen der externen Zeichensätze gebildet, indem man den Schriftgrad an eine Zeichenfolge für die Schrift anhängt. So sind z.B. `cmtt8`, `cmtt9` und `cmtt10` die externen Namen für den Font Computer Modern Typewriter in den Schriftgrößen 8, 9 und 10pt. Mit der `gen`-Funktion kann man die Einträge für alle Zeichensatznamen verkürzen, die nach diesem Schema gebildet werden. Sie kombiniert die Zeichensatzdaten mit der gewünschten Größe und generiert (daher `gen`) daraus den externen Zeichensatznamen. Daher kann man

```
<8> <9> <10> gen * cmtt
```

als Abkürzung für

```
<8> cmtt8 <9> cmtt9 <10> cmtt10
```

verwenden, und damit in den internen NFSS-Tabellen acht Zeichen sparen. Die beiden Teile werden wortwörtlich zusammengefügt. Daher kann man die Funktion nicht für dezimale Schriftgrade wie 14.4 nutzen. Außerdem sollte man darauf achten, dass die Zahlen im externen Fontnamen auch tatsächlich der Entwurfsgröße entsprechen (`cmr17` steht z.B. in Wirklichkeit für Computer Modern Roman in 17,28pt).

Ansonsten verhält sich die `gen`-Funktion genauso wie die leere Funktion, d.h. das optionale Argument enthält, sofern vorhanden, einen Skalierungsfaktor und löst bei Verwendung eine Mitteilung aus.

Die Funktion „sgen“ Die `sgen`-Funktion ist die stille Variante der `gen`-Funktion. Ihre Meldungen werden nur in der Protokolldatei aufgezeichnet.

Die Funktion „genb“ Diese Größenfunktion gleicht der Funktion `gen`, ist jedoch für Zeichensätze gedacht, deren Schriftgrad im Fontnamen in hundertstel Punkten (Centipoints) angegeben wird, wie zum Beispiel die EC Fonts. Eine Zeile wie etwa

```
<9> <10> <10.95> <12> genb * ecrm
```

dient daher als Abkürzung für

```
<9> ecrm0900 <10> ecrm1000 <10.95> ecrm1095 <12> ecrm1200
```

Ein optionales Argument hat, sofern es vorhanden ist, die gleichen Auswirkungen wie bei der leeren Funktion: Es gibt einen Skalierungsfaktor an und erzeugt bei Verwendung eine Mitteilung.

Die Funktion „sgenb“ Die sgenb-Funktion ist die stille Variante der genb-Funktion. Ihre Meldungen werden nur in der Protokolldatei aufgezeichnet.

Die Funktion „sub“ Mithilfe der Funktion sub wird eine Gruppe von Schriftformen durch eine andere ersetzt, wenn für die aktuelle Gruppe keine externen Zeichensätze existieren. In diesem Fall stellt das Argument keinen externen Zeichensatznamen dar, sondern eine durch Schrägstriche getrennte Kombination von Schriftfamilie, -serie und -form (die Kodierung ändert sich aus den bereits erwähnten Gründen nicht). In der Schriftfamilie Computer Modern Sans gibt es z.B. keine kursive sondern nur eine geneigte Form. Es ist daher sinnvoll, die geneigte Schrift als Ersatz für die Kursivschrift zu deklarieren:

```
\DeclareFontShape{OT1}{cmss}{m}{it}{ <-> sub * cmss/m/sl }{}
```

Ohne diese Deklaration würde der automatische Ersetzungsmechanismus von $\text{\LaTeX}$ (siehe Abschnitt 7.9.3) auf die voreingestellte Schriftform Computer Modern Sans aufrecht umschalten.

Neben dem Ersetzen ganzer Gruppen von Schriftformen gibt es aber noch andere gute Verwendungsmöglichkeiten für die sub-Funktion. Man betrachte den folgenden Programmcode:

```
\DeclareFontShape{OT1}{cmss}{m}{sl}{ <-8> sub * cmss/m/n
<8> cmssi8 <9> cmssi9 <10><10.95> cmssi10
<12><14.4> cmssi12 <17.28><20.74><24.88> cmssi17 }{}
```

Diese Deklaration besagt, dass $\text{\LaTeX}$ bei Schriftgraden unter 8pt in der Schriftformdeklaration von OT1/cmss/m/n nachsehen soll. Solche Ersetzungen können über mehrere Stufen erfolgen. Wer sich mit den Standardzeichensätzen auskennt, weiß, dass es keinen Computer Modern Sans Zeichensatz gibt, der kleiner ist als 8pt. Die Ersatzgruppe der Schriftformen wird also wahrscheinlich einen weiteren Ersetzungseintrag enthalten. Das mag zunächst ein wenig seltsam erscheinen, hat jedoch den Vorteil, dass bei Hinzufügen neuer Zeichensätze nur die Deklaration einer einzigen Gruppe von Schriftformen geändert werden muss. Alle weiteren Deklarationen, die indirekt auf diese Fonts zurückgreifen, sind damit automatisch angepasst.

Die Funktion „ssub“ Die ssub-Funktion bewirkt im Prinzip dasselbe wie die sub-Funktion; sie gibt jedoch keine Warnungen auf dem Bildschirm aus (das s steht für „still“).

Die Funktion „subf“ Die subf-Funktion ist eine Mischung zwischen der leeren und der sub-Funktion, denn sie lädt die Zeichensätze auf die gleiche Weise wie die leere Funktion, gibt aber eine Warnung aus, dass eine Ersetzung erfolgte, weil der gewünschte Schriftschnitt nicht vorhanden ist. Man kann sie also benutzen, um externe Zeichensätze zu ersetzen, ohne für sie eine separate Gruppe von Schriftformen definieren zu müssen, wie es bei der sub-Funktion der Fall ist. Das Beispiel

```
\DeclareFontShape{OT1}{ptm}{bx}{n}{ <-> subf * ptmb7t }{}
```

würde den Benutzer warnen, dass die gewünschte Kombination nicht verfügbar ist und dass deshalb an ihrer Stelle der Font ptmb7t geladen wurde. Da diese Meldung weniger informativ ist als die der sub-Funktion, sollte letztere bevorzugt werden.

Die Funktion „ssubf“ Als stille Variante von subf schreibt diese Funktion ihre Meldungen nur in die Protokolldatei.

Die Funktion „fixed“ Diese Funktion ignoriert die vom Benutzer angeforderte Größe und lädt stattdessen den als Argument angegebenen externen Zeichensatz. Wenn vorhanden, gibt das optionale Argument die Größe (in Punkten) an, in welcher der Zeichensatz geladen wird. Mit dieser Funktion kann man also Größenbereiche festlegen, innerhalb deren ein bestimmter Zeichensatz immer mit demselben Schriftgrad geladen wird.

Die Funktion „sfixed“ Als stille Variante der Funktion fixed wird diese Funktion z.B. verwendet, um den Zeichensatz mit den großen mathematischen Symbolen zu laden, der häufig nur in einer Größe vorhanden ist.

Ladeoptionen für Zeichensätze

Wie bereits erwähnt, muss jede Schriftfamilie zunächst mit dem Befehl `\DeclareFontFamily` deklariert werden. Das dritte Argument dieses Befehls kann, genau wie das sechste Argument von `\DeclareFontShape`, dazu verwendet werden, spezielle Operationen festzulegen, die beim Laden eines Zeichensatzes ausgeführt werden. Auf diese Weise lassen sich Parameter ändern, die sich auf eine gesamte Schriftfamilie beziehen.

Neben der Information über jedes Zeichen führt (L^A)T_EX für jeden externen Zeichensatz einige globale Maße und andere Werte, die mit dem jeweiligen Zeichensatz verbunden sind. So hat z.B. jeder Zeichensatz sein eigenes „Trennzeichen“, also das Zeichen, das automatisch eingefügt wird, wenn (L^A)T_EX ein Wort trennt. Fontspezifisch sind auch die normale Breite und die Dehnbarkeit des „Wortzwischenraums“ (Leerraum zwischen den Wörtern), deren Werte ebenfalls gespeichert werden und sich ändern, wenn (L^A)T_EX auf einen anderen Zeichensatz umschaltet. Durch Ändern dieser Werte beim Laden eines Fonts lassen sich besondere Effekte erzielen.

In der Regel gelten Änderungen immer für eine ganze Schriftfamilie. Man kann beispielsweise die Silbentrennung aller Wörter unterdrücken, die in Schreibmaschinenschrift gesetzt sind. Dazu sollte das dritte Argument

der `\DeclareFontFamily`-Deklaration verwendet werden. Wenn die Änderung aber nur für eine spezielle Gruppe von Schriftschnitten gelten soll, benötigt man das sechste Argument von `\DeclareFontShape`. Anders ausgedrückt: Beim Laden eines Zeichensatzes führt NFSS zuerst das dritte Argument von `\DeclareFontFamily` und danach das sechste Argument von `\DeclareFontShape` aus. Sofern es erforderlich ist, lassen sich mittels dieses Verfahrens die Ladeoptionen überschreiben, die für die gesamte Schriftfamilie festgelegt wurden.

Im Folgenden wird untersucht, was auf diese Weise eingestellt werden kann (leider lässt sich nicht alles ändern), und es werden einige nützliche Beispiele besprochen. Dieser Teil der Schnittstelle arbeitet mit den untersten Low-Level-Befehlen von $\TeX$. Da sie sehr spezielle Funktionen bedient, hat man sich nicht bemüht, die Syntax an $\LaTeX$ anzupassen. Das Verfahren für die Zuweisung von ganzen Zahlen und Maßen an Variablen ist daher etwas ungewöhnlich.

Mit `\hyphenchar\font=<zahl>` bestimmt $(\LaTeX)$ das Zeichen, das bei der Silbentrennung eines Wortes als Trennzeichen eingefügt wird. Der Wert `<zahl>` gibt die Position des Zeichens im Kodierschema an. Als Voreinstellung dient der Wert von `\defaulthyphenchar`, der 45 entspricht und in den meisten Kodierungen auf die Position des Zeichens „-“ verweist. Wird diese Zahl auf -1 gesetzt, so wird die Silbentrennung unterdrückt. Mit der Deklaration

Ändern des
Trennzeichens

```
\DeclareFontFamily{OT1}{cmmt}{\hyphenchar\font=-1}
```

wird demnach die Trennung in allen Zeichensätzen der `cmmt`-Schriftfamilie mit dem Kodierschema OT1 unterdrückt. Zeichensätze der T1-Kodierung enthalten an Position 127 ein alternatives Trennzeichen. Man kann also zum Beispiel eingeben

```
\DeclareFontFamily{T1}{cmr}{\hyphenchar\font=127}
```

Der von $(\LaTeX)$ eingefügte Trennstrich unterscheidet sich dann von einem Bindestrich in zusammengesetzten Wörtern wie „C-Dur“. $(\LaTeX)$ trennt keine Wörter, die bereits ein explizites Trennzeichen enthalten (außer direkt nach diesem Trennzeichen). Dies kann in Sprachen, in denen die durchschnittliche Wortlänge sehr viel größer ist als im Englischen, zu einem echten Problem werden. Dieses Problem lässt sich mit der beschriebenen Einstellung beheben.

Mit jedem $(\LaTeX)$ -Font sind eine Reihe von Längenparametern verknüpft, die durch Zuweisungen in der Form `\fontdimen<zahl>\font=<länge>` geändert werden, wobei `<zahl>` die Bezugszahl für den Parameter und `<länge>` den zuzuweisenden Wert darstellt. Die voreingestellten Werte werden beim Laden des Zeichensatzes der `.tfm`-Datei entnommen. Jeder Zeichensatz verfügt über mindestens sieben solcher Parameter:

`\fontdimen1` Spezifiziert die Neigung der Zeichen als den Abstand von der Senkrechten in der Höhe von 1 pt. Ist der Wert null, so handelt es sich um einen aufrechten Zeichensatz.

`\fontdimen2` Gibt den normalen Abstand zwischen Wörtern an (Wortzwischenraum).

`\fontdimen3` Gibt die zusätzliche Dehnbarkeit des Wortzwischenraums an, d.h. den zusätzlichen Leerraum, den (L^A)T_EX für den Randausgleich zwischen Wörtern einfügen darf. Im Notfall kann (L^A)T_EX auch über diesen erlaubten Wert hinaus Leerraum einfügen; in diesem Fall wird die Meldung „Underfull box“ (d.h. Zeile mit zuviel Leerraum zwischen den Wörtern) ausgegeben.

`\fontdimen4` Gibt den Wert für die erlaubte Stauchung des Wortzwischenraums an, d.h. den Leerraum, den (L^A)T_EX für den Randausgleich vom normalen Wortzwischenraum (`\fontdimen2`) abziehen kann. Der Mindestwert wird dabei niemals unterschritten.

`\fontdimen5` Gibt die x-Höhe (Mittellänge) an. Diese definiert das fontspezifische Maß 1ex.

`\fontdimen6` Gibt die Breite des Gevierts an. Sie definiert die fontspezifische Größe 1em.

`\fontdimen7` Gibt den zusätzlichen Leerraum an, der hinter bestimmten Satzendezeichen eingefügt werden soll, wenn `\nonfrenchspacing` eingeschaltet ist. Die genauen Regeln, nach denen T_EX diese Länge (ganz oder teilweise) verwendet, sind recht kompliziert und werden in dem Buch *The T_EXbook* [87] näher erläutert. Sie wird ignoriert bzw. durch den Wert `\xspaceskip` ersetzt, wenn dieser nicht null beträgt.

Wenn man den Wortzwischenraum eines Zeichensatzes ändern will, kann man keinen absoluten Wert angeben, da ein solcher Wert für alle Größen innerhalb einer Gruppe von Schriftschnitten gültig sein muss. Er muss deshalb mithilfe eines anderen, vom jeweiligen Zeichensatz abhängigen Parameters definiert werden. Eine solche Definition kann beispielsweise folgendermaßen aussehen:

```
\DeclareFontShape{OT1}{cmr}{m}{n}{...}
{\fontdimen2\font=.7\fontdimen2\font}
```

Diese Deklaration reduziert den normalen Wortzwischenraum auf 70% seines ursprünglichen Wertes. Dehnung und Stauchung lassen sich auf ähnliche Weise ändern.

Einige Zeichensätze, die in Formeln verwendet werden, benötigen mehr als sieben Fontparameter. Es handelt sich hierbei um die Symbolfonts namens „symbols“ und „largesymbols“ (siehe Abschnitt 7.10.7). T_EX wird eine Formel nicht setzen, wenn die entsprechenden Symbolfonts weniger als 22 bzw. 13 `\fontdimen`-Parameter enthalten. Die Werte dieser Parameter werden verwendet, um die Zeichen in den Formeln zu positionieren. Die Bedeutung sämtlicher `\fontdimen`-Parameter zu erklären, würde den Rahmen dieses Buches sprengen; nähere Erläuterungen hierzu finden sich in Anhang G des Buches *The T_EXbook* [87].

Einen kleinen Schönheitsfehler hat das T_EX-System allerdings: T_EX lädt jede `.tfm`-Datei für einen bestimmten Schriftgrad nur ein einziges Mal. Es ist daher nicht möglich, eine Gruppe von Schriftformen (mit dem Befehl `\DeclareFontShape`) zu definieren, um einen externen Zeichensatz – z.B.

`cmtt10` – zu laden und dann mit einem weiteren `\DeclareFontShape`-Befehl denselben externen Zeichensatz noch einmal zu laden und dabei einige der `\fontdimen`-Parameter oder andere, mit dem Zeichensatz verbundene Parameter zu ändern. In diesem Fall werden die Werte für beide Gruppen von Schriftformen geändert.

Angenommen, man definiert einen engen Schriftschnitt, indem man den Wortzwischenraum verkleinert:

```
\DeclareFontShape{T1}{ptm}{m}{n}{ <-> ptmr8t }{ }
\DeclareFontShape{T1}{ptm}{c}{n}{ <-> ptmr8t }
      {\fontdimen2\font=.7\fontdimen2\font}
```

Diese Deklaration bringt nicht das gewünschte Ergebnis. Sobald der schmale Schriftschnitt geladen wird, würde sich auch der Wortzwischenraum für den normalen Schriftschnitt ändern, und zwar auf den Wert, der für den schmalen Schriftschnitt angegeben wurde. Der beste Ausweg aus diesem Dilemma ist die Definition eines virtuellen Fonts, der dieselben Zeichen enthält wie der ursprüngliche Zeichensatz, jedoch andere Werte für die Fontparameter verwendet (siehe [74, 77, 95]). Eine weitere mögliche Lösung ist das Laden des Fonts in einer geringfügig anderen Größe, wie bei der folgenden Deklaration:

```
\DeclareFontShape{T1}{ptm}{c}{n}{ <-> [0.9999] ptmr8t }
      {\fontdimen2\font=.7\fontdimen2\font}
```

Durch diese Strategie handelt es sich für $\TeX$ um unterschiedliche Zeichensätze mit unterschiedlichen `\fontdimen`-Parametern. In diesem speziellen Fall kann man ansonsten den Wortzwischenraum auch noch mit `\spaceskip` steuern und damit die Werte für den Zeichensatz überschreiben. Dieser Parameter wird in Abschnitt 3.1.12 angesprochen.

7.10.4 Verändern von Schriftfamilien und Schriftschnitten

Wenn man in einem bestimmten Dokument eine geänderte Deklaration für eine Gruppe von Schriftschnitten verwenden möchte, braucht man sie lediglich in einem Paket oder der Präambel des Dokumentes zu definieren. Damit wird jede bereits existierende Deklaration dieser Art überschrieben. Es ist jedoch zu beachten, dass die Verwendung des Befehls `\DeclareFontFamily` ein späteres Laden der zugehörigen `.fd`-Datei verhindert (siehe Abschnitt 7.10.6). Außerdem hat die neue Deklaration keinen Einfluss auf bereits geladene Zeichensätze.

Das heutige $\LaTeX$ ist so voreingestellt, dass es nur wenige Zeichensätze vorab lädt. Über die Konfigurationsdatei `preload.cfg` lassen sich jedoch mehr oder weniger Zeichensätze laden, wenn das Format erzeugt wird. Vorab geladene Fonts können nicht mithilfe von Deklarationen für Schriftfamilien oder -formen verändert werden. Wenn man also für die zentralen Zeichensätze besondere Einstellungen vornehmen will, muss man sichergehen, dass keiner dieser Fonts vorab geladen wird. Das Dokument `cfgguide.tex` [110], das Bestandteil jeder $\LaTeX$ -Distribution ist, enthält weitere Informationen darüber, wie man eine $\LaTeX$ -Installation anpassen kann.

7.10.5 Einbindung neuer Kodierschemata

Zeichensatzwechsel, die Änderungen im Kodierschema beinhalten, sind mit Vorsicht zu genießen. In der T1-Kodierung sind beispielsweise die meisten Akzentbuchstaben separate Glyphen, während sie in der traditionellen T_EX-Kodierung (OT1) mit dem Basisbefehl `\accent` aus Buchstaben und Akzenten zusammengesetzt werden müssen. (Es ist wünschenswert, Akzentbuchstaben durch eigene Glyphen darzustellen und sie nicht mit dem Befehl `\accent` zusammenzusetzen, da so unter anderem bei Wörtern mit Akzenten eine korrekte Silbentrennung möglich ist.) Wenn diese beiden Vorgehensweisen miteinander kombiniert werden müssen, etwa weil ein Zeichensatz nur in einer der beiden Kodierungen vorhanden ist, müssen Befehle wie `\` so definiert sein, dass sie je nach Fontkodierung eine andere Bedeutung haben.

Daher muss jedes Kodierschema zunächst formal in L^AT_EX eingeführt werden, und zwar mithilfe einer `\DeclareFontEncoding`-Deklaration. Diese Deklaration hat drei Argumente. Das erste Argument enthält den Namen, unter dem mit dem Befehl `\fontencoding` später auf die Kodierung zugegriffen wird. Tabelle 7.27 auf Seite 428 zeigt eine Liste der Standardkodierschemata und ihrer internen NFSS-Bezeichnungen.

Das zweite Argument enthält beliebigen Code (z.B. Definitionen), der jedes Mal ausgeführt wird, wenn L^AT_EX mithilfe des Befehls `\fontencoding` von einer Kodierung auf eine andere wechselt. Das letzte Argument enthält den Code, der verwendet wird, wenn man den Zeichensatz als Mathematikalphabet innerhalb von Formeln verwendet. Diese drei Argumente können also dazu benutzt werden, Befehle umzudefinieren, die sich auf Zeichenpositionen in der Kodierung beziehen. Um zu vermeiden, dass durch zusätzliche Leerzeichen innerhalb der Argumente störende Leerräume in der Ausgabe entstehen, werden diese Leerzeichen ignoriert. Für den unwahrscheinlichen Fall, dass in einer der Definitionen eines Argumentes doch ein Leerzeichen benötigt wird, kann man den Befehl `\space` verwenden.

Das L^AT_EX3-Projekt behält sich die Definition neuer Kodierungen mit den folgenden Anfangsbuchstaben vor: T (Standard-Textkodierungen mit 256 Zeichen), TS (Symbole zur Erweiterung der entsprechenden T-Kodierung), X (Textkodierungen, die nicht den strengen Anforderungen für T-Kodierungen entsprechen), M (Standard-Formelkodierungen mit 256 Zeichen), S (andere Symbolkodierungen), A (andere Sonderfälle), OT (Standard-Textkodierungen mit 128 Zeichen) und OM (Standard-Formelkodierungen mit 128 Zeichen). Der Buchstabe O für Kodierungen soll betonen, dass die 128-Zeichen-Kodierungen *veraltet* (Englisch: obsolete) sind. Im Idealfall werden diese Kodierungen durch Standards ersetzt, die von den T_EX-Anwendergruppen definiert werden, so dass in Zukunft nur noch dann Kodierungswechsel erforderlich sind, wenn man von einer Sprache zur anderen wechselt.

Für eigene lokale Kodierungen sollte man die Anfangsbuchstaben L für „lokal“ oder E für „experimentell“ verwenden. Kodierungen, die mit U beginnen, sind „unbekannt“ oder nicht klassifiziert, d.h. ihre Fonts passen in keines der herkömmlichen Kodierschemata. Diese Vorgehensweise sorgt dafür, dass Dateien mit offiziellen Kodierungen portabel sind. Neue Standardkodierungen werden sofort nach Erscheinen in die L^AT_EX-Dokumentation aufgenommen. So sind z.B. seit der ersten Ausgabe dieses Buches die Kodierungen T2* und T5 hinzugekommen.

Der Befehl `\DeclareFontEncoding` speichert den Namen der neu deklarierten Kodierung in dem Befehl `\LastDeclaredEncoding`. Dies ist manchmal hilfreich und wird z.B. in den Dateien zur Deklaration der Kodierungen für die kyrillischen Sprachen genutzt.

Wie bereits in Abschnitt 7.9.3 über die Fontersetzung besprochen, können die Voreinstellungen für die Schriftfamilie, -serie und -form für verschiedene Kodierungen unterschiedlich sein. Aus diesem Grund verfügt NFSS über den Befehl `\DeclareFontSubstitution`, der wiederum als erstes Argument die Kodierung erwartet. In den drei weiteren Argumenten stehen die (für diese Kodierung) voreingestellten Werte für die Schriftfamilie, -serie und -form für die Fontersetzung (siehe dazu Abschnitt 7.9.3). Es ist wichtig, dass diese drei Argumente gemeinsam einen gültigen Schriftschnitt bezeichnen, also in anderen Worten, dass für sie eine `\DeclareFontShape`-Deklaration existiert. Ansonsten gibt NFSS eine Fehlermeldung aus, sobald bei `\begin{document}` die internen Tabellen überprüft werden.

7.10.6 Interne Dateistruktur

Die Schriftfamilien können beim Generieren einer Formatdatei oder in der Präambel des Dokumentes deklariert werden. Man kann sie auch nach Bedarf laden, wenn ein Fontwechselbefehl eine Kombination anfordert, die bisher noch nicht verwendet wurde. Die erste Variante verbraucht in jedem $\text{\LaTeX}$ -Lauf internen Speicher, auch wenn der Font gar nicht verwendet wird. Die zweite und dritte Möglichkeit benötigen etwas mehr Zeit beim Formatieren des Dokumentes, da die Fontdefinitionen während der Verarbeitung gelesen werden müssen. Für die meisten Gruppen von Schriftformen sind trotzdem die beiden letzten Lösungen vorzuziehen, da sich damit eine Vielzahl verschiedener Dokumente mit ein und demselben $\text{\LaTeX}$ -Format setzen lässt.

Während das $\text{\LaTeX}$ -Format erzeugt wird, liest $\text{\LaTeX}$ eine Datei mit Namen `fonttext.ltx`, die einen Standardsatz an Definitionen von Schriftfamilien sowie einige weitere Deklarationen für Textfonts enthält. Mit einigen Einschränkungen¹ lässt sich diese Zusammenstellung durch Hinzufügen einer Konfigurationsdatei namens `fontdef.cfg` ändern; siehe hierzu die Dokumentation `cfgguide.tex`.

Alle anderen Definitionen für Schriftfamilien sollte man in externen Dateien deklarieren, die nur bei Bedarf geladen werden, also entweder in Paketdateien oder in Fontdefinitionsdateien (mit der Erweiterung `.fd`). Wenn man Fontdefinitionen in einer Paketdatei gespeichert hat, muss diese nach dem `\documentclass`-Befehl explizit geladen werden. Es gibt jedoch auch noch eine dritte Möglichkeit: Wenn das NFSS eine Anforderung für eine Schriftfamilie `foo` mit dem Kodierschema `BAR` erhält, die in dieser Kombination nicht bekannt ist, versucht das System, eine Datei namens `barfoo.fd` (alles in Kleinbuchstaben) zu laden. Existiert diese Datei, dann wird angenommen, dass sie Definitionen für die Gruppen von Schriftformen der Familie `foo` im Kodier-

¹ Jede derartige Anpassung sollte mit Sorgfalt durchgeführt werden, da man leider sehr einfach ein $\text{\LaTeX}$ -Format erzeugen kann, das eine unterschwellige oder sogar eine krasse Inkompatibilität zu anderen Installationen aufweist.

schema BAR enthält – also Deklarationen in folgender Form:

```
\DeclareFontFamily{BAR}{foo}{...}
\DeclareFontShape{BAR}{foo}{...}{...}{...}{...}
...
\endinput
```

Auf diese Weise lassen sich eine große Anzahl von Schriftfamilien für L^AT_EX deklarieren, ohne wertvollen Speicherplatz für Informationen zu verschwenden, die selten bis gar nicht verwendet werden.¹

Eine .fd-Datei sollte immer alle benötigten Fontdefinitionen für eine Schriftfamilie einer bestimmten Kodierung enthalten. Sie sollte aus einer oder mehreren \DeclareFontShape-Deklarationen sowie aus genau einer \DeclareFontFamily-Deklaration bestehen. Darüber hinaus sollten keine weiteren Definitionen in diesen Dateien erscheinen, außer vielleicht noch eine \ProvidesFile-Deklaration oder ein \typeout-Statement, das den Anwender darüber informiert, dass der Font geladen wird. Anstelle des \typeout-Befehls kann man auch den Plain-T_EX-Befehl \wlog verwenden, der sein Argument nur in die Protokolldatei schreibt. Prinzipiell sollten alle eingesetzten .fd-Dateien detaillierte Informationen in die Protokolldatei schreiben, da es bei der Fehlersuche sehr hilfreich ist zu wissen, welche Dateien in welcher Version an einem bestimmten L^AT_EX-Lauf beteiligt waren. Bei der Verwendung von \typeout- oder \wlog-Befehlen sollte man bedenken, dass Leerzeichen und -zeilen in einer .fd-Datei ignoriert werden. Wenn auf dem Bildschirm oder in der Protokolldatei Leerzeichen ausgegeben werden sollen, muss dazu in dem Argument der Befehl \space benutzt werden.

Über den .fd-Mechanismus können keine neuen Kodierschemata eingeführt werden. NFSS weist alle Anfragen nach Kodierschemata zurück, die nicht ausdrücklich im L^AT_EX-Format (also in der Datei fonttext.ltx), einer Paketdatei oder der Präambel des Dokumentes deklariert wurden.

7.10.7 Deklarieren neuer Fonts für Formeln

Festlegen der Schriftgrade

Für jede Textgröße führt NFSS drei Schriftgrade zur Verwendung in Formeln (vgl. Abschnitt 8.7.1): den Schriftgrad, in dem die meisten Symbole gesetzt werden (ausgewählt durch \textstyle oder \displaystyle), den Schriftgrad zum Setzen von Hoch- oder Tiefstellungen erster Ordnung (\scriptstyle) und denjenigen für Hoch- und Tiefstellungen höherer Ordnung (\scriptscriptstyle). Beim Umschalten auf eine neue Schriftgröße mit noch unbekannten Formelschriftgraden versucht NFSS diese aus der Schriftgröße des Textes zu berechnen. Stattdessen kann man die korrekten Werte aber auch selbst mithilfe der Deklaration \DeclareMathSizes festlegen. Sie besitzt vier Argumente: den Schriftgrad des umgebenden Textes und

¹Bei (L^A)T_EX-Installationen, die für die Befehle \input und \openin unterschiedliche Suchpfade verwenden, ist diese Möglichkeit jedoch leider nicht uneingeschränkt verfügbar. Dort kann man den .fd-Mechanismus während der Installation aktivieren, indem man NFSS den absoluten Pfad zu den Verzeichnissen mit all den .fd-Dateien übergibt. Lokale .fd-Dateien, also jene, die im aktuellen Verzeichnis abgelegt sind, können demnach in solchen Systemen möglicherweise nicht verwendet werden.

die drei dazu passenden Formelschriftgrade. Die Klassendatei für dieses Buch enthält beispielsweise Einstellungen wie die folgenden:

```
\DeclareMathSizes{14}{14}{10}{7}    \DeclareMathSizes{36}{-}{-}{-}
```

Die erste Deklaration legt fest, dass für die Überschriftengröße 14pt die Formelschriftgrade 14pt, 10pt bzw. 7pt benutzt werden. Die zweite Deklaration teilt NFSS mit, dass für den Schriftgrad 36pt (für Kapitelüberschriften) keine Formelschriftgrade benötigt werden. Dadurch wird das unnötige Laden von mehr als 30 zusätzlichen Fonts vermieden. Solche Deklarationen waren besonders wichtig für die erste Auflage dieses Buches, damit es mit allen Beispielen als ein einzelnes Dokument verarbeitet werden konnte (damit wurden 228 von maximal 255 Fonts geladen). Heutige T_EX-Installationen werden normalerweise mit größeren internen Tabellen kompiliert, so dass ein sparsamer Umgang mit Speicherplatz nicht mehr zu den wichtigsten Problemen gehört. (Die Laptop-Installation, mit der dieses Kapitel geschrieben wurde, erlaubt z.B. 1000 Fonts.) Beim Abschalten der Formelschriftgrade sollte man jedoch sehr vorsichtig sein, denn wenn letztendlich doch eine Formel in dem entsprechenden Schriftgrad gesetzt wird, geschieht dies in den Formelschriftgraden, die noch für eine andere, zuvor benutzte Schriftgröße aktiviert wurden.

Einbinden neuer Symbolfonts

In einem früheren Abschnitt wurde bereits erläutert, wie man durch die Alphabetbefehle in Formeln Buchstaben mit besonderen Formen erzeugen kann. An dieser Stelle soll nun besprochen werden, wie man Zeichensätze, die Sonderzeichen enthalten, so genannte „Symbolfonts“, einbindet und diese dann in Formeln verwendet.

Das Verfahren zum Einbinden neuer Symbolfonts ist vergleichbar mit dem Deklarieren neuer Alphabetbefehle: `\DeclareSymbolFont` definiert die Voreinstellung für alle Formellayouts und mit `\SetSymbolFont` wird die Voreinstellung eines bestimmten Layouts überschrieben.

Die Symbolfonts werden mit einem symbolischen Namen angesprochen, der aus einer Folge von Buchstaben besteht. Wenn man z.B. den AMS-Font `msbm10`, der in Tabelle 7.29 auf der nächsten Seite abgebildet ist, integrieren will, muss man die Schrift NFSS zunächst mithilfe der Deklarationen bekannt machen, die in den vorangegangenen Abschnitten vorgestellt wurden. Diese Anweisungen würden etwa folgendermaßen aussehen:

```
\DeclareFontFamily{U}{msb}{-}
\DeclareFontShape{U}{msb}{m}{n}{
  <5> <6> <7> <8> <9> gen * msbm
  <10> <10.95> <12> <14.4> <17.28> <20.74> <24.88> msbm10}{-}
```

Sie werden normalerweise in eine `.fd`-Datei geschrieben. Dann muss dieser Symbolfont mit folgendem Befehl für alle Formellayouts deklariert werden:

```
\DeclareSymbolFont{AMSb}{U}{msb}{m}{n}
```

Damit wird die Gruppe von Schriftformen `U/msb/m/n` unter dem symbolischen Namen `AMSb` als Symbolfont zur Verfügung gestellt. Wenn in dieser

	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	$\leq$	$\geq$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$	"0x
'01x	$\leq$	$\geq$	$\neq$	$\neq$	$\leq$	$\geq$	$\neq$	$\neq$	
'02x	$\approx$	$\approx$	$\approx$	$\approx$	$\neq$	$\neq$	$\approx$	$\approx$	"1x
'03x	$\approx$	$\approx$	$\approx$	$\approx$	$\approx$	$\neq$	/	\	
'04x	$\subseteq$	$\supseteq$	$\not\subseteq$	$\not\supseteq$	$\subseteq$	$\supseteq$	$\subseteq$	$\supseteq$	"2x
'05x	$\subseteq$	$\supseteq$	$\not\subseteq$	$\not\supseteq$	$\mathbb{H}$	$\mathbb{I}$	$\mathbb{J}$	$\mathbb{K}$	
'06x	$\mathbb{L}$	$\mathbb{M}$	$\mathbb{N}$	$\mathbb{O}$	$\mathbb{P}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{S}$	"3x
'07x	$\leftrightarrow$	$\rightarrow$	$\nrightarrow$	$\nrightarrow$	$\nrightarrow$	$\leftrightarrow$	*	$\emptyset$	
'10x	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{C}$	$\mathbb{D}$	$\mathbb{E}$	$\mathbb{F}$	$\mathbb{G}$	$\mathbb{H}$	"4x
'11x	$\mathbb{I}$	$\mathbb{J}$	$\mathbb{K}$	$\mathbb{L}$	$\mathbb{M}$	$\mathbb{N}$	$\mathbb{O}$	$\mathbb{P}$	
'12x	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{S}$	$\mathbb{T}$	$\mathbb{U}$	$\mathbb{V}$	$\mathbb{W}$	$\mathbb{X}$	"5x
'13x	$\mathbb{Y}$	$\mathbb{Z}$	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{C}$	$\mathbb{D}$	$\mathbb{E}$	$\mathbb{F}$	
'14x	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{I}$	$\mathbb{J}$	$\mathbb{K}$	$\mathbb{L}$	$\mathbb{M}$	$\mathbb{N}$	"6x
'15x	$\mathbb{O}$	$\mathbb{P}$	$\mathbb{Q}$	$\mathbb{R}$	$\mathbb{S}$	$\mathbb{T}$	$\mathbb{U}$	$\mathbb{V}$	
'16x	$\mathbb{W}$	$\mathbb{X}$	$\mathbb{Y}$	$\mathbb{Z}$	$\mathbb{A}$	$\mathbb{B}$	$\mathbb{C}$	$\mathbb{D}$	"7x
'17x	$\mathbb{E}$	$\mathbb{F}$	$\mathbb{G}$	$\mathbb{H}$	$\mathbb{I}$	$\mathbb{J}$	$\mathbb{K}$	$\mathbb{L}$	
	"8	"9	"A	"B	"C	"D	"E	"F	

Tabelle 7.29: Glyphen des msbm10 Fonts

Schriftfamilie eine fette Schriftserie vorhanden wäre (was leider nicht der Fall ist), könnte man danach die Einstellung für das Formellayout `bold` auf folgende Weise ändern:

```
\SetSymbolFont{AMSb}{bold}{U}{msb}{b}{n}
```

Nachdem der Symbolfont deklariert wurde, kann er in Formeln verwendet werden. Wie aber teilt man NFSS mit, dass beispielsweise `$a\lessdot b$` die Formel $a < b$ erzeugen soll? Dazu muss man in NFSS über den Befehl `\DeclareMathSymbol` eigene Symbolnamen festlegen.

```
\DeclareMathSymbol{befehl}{typ}{symbol-font}{slot}
```

Das erste Argument des Befehls `\DeclareMathSymbol` ist der gewählte Befehlsname. Das zweite Argument enthält einen der Befehle aus Tabelle 7.30 auf der nächsten Seite, der die Art des Symbols beschreibt, also ob es sich um einen binären Operator, eine Relation usw. handelt. (L)TeX bestimmt anhand dieser Information, wie viel umgebenden Leerraum das Symbol in einer Formel benötigt. Diese Befehle können (mit Ausnahme von `\mathalpha`) auch direkt in Formeln als Funktionen mit einem Argument verwendet werden. In diesem Falle wird für das (möglicherweise recht komplexe) Argument soviel

Typ	Bedeutung	Beispiel	Typ	Bedeutung	Beispiel
<code>\mathord</code>	Einfaches Symbol	/	<code>\mathopen</code>	Öffnendes Symbol	(
<code>\mathop</code>	Großer Operator	<code>\sum</code>	<code>\mathclose</code>	Schließendes Symbol	)
<code>\mathbin</code>	Binärer Operator	+	<code>\mathpunct</code>	Interpunktion	,
<code>\mathrel</code>	Relation	=	<code>\mathalpha</code>	Alphabetzeichen	A

Tabelle 7.30: Klassifizierung mathematischer Symbole

Zwischenraum eingefügt, als würde es zu der entsprechenden Kategorie gehören (siehe Abschnitt 8.9 auf Seite 540).

Das dritte Argument bestimmt den Symbolfont, aus dem das Zeichen geladen werden soll, d.h. es enthält den symbolischen Namen, der mit dem Befehl `\DeclareSymbolFont` festgelegt wurde. Das vierte und letzte Argument gibt die Position des Zeichens in der Fontkodierung an, und zwar entweder als Dezimal-, Oktal- oder Hexadezimalzahl. Oktal- (Basis 8) und Hexadezimalzahlen (Basis 16) werden durch ' bzw. " eingeleitet. Tabelle 7.29 auf der vorherigen Seite gibt einen Überblick über die Positionen aller Zeichen dieses Zeichensatzes. Solche Tabellen lassen sich mit dem $\text{\LaTeX}$ -Programm `nfsfont.tex` ausdrucken, das Bestandteil der $\text{\LaTeX}$ -Distribution ist; siehe dazu Abschnitt 7.5.7 auf Seite 380. `\lessdot` würde z.B. folgendermaßen deklariert:

```
\DeclareMathSymbol{\lessdot}{\mathbin}{AMSb}{"6C}
```

Anstelle eines Befehlsnamens kann im ersten Argument aber auch ein einzelnes Zeichen angegeben werden. Im `eulervm`-Paket gibt es beispielsweise mehrere Deklarationen der Form

```
\DeclareMathSymbol{0}{\mathalpha}{letters}{"30}
```

die festlegen, von wo die Ziffern geladen werden sollen.

Da `\DeclareMathSymbol` eine bestimmte Position innerhalb eines Symbolfonts adressiert, muss man darauf achten, dass alle externen Fonts, die über die Deklarationen `\DeclareSymbolFont` und `\SetSymbolFont` mit diesem Symbolfont verknüpft sind, das gleiche Zeichen an der gleichen Position aufweisen. Diese Einheitlichkeit lässt sich am einfachsten garantieren, wenn man nur Fonts mit der gleichen Kodierung einsetzt (es sei denn, es handelt sich um die Kodierung U, also die unbekannte Kodierung, die von Zeichensatz zu Zeichensatz verschieden sein kann).

Neben `\DeclareMathSymbol` kennt $\text{\LaTeX}$ auch `\DeclareMathAccent`, `\DeclareMathDelimiter` sowie `\DeclareMathRadical` zum Einrichten der Unterstützung für Mathematikfonts. Nähere Einzelheiten zu diesen etwas spezielleren Deklarationen finden sich in $\text{\LaTeX 2}_{\epsilon}$ *font selection* [109], das Bestandteil jeder $\text{\LaTeX}$ -Distribution ist.

Wer sich die Zeichentabelle für `msbm10` (Tabelle 7.29 auf der vorherigen Seite) noch einmal genau anschaut, sieht, dass dieser Font auch über „Blackboard Bold“-Buchstaben wie z.B. $\mathbb{A}$ $\mathbb{B}$ $\mathbb{C}$ verfügt. Wenn man diese Buchstaben als Mathematikalphabet verwenden will, kann man sie mithilfe des Befehls `\DeclareMathAlphabet` definieren; ist dieser Symbolfont für den Zugriff auf

einzelne Symbole jedoch bereits geladen, verwendet man besser eine Abkürzung:

```
\DeclareSymbolFontAlphabet{\mathbb}{AMSb}
```

Hier wird der Name des Alphabetbefehls und der symbolische Name des zuvor deklarierten Symbolfonts angegeben.

Ein wichtiger Grund, Symbolfonts nicht unnötig doppelt zu laden, ist die Obergrenze von 16 Mathematikfonts, die gleichzeitig in $\text{\LaTeX}$ aktiv sein können. In diese Summe fließen alle Symbolfonts ein; Mathematikalphabete zählen nur, wenn sie im Dokument tatsächlich verwendet werden, und sie zählen lokal in jedem Formellayout. Wenn also z.B. acht Symbolfonts deklariert wurden, kann man in jedem Formellayout maximal noch acht (durchaus verschiedene) Alphabetbefehle verwenden.

Zusammenfassend lässt sich Folgendes sagen: Zum Einführen neuer Symbolfonts benötigt man eine geringe Anzahl von `\DeclareSymbolFont`- und `\SetSymbolFont`-Deklarationen sowie eine möglicherweise größere Anzahl von `\DeclareMathSymbol`-Deklarationen. Daher sollten derartige Fonts am besten durch ein Paket hinzugefügt werden.

Definieren neuer Formellayouts

Wie bereits erwähnt, richtet das Standard-Setup automatisch zwei Formellayouts ein – ein normales (`normal`) und ein fettes (`bold`). Weitere Layouts lassen sich mithilfe der Deklaration `\DeclareMathVersion` einführen, die als Argument den Namen des neuen Formellayouts besitzt. Sämtliche Symbolfonts und Alphabetbefehle, die zuvor deklariert wurden, sind in diesem Formellayout automatisch verfügbar. Für sie werden die voreingestellten Schriftschnitte benutzt, die ihnen bei der Definition mit `\DeclareMathAlphabet` oder `\DeclareSymbolFont` zugewiesen wurden.

Wie in den vorangegangenen Abschnitten (Seiten 364 und 446) anhand des Formellayouts „bold“ bereits gezeigt, können die Einstellungen des neuen Formellayouts dann mit geeigneten `\SetMathAlphabet`- und `\SetSymbolFont`-Befehlen geändert werden. Auch die Einführung neuer Formellayouts wird normalerweise mithilfe einer Paketdatei vorgenommen.

Ändern der Symbolfonteinstellung

Die soeben besprochenen Befehle dienen nicht nur dem Laden neuer Symbolfonts, um zusätzliche Zeichen einzubinden, sondern können auch verwendet werden, um existierende Einstellungen zu ändern. Diese Möglichkeit ist besonders interessant, wenn man in einigen oder allen Formellayouts besondere Fonts einsetzen möchte. $\text{\LaTeX}$ verfügt über die folgenden Voreinstellungen:

```
\DeclareMathVersion{normal} \DeclareMathVersion{bold}

\DeclareSymbolFont{operators} {OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}   {OML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}   {OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols} {OMX}{cmex}{m}{n}
```

```
% Besondere fettdruckende Fonts nur für:
\SetSymbolFont    {operators}{bold}{OT1}{cmr}{bx}{n}
\SetSymbolFont    {letters}  {bold}{OML}{cmm}{b}{it}
```

Im Standard-Setup werden Ziffern und Texte, die von Befehlen wie `\log` oder `\max` erzeugt werden, aus dem Symbolfont `operators` entnommen. Um diese Elemente an die Grundschrift anzupassen, also z.B. Computer Modern Sans anstelle von Computer Modern Roman zu verwenden, kann man die folgenden Befehle formulieren:

```
\SetSymbolFont{operators}{normal}{OT1}{cmss}{m} {n}
\SetSymbolFont{operators}{bold}  {OT1}{cmss}{bx}{n}
```

Die Symbolfonts `symbols` und `largesymbols` haben in T_EX eine Sonderstellung, und benötigen eine Reihe zusätzlicher `\fontdimen`-Parameter. Deshalb können für sie nur speziell ausgearbeitete Fonts verwendet werden. Prinzipiell können jedem beliebigen Zeichensatz während des Ladevorgangs solche zusätzlichen Parameter zugewiesen werden und zwar mithilfe des dritten Argumentes von `\DeclareFontFamily` oder des sechsten Argumentes von `\DeclareFontShape`. Informationen zu den speziellen Parametern für diese Symbolfonts befinden sich in Anhang G von [87].

7.10.8 Beispiel: Definieren eigener .fd-Dateien

Wer neue Fonts und die dafür notwendigen .fd-Dateien erstellen möchte, sollte dafür das in diesem Abschnitt bereits erläuterte Verfahren anwenden. Wenn man die erforderlichen Fontmetrikdateien mithilfe des `fontinst`-Paketes [74] erzeugt, dann werden die entsprechenden .fd-Dateien automatisch ebenfalls generiert. Eine .fd-Datei für eine einzelne Schriftfamilie lässt sich jedoch auch leicht von Hand erstellen, sofern man die Zeichensatzkodierung kennt. Als Beispiel soll hier die Deklarationsdatei `t1bch.fd` für die Schrift Bitstream Charter in der T1-Kodierung untersucht werden:

```
\ProvidesFile{t1bch.fd}[2001/06/04 font definitions for T1/bch.]
% Primäre Deklarationen
\DeclareFontFamily{T1}{bch}{}
\DeclareFontShape{T1}{bch}{m}{n}{<-> bchr8t}{}
\DeclareFontShape{T1}{bch}{m}{sc}{<-> bchrc8t}{}
\DeclareFontShape{T1}{bch}{m}{sl}{<-> bchro8t}{}
\DeclareFontShape{T1}{bch}{m}{it}{<-> bchri8t}{}
\DeclareFontShape{T1}{bch}{b}{n}{<-> bchb8t}{}
\DeclareFontShape{T1}{bch}{b}{sc}{<-> bchbc8t}{}
\DeclareFontShape{T1}{bch}{b}{sl}{<-> bchbo8t}{}
\DeclareFontShape{T1}{bch}{b}{it}{<-> bchbi8t}{}
% Ersetzungen
\DeclareFontShape{T1}{bch}{bx}{n}{<->ssub * bch/b/n}{}
\DeclareFontShape{T1}{bch}{bx}{sc}{<->ssub * bch/b/sc}{}
\DeclareFontShape{T1}{bch}{bx}{sl}{<->ssub * bch/b/sl}{}
\DeclareFontShape{T1}{bch}{bx}{it}{<->ssub * bch/b/it}{}
\endinput
```

Die Datei beginnt mit einer Identifikationszeile und deklariert danach mithilfe von `\DeclareFontFamily` die Schriftfamilie und -kodierung (also bch in T1). Die Argumente dieses Befehls sollten dabei dem Namen der `.fd`-Datei entsprechen, außer dass dort die Kodierung den Konventionen entsprechend klein geschrieben wird. Als nächstes wird jede Kombination von Schriftserie und -form einer `.tfm`-Datei zugewiesen. Diese Fonts sind beliebig skalierbar, so dass ihre `\DeclareFontShape`-Befehle die Deklaration `<->` enthalten. Im zweiten Teil der Datei wird festgelegt, welcher Ersatz verwendet werden soll, wenn eine Kombination nicht verfügbar ist (d.h. Ersetzen der breitfetten Serie durch die fett gedruckte).

Sofern man die zusätzlichen Charter Fonts (Black und BlackItalic) erworben hat, die *nicht* kostenlos erhältlich sind, möchte man die `.fd`-Datei vielleicht um die entsprechenden Deklarationen ergänzen. Zunächst muss man allerdings (z.B. mithilfe von `fontinst`) geeignete virtuelle Fonts bereitstellen, um die T1-Kodierung nachzubilden. Glücklicherweise können diese für eine Vielzahl von Schriften aus dem Internet heruntergeladen werden.¹

*Besondere
Lizenzbedingungen
für .fd-Dateien*

Im Gegensatz zu den meisten anderen Dateien in der $\text{\LaTeX}$ -Welt, darf man `.fd`-Dateien gemäß der für sie üblichen Lizenz verändern, ohne sie umbenennen zu müssen. Allerdings darf man diese veränderten Dateien normalerweise nicht in Umlauf bringen!

Ein weiterer möglicher Grund zum Erstellen eigener `.fd`-Dateien kann darin liegen, dass man Schriften mehrerer Familien kombinieren und $\text{\LaTeX}$ als eine einzige neue Schriftfamilie vorstellen möchte. So entwarf Hermann Zapf beispielsweise 1954 die Schriftfamilie Aldus als Begleitung für seine Palatino-Schrift (die ursprünglich als reine Plakatschrift gestaltet worden war). Da Aldus nicht über eine fett gedruckte Serie verfügt, ist Palatino ein nahe liegender Ersatz für die fette Variante. Im nächsten Beispiel wird Aldus (mit Mediävalziffern) in normaler Serie mit Palatino in fetter Serie kombiniert. Das Ergebnis ist die „Schriftfamilie“ `zasj`. Hier nur ein Ausschnitt der vollständigen `.fd`-Datei, mit deren Hilfe Beispiel 7-10-1 auf der gegenüberliegenden Seite gesetzt wurde:

```
\ProvidesFile{t1zasj.fd}[2003/10/12 Fontdefinitionen für die
                               T1 Aldus/Palatino-Kombination]

\DeclareFontFamily{T1}{zasj}{}
% Normale Schriftserie
\DeclareFontShape{T1}{zasj}{m}{n} {<->pasr9d}{}
\DeclareFontShape{T1}{zasj}{m}{sc}{<->pasrc9d}{}
\DeclareFontShape{T1}{zasj}{m}{it}{<->pasri9d}{}
\DeclareFontShape{T1}{zasj}{m}{sl}{<->ssub * pasj/m/it}{}
% Fette Schriftserie
\DeclareFontShape{T1}{zasj}{b}{n}{<-> pplb8t}{}
\DeclareFontShape{T1}{zasj}{b}{sc}{<->pplbc8t}{}
\DeclareFontShape{T1}{zasj}{b}{sl}{<->pplbo8t}{}
\DeclareFontShape{T1}{zasj}{b}{it}{<->pplbi8t}{}

```

¹Eine gute Quelle für diesen Zweck ist die Homepage von Walter Schmidt: <http://home.vr-web.de/~was/fonts.html>.

Um auf diese „Pseudofamilie“ zugreifen zu können, muss man `zasj` in der T1-Kodierung auswählen. Außerdem muss garantiert sein, dass `\textbf` zur fetten und nicht zur breitfetten Serie wechselt, da diese `.fd`-Datei keine Ersetzungen vorsieht. All das lässt sich automatisch über ein kleines Paket wie das folgende (namens `fontmix.sty`) bewirken:

```
\ProvidesPackage{fontmix}[2003/10/12 T1 Aldus/Palatino-Kombination.]
\RequirePackage[T1]{fontenc}
\renewcommand\rmdefault{zasj} \renewcommand\bfdefault{b}
```

Durch Laden von `fontmix` erhält man nun also Aldus mit Überschriften in Palatino Bold. In vielen Fällen stellt eine solche Kombination keine Verbesserung für den Text dar. Dieses Beispiel sollte daher nicht als Ermutigung verstanden werden, beliebige Schriften miteinander zu mischen.

Palatino und Aldus von Zapf

Dieser Text ist in der Schrift Aldus mit passenden *Mediävalziffern* gesetzt „1234567890“.

Als begleitende **fette Schrift** wurde die Palatino von Zapf gewählt.

```
\usepackage{fontmix}
% t1zasj.fd und fontmix.sty wie definiert
\section*{Palatino und Aldus von Zapf}
Dieser Text ist in der Schrift Aldus mit
passenden \emph{Mediävalziffern} gesetzt
"‘1234567890’”.
```

Als begleitende `\textbf{fette Schrift}` wurde die Palatino von Zapf gewählt.

Bsp.
7-10-1

7.10.9 Die Reihenfolge der Deklarationen

NFSS schreibt für alle Deklarationen eine bestimmte Reihenfolge vor, so dass das System prüfen kann, ob alle erforderlichen Angaben gemacht wurden. Wird die Reihenfolge nicht eingehalten, so erfolgt eine Fehlermeldung. Die folgenden Beziehungen der Deklarationen zueinander müssen beachtet werden:

- `\DeclareFontFamily` überprüft, ob das Kodierschema zuvor mit dem Befehl `\DeclareFontEncoding` deklariert wurde.
- `\DeclareFontShape` überprüft, ob die verwendete Schriftfamilie so deklariert wurde, dass sie in der angeforderten Kodierung verfügbar ist (`\DeclareFontFamily`).
- `\DeclareSymbolFont` überprüft, ob das Kodierschema gültig ist.
- `\SetSymbolFont` stellt zusätzlich sicher, dass sowohl das angeforderte Formellayout (`\DeclareMathVersion`) als auch der angeforderte Symbolfont (`\DeclareSymbolFont`) bereits deklariert sind.
- `\DeclareSymbolFontAlphabet` überprüft, ob der Befehlsname für den Alphabetbefehl verwendet werden kann und ob der Symbolfont zuvor deklariert wurde.
- `\DeclareMathAlphabet` überprüft, ob der ausgewählte Befehlsname verwendet werden kann und ob das Kodierschema bereits deklariert wurde.

- `\SetMathAlphabet` überprüft, ob der verwendete Alphabetbefehl zuvor mit `\DeclareMathAlphabet` oder `\DeclareSymbolFontAlphabet` deklariert wurde und ob das Formellayout und das Kodierschema bekannt sind.
- `\DeclareMathSymbol` stellt sicher, dass der Befehlsname verwendet werden kann (d.h. dass er entweder noch nicht definiert ist oder vorher als mathematisches Symbol deklariert war) und dass der Symbolfont zuvor deklariert wurde.
- Bei Erreichen des Befehls `\begin{document}` führt NFSS schließlich noch einige zusätzliche Überprüfungen durch, z.B. ob die Voreinstellungen zur Fontersetzung für jedes Kodierschema auch tatsächlich auf bekannte Deklarationen von Schriftformgruppen verweisen.

7.11 Kodierungsmodelle in $\text{\LaTeX}$

Den meisten Anwendern wird wahrscheinlich das Wissen genügen, dass es bestimmte Eingabe- und Fontkodierungen gibt und wie man im Allgemeinen, entsprechend der Beschreibungen in den vorherigen Abschnitten, mit ihnen arbeiten kann. Manchmal ist es jedoch hilfreich, sich intensiver mit der Materie zu beschäftigen, um beispielsweise eine eigene Kodierung einrichten zu können oder besser zu verstehen, wie besondere Funktionen in Pakete oder Klassen umgesetzt sind. Darum erklärt dieser Abschnitt alles, was man schon immer über Kodierungen in $\text{\LaTeX}$ wissen wollte.

Zunächst wird der generelle Datenfluss der Zeichen im $\text{\LaTeX}$ -System beschrieben, aus dem die grundlegenden Anforderungen für verschiedene Kodierungen und ihre Zuordnung zueinander abgeleitet werden. Dann wird das interne Darstellungsmodell für die Zeichendaten in $\text{\LaTeX}$ genauer betrachtet, gefolgt von einer Besprechung der Mechanismen, mit deren Hilfe eintreffende Daten durch Eingabekodierungen ihren internen Darstellungen zugeordnet werden.

Und schließlich wird erklärt, wie die internen Darstellungen mittels der Fontkodierungen in die Form übersetzt werden, die schließlich für die eigentliche Ausgabe benötigt wird.

Dargestellte Zeichen

Ziffern:	0 1 2 3 4 5 6 7 8 9
Kleinbuchstaben:	a b c d e f g h i j k l m n o p q r s t u v w x y z
Großbuchstaben:	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Satzzeichen:	. , ; : ? ! ' '
Verschiedene Symbole:	* + - = () [] / @

Nicht dargestellte Zeichen

$\text{\TeX}$ -Syntaxzeichen:	\$ ^ _ { } # & % \ ~
In (einigen) OT1-Fonts fehlen	< > "

Tabelle 7.31: LICR-Objekte, die durch einzelne Zeichen dargestellt werden

7.11.1 Zeichendaten im L^AT_EX-System

Die Verarbeitung eines Dokumentes im L^AT_EX-System beginnt mit der Auswertung von Daten aus einer oder mehreren Quelldateien. Diese Daten, die den Inhalt des Dokumentes repräsentieren, sind in den Dateien in Form von Oktetten gespeichert, welche die jeweiligen Zeichen darstellen. Um diese Oktette richtig interpretieren zu können, muss L^AT_EX (oder jedes andere Programm, mit dem die Datei verarbeitet wird, wie etwa ein Editor) wissen, mit welcher Kodierung die Datei erstellt wurde. In anderen Worten: Es muss die Zuordnung zwischen den abstrakten Zeichen und den sie repräsentierenden Oktetten kennen.

Bei einer fehlerhaften Zuordnung erfolgt die weitere Verarbeitung mehr oder weniger fehlerhaft, es sei denn, die Datei verwendet nur einen Zeichenvorrat, der in beiden Kodierungen identisch ist.¹

An diesem Punkt geht L^AT_EX grundsätzlich von der Annahme aus, dass (fast) alle sichtbaren ASCII-Zeichen (Dezimal 32–126) durch ihre Nummer in der ASCII-Zeichensatztafel dargestellt werden (siehe Tabelle 7.31 auf der vorherigen Seite).

Für diese Annahme gibt es sowohl einen praktischen als auch einen T_EXnischen Grund. Der praktische Grund ist, dass die meisten heutzutage gebräuchlichen 8-Bit-Kodierungen eine gemeinsame 7-Bit-Ebene haben. Der T_EXnische Grund liegt darin, dass für eine effektive Nutzung von T_EX² der Großteil des sichtbaren ASCII-Bereiches entweder als Zeichen der Kategorie „letter“ (Buchstaben) verarbeitet werden muss, da nur Zeichen dieser Kategorie in mehrbuchstabigen T_EX-Befehlsnamen verwendet werden dürfen; oder sie müssen zur Kategorie „other“ (andere) gehören, da T_EX z.B. dezimale Ziffern nicht als Teil einer Zahl erkennt, wenn sie nicht diese Kategorie haben.

Wird ein Zeichen – genauer gesagt eine 8-Bit-Zahl – in T_EX mit der Kategorie „letter“ oder „other“ deklariert, so wird diese 8-Bit-Ziffer einfach transparent durchgereicht. T_EX setzt also in der Ausgabe jedes beliebige Zeichen, das gerade im Zeichensatz unter dieser Nummer geführt wird.

Als Konsequenz dieser zuvor erwähnten Annahme müssen Zeichensätze für allgemeinen Text die (meisten) sichtbaren ASCII-Zeichen enthalten und diese müssen der ASCII-Kodierung entsprechend positioniert sein. Die genaue Liste ist in Tabelle 7.31 auf der gegenüberliegenden Seite dargestellt.

Allen anderen 8-Bit-Zahlen (also denjenigen außerhalb des sichtbaren ASCII-Bereiches), die sich in der Eingabedatei befinden können, wird die Kategorie „active“ (aktiv) zugewiesen, so dass sie sich innerhalb von T_EX wie Befehle verhalten. Dadurch kann L^AT_EX sie mithilfe der Eingabekodierung in ei-

*L^AT_EX-interne
Zeichendarstellung
(LICR)*

¹Da die meisten westlichen Kodierungen einen Großteil der ASCII-Zeichen (d.h. weitgehend die 7-Bit-Ebene) gemeinsam haben, sind Dokumente, die hauptsächlich aus akzentlosen, lateinischen Buchstaben bestehen, auch dann noch verständlich, wenn sie mit einer anderen als ihrer ursprünglichen Kodierung betrachtet oder verarbeitet werden. Je mehr Zeichen jedoch außerhalb der sichtbaren ASCII-Kodierung verwendet werden, desto unverständlicher wird der Text. Ein Text kann völlig sinnlos werden, wenn z.B. griechische oder russische Dokumente in der Annahme verarbeitet werden, sie verwendeten die Kodierung für das US-amerikanische Windows.

²Das traf zumindest zu, als diese Schnittstelle gestaltet wurde. Mit den heutigen, wesentlich schnelleren Computern könnte man das Eingabeverfahren in T_EX radikal verändern, indem man es im Prinzip komplett abschaltet und die Eingabedaten manuell, also Zeichen für Zeichen verarbeitet.

ne Form überführen, die $\text{\LaTeX}$ Internal Character Representation (LICR, $\text{\LaTeX}$ -interne Zeichendarstellung) genannt wird.

Die Unicode UTF-8-Kodierung wird ebenso gehandhabt: Die ASCII-Zeichen repräsentieren sich selbst und die Anfangsoktette der Mehrbyte-Entsprechungen suchen als aktive Zeichen die Eingabedaten nach den verbleibenden Oktetten ab. Das Ergebnis wird in ein LICR-Objekt umgewandelt, sofern eine Zuordnung existiert, oder es erzeugt eine Fehlermeldung, wenn das entsprechende Unicode-Zeichen nicht zugeordnet ist.

Die wichtigste Eigenschaft aller LICR-Objekte ist, dass ihre Darstellung in 7-Bit-ASCII erfolgt (typischerweise als Befehle) und damit von jeder Änderung der Kodierung unabhängig ist, da sich alle Eingabekodierungen hinsichtlich der sichtbaren ASCII-Zeichen transparent verhalten sollen. Dadurch kann $\text{\LaTeX}$ beispielsweise Hilfsdateien (z.B. `.toc`-Dateien) in der LICR-Darstellung verfassen, und sie in einem anderen Kontext (möglicherweise sogar einer anderen Kodierung) zurück übertragen, ohne dass dabei Fehlinterpretationen auftreten.

Aufgabe der Font- (oder Ausgabe-) Kodierung ist es dann, die internen Zeichendarstellungen den Glyphenpositionen des aktuellen Zeichensatzes für die Ausgabe zuzuordnen, oder in einigen Fällen auch komplexere Vorgänge auszulösen. Sie könnte z.B. ein Akzentzeichen (von einer Position im aktuellen Font) über einer Glyphe (von einer anderen Position im aktuellen Font) platzieren, um damit ein gedrucktes Bild des abstrakten Zeichens zu erhalten, das durch den oder die Befehle der internen Zeichenkodierung dargestellt wurde.

Da die LICR alle Zeichen kodiert, die in $\text{\LaTeX}$ adressiert werden können, ist sie wesentlich umfangreicher als ein einzelner $\text{\TeX}$ -Font (der maximal 256 Glyphen enthalten kann). In einigen Fällen lassen sich Zeichen der internen Kodierung durch Kombinationen von Glyphen eines Fonts nachbilden, wie etwa die zuvor erwähnten Akzentbuchstaben. Wenn dieses interne Zeichen jedoch eine besondere Form benötigt (wie z.B. das Währungssymbol „ $\text{\textcircled{C}}$ “), lässt es sich auch nicht imitieren, wenn es im Zeichensatz nicht vorhanden ist.

Das $\text{\LaTeX}$ -Modell für die Zeichenkodierung unterstützt jedoch auch automatische Abläufe, in denen Glyphen aus anderen Fonts herangezogen werden, damit selbst ein im aktuellen Font fehlendes Zeichen gesetzt werden kann – natürlich vorausgesetzt, es ist ein geeigneter zusätzlicher Font mit dem betreffenden Zeichen verfügbar.

7.11.2 Die $\text{\LaTeX}$ -interne Zeichendarstellung (LICR)

Technisch ausgedrückt, werden Textzeichen $\text{\LaTeX}$ -intern auf eine von drei Arten dargestellt. Diese Arten werden in den folgenden Abschnitten besprochen.

Darstellung als Zeichen

Eine kleine Gruppe von Zeichen wird durch „sich selbst“ dargestellt, das lateinische A z.B. durch das Zeichen „A“. Die auf diese Art dargestellten Zeichen sind in Tabelle 7.31 auf Seite 452 aufgeführt. Sie bilden eine Untergruppe der ASCII-Zeichen und sind in $\text{\TeX}$ alle der Kategorie „letter“ oder „other“ zugeordnet. Einige der sichtbaren ASCII-Zeichen werden nicht auf diese Art dargestellt,

entweder weil es sich um Zeichen der T_EX-Syntax handelt¹, oder weil sie nicht in allen Zeichensätzen vorkommen. Wenn man beispielsweise das Zeichen „<“ im Text benutzt, bestimmt die aktuelle Fontkodierung, ob man im Ausdruck ein „<“ (T1) oder vielleicht ein „ı“ (OT1) erhält.²

Darstellung als Zeichenfolge

Der interne Ligaturmechanismus von T_EX unterstützt die Bildung neuer Zeichen aus einer Folge eingegebener Zeichen. Obwohl es sich hier eigentlich um eine Fonteigenschaft handelt, wurden einige dieser Zeichenfolgen ausdrücklich als Kürzel für Zeichen definiert, deren Eingabe über die meisten Tastaturen ansonsten problematisch wäre. Nur sehr wenige der so erzeugten Zeichen werden als Bestandteil der L^AT_EX-internen Zeichendarstellung angesehen. Dazu gehören der Halbgeviertstrich (en-Strich, Gedankenstrich) und der Geviertstrich (em-Strich), welche durch die Ligaturen „--“ und „---“ erzeugt werden, sowie die öffnenden und schließenden englischen Anführungszeichen, die durch ‘ ‘ und ’ ’ erzeugt werden.³ Manche Fonts setzen außerdem ! ‘ und ? ‘ ein, um ı und ı zu erzeugen; diese Eigenschaft wird aber nicht generell von allen Zeichensätzen unterstützt. Aus diesem Grund gibt es für *alle* diese Zeichen eine alternative interne Darstellung als Befehl (z.B. `\textendash` oder `\textexclamdown`).

Darstellung als „fontkodierungsspezifische“ Befehle

Des Weiteren lassen sich Zeichen (und dies gilt für die Mehrzahl der Zeichen) L^AT_EX-intern als besondere L^AT_EX-Befehle (oder Befehlsfolgen) darstellen, deren Ausführung unterbunden wird, wenn sie in eine Datei geschrieben werden oder in einem bewegten Argument vorkommen. Diese Spezialbefehle werden manchmal auch „fontkodierungsspezifische“ Befehle genannt, da ihre Bedeutung von der Fontkodierung abhängt, die aktiv ist, wenn L^AT_EX sie ausführt. Derartige Befehle werden, wie im Folgenden noch besprochen, mittels besonderer Deklarationen definiert. Diese Definitionen sind dabei normalerweise für jede Fontkodierung unterschiedlich. Wenn für die aktive Kodierung keine Definition existiert, wird, soweit vorhanden, auf eine Voreinstellung zurückgegriffen oder der Anwender erhält eine Fehlermeldung.

Technisch gesehen ändern sich die Definitionen der kodierungsspezifischen Befehle nicht unmittelbar, sobald die Kodierung gewechselt wird, da sonst auf einen Schlag eine Vielzahl von Befehlen geändert werden müsste. Stattdessen wurden die Befehle so angelegt, dass sie bei ihrer Ausführung bemerken, ob ihre aktuelle Definition noch zu der aktiven Fontkodierung passt.

¹Die L^AT_EX-Syntax kennt noch einige weitere Zeichen, wie etwa *[] . Sie spielen eine Doppelrolle, da sie auch verwendet werden, um normale Textzeichen darzustellen. Es ist manchmal nicht so einfach, diese beiden Bedeutungen auseinander zu halten, was zu Problemen führen kann. So ist etwa ein] in einem optionalen Argument nur zulässig, wenn es durch geschweifte Klammern abgesichert wird; andernfalls wird es von L^AT_EX als Ende des Argumentes interpretiert.

²Diese Situation trifft für Texte zu. In Formeln hat „<“ eine genau festgelegte Bedeutung: „Erzeuge ein kleiner-als-Zeichen“.

³Die Benutzung des einzelnen "-Zeichens als schließendes Anführungszeichen ist nicht korrekt, da es in manchen Zeichensätzen ein senkrechtes Anführungszeichen, also " anstelle von ", erzeugt!

Wenn diese nicht passt, rufen sie ihre Entsprechungen für die aktive Fontkodierung auf, die dann die eigentliche Aufgabe erledigen.

Es gibt keinen fest definierten Satz fontkodierungsspezifischer Befehle in $\text{\LaTeX}$. Sie sind implizit definiert als die Gesamtheit aller Befehle, die kodierungsabhängige Definitionen besitzen. Daher können durch neue Fontkodierungen mit zusätzlichen Symbolen auch neue fontkodierungsspezifische Befehle hinzukommen.

7.11.3 Eingabekodierungen

Sobald das Paket `inputenc` (mit oder ohne Optionen) geladen ist, sind die beiden Deklarationen `\DeclareInputText` und `\DeclareInputMath` für die Zuordnung von 8-Bit-Eingabezeichen zu LICR-Objekten verfügbar. Sie sollten nur in Eingabekodierungsdateien (im Folgenden beschrieben), Paketen oder, soweit erforderlich, der Präambel von Dokumenten verwendet werden.

Ihr erstes Argument besteht aus einer 8-Bit-Zahl, bei der es sich um eine Dezimalzahl (z.B. 239), eine Oktalzahl (z.B. '357) oder einen Hexadezimalwert (z.B. "EF) handeln kann. Es ist ratsam, die dezimale Schreibweise zu verwenden, da die Zeichen ' und " durch ein Paket für eine bestimmte Sprache eine besondere Bedeutung, z.B. als Kürzel für Akzente, erhalten können, so dass die oktale oder hexadezimale Schreibweise nicht mehr richtig interpretiert wird, wenn die Pakete in der falschen Reihenfolge geladen werden.

$$\backslash\text{DeclareInputText}\{zahl\}\{LICR\text{-}objekt\}$$

Der Befehl `\DeclareInputText` deklariert Tastaturcodezuordnungen für Textzeichen. Sein zweites Argument enthält den kodierungsspezifischen Befehl (oder die Befehlsfolge), bei dem es sich um das LICR-Objekt handelt, welchem der Tastaturcode zugeordnet werden soll. Das Beispiel

$$\backslash\text{DeclareInputText}\{239\}\{"\text{\i}\}$$

ordnet der Zahl 239 die kodierungsspezifische Darstellung des Zeichens $\text{\i}$ zu, nämlich `"\i`. Auf diese Weise deklarierte Tastaturcodes lassen sich nicht in Formeln verwenden.

$$\backslash\text{DeclareInputMath}\{zahl\}\{formel\text{-}objekt\}$$

Tastaturcodes, die sich auf Zeichen für mathematische Formeln beziehen, müssen mit der Deklaration `\DeclareInputMath` definiert werden. In der Eingabekodierung `cp437de` (deutsche MS-DOS-Tastatur) verknüpft zum Beispiel die Deklaration

$$\backslash\text{DeclareInputMath}\{224\}\{\backslash\alpha\}$$

die Nummer 224 mit dem Befehl `\alpha`. Dabei ist zu beachten, dass durch diese Deklaration die Taste, mit welcher diese Zahl erzeugt wird, nur noch im Formelmodus verwendet werden kann, da `\alpha` dann nirgendwo anders mehr zulässig ist.

`\DeclareUnicodeCharacter{hex-zahl}{LICR-objekt}`

Diese Deklaration ist nur bei Verwendung der Option `utf8` verfügbar. Sie ordnet Unicode-Zahlen passende LICR-Objekte (d.h. Zeichen für Texte) zu, wie zum Beispiel:

```
\DeclareUnicodeCharacter{00A3}{\textsterling}
\DeclareUnicodeCharacter{011A}{\v E}
\DeclareUnicodeCharacter{2031}{\textpertenthousand}
```

Theoretisch sollte es nur eine einzige, eindeutige bidirektionale Zuordnung der beiden Namensräume zueinander geben, so dass all diese Deklarationen schon automatisch vorgenommen werden könnten, sobald die Option `utf8` ausgewählt wird. In der Praxis stellt sich die Situation jedoch etwas komplizierter dar. Zum einen wäre es nicht vernünftig, automatisch die gesamte Tabelle zur Verfügung zu stellen, da sie einen riesigen Teil des T_EX-Speichers belegen würde. Außerdem gibt es viele Unicode-Zeichen, für die (bisher) noch kein LICR-Objekt existiert, und umgekehrt haben viele LICR-Objekte keine Entsprechung in Unicode.¹ Das `inputenc`-Paket löst dieses Problem, indem es nur diejenigen Unicode-Zuordnungen lädt (soweit sie bekannt sind), die zu den Fontkodierungen passen, die in einem bestimmten Dokument verwendet werden. Auf Anfragen nach anderen Unicode-Zeichen reagiert es mit einer geeigneten Fehlermeldung. Damit ist es dem Anwender überlassen, entweder die richtigen Zuordnungsinformationen bereitzustellen, oder, falls erforderlich, eine weitere Fontkodierung zu laden.

Wie bereits erwähnt, kann man Deklarationen für Eingabekodierungen auch in Paketen oder der Präambel eines Dokumentes einsetzen. Dazu muss zuerst das `inputenc`-Paket geladen werden, womit eine geeignete Kodierung ausgewählt wird. Die darauf folgenden Deklarationen für Eingabekodierungen ersetzen (oder ergänzen) die Definitionen der aktiven Eingabekodierung.

Wenn man das `inputenc`-Paket benutzt, begegnet man möglicherweise zwei internen Befehlen. Der Befehl `\IeC` wird unter bestimmten Umständen intern von der Deklaration `\DeclareInputText` verwendet. Er sorgt dafür, dass beim Schreiben eines kodierungsspezifischen Befehls in eine Datei ein ihm folgendes Leerzeichen nicht einfach verschluckt wird, wenn die Datei wieder eingelesen wird. Dieser Ablauf erfolgt automatisch, so dass ein Anwender diesen Befehl niemals zu schreiben braucht. Er wird hier nur erwähnt, da er in `.toc`- oder anderen Hilfsdateien auftauchen kann.

Der andere Befehl, `\@tabacckludge`, ist eine Abkürzung für „tabbing accent kludge“ („schlechte“ Behelfslösung für Akzente in tabbing-Umgebungen). Er wird (leider) benötigt, da die aktuelle L^AT_EX-Version eine Mehrfachbedeutung der Befehle `\=`, `\‘` und `\’` geerbt hat, die normalerweise für bestimmte Akzente stehen (also kodierungsspezifische Befehle sind), jedoch in der tabbing-Umgebung eine andere Aufgabe haben. Darum müssen Zuordnungen im Zusammenhang mit diesen Akzenten besonders kodiert

¹Das mag eine überraschende Aussage sein, aber man bedenke, dass beispielsweise Akzentbefehle wie `\"` in Kombination mit einem anderen Zeichen ein eigenes LICR-Objekt wie `\"d` bilden (egal ob das sinnvoll ist oder nicht). Viele dieser Kombinationen kommen in Unicode nicht vor.

werden. Wenn man z.B. die Zahl 232 dem Zeichen è zuordnen möchte, das intern als \’e dargestellt wird, darf man nicht schreiben:

```
\DeclareInputText{232}{\’e}
```

sondern:

```
\DeclareInputText{232}{\@tabacckludge’e}
```

Die zweite Deklaration funktioniert überall, auch innerhalb der tabbing-Umgebung.

Zuordnung zu Text und/oder Formeln

Aus technischen und konzeptionellen Gründen unterscheidet T_EX sehr streng zwischen Zeichen, die in Texten bzw. solchen, die in Formeln verwendet werden können. Anders als die sichtbaren ASCII-Zeichen, können Befehle, die Zeichen erzeugen, normalerweise nur entweder im Text- oder im Formelmodus genutzt werden, nicht aber in beiden.

Bei einigen Tasten der Tastatur lässt sich aber leider nicht eindeutig bestimmen, ob die von ihnen erzeugten Zeichen als Eingaben für Formeln oder für Texte angesehen werden sollten. Sollte beispielsweise der Tastaturcode, der das Zeichen ± erzeugt, dem Befehl \textpm zugeordnet werden, der kodierungsspezifisch ist und damit nur im Text verwendet werden kann, oder eher dem Befehl \pm, womit die Taste sich nur in Formeln nutzen lässt?

Die frühen Versionen des Paketes inputenc arbeiteten nach der folgenden Strategie: Alle Tastaturcodes, die in T_EX-Standardzeichensätzen für Text verfügbar waren (d.h. in Fonts mit der Kodierung OT1 oder T1), wurden kodierungsspezifischen Befehlen zugeordnet; für die übrigen Codes erfolgte eine Zuordnung zu Formelbefehlen. Eine solche Strategie, die sich allein an der Verfügbarkeit von Glyphen orientierte, hatte jedoch den Nachteil, dass nur Anwender mit tief gehenden T_EX-Kenntnissen auf Anhieb wussten, ob z.B. ein Code namens „¾“ oder „³“ nur für Text oder nur für Formeln zulässig war.¹

Wie lässt sich diese Problematik nun elegant lösen? Im babel-Paket wird durch den Befehl \textormath der aktuelle Modus abgefragt:

```
\ifmmode \ddots a\else \"a\fi
```

Dieser Ansatz schlägt jedoch in Ausrichtungsstrukturen für Formeln fehl, da dort der falsche Teil der Bedingung ausgewählt wird, was gewöhnlich zu einer abstrusen T_EX-Fehlermeldung führt. Wenn man dieses Problem behebt, indem man der obigen Konstruktion den Befehl \relax voranstellt, werden Unterschneidungen und Ligaturen verhindert, die wiederum durchaus in einem Wort vorkommen können. Dieses Problem lässt sich tatsächlich nicht in T_EX lösen. Es lässt sich jedoch beheben, wenn man eT_EX als Basis-Satzprogramm für L^AT_EX verwendet, und da eT_EX heutzutage auf fast jedem T_EX-System vorhanden ist, gibt es Pläne, dieses Programm als Basis für zukünftige L^AT_EX-Updates einzusetzen.

¹In den ersten Versionen des inputenc-Paketes war „¾“ ein Textzeichen, „³“ jedoch ein Formelzeichen – alles klar?

Als dieses Buch geschrieben wurde, waren die Arbeiten an einer Erweiterung des Paketes `inputenc` (auf der Basis von e^T_EX) bereits in vollem Gange. Diese vorgesehene Erweiterung wird automatisch alle verfügbaren Tastatursymbole in Texten und Formeln unterstützen. Sobald es veröffentlicht ist, kann man Formeln ganz bequem setzen, indem man das `inputenc`-Paket einfach mit der Option `math` lädt.

Eingabekodierungsdateien für 8-Bit-Kodierungen

Eingabekodierungen werden in Dateien mit dem Namen der Eingabekodierung und der Erweiterung `.def` gespeichert (z.B. `latin1.def`). Diese Dateien sollten nur die in diesem Abschnitt beschriebenen Befehle enthalten.

Sie sollten mit einer `\ProvidesFile`-Deklaration beginnen, in der die Art der Datei beschrieben ist, wie zum Beispiel:

```
\ProvidesFile{latin1.def}[2000/07/01 v0.996 Input encoding file]
```

Wenn es Zuordnungen zu kodierungsspezifischen Befehlen gibt, die nur verfügbar sind, wenn zusätzliche Pakete geladen werden, kann man ihre Voreinstellungen mit dem Befehl `\ProvideTextCommandDefault` deklarieren, wie zum Beispiel:

```
\ProvideTextCommandDefault{\textonehalf}{\ensuremath{\frac{1}{2}}}  
\ProvideTextCommandDefault{\textcent}{\TextSymbolUnavailable\textcent}
```

Der hier verwendete Befehl `\TextSymbolUnavailable` gibt eine Warnung aus, die darauf hinweist, dass ein bestimmtes Zeichen in den aktuellen Fonts nicht verfügbar ist. Er kann als Voreinstellung nützlich sein, wenn die betreffenden Zeichen nur nach Laden spezieller Fonts verfügbar sind und sie nicht auf geeignete Weise aus vorhandenen Zeichen nachgebildet werden können (wie es z.B. bei der ebenfalls aufgeführten Voreinstellung für `\textonehalf` möglich war).

Der übrige Teil der Datei sollte nur aus Deklarationen für die Eingabekodierung durch `\DeclareInputText` oder `\DeclareInputMath` bestehen. Wie bereits erwähnt, wird von dem Gebrauch des letzteren Befehls abgeraten, auch wenn er zulässig ist. Eine Eingabekodierungsdatei sollte keine weiteren Befehle enthalten, insbesondere keine Befehle, die verhindern, dass die Datei mehrmals gelesen werden kann (wie z.B. `\newcommand`), da Kodierungsdateien häufig mehrmals im gleichen Dokument geladen werden!

Eingabezuordnungsdateien für UTF-8

Wie zuvor erwähnt, erfolgt die Zuordnung von Unicode zu LICR-Objekten nicht in einer einzelnen großen Zuordnungsdatei. Sie ist vielmehr so organisiert, dass L^AT_EX gezielt nur die Zuordnungen laden kann, die für die Fontkodierungen im aktuellen Dokument von Bedeutung sind. Dazu wird versucht, für jede Kodierung namens *<name>* eine Datei mit Namen *<name>.enc.dfu* zu laden, die, sofern sie existiert, die Zuordnungsdaten für die Unicode-Zeichen der betreffenden Kodierung enthält. Außer einer Reihe von `\DeclareUnicodeCharacter`-Deklarationen sollten solche Dateien nur noch eine `\ProvidesFile`-Zeile enthalten.

Da verschiedene Fontkodierungen häufig bis zu einem gewissen Anteil die gleichen Zeichen enthalten, ist es ganz normal, dass Deklarationen für die gleichen Unicode-Zeichen in mehreren .dfu-Dateien vorkommen. Darum ist es sehr wichtig, dass diese Deklarationen in allen Dateien identisch sind (was sie ohnehin sein sollten, aber...). Ansonsten ist die zuletzt geladene Deklaration gültig, und das kann in jedem Dokument eine andere sein.

Wer also eine .dfu-Datei für eine bisher nicht erfasste Kodierung bereitstellen will, sollte zunächst sorgfältig die bereits vorhandenen Definitionen in den .dfu-Dateien verwandter Kodierungen studieren. Die Standarddateien des inputenc-Paketes enthalten auf jeden Fall einheitliche Definitionen – sie wurden nämlich alle auf Basis einer einzelnen Liste, die in geeignete Abschnitte aufgeteilt wurde, erzeugt. Die Datei `utf8enc.dfu` enthält eine vollständige Liste der zur Zeit existierenden Zuordnungen.

7.11.4 Fontkodierungen

Wie bereits besprochen, legen Fontkodierungen die Zuordnung zwischen der LICR und den Glyphen (oder Glyphenkonstruktionen) fest, die in den jeweiligen Fonts verfügbar sind, welche für den Satz verwendet werden. Diese Zuordnungen werden in $\LaTeX$ mit zwei- oder dreibuchstabigen Bezeichnungen benannt (z.B. OT1 und T3). Man sagt, ein bestimmter Zeichensatz gehört zu einer bestimmten Kodierung, wenn ihre LICR-Zuordnung genau den Positionen der Glyphen in dem jeweiligen Zeichensatz entspricht. Woraus genau besteht nun aber eine solche Zuordnung?

Zeichen, die intern durch ASCII-Zeichen dargestellt sind, werden einfach an den Zeichensatz durchgereicht. Anders ausgedrückt verwendet $\TeX$ den ASCII-Code um eine Glyphe aus dem aktuellen Zeichensatz auszuwählen. So bewirkt z.B. das Zeichen „A“ mit dem ASCII-Code 65, dass die Glyphe von Position 65 des aktuellen Fonts gesetzt wird. Darum müssen alle ASCII-Buchstaben bei Textfonts in $\LaTeX$ an ihren ASCII-Positionen stehen, da es keine Möglichkeit gibt, auf diesen Basismechanismus in $\TeX$ einzuwirken (außer ihn komplett abzuschalten und alles „manuell“ abzarbeiten). Dadurch sind die sichtbaren ASCII-Zeichen in allen Fontkodierungen implizit eins zu eins zugeordnet.

Intern durch ASCII-Zeichenfolgen dargestellte Zeichen werden folgendermaßen gehandhabt: Wenn der aktuelle Zeichensatz zum ersten Mal geladen wird, erhält $\TeX$ die Information, dass dieser Zeichensatz eine Reihe so genannter Ligaturprogramme enthält. Diese definieren bestimmte Zeichenfolgen, die nicht direkt ausgegeben, sondern durch andere Glyphen des Zeichensatzes ersetzt¹ werden sollen (die genaue Position der ersetzenden Glyphe ist Zeichensatzabhängig und hier nicht von Bedeutung). Wenn $\TeX$ zum Beispiel in der Eingabe auf die Zeichenfolge „--“ trifft (d.h. zweimal hintereinander ASCII-Code 45), könnte ein Ligaturprogramm es auf die Glyphe an Position 123 umleiten (bei der es sich dann um die Glyphe „-“ handeln würde). Auch auf diesen Mechanismus lässt sich kein Einfluss nehmen. Einige dieser Ligaturen dienen rein ästhetischen Zwecken und sind möglicherweise in bestimmten Fonts gar nicht vorhanden (wie z.B. `ff` zum Erzeugen von „ff“ anstelle von „ff“). Andere

¹Ein Fontligatur-Programm kann noch wesentlich komplexere Schritte ausführen. Für die hier besprochene Thematik ist diese vereinfachte Betrachtung jedoch völlig ausreichend. Sie werden in Knuths Abhandlung über virtuelle Fonts [95] ausführlich besprochen.

sollten in bestimmten Kodierungen vorhanden sein (z.B. „--“ für \emdash).

Bei der großen Masse der internen Zeichendarstellungen handelt es sich jedoch um fontkodierungsspezifische Befehle. Sie werden mithilfe der im Folgenden beschriebenen Deklarationen zugeordnet. Die ersten beiden Argumente weisen bei allen diesen Deklarationen die gleiche Struktur auf: Zunächst kommt der fontkodierungsspezifische Befehl (oder sein erster Teil, wenn es sich um eine Befehlssequenz handelt), danach folgt der Name der Kodierung. Alle weiteren Argumente sind von der jeweiligen Deklaration abhängig.

Eine Kodierung XYZ wird also durch ein Bündel von Deklarationen definiert, deren zweites Argument immer der Name XYZ ist. Damit das Ganze von Nutzen ist, müssen natürlich auch einige Fonts in dieser Kodierung existieren. Die Entwicklung von Fontkodierungen erfolgt ja normalerweise auch in umgekehrter Reihenfolge: Jemand geht von einem existierenden Font aus und formuliert dann die entsprechenden Deklarationen, um ihn verwenden zu können. Diese Sammlung von Deklarationen wird dann mit einem geeigneten Namen versehen, wie z.B. OT1. Im nächsten Abschnitt wird der Zeichensatz ecrm1000 vorgestellt, der in Tabelle 7.32 auf der nächsten Seite abgebildet ist, und dessen Fontkodierung in L^AT_EX T1 heißt. Für ihn werden geeignete Deklarationen formuliert, um auf die Glyphen eines so kodierten Zeichensatzes zugreifen zu können. In der Tabelle sind diejenigen Zeichen blau dargestellt, die in jeder Textkodierung an den gleichen Positionen stehen müssen, da sie transparent durch T_EX hindurchgereicht werden.

Fontkodierungsdateien

Genau wie Eingabekodierungsdateien, sind auch Fontkodierungsdateien an der Erweiterung .def zu erkennen. Der eigentliche Name der Datei ist jedoch etwas weiter strukturiert: Er besteht aus dem kleingeschriebenen Namen der Kodierung, gefolgt von den Buchstaben enc (z.B. t1enc.def für die T1-Kodierung).

Diese Dateien sollten nur die in diesem Abschnitt beschriebenen Deklarationen enthalten. Da Fontkodierungsdateien unter Umständen mehrmals von L^AT_EX gelesen werden, ist es besonders wichtig, sich streng an diese Regel zu halten und von anderen Befehlen, wie z.B. \newcommand, abzusehen, die verhindern, dass eine Datei mehrfach gelesen werden kann.

Zur besseren Identifizierung sollten Fontkodierungsdateien mit einer \ProvidesFile-Deklaration beginnen, in der die Art der Datei beschrieben ist, wie z.B.:

```
\ProvidesFile{t1enc.def}[2001/06/05 v1.94 Standard LaTeX file]
```

Bevor man kodierungsspezifische Befehle für eine bestimmte Kodierung deklarieren kann, muss diese L^AT_EX erst bekannt gemacht werden. Das geschieht mithilfe einer \DeclareFontEncoding-Deklaration. An dieser Stelle ist es überdies sinnvoll, mit dem Befehl \DeclareFontSubstitution die Voreinstellungen für die Ersetzungsregeln zu dieser Kodierung zu deklarieren. Beide Deklarationen sind in Abschnitt 7.10.5 ab Seite 442 ausführlich beschrieben.

```
\DeclareFontEncoding{T1}{}{}
\DeclareFontSubstitution{T1}{cmr}{m}{n}
```


	'0	'1	'2	'3	'4	'5	'6	'7	
'00x	`	´	^	~	¨	˘	°	˘	"0x
'01x	˘	–	·	˘	˘	˘	˘	˘	
'02x	“	”	„	«	»	–	—		"1x
'03x	o	l	J	ff	fi	fl	ffi	ffl	
'04x	˘	!	"	#	\$	%	&	'	"2x
'05x	(	)	*	+	,	-	.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;	<	=	>	?	
'10x	@	A	B	C	D	E	F	G	"4x
'11x	H	I	J	K	L	M	N	O	
'12x	P	Q	R	S	T	U	V	W	"5x
'13x	X	Y	Z	[	\	]	^	_	
'14x	‘	a	b	c	d	e	f	g	"6x
'15x	h	i	j	k	l	m	n	o	
'16x	p	q	r	s	t	u	v	w	"7x
'17x	x	y	z	{		}	~	-	
'20x	Ă	Ą	Ć	Č	Ď	Ě	Ė	Ğ	"8x
'21x	Ĺ	Ł	Ł	Ń	Ň	Đ	Œ	Ř	
'22x	Ŕ	Ś	Š	Ş	Ť	Ț	Ů	Ű	"9x
'23x	Ÿ	Ž	Ž	Ž	IJ	İ	đ	§	
'24x	ă	ą	ć	č	ď	ě	ė	ğ	"Ax
'25x	í	ı	ı	ń	ň	ŋ	ő	í	
'26x	ř	ś	š	ş	ť	ț	ű	ű	"Bx
'27x	ÿ	ž	ž	ž	ij	i	ı	ı	
'30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	"Cx
'31x	È	É	Ê	Ë	Ì	Í	Î	Ï	
'32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	Œ	"Dx
'33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ŠŠ	
'34x	à	á	â	ã	ä	å	æ	ç	"Ex
'35x	è	é	ê	ë	ì	í	î	ï	
'36x	ð	ñ	ò	ó	ô	õ	ö	œ	"Fx
'37x	ø	ù	ú	û	ü	ý	þ	ß	
	"8	"9	"A	"B	"C	"D	"E	"F	

Blau dargestellte Zeichen müssen in jeder Textkodierung (an den gleichen Positionen) vorhanden sein, da sie transparent durch T_EX hindurchgereicht werden.

Tabelle 7.32: Glyphentabelle eines T1-kodierten Fonts (ecrm1000)

Nachdem die T1-Kodierung durch dieses Vorgehen in L^AT_EX bekannt ist, können nun die Deklarationen folgen, die festlegen, wie sich fontkodierungsspezifische Befehle in dieser Kodierung verhalten sollen.

```
\DeclareTextSymbol{LICR-objekt}{kodierung}{slot}
```

Am einfachsten lassen sich Textsymbole deklarieren, deren interne Darstellung direkt einer einzelnen Glyphen im Zielfont zugeordnet werden kann. Dies geschieht mithilfe der Deklaration `\DeclareTextSymbol`, deren drittes Argument, die Position im Font, als dezimale, hexadezimale oder oktale Zahl angegeben werden kann. Die Beispiele

```
\DeclareTextSymbol{\ss}{T1}{255}
\DeclareTextSymbol{\AE}{T1}{'306} % Fontposition als Oktalzahl
\DeclareTextSymbol{\ae}{T1}{"E6}  % ... als Hexadezimalzahl
```

deklarieren, dass die fontkodierungsspezifischen Befehle `\ss`, `\AE` und `\ae` in einem T1-kodierten Font den (dezimalen) Fontpositionen 255, 198 bzw. 230 zugeordnet werden. Wie bereits erwähnt, ist in solchen Deklarationen die dezimale Schreibweise die sicherste, auch wenn oktale oder hexadezimale Werte in Zeichensatztabellen wie derjenigen auf der vorherigen Seite leichter auszumachen sind. Sie wie im vorigen Beispiel zu mischen ist auf jeden Fall ein schlechter Stil. Für die T1-Kodierung existieren insgesamt 49 dieser Deklarationen.

```
\DeclareTextAccent{LICR-akzent}{kodierung}{slot}
```

Oftmals enthalten Fonts diakritische Zeichen als einzelne Glyphen, damit Akzentbuchstaben aus diesen Zeichen in Kombination mit einer weiteren Glyphen zusammengesetzt werden können. Solche Akzente werden (solange sie oberhalb anderer Glyphen platziert werden sollen) mit dem Befehl `\DeclareTextAccent` deklariert. Dabei gibt das dritte Argument *slot* die Position des diakritischen Zeichens im Font an. Das Beispiel

```
\DeclareTextAccent{\"}{T1}{4}
```

definiert den „Umlautakzent“. Ab diesem Zeitpunkt hat eine interne Darstellung wie `\"a` in der T1-Fontkodierung die folgende Bedeutung: Um „ä“ zu setzen, wird der Akzent von Position 4 über der Glyphen von Position 97 (ASCII-Code des Zeichens a) platziert. Eine solche Deklaration definiert implizit sogar eine ganze Menge interner Zeichendarstellungen, nämlich alle der Art `\"⟨basis-glyphe⟩`, wobei *⟨basis-glyphe⟩* durch `\DeclareTextSymbol` definiert wird, oder irgendein ASCII-Zeichen der LICR ist, wie beispielsweise „a“.

Selbst scheinbar sinnlose Kombinationen, wie etwa `\"P` (also das Absatzzeichen mit Umlaut ¶) werden damit im Prinzip Teil der Sammlung von fontkodierungsspezifischen Befehlen. In der T1-Kodierung gibt es insgesamt 11 derartige Deklarationen.

```
\DeclareTextComposite
      {LICR-akzent}{kodierung}{einfaches-LICR-objekt}{slot}
```

Die Zeichensatztabelle auf Seite 462 enthält eine Vielzahl von Akzentbuchstaben als eigenständige Glyphen, wie z.B. „ä“ an der Position '344 (oktal). In der T1-Kodierung sollte der kodierungsspezifische Befehl `\a` also nicht ein Akzentzeichen über das Zeichen „a“ setzen, sondern direkt auf die Glyphen an der entsprechenden Fontposition zugreifen. Das wird mit folgender Deklaration bewirkt:

```
\DeclareTextComposite{"}{T1}{a}{228}
```

Sie besagt, dass der kodierungsspezifische Befehl `\a` die Glyphen 228 druckt, und setzt somit die vorherige Akzentdeklaration außer Kraft. Für alle anderen kodierungsspezifischen Befehle, die mit `\` beginnen, behält die Akzentdeklaration ihre Gültigkeit. Der Befehl `\b` wird beispielsweise weiterhin das Zeichen „b“ ausgeben, indem er einen Akzent über dem Basiszeichen `b` platziert.

Das dritte Argument, *einfaches-LICR-objekt*, sollte aus einem einzelnen Buchstaben wie z.B. „a“ bestehen, oder aus einem einzelnen Befehl wie `\j` oder `\oe`. Für die T1-Kodierung sind 110 solcher Kombinationen deklariert.

Zu `\DeclareTextComposite` gibt es auch noch eine allgemeinere Variante, die jedoch für die T1-Kodierung nicht zur Anwendung kommt.

```
\DeclareTextCompositeCommand
      {LICR-objekt}{kodierung}{einfaches-LICR-objekt}{code}
```

Mit ihrer Hilfe kann man einer Position im Zeichensatz einen beliebigen Programmcode zuweisen. Bei der OT1-Kodierung wird mithilfe dieser Deklaration z.B. der nordische (ringförmige) Akzent über dem „A“ weiter nach unten gesetzt als durch den $\TeX$ -Basisbefehl `\accent`. Die Akzente über dem „i“ werden ebenfalls mithilfe dieser Deklaration eingerichtet:

```
\DeclareTextCompositeCommand{'}{OT1}{i}{\@tabacckludge'i}
\DeclareTextCompositeCommand{^}{OT1}{i}{\^i}
```

Welche weiteren Deklarationen gibt es noch für die T1-Kodierung? Eine Reihe diakritischer Zeichen wird nicht oberhalb sondern unterhalb von Buchstaben platziert. Für solche Auszeichnungen gibt es keine besondere Deklaration, da die eigentliche Positionierung normalerweise mithilfe von Low-Level-Befehlen des $\TeX$ -Systems erfolgt. Stattdessen lässt sich auch hier der generische Befehl `\DeclareTextCommand` verwenden.

```
\DeclareTextCommand{LICR-objekt}{kodierung}[anz][voreinst]{code}
```

Der „Unterstrich“-Akzent `\b` in der T1-Kodierung wird z.B. mit dem folgenden Meisterwerk an Prosa definiert:

```
\DeclareTextCommand{\b}{T1}[1]
  {\hmode\bgroup\o@lign{\relax#1\cr cr\hidewidth\sh@ft{29}%
   \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
```

Ohne im Einzelnen auf die detaillierte Bedeutung des Programmcodes einzugehen, kann man doch sehen, dass die Deklaration `\DeclareTextCommand` in ihrer Struktur dem Befehl `\newcommand` ähnelt. Das heißt, es gibt ein optionales Argument *anz*, das die Anzahl der Argumente (hier: eins) festlegt, ein zweites optionales Argument *voreinst* (hier nicht genutzt) und schließlich ein obligatorisches Argument, das den Programmcode enthält. Innerhalb des Programmcodes erfolgt der Bezug auf die vorhandenen Argumente über #1, #2 usw. T1 kennt vier derartige Deklarationen, und zwar für `\b`, `\c`, `\d` und `\k`.

Mithilfe von `\DeclareTextCommand` kann man auch fontkodierungsspezifische Befehle ohne Argument bilden. In diesem Falle wird die Deklaration ohne optionales Argument verwendet, definiert also einen Befehl ohne Argumente. So verfügt die T1-Kodierung beispielsweise über kein `%`-Zeichen, es gibt jedoch ein seltsames kleines „o“ an Position '30, mit welchem sich die korrekte Glyph e erstellen lässt, wenn es direkt hinter einem `%` platziert wird. Man kann also schreiben:

```
\DeclareTextCommand{\textperthousand}{T1}{\%\char 24 }
\DeclareTextCommand{\textpertenthousand}{T1}{\%\char 24\char 24 }
```

Damit sind alle erforderlichen Befehle besprochen, um die fontkodierungsspezifischen Befehle für eine neue Kodierung zu deklarieren. Wie bereits betont, sollten auch nur diese Befehle in Definitionsdateien für Kodierungen auftauchen.

Voreinstellungen für die Fontkodierung

Was geschieht, wenn ein kodierungsspezifischer Befehl verwendet wird, für den es in der aktuellen Fontkodierung keine Deklaration gibt? Dann gibt es zwei Möglichkeiten: Entweder verfügt L^AT_EX über eine voreingestellte Definition für das besagte LICR-Objekt, die in diesem Falle dann auch zum Einsatz kommt, oder der Anwender erhält die Fehlermeldung, dass das angeforderte LICR-Objekt in der aktuellen Kodierung nicht verfügbar ist. Man kann solche Voreinstellungen auf verschiedene Weise vornehmen.

```
\DeclareTextCommandDefault{LICR-objekt}[anz][voreinst]{code}
```

Der Befehl `\DeclareTextCommandDefault` stellt die voreingestellte Definition für ein *LICR-objekt* zur Verfügung, die immer dann verwendet werden soll, wenn es in der aktuellen Kodierung keine besonderen Einstellungen für das Objekt gibt. Solche Definitionen können zum Beispiel ein bestimmtes Zeichen nachahmen. Für `\textregistered` gibt es beispielsweise eine voreingestellte Definition, durch die das Zeichen folgendermaßen aus zwei anderen zusammengesetzt wird.

```
\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
```

Technisch gesehen werden diese Definitionen als Kodierung unter dem Namen „?“ gespeichert. Man sollte sich zwar nicht darauf verlassen, da sich diese Umsetzung zukünftig noch ändern kann. Trotzdem bedeutet es, dass man keine Kodierung mit diesem Namen deklarieren kann.

```
\DeclareTextSymbolDefault{LICR-objekt}{kodierung}
```

In den meisten Fällen erfordert eine voreingestellte Definition keine Programmierung, sondern sie weist $\LaTeX$ einfach an, das Zeichen aus einer Kodierung zu nehmen, in der es bekanntermaßen existiert. Das Paket `textcomp`, z.B., besteht aus einer Vielzahl von voreingestellten Deklarationen, die alle auf die Kodierung TS1 verweisen, wie etwa die folgende:

```
\DeclareTextSymbolDefault{\texteuro}{TS1}
```

Mit dem Befehl `\DeclareTextSymbolDefault` lassen sich Voreinstellungen für alle LICR-Objekte ohne Argument vornehmen, nicht nur für solche, die in anderen Kodierungen mit `\DeclareTextSymbol` deklariert wurden.

```
\DeclareTextAccentDefault{LICR-akzent}{kodierung}
```

Es gibt eine ähnliche Deklaration für LICR-Objekte mit einem Argument, wie etwa Akzente (welche dieser Deklaration ihren Namen gaben). Sie lässt sich für alle LICR-Objekte mit einem Argument verwenden. Der $\LaTeX$ -Kern enthält zum Beispiel eine ganze Reihe von Deklarationen der folgenden Art:

```
\DeclareTextAccentDefault{"}{OT1}
\DeclareTextAccentDefault{\t}{OML}
```

Das bedeutet: Wenn das Zeichen `"` in der aktuellen Kodierung nicht definiert ist, dann soll es aus einem OT1-kodierten Zeichensatz herangezogen werden. Ebenso soll der Verbindungsakzent aus der OML-Kodierung¹ verwendet werden, wenn nichts Besseres verfügbar ist.

```
\ProvideTextCommandDefault{LICR-objekt}[anz][voreinst]{code}
```

Mithilfe der Deklaration `\ProvideTextCommandDefault` lässt sich eine weitere Art von Voreinstellungen „bereitstellen“. Wie der Name schon andeutet, übt es die gleiche Funktion aus wie `\DeclareTextCommandDefault`, wobei diese Voreinstellung jedoch nur aktiviert wird, wenn zuvor noch keine Voreinstellung definiert wurde. Dieses Verfahren wird hauptsächlich in Dateien für Eingabekodierungen verwendet, um eine Art einfacher Voreinstellungen für ungewöhnliche LICR-Objekte zu schaffen, wie z.B.:

```
\ProvideTextCommandDefault{\textonequarter}{\ensuremath{\frac{1}{4}}}
\ProvideTextCommandDefault{\textcent}
{\TextSymbolUnavailable\textcent}
```

Pakete wie `textcomp` können solche Definitionen dann durch Deklarationen ersetzen, die auf echte Glyphen verweisen. Wenn man `\Provide..` anstelle von `\Declare..` verwendet, kann man damit vermeiden, dass eine bessere Voreinstellung beim Lesen der Datei für die Eingabekodierung versehentlich überschrieben wird.

¹OML ist eine Formelkodierung, die jedoch eben dieses Akzentzeichen enthält.

```
\UndeclareTextCommand{LICR-objekt}{kodierung}
```

In manchen Fällen muss eine vorhandene Deklaration erst entfernt werden, damit auch garantiert eine voreingestellte Deklaration verwendet wird. Diese Aufgabe lässt sich mit dem Befehl `\UndeclareTextCommand` bewerkstelligen. Das Paket `textcomp` entfernt z.B. die Definitionen von `\textdollar` und `\textsterling` aus der OT1-Kodierung, da nicht jeder OT1-kodierte Zeichensatz wirklich über diese Symbole verfügt.¹

```
\UndeclareTextCommand{\textsterling}{OT1}
\UndeclareTextCommand{\textdollar} {OT1}
```

Ohne dieses Entfernen würden die neuen voreingestellten Deklarationen, die auf TS1-Symbole zurückgreifen sollen, bei OT1-kodierten Fonts nicht funktionieren.

```
\UseTextSymbol{kodierung}{LICR-objekt}
\UseTextAccent{kodierung}{LICR-objekt}{einfaches-LICR-objekt}
```

Auf die Funktionen, die sich hinter `\DeclareTextSymbolDefault` und `\DeclareTextAccentDefault` verbergen, kann man auch direkt zugreifen. Angenommen, die aktuelle Kodierung ist beispielsweise U. In diesem Fall haben die Befehle

```
\UseTextSymbol{OT1}{\ss}
\UseTextAccent{OT1}{\'}{a}
```

den gleichen Effekt wie die nächsten Befehlszeilen. Dabei ist besonders zu beachten, dass der Buchstabe „a“ weiterhin in der Kodierung U gesetzt wird – lediglich der Akzent entstammt einer anderen Kodierung.

```
{\fontencoding{OT1}\selectfont\ss}
{\fontencoding{OT1}\selectfont\'\fontencoding{U}\selectfont a}}
```

Eine Liste von Standard-LICR-Objekten

Tabelle 7.33 zeigt eine umfassende Übersicht der L^AT_EX-internen Darstellungen, die in den drei wichtigsten Kodierungen für lateinische Sprachen verfügbar sind: OT1 (die originale Textfontkodierung in T_EX), T1 (die L^AT_EX-Standardkodierung, auch bekannt als Cork-Kodierung) und LY1 (eine alternative 8-Bit-Kodierung von Y&Y). Außerdem zeigt sie alle LICR-Objekte, die von TS1 (der Standardkodierung für Textsymbole in L^AT_EX) deklariert werden und über das `textcomp`-Paket verfügbar sind.

In der ersten Spalte sind die Namen der LICR-Objekte in alphabetischer Reihenfolge aufgeführt. Dabei sind Akzente als solche gekennzeichnet. In der zweiten Spalte sind die Glyphendarstellungen der Objekte zu sehen.

Die dritte Spalte gibt Aufschluss, ob für das jeweilige Objekt eine voreingestellte Deklaration existiert. Wenn an dieser Stelle eine Kodierung erscheint, bedeutet dies, dass die Glyphe per Voreinstellung aus einem geeigneten Font

¹Das ist einer der Nachteile der alten T_EX-Kodierungen: Abgesehen von den fehlenden Akzentbuchstaben stimmen sie auch nicht miteinander überein.

dieser Kodierung herangezogen wird; $\not\subset$ bedeutet, dass die Voreinstellung mit Hilfe einer Low-Level-Programmierung in T_EX erzeugt wird; ein leerer Eintrag weist darauf hin, dass es für dieses LICR-Objekt keine Voreinstellung gibt. In letzterem Falle führt der Einsatz des Objektes in einer Kodierung, die keine explizite Definition dafür enthält, zu der Fehlermeldung „Symbol unavailable“ (Symbol nicht verfügbar). Wenn das Objekt als Alias für ein anderes LICR-Objekt fungiert, wird in dieser Spalte der alternative Name aufgeführt.

Die Spalten vier bis sieben zeigen, ob das Objekt in der jeweiligen Kodierung zur Verfügung steht. Das Symbol $\times$ bedeutet, dass dieses Objekt originär (als Glyph) in Fonts dieser Kodierung enthalten ist, das Zeichen $\circ$ weist darauf hin, dass es über die Voreinstellungen für alle Kodierungen verfügbar ist und $\not\subset$ bedeutet, dass es aus mehreren Glyphen, Akzentzeichen oder anderen Elementen zusammengesetzt ist. Wenn sich die Voreinstellung auf die TS1-Kodierung bezieht, ist das LICR-Objekt nur bei geladenem textcomp-Paket verfügbar.

Tabelle 7.33: Standard LICR-Objekte

LICR-Objekt		Glyphe	Voreinstellung	OT1	T1	LY1	TS1
ABC..XYZ	(Großbuchstaben)	ABC..XYZ		$\times$	$\times$	$\times$	
abc..xyz	(Kleinbuchstaben)	abc..xyz		$\times$	$\times$	$\times$	
0123..9	(Ziffern)	0123..9		$\times$	$\times$	$\times$	$\times$
.,/	(Satzzeichen)	.,/		$\times$	$\times$	$\times$	$\times$
;:?!"‘’	(weitere Satzzeichen)	;:?!"‘’		$\times$	$\times$	$\times$	
*+==() []	(Verschiedenes)	*+==() []		$\times$	$\times$	$\times$	
# & %		#&%		$\times$	$\times$	$\times$	
\"	(Akzent)	..	OT1	$\times$	$\times$	$\times$	
\"A		Ä		$\not\subset$	$\times$	$\times$	
\"E		Ë		$\not\subset$	$\times$	$\times$	
\"I		Ï		$\not\subset$	$\times$	$\times$	
\"O		Ö		$\not\subset$	$\times$	$\times$	
\"U		Ü		$\not\subset$	$\times$	$\times$	
\"Y		ÿ		$\not\subset$	$\times$	$\times$	
\"a		ä		$\not\subset$	$\times$	$\times$	
\"e		ë		$\not\subset$	$\times$	$\times$	
\"i		ï		$\not\subset$	$\times$	$\times$	
\"i	(Alias)	ï	\"i	$\not\subset$	$\times$	$\times$	
\"o		ö		$\not\subset$	$\times$	$\times$	
\"u		ü		$\not\subset$	$\times$	$\times$	
\"y		ÿ		$\not\subset$	$\times$	$\times$	
\\$	(Alias)	\$	\textdollar	$\circ$	$\times$	$\times$	$\times$
\'	(Akzent)	´	OT1	$\times$	$\times$	$\times$	
\'A		Á		$\not\subset$	$\times$	$\times$	
\'C		Ć		$\not\subset$	$\times$	$\not\subset$	
\'E		É		$\not\subset$	$\times$	$\times$	
\'I		Í		$\not\subset$	$\times$	$\times$	
$\times$ in Kodierung definiert			$\circ$ über Voreinstellung definiert	$\not\subset$ konstruiert			

LICR-Objekt	Glyphe	Voreinstellung	OT1	T1	LY1	TS1
\’L	Ĺ		☐	✕	☐	
\’N	Ń		☐	✕	☐	
\’O	Ó		☐	✕	✕	
\’R	Ř		☐	✕	☐	
\’S	Ŝ		☐	✕	☐	
\’U	Ů		☐	✕	✕	
\’Y	Ÿ		☐	✕	✕	
\’Z	Ž		☐	✕	☐	
\’a	á		☐	✕	✕	
\’c	ć		☐	✕	☐	
\’e	é		☐	✕	✕	
\’\i	í		☐	✕	✕	
\’i	í	\’\i	☐	✕	✕	
\’l	ĺ		☐	✕	☐	
\’n	ń		☐	✕	☐	
\’o	ó		☐	✕	✕	
\’r	ř		☐	✕	☐	
\’s	ś		☐	✕	☐	
\’u	ú		☐	✕	✕	
\’y	ý		☐	✕	✕	
\’z	ž		☐	✕	☐	
\.	·	OT1	✕	✕	✕	
\.I	İ		☐	✕	☐	
\.Z	Ž		☐	✕	☐	
\.\i	i		✕	✕	☐	
\.i	i	\.\i	✕	✕	☐	
\.z	ž		☐	✕	☐	
\=	-	OT1	✕	✕	✕	
\AE	Æ	OT1	✕	✕	✕	
\DH	Ð			✕	✕	
\DJ	Đ			✕		
\H	˜	OT1	✕	✕	✕	
\H O	Õ		☐	✕	☐	
\H U	Ũ		☐	✕	☐	
\H o	õ		☐	✕	☐	
\H u	ũ		☐	✕	☐	
\L	Ł	OT1	✕	✕	✕	
\NG	Ŋ			✕		
\O	Ø	OT1	✕	✕	✕	
\OE	Œ	OT1	✕	✕	✕	
\P	¶	\textparagraph	○	○	✕	✕
\S	§	\textsection	○	✕	✕	✕
\SS	Š	☐	○	✕	○	
\TH	Þ			✕	✕	

✕ in Kodierung definiert

○ über Voreinstellung definiert

☐ konstruiert

LICR-Objekt		Glyphe	Voreinstellung	OT1	T1	LY1	TS1
\^	(Akzent)	^	OT1	✗	✗	✗	
\^A		Â		✗	✗	✗	
\^E		Ê		✗	✗	✗	
\^I		Î		✗	✗	✗	
\^O		Ô		✗	✗	✗	
\^U		Û		✗	✗	✗	
\^a		â		✗	✗	✗	
\^e		ê		✗	✗	✗	
\^i		î		✗	✗	✗	
\^i	(Alias)	î	\^i	✗	✗	✗	
\^o		ô		✗	✗	✗	
\^u		û		✗	✗	✗	
_	(Alias)	—	\textunderscore	○	✗	✗	
\‘	(Akzent)	`	OT1	✗	✗	✗	
\‘A		À		✗	✗	✗	
\‘E		È		✗	✗	✗	
\‘I		Ì		✗	✗	✗	
\‘O		Ò		✗	✗	✗	
\‘U		Û		✗	✗	✗	
\‘a		à		✗	✗	✗	
\‘e		è		✗	✗	✗	
\‘i		ì		✗	✗	✗	
\‘i	(Alias)	ì	\‘i	✗	✗	✗	
\‘o		ò		✗	✗	✗	
\‘u		ù		✗	✗	✗	
\ae		æ	OT1	✗	✗	✗	
\b	(Akzent)	–	OT1	✗	✗	✗	
\c	(Akzent)	ç	OT1	✗	✗	✗	
\c C		Ç		✗	✗	✗	
\c S		Ş		✗	✗	✗	
\c T		Ţ		✗	✗	✗	
\c c		ç		✗	✗	✗	
\c s		ş		✗	✗	✗	
\c t		ţ		✗	✗	✗	
\capitalacute	(Akzent)	ˆ	TS1	○	○	○	✗
\capitalcaron	(Akzent)	ˇ	TS1	○	○	○	✗
\capitaldieresis	(Akzent)	¨	TS1	○	○	○	✗
\capitalgrave	(Akzent)	`	TS1	○	○	○	✗
\capitalmacron	(Akzent)	ˉ	TS1	○	○	○	✗
\capitalogonek	(Akzent)	˛	TS1	○	○	○	✗
\capitalring	(Akzent)	˚	TS1	○	○	○	✗
\capitaltilde	(Akzent)	˜	TS1	○	○	○	✗
\copyright	(Alias)	©	\textcopyright	○	○	✗	✗
\d	(Akzent)	.	OT1	✗	✗	✗	

✗ in Kodierung definiert

○ über Voreinstellung definiert

✗ konstruiert

LICR-Objekt		Glyphe	Voreinstellung	OT1	T1	LY1	TS1
<code>\dag</code>	(Alias)	†	<code>\textdagger</code>	○	○	✗	✗
<code>\ddag</code>	(Alias)	‡	<code>\textdaggerdbl</code>	○	○	✗	✗
<code>\dh</code>		ð			✗	✗	
<code>\dj</code>		đ			✗		
<code>\dots</code>	(Alias)	...	<code>\textellipsis</code>	○	○	✗	
<code>\guillemotleft</code>		«		babel	✗	✗	
<code>\guillemotright</code>		»		babel	✗	✗	
<code>\guilsinglleft</code>		<		babel	✗	✗	
<code>\guilsinglright</code>		>		babel	✗	✗	
<code>\i</code>		ı	OT1	✗	✗	✗	
<code>\j</code>		Ј	OT1	✗	✗	✗	
<code>\k</code>	(Akzent)	ˆ			✗	✗	
<code>\k A</code>		Ā			✗	↯	
<code>\k E</code>		Ē			✗	↯	
<code>\k O</code>		Ō			✗	↯	
<code>\k a</code>		ā			✗	↯	
<code>\k e</code>		ē			✗	↯	
<code>\k o</code>		ō			✗	↯	
<code>\l</code>		ł	OT1	✗	✗	✗	
<code>\ng</code>		ŋ			✗		
<code>\o</code>		ø	OT1	✗	✗	✗	
<code>\oe</code>		œ	OT1	✗	✗	✗	
<code>\pounds</code>	(Alias)	£	<code>\textsterling</code>	○	✗	✗	✗
<code>\quotedblbase</code>		”			✗	✗	
<code>\quotesinglbase</code>		‘			✗	✗	
<code>\r</code>	(Akzent)	˚	OT1	✗	✗	✗	
<code>\r A</code>		Ā		↯	✗	↯	
<code>\r U</code>		Ū		↯	✗	↯	
<code>\r a</code>		ā		↯	✗	↯	
<code>\r u</code>		ū		↯	✗	↯	
<code>\ss</code>		ß	OT1	✗	✗	✗	
<code>\t</code>	(Akzent)	˘	OML	✗	✗	○	
<code>\textacutedbl</code>		”	TS1	○	○	○	✗
<code>\textascendercompwordmark</code>		unsichtbar	TS1	○	○	○	✗
<code>\textasciicircum</code>		ˆ	TS1	○	○	○	✗
<code>\textasciibreve</code>		˘	TS1	○	○	○	✗
<code>\textasciicaron</code>		ˇ	TS1	○	○	○	✗
<code>\textasciicircum</code>		ˆ	↯	○	✗	✗	
<code>\textasciidieresis</code>		¨	TS1	○	○	○	✗
<code>\textasciigrave</code>		`	TS1	○	○	○	✗
<code>\textasciimacron</code>		—	TS1	○	○	○	✗
<code>\textasciitilde</code>		˜	↯	○	✗	✗	
<code>\textasteriskcentered</code>		*	OMS/TS1	○	○	○	✗
<code>\textbackslash</code>		\	OMS	○	✗	✗	
✗ in Kodierung definiert		○ über Voreinstellung definiert	↯ konstruiert				

LICR-Objekt	Glyphe	Voreinstellung	OT1	T1	LY1	TS1
\textbaht	฿	TS1	○	○	○	✗
\textbar		OMS	○	✗	✗	
\textbardbl		TS1	○	○	○	✗
\textbigcircle	◯	TS1	○	○	○	✗
\textblank		TS1	○	○	○	✗
\textborn	★	TS1	○	○	○	✗
\textbraceleft	{	OMS	○	✗	✗	
\textbraceright	}	OMS	○	✗	✗	
\textbrokenbar	̸	TS1	○	○	✗	✗
\textbullet	•	OMS/TS1	○	○	✗	✗
\textcapitalcompwordmark	unsichtbar	TS1	○	○	○	✗
\textcelsius	°C	⚠/TS1	○	○	○	✗
\textcent	¢	TS1	○	○	✗	✗
\textcentoldstyle	¢	TS1	○	○	○	✗
\textcircled (Akzent)	◯	OMS/TS1	○	○	○	✗
\textcircledP	Ⓟ	TS1	○	○	○	✗
\textcolonmonetary	₯	TS1	○	○	○	✗
\textcompwordmark	unsichtbar	⚠	○	✗	○	
\textcopyleft	©	TS1	○	○	○	✗
\textcopyright	©	⚠/TS1	○	○	✗	✗
\textcurrency	₭	TS1	○	○	✗	✗
\textdagger	†	OMS/TS1	○	○	✗	✗
\textdaggerdbl	‡	OMS/TS1	○	○	✗	✗
\textdblhyphen	=	TS1	○	○	○	✗
\textdblhyphenchar	=	TS1	○	○	○	✗
\textdegree	°	TS1	○	○	✗	✗
\textdied	†	TS1	○	○	○	✗
\textdiscount	ℳ	TS1	○	○	○	✗
\textdiv	÷	TS1	○	○	✗	✗
\textdivorced	⚭	TS1	○	○	○	✗
\textdollar	\$	OT1/TS1	○	✗	✗	✗
\textdollaroldstyle	\$	TS1	○	○	○	✗
\textdong	₫	TS1	○	○	○	✗
\textdownarrow	↓	TS1	○	○	○	✗
\texteightoldstyle	8	TS1	○	○	○	✗
\textellipsis	...	⚠	○	○	✗	
\textemdash	—	OT1	✗	✗	✗	
\textendash	–	OT1	✗	✗	✗	
\textestimated	€	TS1	○	○	○	✗
\texteuro	€	TS1	○	○	✗	✗
\textexclamdown	¡	OT1	✗	✗	✗	
\textfiveoldstyle	5	TS1	○	○	○	✗
\textflorin	f	TS1	○	○	✗	✗
\textfouroldstyle	4	TS1	○	○	○	✗

✗ in Kodierung definiert

○ über Voreinstellung definiert

⚠ konstruiert

LICR-Objekt	Glyphe	Voreinstellung	OT1	T1	LY1	TS1
<code>\textfractionsolidus</code>	/	TS1	○	○	○	✕
<code>\textgravedbl</code>	“	TS1	○	○	○	✕
<code>\textgreater</code>	>	OML	○	✕	✕	
<code>\textguarani</code>	Ḡ	TS1	○	○	○	✕
<code>\textinterrobang</code>	‡	TS1	○	○	○	✕
<code>\textinterrobangdown</code>	‡↓	TS1	○	○	○	✕
<code>\textlangle</code>	<	TS1	○	○	○	✕
<code>\textlbrackdbl</code>	⌈	TS1	○	○	○	✕
<code>\textleaf</code>	🍃	TS1	○	○	○	✕
<code>\textleftarrow</code>	←	TS1	○	○	○	✕
<code>\textless</code>	<	OML	○	✕	✕	
<code>\textlira</code>	₧	TS1	○	○	○	✕
<code>\textlnot</code>	¬	TS1	○	○	✕	✕
<code>\textlquill</code>	{	TS1	○	○	○	✕
<code>\textmarried</code>	∞	TS1	○	○	○	✕
<code>\textmho</code>	℧	TS1	○	○	○	✕
<code>\textminus</code>	—	TS1	○	○	○	✕
<code>\textmu</code>	μ	TS1	○	○	✕	✕
<code>\textmusicalnote</code>	♪	TS1	○	○	○	✕
<code>\textnaira</code>	₦	TS1	○	○	○	✕
<code>\textnineoldstyle</code>	9	TS1	○	○	○	✕
<code>\textnumero</code>	№	TS1	○	○	○	✕
<code>\textogonekcentered</code> (Akzent)	◌̈			✕		
<code>\textohm</code>	Ω	TS1	○	○	○	✕
<code>\textonehalf</code>	½	TS1	○	○	✕	✕
<code>\texttoneoldstyle</code>	1	TS1	○	○	○	✕
<code>\texttonequarter</code>	¼	TS1	○	○	✕	✕
<code>\texttonesuperior</code>	¹	TS1	○	○	○	✕
<code>\textopenbullet</code>	◦	TS1	○	○	○	✕
<code>\textordfeminine</code>	ª	ℳ/TS1	○	○	✕	✕
<code>\textordmasculine</code>	º	ℳ/TS1	○	○	✕	✕
<code>\textparagraph</code>	¶	OMS/TS1	○	○	✕	✕
<code>\textperiodcentered</code>	·	OMS/TS1	○	○	✕	✕
<code>\textpertenthousand</code>	‰	TS1	○	ℳ	○	✕
<code>\textperthousand</code>	‱	TS1	○	ℳ	✕	✕
<code>\textpeso</code>	₱	TS1	○	○	○	✕
<code>\textpilcrow</code>	¶	TS1	○	○	○	✕
<code>\textpm</code>	±	TS1	○	○	✕	✕
<code>\textquestiondown</code>	¿	OT1	✕	✕	✕	
<code>\textquotedbl</code>	”			✕	✕	
<code>\textquotedblleft</code>	“	OT1	✕	✕	✕	
<code>\textquotedblright</code>	”	OT1	✕	✕	✕	
<code>\textquoteleft</code>	‘	OT1	✕	✕	✕	
<code>\textquoteright</code>	’	OT1	✕	✕	✕	

✕ in Kodierung definiert

○ über Voreinstellung definiert

ℳ konstruiert

LICR-Objekt	Glyphe	Voreinstellung	OT1	T1	LY1	TS1
\textquotesingle	'	TS1	○	○	○	✕
\textquotestraightbase	'	TS1	○	○	○	✕
\textquotestraightdblbase	"	TS1	○	○	○	✕
\texttriangle	>	TS1	○	○	○	✕
\texttrbrackdbl	]]	TS1	○	○	○	✕
\textrecipe	R	TS1	○	○	○	✕
\textreferencemark	※	TS1	○	○	○	✕
\textregistered	®	™/TS1	○	○	✕	✕
\textrightarrow	→	TS1	○	○	○	✕
\texttrquill	}	TS1	○	○	○	✕
\textsection	§	OMS/TS1	○	✕	✕	✕
\textservicemark	SM	TS1	○	○	○	✕
\textsevenoldstyle	7	TS1	○	○	○	✕
\textsixoldstyle	6	TS1	○	○	○	✕
\textsterling	£	OT1/TS1	○	✕	✕	✕
\textsurd	√	TS1	○	○	○	✕
\textthreeoldstyle	3	TS1	○	○	○	✕
\textthreequarters	$\frac{3}{4}$	TS1	○	○	✕	✕
\textthreequartersemdash	—	TS1	○	○	○	✕
\textthreesuperior	³	TS1	○	○	○	✕
\texttildelow	~	TS1	○	○	○	✕
\texttimes	×	TS1	○	○	✕	✕
\texttrademark	™	™/TS1	○	○	✕	✕
\texttwelveudash	—	TS1	○	○	○	✕
\texttwooldstyle	2	TS1	○	○	○	✕
\texttwosuperior	²	TS1	○	○	○	✕
\textunderscore	—	™	○	✕	✕	
\textuparrow	↑	TS1	○	○	○	✕
\textvisiblespace	␣	™	○	✕	○	
\textwon	₩	TS1	○	○	○	✕
\textyen	¥	TS1	○	○	✕	✕
\textzerooldstyle	o	TS1	○	○	○	✕
\th	þ			✕	✕	
\u (Akzent)	˘	OT1	✕	✕	✕	
\u A	Ä		™	✕	™	
\u G	Ç		™	✕	™	
\u a	ä		™	✕	™	
\u g	ç		™	✕	™	
\v (Akzent)	˘	OT1	✕	✕	✕	
\v C	Č		™	✕	™	
\v D	Ď		™	✕	™	
\v E	Ě		™	✕	™	
\v L	Ĺ		™	✕	™	
\v N	Ň		™	✕	™	
✕ in Kodierung definiert			○ über Voreinstellung definiert			™ konstruiert

LICR-Objekt		Glyphe	Voreinstellung	OT1	T1	LY1	TS1
<code>\v R</code>		Ř		☞	✗	☞	
<code>\v S</code>		Š		☞	✗	✗	
<code>\v T</code>		Ť		☞	✗	☞	
<code>\v Z</code>		Ž		☞	✗	✗	
<code>\v c</code>		č		☞	✗	☞	
<code>\v d</code>		ď		☞	✗	☞	
<code>\v e</code>		ě		☞	✗	☞	
<code>\v l</code>		ĺ		☞	✗	☞	
<code>\v n</code>		ň		☞	✗	☞	
<code>\v r</code>		ř		☞	✗	☞	
<code>\v s</code>		š		☞	✗	✗	
<code>\v t</code>		ť		☞	✗	☞	
<code>\v z</code>		ž		☞	✗	✗	
<code>\{</code>	(Alias)	{	<code>\textbraceleft</code>	○	✗	✗	
<code>\}</code>	(Alias)	}	<code>\textbraceright</code>	○	✗	✗	
<code>\~</code>	(Akzent)	˘	OT1	✗	✗	✗	
<code>\~A</code>		Ā		☞	✗	✗	
<code>\~N</code>		Ñ		☞	✗	✗	
<code>\~O</code>		Ō		☞	✗	✗	
<code>\~a</code>		ā		☞	✗	✗	
<code>\~n</code>		ñ		☞	✗	✗	
<code>\~o</code>		ō		☞	✗	✗	
✗ in Kodierung definiert ○ über Voreinstellung definiert ☞ konstruiert							

7.12 Dokumente aus dem letzten Jahrtausend

Die Fontschnittstelle in $\text{\LaTeX}$ hat sich von einer starren Struktur mit fest eingerichteten Fonts ($\text{\LaTeX}$ 2.09 vor 1990) zu einem flexiblen System ($\text{\LaTeX}$ 2 ϵ mit NFSS Version 2, 1994 integriert) gewandelt. Von 1990 bis 1993 war das NFSS in der Version 1 in Europa weit verbreitet. Auch wenn die Unterschiede zwischen den Versionen 1 und 2 nicht übermäßig groß waren, sind sie doch dafür verantwortlich, dass man Dokumente aus dieser Zeit mit dem heutigen $\text{\LaTeX}$ nicht mehr sinnvoll verarbeiten kann. Aus diesem Grunde wurden eine Reihe von Paketen entwickelt, die wieder eine gewisse Kompatibilität herstellen, damit auch Dokumente, die für $\text{\LaTeX}$ 2.09 mit oder ohne NFSS 1 geschrieben wurden, verarbeitet werden können.

7.12.1 oldfont, rawfonts, newfont – Kompatibilitätspakete

Wie bereits gesehen, unterscheidet sich das NFSS, und damit auch $\text{\LaTeX}$ 2 ϵ , von $\text{\LaTeX}$ 2.09 in verschiedenen Punkten des Umgangs mit Fontbefehlen. Am deutlichsten treten diese Unterschiede in mathematischen Formeln zutage, in denen Befehle wie `\bfseries` nicht unterstützt werden. Trotzdem lassen sich auch ältere Dokumente sehr einfach mit NFSS setzen. *Rückwärtskompatibilität bis 1993 und früher*

Wenn das Dokument lediglich erneut gedruckt werden soll, schaltet $\text{\LaTeX}$ anhand des Befehls `\documentstyle` automatisch in den Kompatibilitätsmodus, wodurch das alte Zeichensatz-Auswahlverfahren von $\text{\LaTeX}$ 2.09 emuliert wird, wie es in der ersten Ausgabe des *$\text{\LaTeX}$ -Manuals* beschrieben ist. Alternativ dazu kann man auch nach dem `\documentclass`-Befehl das Paket `oldfont` laden. Dadurch werden alle alten Fontauswahlbefehle bereitgestellt, die Fontwechselbefehle heben sich gegenseitig auf und alle diese Befehle können auch in mathematischen Formeln verwendet werden.

Einige alte Dokumente beziehen sich auf interne Fontbefehle von $\text{\LaTeX}$ 2.09, wie z.B. `\twlrm` oder `\nintt`. Diese Befehle rufen nun Fehlermeldungen hervor, da sie (selbst im Kompatibilitätsmodus) nicht mehr definiert sind. Sie werden unter anderem deswegen nicht mehr unterstützt, weil sie nie auf allen Installationen verfügbar waren. Um ein Dokument verarbeiten zu können, das solche expliziten Fontwechselbefehle enthält, muss man diese mithilfe der in Abschnitt 7.9 beschriebenen Befehle in der Präambel definieren. Für die zuvor genannten Befehle würde es z.B. ausreichen, die Präambel um folgende Definitionen zu erweitern:

```
\newcommand\twlrm{\fontsize{12pt}{14pt}\normalfont\rmfamily}
\newcommand\nintt{\fontsize{9pt}{11pt}\normalfont\ttfamily}
```

Auch für diese Aufgabe gibt es Unterstützung: Wenn man das Paket `rawfonts` mit den Optionen `only`, `twlrm` und `nintt` lädt, nimmt es selbst die entsprechenden Deklarationen vor. Lädt man es ohne jegliche Option, so definiert es selbständig alle festen Fontbefehle aus $\text{\LaTeX}$ 2.09.

Auch Teile von Dokumenten lassen sich sehr einfach wiederverwenden: Man muss sie nur in das neue Dokument einfügen und sehen was geschieht. Die Wahrscheinlichkeit ist groß, dass $\text{\LaTeX}$ die Fragmente des alten Dokumentes einfach problemlos verarbeitet. Andernfalls gibt es explizite Informationen aus, wo das Quelldokument überarbeitet werden muss, z.B. wo `\it`-, `\sf`- und ähnliche Befehle in Formeln zu den entsprechenden Alphabetbefehlen `\mathit`, `\mathsf` usw. geändert werden müssen.

Rückwärtskompatibilität
mit der ersten
NFSS-Version

In der ersten Version von NFSS wurden die zweibuchstabigen Fontwechselbefehle so umdefiniert, dass sie sich nur auf einzelne Attribute auswirkten. So verhielten sich beispielsweise `\sf` und `\it` genau wie die NFSS2-Befehle `\sffamily` bzw. `\itshape`. Wenn man ein altes Dokument verarbeiten möchte, das nach diesen Konventionen erstellt wurde, kann man das `newfont`-Paket in der Dokumentenpräambel laden, um sie erneut einzurichten.

7.12.2 latexsym – Zugriff auf $\text{\LaTeX}$ 2.09 Lasy Fonts

Elf der mathematischen Zeichen aus $\text{\LaTeX}$ 2.09 sind in der Basisinstallation von NFSS nicht mehr definiert.

$\square$	$\diamond$	$\bowtie$	$\leadsto$	<code>\usepackage{latexsym}</code>	<code>\newcommand\Q[1]{\\$#1\\$ \quad}</code>
$\triangleleft$	$\cup$	$\triangleright$	$\square$	<code>\Q{\Box}</code>	<code>\Q{\Diamond}</code>
$\trianglelefteq$	$\supseteq$			<code>\Q{\Join}</code>	<code>\Q{\leadsto}</code>
				<code>\Q{\lhd}</code>	<code>\Q{\mho}</code>
				<code>\Q{\rhd}</code>	<code>\Q{\sqsubset}</code>
				<code>\Q{\sqsupset}</code>	<code>\Q{\unlhd}</code>
				<code>\Q{\unrhd}</code>	

Bsp.
7-12-1

Bei Bedarf können diese Sonderzeichen durch Laden des Paketes `latexsym` verfügbar gemacht werden. Sie lassen sich außerdem durch die Pakete `amsfonts` oder `amssymb` bereitstellen (siehe Abschnitt 8.9).

Höhere Mathematik

8.1 Eine Einführung in $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\LaTeX}$	478
8.2 Umgebungen für abgesetzte Formeln	481
8.3 Matrixähnliche Umgebungen	498
8.4 Komplexere Gebilde und Beschriftungen	502
8.5 Symbole mit variablen Formen	508
8.6 Text in Formeln	512
8.7 Feinabstimmung des Formellayouts	514
8.8 Fonts in Formeln	522
8.9 Symbole in Formeln	540

Die Standardversion von $\text{\LaTeX}$ eignet sich hervorragend, um einfache mathematische Formeln in Dokumente zu integrieren. Wenn man jedoch zahlreiche komplexe abgesetzte Formeln oder sehr anspruchsvolle mathematische Konstrukte setzen will, reicht $\text{\LaTeX}$ allein nicht aus. Man kann zwar neue Befehle oder Umgebungen definieren, die das Eintippen der Formeln erleichtern, das ist jedoch nicht die beste Lösung. Die American Mathematical Society (AMS) stellt ein umfangreiches Paket namens `amsmath` zur Verfügung, das die Arbeit an mathematischen Dokumenten zeitsparender und einheitlicher gestaltet.¹ Es bildet das Kernstück einer Sammlung von Paketen, die unter dem Namen $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\LaTeX}$ [8] geführt werden und ist Hauptthema dieses Kapitels. Ein nützliches Buch von George Grätzer [62] beschäftigt sich ebenfalls im Detail mit diesen Paketen.

Dieses Kapitel gibt anhand von Beispielen Einblick in einen Großteil der zahlreichen Funktionen dieser sowie einiger nahe verwandter Pakete. Außerdem verweist es auf einzelne andere relevante Pakete und führt in wichtige Hintergründe für das Setzen von Formeln in $\text{\TeX}$ ein. Damit deckt es auch einige der Standard- $\text{\LaTeX}$ -Funktionen für das Setzen und Gestalten von Formeln ab, und enthält einige allgemeine Tipps für den Formelsatz, obgleich dies nicht das Hauptziel des Kapitels ist.

¹Dieses Paket basiert auf den Erweiterungen zu `plain\TeX`, die als $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\text{\TeX}$ bekannt sind.

Es ist auch definitiv kein umfassendes Handbuch für den fachgerechten Formelsatz mit $\text{\LaTeX}$. Viele der Beispiele sollen spezielle Sachverhalte anschaulicher machen und sind deshalb weder ein Maßstab für ansprechende Gestaltung noch für gute Mathematik und auch nicht unbedingt für vorbildliche $\text{\LaTeX}$ -Programmierung.

Das Buch *Math into Type* [157] von Ellen Swanson ist eine Anleitung, wie man Formeln entsprechend der Ende des 20. Jahrhunderts vorherrschenden Praxis setzt. Wie man diese Anleitung mit $\text{\TeX}$ oder ebenso mit Standard- $\text{\LaTeX}$ umsetzen kann, darüber geben die Kapitel 16–18 des Buches *The $\text{\TeX}$ book* [87] von Donald Knuth detailliert Aufschluss.

Für den Großteil des in diesem Kapitel beschriebenen Materials muss man mindestens das `amsmath`-Paket in der Präambel seines Dokumentes laden. Wenn weitere Pakete erforderlich sind, enthalten die jeweiligen Beispiele einen entsprechenden Hinweis. Eine detaillierte Installations- und Gebrauchsanleitung ist in den einzelnen Paketen enthalten.

8.1 Eine Einführung in $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$

Das $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$ -Projekt wurde 1987 ins Leben gerufen und drei Jahre später wurde $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$ in der Version 1.0 veröffentlicht. Es handelte sich um die originale Umsetzung der mathematischen Funktionalitäten von $\mathcal{A}\mathcal{M}\mathcal{S}\text{\TeX}$ von Michael Spivak in $\text{\LaTeX}$. Sie wurde von Frank Mittelbach und Rainer Schöpf durchgeführt, die als Berater für die American Mathematical Society arbeiteten und dabei von Michael Downes unterstützt wurden, der bei AMS im technischen Bereich tätig war. 1994 wurden zusammen mit David Jones weitere Arbeiten vorgenommen. Diese Arbeiten wurden von Michael Downes koordiniert, und die Pakete wurden seitdem unter seiner Anleitung und der Schirmherrschaft der AMS weiter unterstützt und wesentlich verbessert.¹

Ein Dankeschön an
einen großartigen
Menschen!

Michael Downes hätte dieses Kapitel verfasst, wenn er nicht im Frühjahr 2003 gestorben wäre. Ein Großteil des Textes basiert auf der Dokumentation, die er für $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$ geschrieben hat. Darum ist dies ein aus tiefstem Herzen ausgesprochenes Zeichen der Anerkennung der verbliebenen Autoren dieses Buches, um das Leben und das Werk eines großartigen Freundes und Kollegen zu ehren, mit dem wir gemeinsam viele spannende Programmierabenteuer in den unerforschten Tiefen von $\text{\TeX}$ bestanden haben.

Verfügbare
Paketoptionen

Das `amsmath`-Paket kennt nur wenige Optionen. Die meisten betreffen lediglich die genaue Positionierung der „Grenzen“ für verschiedene Arten mathematischer Operatoren (Abschnitt 8.4.4) oder die Platzierung der Formeltags (Abschnitt 8.2.4).

Die folgenden drei Optionen werden oft im `\documentclass`-Befehl als Klassenoptionen verwendet. Sie werden jedoch auch richtig interpretiert, wenn man das `amsmath`-Paket mit dem `\usepackage`-Befehl lädt.

`reqno` Gleichungsnummern (Tags) rechts positionieren (Voreinstellung).

`leqno` Gleichungsnummern (Tags) links positionieren.²

¹Einige Informationen in diesem Kapitel basieren auf der Dokumentation, die mit $\mathcal{A}\mathcal{M}\mathcal{S}\text{\TeX}$ verteilt wird (mit freundlicher Genehmigung der American Mathematical Society).

²Die $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$ -Dokumentenklassen verwenden die Voreinstellung `leqno`.

`fleqn` Gleichungen mit einem festen Einzug vom linken Rand positionieren anstatt diese zu zentrieren.

Die $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ -Distribution enthält auch Komponenten, die mithilfe des `\usepackage`-Befehls einzeln geladen werden können. Dies sind zum einen kleinere Pakete, die einige der Funktionen des `amsmath`-Paketes separat verfügbar machen:

*Verfügbare
Unterpakete*

`amsofn` Stellt `\DeclareMathOperator` zum Definieren neuer Operatornamen bereit, wie etwa `\Ker` und `\esssup`.

`amstext` Stellt den Befehl `\text` zur Verfügung, um Textfragmente im richtigen Schriftgrad zu formatieren.

Die folgenden Pakete erweitern `amsmath` um zusätzliche Funktionen und müssen explizit geladen werden. Sie werden hier lediglich der Vollständigkeit halber aufgeführt.

Erweiterungspakete

`amscd` Definiert einige Befehle, mit denen sich kommutative Diagramme einfacher erzeugen lassen. Dazu wird die `CD`-Umgebung bereitgestellt (siehe Abschnitt 8.3.4 auf Seite 500). Diagonale Pfeile werden nicht unterstützt.

`amsthm` Bietet eine Methode zum Deklarieren theoremähnlicher Gebilde sowie eine `proof`-Umgebung. Es wurde in Abschnitt 3.3.3 auf Seite 146 besprochen.

`amsxtra` Stellt verschiedene Kleinigkeiten bereit um die Rückwärtskompatibilität zu gewährleisten, wie zum Beispiel `\fracwithdelims`, `\accentedsymbol` sowie Befehle, die Akzente als Hochstellungen positionieren.

`upref` Lässt `\ref` alle Querverweisnummern in einer aufrechten Serifenschrift setzen, unabhängig vom Kontext.

Die wichtigste Dokumentation zu diesen Paketen ist der *User's Guide for the amsmath Package (Version 2.0)* [8].

Die aktuelle $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ -Sammlung umfasst drei Dokumentenklassen: `amsart`, `amsproc` und `amsbook`, entsprechend den $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ -Klassen `article`, `proc` beziehungsweise `book`. Sie wurden zur Formatierung von Manuskripten entwickelt, die bei der AMS [6] eingereicht werden sollen. Es spricht jedoch nichts gegen eine Verwendung für andere Zwecke. Die Klassendateien laden automatisch das `amsmath`-Paket, so dass ein Dokument einfach mit `\documentclass{amsart}` eingeleitet werden kann. Sie werden in diesem Buch nicht eigens beschrieben, da ihre Schnittstelle den $\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ -Standardklassen sehr ähnelt. Näheren Aufschluss über ihre Verwendung gibt [6].

*Die $\mathcal{A}_{\mathcal{M}}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$ -
Dokumentenklassen*

Einige Informationen in diesem Kapitel beziehen sich auf eine weitere Paketsammlung der American Mathematical Society und zwar auf die `AMSfonts`-Distribution. Ihre Pakete, die im Folgenden aufgeführt sind, richten verschiedene Schriften und Befehle für den Formelsatz ein.

*Die
AMSfonts-Sammlung*

amsfonts Definiert die Befehle `\mathfrak` und `\mathbb` und richtet folgende Fonts ein: **msam** (zusätzliche mathematische Zeichen A), **msbm** (zusätzliche mathematische Zeichen B und Blackboard Bold), **eufm** (Euler Fraktur) sowie zusätzliche Schriftgrade von **cmmb** (fettkursiver Mathematikfont und griechische Kleinbuchstaben in Fettdruck) und **cmbsy** (fette mathematische Zeichen und fette Schreibschrift).

amssymb Definiert die Namen aller mathematischen Symbole der AMSfonts-Sammlung. Diese Befehle werden in Abschnitt 8.9 besprochen. Das Paket lädt automatisch das **amsfonts**-Paket.

eufrak Richtet, wie in Abschnitt 7.7.10 besprochen, die Fonts für die Buchstaben der Euler Fraktur ein (`\mathfrak`). Dieses Alphabet wird auch durch das **amsfonts**-Paket bereitgestellt.

eucal Lässt `\mathcal` anstelle der üblichen Computer Modern Schreibschrift die Euler Schreibschrift verwenden (nähere Einzelheiten in Abschnitt 7.7.10).

Alle diese Pakete kennen die Option **psamsfonts**, die $\text{\LaTeX}$ anweist, die Y&Y/Blue Sky Research-Version dieser Schriften aus der AMSfonts-Sammlung zu verwenden. Das ist nur sinnvoll, wenn die Fonts auch installiert sind; man erhält sie über CTAN und sie sind in vielen neueren $\text{\LaTeX}$ -Distributionen bereits per Voreinstellung enthalten. Die vorgenannten Pakete sind hauptsächlich im *User's Guide to AMSFonts Version 2.2d* [9] dokumentiert.

Einige wichtige Warnhinweise

Vorsicht
bei zerbrechlichen
Befehlen

Viele der in diesem Kapitel beschriebenen Befehle sind zerbrechlich und müssen daher in bewegten Argumenten mit dem Befehl `\protect` geschützt werden (siehe Anhang B.1 auf Seite 927). Wenn also seltsame Fehlermeldungen auftreten, ist die Ursache wahrscheinlich ein fehlendes `\protect`.

Keine Kurzformen
für Umgebungen

Es ist absolut nicht ratsam, Kurzformen für $\text{\LaTeX}$ -Umgebungen zu verwenden. Bei den in diesem Kapitel beschriebenen Formelumgebungen von **amsmath** wirken sich solche Kurzformen immer verheerend aus – Finger weg! Aus sehr ähnlichen Gründen lassen sich auch keine Verbatim-Umgebungen oder -Befehle in diesen Umgebungen verwenden. Hier einige Beispiele für verhängnisvolle Deklarationen:

```
\newenvironment{mlt}{\begin{multline}}{\end{multline}}
\newcommand\bga{\begin{gather}} \newcommand\ega{\end{gather}}
```

Beide erzeugen Fehlermeldungen der Art „`\begin{...} ended by ...`“. Man kann aber folgendermaßen Synonyme und Varianten für diese Umgebungen definieren:

```
\newenvironment{mlt}{\multline}{\endmultline}
\newenvironment{longgather}{\allowdisplaybreaks\gather}{\endgather}
```

Dabei müssen sie jedoch die Befehlsform einer existierenden Umgebung als letzten Befehl im „begin-Code“ und den entsprechenden Befehl `\end...` am Anfang des „end-Codes“ aufweisen. Nähere Einzelheiten hierzu sind in Abschnitt A.1.3 erläutert.

<code>equation</code>	<code>equation*</code>	Eine Zeile, eine Formel
<code>multline</code>	<code>multline*</code>	Eine nicht ausgerichtete mehrzeilige Formel, eine Formelnummer
<code>gather</code>	<code>gather*</code>	Mehrere nicht ausgerichtete Formeln
<code>align</code>	<code>align*</code>	Mehrere Formeln mit unterschiedlicher Ausrichtung
<code>flalign</code>	<code>flalign*</code>	Mehrere Formeln: horizontal gedehnte Form von <code>align</code>
<code>split</code>		Eine einfache Ausrichtung innerhalb einer mehrzeiligen Formel
<code>gathered</code>		Eine „Miniseite“ mit nicht ausgerichteten Formeln
<code>aligned</code>		Eine „Miniseite“ mit mehrfacher Ausrichtung

Tabelle 8.1: Formel-Umgebungen des `amsmath`-Paketes

8.2 Umgebungen für abgesetzte Formeln

Das `amsmath`-Paket definiert verschiedene Umgebungen für abgesetzte Formeln. Sie umfassen ein- und mehrzeilige Formeln mit einem oder mehreren Ausrichtungspunkten und verschiedenen Nummerierungsvarianten für Formelteile innerhalb der Umgebungen.

In diesem Abschnitt hat der Begriff „Formel“ eine ganz bestimmte Bedeutung: Er bezeichnet einen *logisch* separaten Abschnitt einer Formelgruppe, der häufig aus Gründen der Bezugnahme nummeriert und außerdem mit einem Label (normalerweise der Nummer in Klammern) versehen ist, das häufig als *Tag* bezeichnet wird.

Tabelle 8.1 zeigt eine vollständige Liste aller Umgebungen, die man zum Setzen abgesetzter Formeln (oder Formelgruppen) benötigt. Die Mehrzahl dieser Umgebungen wird zusammen mit entsprechenden Anwendungsbeispielen in diesem Abschnitt abgehandelt. Soweit das sinnvoll ist, existieren sie auch als Sternformen ohne Nummerierungen oder Tags.

Die Beispiele zu den Ausrichtungsumgebungen verwenden noch weitere Befehle des `amsmath`-Paketes. An dieser Stelle ist ein detailliertes Verständnis ihrer Funktionsweise nicht notwendig, die späteren Abschnitte halten jedoch für interessierte Leser weitere Informationen bereit. Die Darstellungsbreite (`display width`) ist das Maß, das den rechten und linken Rand (oder die Weite) der abgesetzten Formeln definiert. In den Beispielen wird diese Breite durch dünne blue vertikale Linien am linken und rechten Rand der Darstellung angezeigt.

Soweit nicht anders angegeben, sind die Formeln aller Beispiele in diesem Kapitel zentriert und ihre Formelnummern oder Tags stehen rechts (entsprechend der Voreinstellungen des `amsmath`-Paketes). Wenn das `amsmath`-Paket oder die Dokumentenklasse mit der Option `leqno` geladen wird, befinden sich die Tags auf der linken Seite der Formeln.

Bsp.
8-2-1

$$(1) \quad (a+b)^2 = a^2 + 2ab + b^2$$

$$\sin^2 \eta + \cos^2 \eta = 1$$

```
\usepackage[leqno]{amsmath}
\begin{equation} (a+b)^2=a^2+2ab+b^2 \end{equation}
\[ \sin^2\eta+\cos^2\eta = 1 \]
```

Die Option `fleqn` setzt die Formeln mit einem festen Einzug vom linken Rand anstatt in der Kolumne zentriert. Dazu muss normalerweise die Breite

des Einzugs in der Präambel festgelegt werden. Sie hat den Wert der elastischen Länge `\mathindent`, die auf die Breite des Einzugs der ersten Listenebene voreingestellt ist – was wahrscheinlich nicht der gewünschten Größe entspricht! Man beachte die Unterschiede zwischen dem vorigen und dem nächsten Beispiel. In diesem speziellen Fall ist die Option `reqno` eigentlich überflüssig (da sie voreingestellt ist). Sie zwingt die Formelnummer jedoch immer auf die rechte Seite, egal wie die Einstellungen in der Dokumentenklasse lauten.

$$(a + b)^2 = a^2 + 2ab + b^2 \quad (1)$$

$$\sin^2 \eta + \cos^2 \eta = 1$$

```
\usepackage[fleqn,reqno]{amsmath}
\setlength{\mathindent}{1pc}

\begin{equation} (a+b)^2=a^2+2ab+b^2 \end{equation}

\[ \sin^2\eta+\cos^2\eta = 1 \]
```

Bsp.
8-2-2

Wie die weiteren Beispiele noch zeigen, werden `&` und `\`, genau wie in Standard- $\text{\LaTeX}$, als Spalten- und Zeilenbegrenzungen in abgesetzten, ausgerichteten Formeln verwendet. Wie sie genau benutzt werden, ändert sich jedoch je nach `amsmath`-Umgebung (siehe nächster Abschnitt).

8.2.1 Ein Vergleich mit Standard- $\text{\LaTeX}$

Einige der mehrzeiligen Formelumgebungen lassen das Ausrichten von Formelteilen zu. Im Gegensatz zu den $\text{\LaTeX}$ -Umgebungen `eqnarray` und `eqnarray*` verwendet das `amsmath`-Paket eine etwas andere und direktere Methode zum Markieren der Ausrichtungspunkte. Die $\text{\LaTeX}$ -Umgebung `eqnarray*` ähnelt der Umgebung `array` mit `{rcl}` als Präambel und erfordert daher zwei `&`-Zeichen zum Kennzeichnen der beiden Ausrichtungspunkte. In den entsprechenden `amsmath`-Strukturen gibt es nur einen Ausrichtungspunkt (ähnlich wie bei einer `{rl}`-Präambel), so dass nur ein `&`-Zeichen links des auszurichtenden Symbols (normalerweise ein Relationssymbol) verwendet werden sollte.

Die `amsmath`-Strukturen schreiben feste Abstände an den Ausrichtungspunkten vor, während die `eqnarray`-Umgebung je nach den Einstellungen für die Umgebung `array` zusätzlichen Leerraum erzeugt. Die Unterschiede werden im nächsten Beispiel deutlich, in dem dieselbe Gleichung nacheinander mithilfe der Umgebungen `equation`, `align` und `eqnarray` gesetzt wird. Dabei sind die Abstände in der `eqnarray`-Umgebung, gemessen an den üblichen Standards für das Setzen von Formeln, zu groß.

$$x^2 + y^2 = z^2$$

$$x^2 + y^2 = z^2$$

$$x^3 + y^3 < z^3$$

$$x^2 + y^2 = z^2$$

$$x^3 + y^3 < z^3$$

```
(1) \usepackage{amsmath}
    \begin{equation}
      x^2 + y^2 = z^2
    \end{equation}
(2) \begin{align}
    x^2 + y^2 &= z^2 \\
    x^3 + y^3 &< z^3
  \end{align}
(4) \begin{eqnarray}
    x^2 + y^2 &= z^2 \\
    x^3 + y^3 &< z^3
  \end{eqnarray}
```

Bsp.
8-2-3

Genau wie in Standard- $\text{\LaTeX}$ werden die Zeilenenden in einer `amsmath`-Formel durch einen `\`-Befehl (oder das Ende der Umgebung) festgelegt. Da das Umbrechen von Zeilen in Formeln normalerweise voraussetzt, dass man die Struktur der Formel genau versteht, wird es allgemein als eine Aufgabe angesehen, die über die Fähigkeiten heutiger Software hinausgeht. Eines der letzten größeren Projekte von Michael Downes befasste sich jedoch mit eben diesem Problem. Es führte zu dem Paket `breqn` (nähere Einzelheiten in [43]).

Im Gegensatz zu `eqnarray` erlauben die `amsmath`-Umgebungen entsprechend ihrer Voreinstellung keine Seitenumbrüche zwischen ihren Zeilen (vgl. Abschnitt 8.2.10).

Ein weiterer Unterschied betrifft die Verwendung der Befehle `\*` oder `\[länge]` innerhalb von Umgebungen für abgesetzte Formeln. Bei `amsmath` dürfen zwischen `\` und `[` oder `*` keine Leerzeichen stehen, sonst wird das optionale Argument oder der Stern nicht erkannt. Der Grund liegt darin, dass eckige Klammern und Sterne häufig in Formeln vorkommen, und diese Einschränkung soll verhindern, dass eine echte eckige Klammer der Formel fälschlicherweise als Anfang eines optionalen Argumentes interpretiert wird.

Schließlich gibt es noch eine weniger offensichtliche Änderung, die Anwendern sehr wahrscheinlich keine Probleme bereiten wird: in Standard- $\text{\LaTeX}$ ist der Parameter `\mathindent` keine elastische Länge, in `amsmath` wird er jedoch dazu. Die Ursachen und Konsequenzen dieser Änderung werden in der Datei `amsmath.dtx` besprochen, der dokumentierten Quelldatei des `amsmath`-Paketes.

*Leerzeichen nach `\`
werden nicht ignoriert*

8.2.2 Eine Formel in einer Zeile

Die `equation`-Umgebung erzeugt eine einzelne Formel mit automatisch generierter Nummer (oder generiertem Tag), welche je nach verwendeter Option ganz außen links oder rechts positioniert wird (siehe Abschnitt 8.2.11); `equation*` ergibt dasselbe Ergebnis, jedoch ohne Tag.¹

Die Existenz eines Tags hat dabei keinerlei Auswirkung auf die Positionierung der Inhalte. Wenn in der einen Zeile nicht genügend Platz ist, wird das Tag nach oben oder unten verschoben: in die Zeile davor, wenn die Formelnummern links stehen, und in die nächste Zeile, wenn sie rechts stehen.

Bsp.
8-2-4

$$(1) \quad n^2 + m^2 = k^2 \quad p > 2$$

```
\usepackage[leqno]{amsmath}
\begin{equation*} n^2 + m^2 = k^2 \end{equation*}
\begin{equation}
  n^p + m^p \neq k^p \quad \quad p > 2
\end{equation}
```

8.2.3 Eine Formel in mehreren Zeilen: ohne Ausrichtung

Die `multline`-Umgebung ist eine Abwandlung der `equation`-Umgebung für Formeln, die nicht in eine einzelne Zeile passen. In dieser Umgebung muss man die Zeilenumbrüche mit `\` festlegen, da sie nicht automatisch gefunden werden.

¹Standard- $\text{\LaTeX}$ verfügt ebenfalls über die Umgebung `equation`, nicht aber über die Sternform `equation*`, da letztere der Standardumgebung `displaymath` entspricht.

Die erste Zeile in dieser Umgebung wird mit einem Einzug vom linken Rand ausgerichtet, und die letzte Zeile mit dem gleichen Einzug vom rechten Rand.¹ Die Größe des Einzugs entspricht dabei dem Wert des Längenparameters `\multlinegap`, der mit den L^AT_EX-Befehlen `\setlength` und `\addtolength` verändert werden kann.

In einer `multline`-Umgebung mit mehr als zwei Zeilen wird jede Zeile, außer der ersten und der letzten, einzeln innerhalb der Breite der Darstellung zentriert (es sei denn, man verwendet die Option `fleqn`). Mit dem Befehl `\shoveleft` bzw. `\shoveright` in einer Zeile kann man diese jedoch auch nach links oder rechts ausrichten.

Eine `multline`-Umgebung stellt eine einzige (logische) Formel dar und verfügt daher nur über ein einziges Tag, `multline*` über gar keines. Daher können einzelne Zeilen nicht durch die Befehle `\tag` oder `\notag` verändert werden. Wenn ein Tag vorhanden ist, wird es mit der Voreinstellung `reqno` ganz rechts in der letzten Zeile positioniert, mit der Option `leqno` ganz links in der ersten Zeile.

Erste Zeile einer `multline`

Zentrierte innere Zeile

Eine rechte Innere

Noch eine zentrierte Innere

Weitere zentrierte Innere

Eine linke Innere

Letzte Zeile einer `multline` (1)

```
\usepackage{amsmath}
\begin{multline}
\text{Erste Zeile einer multline} \\
\text{Zentrierte innere Zeile} \\
\shoveright{\text{Eine rechte Innere}} \\
\text{Noch eine zentrierte Innere} \\
\text{Weitere zentrierte Innere} \\
\shoveleft{\text{Eine linke Innere}} \\
\text{Letzte Zeile einer multline}
\end{multline}
```

Bsp.
8-2-5

Das nächste Beispiel zeigt, wie sich `\multlinegap` auswirkt. In der ersten Einstellung stehen die „*dy*“s genau untereinander und es sieht so aus, als fehle bei der ersten Zeile das Tag.

$$\sum_{t \in \mathbf{T}} \int_a^t \left\{ \int_a^t f(t-x)^2 g(y)^2 dx \right\} dy$$

$$= \sum_{t \notin \mathbf{T}} \int_t^a \left\{ g(y)^2 \int_t^a f(x)^2 dx \right\} dy \quad (2)$$

```
\usepackage{amsmath}
\begin{multline} \tag{2}
\sum_{t \in \mathbf{T}} \int_a^t \biggl\lbrace \int_a^t f(t-x)^2 \, , \\
\qquad\qquad\qquad g(y)^2 \, , dx \biggr\rbrace \, dy \\
= \sum_{t \notin \mathbf{T}} \int_t^a \biggl\lbrace g(y)^2 \int_t^a \\
\qquad\qquad\qquad f(x)^2 \, , dx \biggr\rbrace \, dy
\end{multline}
```

Bsp.
8-2-6

Wenn der Wert des Parameters auf null gesetzt wird, ändert sich wegen des Tags an der zweiten Zeile nichts, während die erste Zeile nach links ver-

¹Man sollte die `multline`-Umgebung niemals für einzeilige Formeln verwenden, da sich die Auswirkungen nicht vorhersehen lassen.

schoben wird. Dadurch wird deutlicher, dass es sich nur um eine einzige Formel handelt.

Bsp.
8-2-7

$$\sum_{t \in \mathbf{T}} \int_a^t \left\{ \int_a^t f(t-x)^2 g(y)^2 dx \right\} dy$$

$$= \sum_{t \notin \mathbf{T}} \int_t^a \left\{ g(y)^2 \int_t^a f(x)^2 dx \right\} dy \quad (2)$$

```
\usepackage{amsmath}
\setlength{\multlinegap}{0pt}
\begin{multline} \tag{2}
  \sum_{t \in \mathbf{T}} \int_a^t \biggl\{ \int_a^t f(t-x)^2 \, , \\
    g(y)^2 \, dx \biggr\} dy \\
  = \sum_{t \notin \mathbf{T}} \int_t^a \biggl\{ g(y)^2 \int_t^a \\
    f(x)^2 \, dx \biggr\} dy
\end{multline}
```

8.2.4 Eine Formel in mehreren Zeilen: mit Ausrichtung

Wenn eine einfache Ausrichtung für eine einzelne mehrzeilige Formel benötigt wird, ist die `split`-Umgebung fast immer die beste Wahl. Sie markiert den Ausrichtungspunkt in jeder Zeile mit einem einzelnen `&`-Zeichen.

Bsp.
8-2-8

$$\begin{aligned} (a-b)^3 &= (a-b)(a-b)^2 \\ &= (a-b)(a^2 - 2ab + b^2) \\ &= a^3 - 3a^2b + 3ab^2 - b^3 \end{aligned} \quad (1)$$

```
\usepackage{amsmath}
\begin{equation}
\begin{split}
  (a-b)^3 &= (a-b) (a-b)^2 \\
           &= (a-b)(a^2 - 2ab + b^2) \\
           &= a^3 - 3a^2b + 3ab^2 - b^3
\end{split}
\end{equation}
```

Da ihr Inhalt immer nur aus einer einzigen (logischen) Formel besteht, erzeugt `split` selbst keine Nummerierung, so dass es auch keine Sternvariante gibt. Soweit erforderlich stellt die äußere Umgebung für abgesetzte Formeln die benötigten Tags zur Verfügung.

Abgesehen von Befehlen wie `\label` oder `\notag`, die kein sichtbares Material erzeugen, sollte eine `split`-Struktur normalerweise die gesamte aufzuteilende Formel umfassen. Diese kann entweder aus einer ganzen `equation`- oder `equation*`-Umgebung bestehen oder aus einer ganzen Zeile einer `gather`- oder `gather*`-Umgebung (siehe Abschnitt 8.2.5).

Durch die (voreingestellte) Option `centertags` wird das Tag (und alles weitere Material der Formel außerhalb der `split`-Umgebung) an der Gesamthöhe des `split`-Materials vertikal zentriert. Durch die Option `tbtags` wird das Tag, sofern es rechts steht, an der letzten Zeile, und wenn es links steht, an der ersten Zeile der `split`-Umgebung ausgerichtet.

Bsp.
8-2-9

$$\begin{aligned} (a+b)^3 &= (a+b)(a+b)^2 \\ &= (a+b)(a^2 + 2ab + b^2) \\ &= a^3 + 3a^2b + 3ab^2 + b^3 \end{aligned} \quad (1)$$

```
\usepackage[tbtags]{amsmath}
\begin{equation}
\begin{split}
  (a+b)^3 &= (a+b) (a+b)^2 \\
           &= (a+b)(a^2 + 2ab + b^2) \\
           &= a^3 + 3a^2b + 3ab^2 + b^3
\end{split}
\end{equation}
```


Im nächsten Beispiel wird die horizontale Position mit dem Befehl `\phantom` justiert. Er wird zunächst in der Präambel verwendet, um ein „unsichtbares Relationssymbol“ in der Breite seines Argumentes (hier: $=$) zu definieren. Innerhalb des Beispiels dient er der Ausrichtung bestimmter Zeilen, indem er sie mit einer „Phantom- oder unsichtbaren Unterformel“ einleitet (siehe Abschnitt 8.7.2 auf Seite 516). Das leere Klammernpaar $\{\}$ entspricht `\mathord{\}` und dient als unsichtbarer „Buchstabe“ der Breite null, der benötigt wird, um innerhalb von $+h$ den richtigen Abstand herzustellen (ohne $\{\}$ ergibt sich folgende Darstellung: $+h$).

```
\usepackage{amsmath}
\newcommand\relphantom[1]{\mathrel{\phantom{#1}}}
\newcommand\ve{\varepsilon} \newcommand\tve{t_{\varepsilon}}
\newcommand\vf{\varphi} \newcommand\yvf{y_{\varphi}}
\newcommand\bfE{\mathbf{E}}
\begin{equation}
\begin{split}
f_{h,\ve}(x,y)
&= \ve \bfE_{\{x,y\}} \int_0^{\tve} L_{\{x,\yvf(\ve u)\}} \vf(x) \, du \quad \backslash\backslash \\
&= h \int L_{\{x,z\}} \vf(x) \rho_x(dz) \quad \backslash\backslash \\
&\quad \relphantom{=} \{ \} + h \biggl[ \\
&\quad \quad \frac{1}{\tve} \\
&\quad \quad \biggl( \bfE_{\{y\}} \int_0^{\tve} L_{\{x,y^x(s)\}} \vf(x) \, ds \\
&\quad \quad \quad - \tve \int L_{\{x,z\}} \vf(x) \rho_x(dz) \\
&\quad \quad \biggr) + \quad \backslash\backslash \\
&\quad \relphantom{=} \{ \} + h \biggl[ \} \\
&\quad \quad \frac{1}{\tve} \\
&\quad \quad \biggl( \bfE_{\{y\}} \int_0^{\tve} L_{\{x,y^x(s)\}} \vf(x) \, ds \\
&\quad \quad \quad - \bfE_{\{x,y\}} \int_0^{\tve} L_{\{x,\yvf(\ve s)\}} \\
&\quad \quad \quad \quad \vf(x) \, ds \\
&\quad \biggr) \biggr] \biggr] \\
\end{split}
\end{equation}
```

Man beachte, dass das Tag mit der Formelnummer in die Zeile unterhalb des dargestellten Materials verschoben wurde. Auch wenn das keine sehr kluge Entscheidung zu sein scheint, ist es doch das beste Ergebnis, das die eingebauten automatischen Routinen des Systems hier erzeugen können.

$$\begin{aligned}
 f_{h,\varepsilon}(x,y) &= \varepsilon \mathbf{E}_{x,y} \int_0^{t_\varepsilon} L_{x,y_\varphi(\varepsilon u)} \varphi(x) \, du \\
 &= h \int L_{x,z} \varphi(x) \rho_x(dz) \\
 &\quad + h \left[\frac{1}{t_\varepsilon} \left(\mathbf{E}_y \int_0^{t_\varepsilon} L_{x,y^x(s)} \varphi(x) \, ds - t_\varepsilon \int L_{x,z} \varphi(x) \rho_x(dz) \right) + \right. \\
 &\quad \quad \left. \frac{1}{t_\varepsilon} \left(\mathbf{E}_y \int_0^{t_\varepsilon} L_{x,y^x(s)} \varphi(x) \, ds - \mathbf{E}_{x,y} \int_0^{t_\varepsilon} L_{x,y_\varphi(\varepsilon s)} \varphi(x) \, ds \right) \right] \\
 &\quad (1)
 \end{aligned}$$

Bsp.
8-2-10

8.2.5 Formelgruppen ohne Ausrichtung

Die Umgebung `gather` kann zwei oder mehr Formeln in einer einzigen Darstellung zusammenfassen, ohne sie zueinander auszurichten. Dabei wird jede Formel separat innerhalb der Darstellungsbreite zentriert und hat, soweit erforderlich, ein eigenes Tag. Jede Zeile einer `gather`-Umgebung stellt eine einzelne (logische) Formel dar.

Bsp.
8-2-11

$$(a+b)^2 = a^2 + 2ab + b^2 \quad (1)$$

$$(a+b) \cdot (a-b) = a^2 - b^2 \quad (2)$$

```
\usepackage{amsmath}
\begin{gather}
(a+b)^2 = a^2 + 2ab + b^2 \\
(a+b) \cdot (a-b) = a^2 - b^2
\end{gather}
```

Mit dem Befehl `\notag` innerhalb einer logischen Zeile lässt sich die Formelnummer für diese Zeile unterdrücken. Die Umgebung `gather*` unterdrückt alle Formelnummern.

Bsp.
8-2-12

$$D(a, r) \equiv \{z \in \mathbf{C} : |z - a| < r\}$$

$$\operatorname{seg}(a, r) \equiv \{z \in \mathbf{C} : \Im z < \Im a, |z - a| < r\} \quad (1)$$

$$C(E, \theta, r) \equiv \bigcup_{e \in E} c(e, \theta, r) \quad (2)$$

```
\usepackage{amsmath}
\begin{gather}
D(a,r) \equiv \{ z \in \mathbf{C} \\
\quad \quad \quad \colon |z - a| < r \} \quad \notag \\
\operatorname{seg}(a, r) \equiv \\
\quad \{ z \in \mathbf{C} \colon \\
\quad \quad \Im z < \Im a, \quad |z - a| < r \} \\
C(E, \theta, r) \equiv \\
\quad \bigcup_{e \in E} c(e, \theta, r)
\end{gather}
```

8.2.6 Formelgruppen mit einfacher Ausrichtung

Die Umgebung `align` sollte für zwei oder mehr Formeln in einer einzigen abgesetzten Darstellung mit vertikaler Ausrichtung verwendet werden. Ihre einfachste Form markiert den Ausrichtungspunkt in jeder Zeile mit einem einzelnen `&`-Zeichen, das normalerweise direkt vor dem Relationssymbol steht.

Bsp.
8-2-13

$$(a+b)^3 = (a+b)(a+b)^2 \quad (1)$$

$$= (a+b)(a^2 + 2ab + b^2) \quad (2)$$

$$= a^3 + 3a^2b + 3ab^2 + b^3 \quad (3)$$

$$x^2 + y^2 = 1 \quad (4)$$

$$x = \sqrt{1 - y^2} \quad (5)$$

```
\usepackage{amsmath}
\begin{align}
(a+b)^3 &= (a+b)(a+b)^2 \\
&= (a+b)(a^2 + 2ab + b^2) \\
&= a^3 + 3a^2b + 3ab^2 + b^3
\end{align}
\begin{align}
x^2 + y^2 &= 1 \\
x &= \sqrt{1-y^2}
\end{align}
```

8.2.7 Mehrfache Ausrichtung: `align` und `flalign`

Eine `align`-Umgebung kann mehr als einen Ausrichtungspunkt enthalten. Sie enthält so viele Spaltenpaare wie nötig und ähnelt einer `array`-Umgebung mit einer Präambel der Form `{r1r1...}`. Wenn sie aus n solcher `r1`-Spaltenpaare

auch nur innerhalb von Formelumgebungen genutzt und sie erzeugen selbst keine Tags.

$$\left. \begin{array}{l} x^2 + y^2 = 1 \\ x = \sqrt{1 - y^2} \\ \text{sowie } y = \sqrt{1 - x^2} \end{array} \right\} \begin{array}{l} (a + b)^2 = a^2 + 2ab + b^2 \\ (a + b) \cdot (a - b) = a^2 - b^2 \end{array} \quad (1)$$

```
\usepackage{amsmath}
\begin{equation}
\begin{aligned}
x^2 + y^2 &= 1 && \\
x &= \sqrt{1-y^2} && \\
\text{\text{sowie } } y &= \sqrt{1-x^2} && \\
\end{aligned}
\quad
\begin{gathered}
(a + b)^2 = a^2 + 2ab + b^2 \\
(a + b) \cdot (a - b) = a^2 - b^2
\end{gathered}
\end{equation}
```

Bsp.
8-2-18

Dieselben Formeln lassen sich, wenn auch nicht sehr schön, ebenso mit unterschiedlicher vertikaler Ausrichtung in den Umgebungen setzen.

$$\left. \begin{array}{l} x^2 + y^2 = 1 \\ x = \sqrt{1 - y^2} \\ \text{sowie } y = \sqrt{1 - x^2} \end{array} \right\} \begin{array}{l} (a + b)^2 = a^2 + 2ab + b^2 \\ (a + b) \cdot (a - b) = a^2 - b^2 \end{array} \quad (1)$$

```
\usepackage{amsmath}
\begin{equation}
\begin{aligned}[b]
x^2 + y^2 &= 1 && \\
x &= \sqrt{1-y^2} && \\
\text{\text{sowie } } y &= \sqrt{1-x^2} && \\
\end{aligned}
\quad
\begin{gathered}[t]
(a + b)^2 = a^2 + 2ab + b^2 \\
(a + b) \cdot (a - b) = a^2 - b^2
\end{gathered}
\end{equation}
```

Bsp.
8-2-19

Sie werden auf vielfältige Weise verwendet, z.B. um berühmte Formeln kreativ und sinnvoll zu gruppieren. Nebenbei gehören diese Formelumgebungen, die Miniseiten erzeugen, zu den wenigen Umgebungen des `amsmath`-Paketes, die so robust sind, dass man sie ohne Probleme innerhalb anderer Definitionen verwenden kann, wie im folgenden Beispiel:

```
\usepackage{amsmath,bm}
\newenvironment{rcase}{\left.\begin{aligned}}
{\end{aligned}\right\}}

\begin{equation*}
\begin{rcase} \bm{B}' &= -c\nabla\times\bm{E} && \\
&\bm{E}' &= c\nabla\times\bm{B} - 4\pi\bm{J} \backslash, && \\
\end{rcase} \quad \text{\text{Maxwell'sche Gleichungen}}
\end{equation*}
```

$$\left. \begin{array}{l} B' = -c\nabla \times E \\ E' = c\nabla \times B - 4\pi J \end{array} \right\} \text{Maxwell'sche Gleichungen}$$

Bsp.
8-2-20

Mithilfe des Befehls `\minalignsep` kann man auch den Abstand zwischen den Spaltenpaaren einer `aligned`-Umgebung einstellen, wie im nachfolgenden Beispiel zu sehen.

Bsp.
8-2-21

$$\begin{aligned}
 V_j &= v_j & X_i &= x_i - q_i x_j & & = u_j + \sum_{i \neq j} q_i \\
 V_i &= v_i - q_i v_j & X_j &= x_j & & U_i = u_i
 \end{aligned}
 \tag{1}$$

```

\usepackage{amsmath}
\renewcommand{\minalignsep}{5pt}
\begin{equation} \begin{aligned}
V_j &= v_j & X_i &= x_i - q_i x_j & & = u_j + \sum_{i \neq j} q_i \\
V_i &= v_i - q_i v_j & X_j &= x_j & & U_i = u_i
\end{aligned} \end{equation}

```

8.2.9 Unterbrechen abgesetzter Formeln: `\intertext`

Der Befehl `\intertext` wird für kurze Textpassagen (normalerweise höchstens ein paar Zeilen) verwendet, die innerhalb einer Anordnung von Formeln erscheinen. Er ist deshalb so wichtig, weil sein Text, der als normaler Absatz in der Darstellungsbreite gesetzt wird, die Ausrichtung der Formeln zueinander nicht beeinträchtigt. Diese Ausrichtung wäre nicht mehr möglich, wenn man eine Darstellung beendet und die nächste erst wieder nach dem Text beginnt. Dieser Befehl darf nur direkt auf einen `\\`-Befehl oder dessen Sternform folgen.

Im nächsten Beispiel befinden sich die Wörter „und schließlich“ am linken Rand außerhalb der Ausrichtung, die drei Formeln stehen jedoch linksbündig untereinander.

Bsp.
8-2-22

und schließlich

$$A_1 = N_0(\lambda; \Omega') - \phi(\lambda; \Omega') \tag{1}$$

$$A_2 = \phi(\lambda; \Omega') \phi(\lambda; \Omega) \tag{2}$$

$$A_3 = \mathcal{N}(\lambda; \omega) \tag{3}$$

```

\usepackage{amsmath}
\begin{align}
A_1 &= N_0 (\lambda ; \Omega') \\
&\quad - \phi ( \lambda ; \Omega') \\
A_2 &= \phi (\lambda ; \Omega') \\
&\quad \phi (\lambda ; \Omega) \\
\intertext{und schließlich}
A_3 &= \mathcal{N} (\lambda ; \omega)
\end{align}

```

8.2.10 Vertikale Abstände und Seitenumbrüche in und um abgesetzte Formeln

Wie in $\text{\LaTeX}$ üblich, fügt das optionale Argument `\\[länge]` in allen `amsmath`-Umgebungen für abgesetzte Formeln einen zusätzlichen vertikalen Abstand zwischen zwei Zeilen ein. Dabei darf zwischen dem `\\` und dem `[-`-Zeichen am Anfang des optionalen Argumentes kein Leerzeichen stehen. Die vertikalen Abstände oberhalb und unterhalb jeder Formelumgebung werden über die folgenden elastischen Längen gesteuert, wobei sich die Werte in Klammern auf den Befehl `\normalsize` mit der (voreingestellten) Option `10pt` in den $\text{\LaTeX}$ -Standardklassen beziehen.¹

*Abstände zwischen
den Formeln...*

*...und um die Formeln
herum*

`\abovedisplayskip`, `\belowdisplayskip` Der normale vertikale Abstand oberhalb und unterhalb abgesetzter Formeln (default 10pt plus 2pt minus 5pt).

¹Die $\mathcal{A}_{\mathcal{M}}\text{\LaTeX}$ -Dokumentenklassen verbessern diese Voreinstellungen erheblich.

`\abovedisplayshortskip`, `\belowdisplayshortskip` Der (normalerweise kleinere) vertikale Abstand, der oberhalb und unterhalb einer „kurzen abgesetzten Formel“ zugegeben wird (0pt plus 3pt bzw. 6pt plus 3pt minus 3pt). Unter einer *kurzen abgesetzten Formel* versteht man eine Formel, die rechts vom Ende der vorhergehenden Textzeile beginnt.

Bei genauerem Hinsehen kann man die Auswirkungen dieser Längenparameter im folgenden Beispiel erkennen. Die zweite Formel weist geringere Abstände auf, da der Text oberhalb nicht mit ihr überlappt.

Bisher gilt das Folgende:

$$X = a \quad a = c$$

und daher gilt

$$X = c \quad (1)$$

Und so ist die Formel nur von wenig Platz umgeben!

`\usepackage{amsmath}`

Bisher gilt das Folgende:

`\[ X = a \quad a = c \]`

und daher gilt

`\begin{equation} X = c \end{equation}`

Und so ist die Formel nur von

wenig Platz umgeben!

Bsp.
8-2-23

Da die vier Parameter `\abovedisplay..` und `\belowdisplay..` vom aktuellen Schriftgrad abhängig sind, können sie nicht mithilfe des Befehls `\setlength` in der Dokumentenpräambel geändert werden. Stattdessen müssen ihre Einstellungen in den Definitionen von `\normalsize`, `\small` usw. angepasst werden, was normalerweise in einer Dokumentenklasse geschieht.

Viele Autoren sind so umsichtig, ihr Quelldokument durch Leerzeilen zwischen bestimmten Teilstücken übersichtlicher zu gestalten. In den meisten Fällen, wie z.B. vor und nach Überschriften, stören diese Leerzeilen auch nicht. Das ist jedoch nicht immer so. Besonders um und innerhalb von Umgebungen zur Darstellung abgesetzter Formeln muss man sehr vorsichtig sein: Eine Leerzeile vor einer solchen Umgebung wird anders formatiert als erwartet, denn sie wird in einen leeren Absatz umgewandelt, der nur die unsichtbare Box für den Einzug des Absatzes enthält. Daher ist die nächste Darstellung von Leeräumen der Größe `\. .displayshortskip` umgeben. Insgesamt ergibt sich also ein ziemlich großer (möglicherweise zu großer) Abstand oberhalb der Formeln (eine ganze Leerzeile plus `\abovedisplayshortskip`) und ein ziemlich kleiner Abstand unterhalb, wie das Beispiel zeigt.

Vorsicht
bei Leerzeilen um
abgesetzte Formeln

Leerzeile oberhalb der Darstellung:

$$a \neq b$$

In beiden Fällen zu viel Platz oberhalb!

$$a \neq b \quad (1)$$

... und nicht viel Platz unterhalb!

`\usepackage{amsmath}`

Leerzeile oberhalb der Darstellung:

`\[ a \neq b \]`

In beiden Fällen zu viel Platz oberhalb!

`\begin{equation} a \neq b \end{equation}`

`\ldots\` und nicht viel Platz unterhalb!

Bsp.
8-2-24

Wenn das `amsmath`-Paket geladen ist, zeigen alle Umgebungen für abgesetzte Formeln dieses Verhalten. In Standard-L^AT_EX erscheinen die Abstände

im Zusammenhang mit `\[` jedoch seltsamerweise mehr oder weniger richtig.

Leerzeile oberhalb der Darstellung:

$$a \neq b$$

Weißraum nun korrekt, aber nicht hier:

$$a \neq b$$

In diesem Fall weniger Platz danach!

Leerzeile oberhalb der Darstellung:

`\[ a \neq b \]`

Weißraum nun korrekt, aber nicht hier:

`\begin{equation} a \neq b \end{equation}`

In diesem Fall weniger Platz danach!

Bsp.
8-2-25

(1)

Insgesamt lässt sich also sagen: Niemals Leerzeilen ober- oder unterhalb abgesetzter Formeln verwenden!

Automatische Seitenumbrüche vor und nach jeder Umgebung für abgesetzte Formeln werden durch zwei interne Penalty-Parameter gesteuert: `\predisplaypenalty` für Umbrüche vor einer Umgebung ist auf den Wert 10000 voreingestellt, das heißt Umbrüche sind dort nicht zulässig; `\postdisplaypenalty` für Umbrüche nach einer Umgebung ist auf 0 voreingestellt, d.h. Umbrüche sind erlaubt. Die Werte entsprechen den Vorgaben von Standard- $\text{\LaTeX}$ und werden durch `amsmath` nicht geändert.

*Seitenumbrüche um
abgesetzte Formeln...*

Im Gegensatz zu Standard- $\text{\LaTeX}$ sind die `amsmath`-Umgebungen für abgesetzte Formeln jedoch so voreingestellt, dass sie keine Seitenumbrüche zwischen den Zeilen einer Umgebung erlauben. Das liegt daran, dass korrekte Seitenumbrüche an diesen Stellen sehr stark vom Aufbau der einzelnen Formeln abhängen, so dass sie häufig der individuellen Aufmerksamkeit des Autors bedürfen.

*...und zwischen den
Formeln*

In `amsmath` lassen sich manuelle Seitenumbrüche am besten mit dem Befehl `\displaybreak` vornehmen. Er sollte jedoch nur dort eingesetzt werden, wo er unbedingt benötigt wird, um innerhalb einer Formeldarstellung eine Seite zu umbrechen. Der Befehl muss vor dem `\`-Befehl stehen, an dem der Seitenumbruch erfolgen darf. Dabei gilt er nur für diese Zeile und darf auch nur in Umgebungen eingesetzt werden, die eine vollständige, abgesetzte Formel erzeugen. Ähnlich wie der Standard- $\text{\LaTeX}$ -Befehl `\pagebreak` (siehe Abschnitt 6.2.2 in [106]) besitzt `\displaybreak` ein optionales Argument, das eine ganze Zahl im Wert von null bis vier sein kann, wobei der Wert die Dringlichkeit des Seitenumbruchs anzeigt: `\displaybreak[0]` bedeutet „hier darf umbrochen werden“ ohne den Seitenumbruch zu forcieren; `\displaybreak` ohne optionales Argument entspricht `\displaybreak[4]` und erzwingt einen Seitenumbruch. Mit dem gleichen Befehl lässt sich jedoch weder die Wahrscheinlichkeit für einen Umbruch senken, noch ein Umbruch verhindern. Man beachte, dass es sinnlos ist einen Umbruch in einer „Miniseiten-Darstellung“ vorzunehmen, da diese Umgebungen niemals auf zwei Seiten verteilt werden.

Genau wie das Einfügen von Zeilen- und Seitenumbrüchen in normalen Texten, geben derlei Korrekturen dem Dokument den letzten Schliff. Sie sollten daher auch erst durchgeführt werden, wenn es so gut wie fertig gestellt ist. Andernfalls kann es passieren, dass man immer und immer wieder nachbessern muss, sobald sich der Inhalt des Dokumentes ändert.

Der Befehl `\allowdisplaybreaks`, der die $\text{\LaTeX}$ -üblichen Geltungsregeln befolgt, wirkt wie ein `\displaybreak`-Befehl vor jedem Zeilenende aller Formelumgebungen. Er besitzt das gleiche optionale Argument wie `\displaybreak`. Seitenumbrüche lassen sich innerhalb des Geltungsbereiches eines `\allowdisplaybreaks`-Befehls mit `\\*` unterdrücken.

Ein `\displaybreak`-Befehl setzt sich sowohl über die Voreinstellung als auch über einen geltenden `\allowdisplaybreaks`-Befehl hinweg.

8.2.11 Formelnummern und Tags

Die Tags für Formeln werden in $\text{\LaTeX}$ normalerweise automatisch generiert und enthalten eine gedruckte Entsprechung des $\text{\LaTeX}$ -Zählers `equation`. Das geschieht in drei Stufen: Zunächst wird der Wert des `equation`-Zählers gesetzt (normalerweise durch Erhöhen), dann wird das Tag formatiert und dann wird es an der richtigen Position gedruckt.

In der Praxis sind die beiden ersten Vorgänge fast immer miteinander verknüpft. Der Wert des `equation`-Zählers wird also nur erhöht, wenn das Tag mit seiner Entsprechung automatisch gedruckt wird. Wenn eine Umgebung für abgesetzte Formeln Varianten mit und ohne Stern enthält, erzeugen diejenigen ohne Stern automatisch ein Tag für jede logische Formel (und erhöhen den `equation`-Zähler), die Sternformen jedoch nicht.

Bei diesen Varianten lässt sich das Tag (und damit die Erhöhung des Zählerwertes) für eine bestimmte logische Formel mithilfe des Befehls `\notag` (oder `\nonumber`¹) vor dem `\\`-Befehl unterdrücken. Mithilfe des Befehls `\tag` vor `\\` kann man das voreingestellte, automatische Tag durch ein selbst gestaltetes ersetzen (oder ein neues einfügen). Das Argument dieses Befehls kann jeden beliebigen normalen Text enthalten, der dann (in normalen Klammern) als Tag für diese Formel gesetzt wird.

Man beachte: Auch durch den Befehl `\tag` wird der Wert des Zählers nicht erhöht. Das voreingestellte Tag hat also nur das gleiche Erscheinungsbild wie `\tag{\theequation}`, es handelt sich jedoch um unterschiedliche Formen. Die Sternvariante `\tag*` setzt den Text ihres Argumentes ohne Klammern (und ohne irgendwelche sonstigen Zeichen, die möglicherweise durch eine bestimmte Dokumentenklasse vorgegeben sind).

`\usepackage{amsmath}`

$x^2 + y^2 = z^2$	(1)	<code>\begin{align}</code>	
$x^3 + y^3 = z^3$		<code>x^2+y^2 &= z^2 \label{eq:A}</code>	<code>\\</code>
$x^4 + y^4 = r^4$	(*)	<code>x^3+y^3 &= z^3 \notag</code>	<code>\\</code>
$x^5 + y^5 = r^5$	*	<code>x^4+y^4 &= r^4 \tag{***}</code>	<code>\\</code>
$x^6 + y^6 = r^6$	(1')	<code>x^5+y^5 &= r^5 \tag*{***}</code>	<code>\\</code>
		<code>x^6+y^6 &= r^6 \tag{\ref{eq:A}\$'\$}</code>	<code>\\</code>
$A_1 = N_0(\lambda; \Omega') - \phi(\lambda; \Omega')$	(2)	<code>A_1 &= N_0 (\lambda ; \Omega')</code>	<code>- \phi (\lambda ; \Omega') \\</code>
$A_2 = \phi(\lambda; \Omega') \phi(\lambda; \Omega)$		<code>A_2 &= \phi (\lambda ; \Omega')</code>	<code>\, , \phi (\lambda ; \Omega)</code>
	AUCH (2)		<code>\tag*{AUCH (\theequation)} \\</code>
$A_3 = \mathcal{N}(\lambda; \omega)$	(3)	<code>A_3 &= \mathcal{N} (\lambda ; \omega)</code>	<code>\\</code>
		<code>\end{align}</code>	

Bsp.
8-2-26

¹Die Befehle `\notag` und `\nonumber` sind austauschbar.

Das Besondere an diesem Beispiel ist, dass mithilfe der Befehle `\label` und `\ref` in Zeilen 1 und 5 versucht wird, eine Art „relativer Nummerierung“ der Formeln herzustellen.

Um Querverweise auf Formeln zu erleichtern, setzt der Befehl `\eqref` (siehe Beispiel 8-2-30 auf Seite 497) die Formelnummer automatisch in Klammern und fügt, falls erforderlich, eine Kursivkorrektur hinzu. In Abschnitt 2.4 auf Seite 69 werden außerdem allgemeinere Lösungen für den Umgang mit Querverweisen behandelt.

*Querverweise auf
Formeln*

8.2.12 Feinpositionierung von Tags

Die optimale Positionierung von Formeltags kann in mehrzeiligen Darstellungen zu einer recht komplexen Aufgabe werden. Die Umgebungen für abgesetzte Formeln versuchen unter allen Umständen zu vermeiden, dass die Formel die Formelnummer überdrückt. Dazu wird das Tag, soweit erforderlich, sogar nach oben oder unten in eine eigene Zeile verschoben. Die Schwierigkeit, eine abgesetzte Formel korrekt zu gestalten, kann hin und wieder dazu führen, dass die Position eines Tags nachgebessert werden muss. Das nächste Beispiel zeigt, was passieren kann, und wie man solche Schwierigkeiten behebt. Die automatisch gewählte Position lässt eindeutig zu wünschen übrig.

```
\usepackage{amsmath}
\begin{equation}
\begin{split}
\lvert I_2 \rvert &= \left\lvert \int_0^T \psi(t) \left\{ u(a, t) - \int_{\gamma(t)}^a \frac{d\theta}{k}(\theta, t) \int_a^\theta c(\xi) u_t(\xi, t) d\xi \right\} dt \right\rvert \\
&\leq C_6 \left\lvert f \int_\Omega \left| \tilde{S}_{a,-}^{-1,0} W_2(\Omega, \Gamma_l) \right| \left| |u| \overset{\circ}{\rightarrow} W_2^{\tilde{A}}(\Omega; \Gamma_r, T) \right| \right\rvert
\end{split}
\end{equation}
```

Bsp.
8-2-27

$$\left| I_2 \right| = \left| \int_0^T \psi(t) \left\{ u(a, t) - \int_{\gamma(t)}^a \frac{d\theta}{k}(\theta, t) \int_a^\theta c(\xi) u_t(\xi, t) d\xi \right\} dt \right|$$

$$\leq C_6 \left\lvert f \int_\Omega \left| \tilde{S}_{a,-}^{-1,0} W_2(\Omega, \Gamma_l) \right| \left| |u| \overset{\circ}{\rightarrow} W_2^{\tilde{A}}(\Omega; \Gamma_r, T) \right| \right\rvert \quad (1)$$

Einen recht einfachen Weg, das Erscheinungsbild einer solchen Formel zu verbessern, bietet die Umgebung `align` mit dem Befehl `\notag` in der ersten Formelzeile:

```
\begin{align}
\lvert I_2 \rvert &= \left\lvert \int_0^T \psi(t) \left\{ u(a, t) - \int_{\gamma(t)}^a \frac{d\theta}{k}(\theta, t) \int_a^\theta c(\xi) u_t(\xi, t) d\xi \right\} dt \right\rvert \\
&\leq C_6 \left\| f \int_\Omega \left| \tilde{S}_{a,-}^{-1,0} W_2(\Omega, \Gamma_l) \right| \left\| |u| \overset{\circ}{\rightarrow} W_2^{\tilde{A}}(\Omega; \Gamma_r, T) \right\| \right\| \quad (1) \\
&\end{align}
```

Das Ergebnis kann man sich ansehen, aber es missbraucht das logische Markup – indem es davon ausgeht, dass die Formelnummern rechts stehen!

$$\left| I_2 \right| = \left| \int_0^T \psi(t) \left\{ u(a, t) - \int_{\gamma(t)}^a \frac{d\theta}{k}(\theta, t) \int_a^\theta c(\xi) u_t(\xi, t) d\xi \right\} dt \right|$$

$$\leq C_6 \left\| f \int_\Omega \left| \tilde{S}_{a,-}^{-1,0} W_2(\Omega, \Gamma_l) \right| \left\| |u| \overset{\circ}{\rightarrow} W_2^{\tilde{A}}(\Omega; \Gamma_r, T) \right\| \right\| \quad (1)$$

Bsp.
8-2-28

Es gibt außerdem den Befehl `\raisetag`, der die vertikale Position der aktuellen Formelnummer weiter justiert, allerdings *nur*, wenn sie automatisch von ihrer „normalen Position“ verschoben wurde. Um ein solches Tag z.B. um 6pt anzuheben¹, könnte man `\raisetag{6pt}` schreiben. Man kann versuchen die vorige Formel mit `\raisetag` anzupassen. Es ist jedoch gar nicht so einfach, den richtigen Wert zu finden: Hier scheint ein Wert von `1.2\baselineskip` zu passen!

Das nächste Beispiel zeigt eine sinnvollere Verwendung: Mithilfe von `\raisetag` mit einem negativen Argument wird das linksseitige Tag nach unten in die abgesetzte Formel verschoben.

(1)

Vorzeichenfunktion: $S(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$

```
\usepackage[leqno]{amsmath}
\begin{gather} \raisetag{-10pt}
\text{Vorzeichenfunktion: } \{
\mathcal{S}(x) = \begin{cases}
-1 & x < 0 \\
0 & x = 0 \\
1 & x > 0
\end{cases} \\
\end{gather}
```

Bsp.
8-2-29

Hier wurde eine `gather`-Umgebung mit einer einzelnen Zeile verwendet, da der `\raisetag`-Befehl (nur) in der `equation`-Umgebung unglücklicherweise keine Auswirkungen zeigt (er ist in Low-Level-TeX programmiert).

Solche Korrekturen gehören zum „letzten Schliff“, genau wie das Einfügen von Zeilen- und Seitenumbrüchen in normalen Texten. Sie sollten daher

¹Die Beschreibung in der Datei `amsmath.dtx` lässt annehmen, dass ein positiver Wert das Tag immer seiner „normalen Position“ annähert, also nach unten für linksseitige Tags. Das ist in der derzeitigen Version jedoch nicht der Fall.

auch erst durchgeführt werden, wenn das Dokument so gut wie fertig ist. Andernfalls kann es passieren, dass man immer und immer wieder nachbessern muss, sobald sich der Inhalt des Dokumentes ändert.

8.2.13 Untersequenzen in der Nummerierung

Das `amsmath`-Paket verfügt über die Umgebung `subequations`, welche die „Unternummerierung von Formeln“ mit Tags der Form (2a), (2b), (2c) usw. unterstützt. Alle mit Tags versehenen Formeln dieser Umgebung werden mithilfe der folgenden zwei normalen $\text{\LaTeX}$ -Zähler nummeriert: `parentequation` und `equation`.

Das nächste Beispiel zeigt, dass sich das Tag bis zu einem gewissen Grad umdefinieren lässt. Dabei ist zu beachten, dass die geänderte Definition für `\theequation` innerhalb der `subequations`-Umgebung stehen muss! (Veränderungen an Zählern werden in Anhang A.1.4 besprochen.)

		<code>\usepackage{amsmath}</code>
		<code>\begin{subequations} \label{eq:1}</code>
$f = g$	(1a)	<code>\begin{align} f &= g \quad \label{eq:1A} \\ f' &= g' \quad \label{eq:1B} \\ \mathcal{L}f &= \mathcal{L}g \quad \label{eq:1C} \end{align}</code>
$f' = g'$	(1b)	
$\mathcal{L}f = \mathcal{L}g$	(1c)	<code>\end{subequations}</code>
		<code>\begin{subequations} \label{eq:2}</code>
		<code>\renewcommand\theequation{\theparentequation\roman{equation}}</code>
$f = g$	(2i)	<code>\begin{align} f &= g \quad \label{eq:2A} \\ f' &= g' \quad \label{eq:2B} \\ \mathcal{L}f &= \mathcal{L}g + K \quad \label{eq:2C} \end{align}</code>
$f' = g'$	(2ii)	
$\mathcal{L}f = \mathcal{L}g + K$	(2iii)	<code>\end{subequations}</code>
Man beachte die Beziehung zwischen (1) und (2): nur 1c und 2iii sind verschieden.		Man beachte die Beziehung zwischen <code>\eqref{eq:1}</code> und <code>\eqref{eq:2}</code> : nur <code>\ref{eq:1C}</code> und <code>\ref{eq:2C}</code> sind verschieden.

Bsp.
8-2-30

Die Umgebung `subequations` muss die abgesetzten Formeln, auf die sie sich auswirkt, *umschließen*. Sie selbst sollte außerdem nicht in sich selbst verschachtelt sein. Jede `subequations`-Umgebung erhöht den „Hauptzähler“ um eins. Ein `\label`-Befehl innerhalb der Umgebung, aber außerhalb der einzelnen (logischen) Formeln, erzeugt einen `\ref`-Verweis auf die Hauptnummer (z.B. auf 2 und nicht auf 2i).

8.2.14 Zurücksetzen des Formelzählers

Es ist üblich, Formeln innerhalb von Abschnitten oder Kapiteln mit Tags wie (1.1), (1.2), (1.3) ..., (2.1), (2.2) ... zu nummerieren. Diese Art der Nummerierung lässt sich in `amsmath` leicht mithilfe der `\numberwithin`-Deklaration erzielen.¹ So ergibt beispielsweise `\numberwithin{equation}{section}` aus

¹Wie der Name andeutet, kann `\numberwithin` auf jedes Zählerpaar angewendet werden, wobei das Ergebnis aufgrund möglicher Komplikationen nicht immer zufrieden stellend ist. Man vergleiche die Informationen zum Befehl `\addtoreset` in Anhang A.1.4.

mehreren Elementen zusammengesetzte Formelzeilen, welche die Abschnittsnummer enthalten, wobei gleichzeitig der Formelzähler am Anfang jedes neuen Abschnitts zurückgesetzt wird.

8.3 Matrixähnliche Umgebungen

Das `amsmath`-Paket bietet eine Reihe von matrixähnlichen Umgebungen, die alle der Umgebung `array` in Syntax und Layout gleichen. Komplexe mathematische Layouts auf diese Weise zu betrachten ist eine gute Übung, da sich recht viele zweidimensionale mathematische Strukturen und tabellenähnliche Anordnungen so beschreiben lassen.

Deaktivierte alte
Befehle

Drei dieser Umgebungen ersetzen alte Befehle, die tief in Standard- $\text{\LaTeX}$ verborgen liegen: die Umgebungen `cases` (im nächsten Abschnitt erläutert) sowie `matrix` und `pmatrix` (im darauf folgenden Abschnitt erläutert). Da die alten Befehlsformen eine völlig andere Schreibweise benutzen, gehören sie nicht wirklich zu $\text{\LaTeX}$ und können auch nicht gemeinsam mit den hier beschriebenen Umgebungen verwendet werden. Das `amsmath`-Paket erzeugt sogar eine ausführliche Fehlermeldung, wenn einer der alten Befehle benutzt wird (siehe Seite 946). Wenn man andererseits den Fehler begeht, die Umgebungen des `amsmath`-Paketes zu verwenden, ohne dieses zuvor geladen zu haben, erhält man wahrscheinlich die folgende Fehlermeldung: „Misplaced alignment tab character &“ (falsch platziertes Ausrichtungszeichen &).

8.3.1 Die `cases`-Umgebung

Konstruktionen wie die folgende, wo eine einzige Gleichung einige Varianten hat, kommen in der Mathematik häufig vor. Für solche Konstruktionen stellt `amsmath` die `cases`-Umgebung bereit. Sie erzeugt ein beschriftetes Array mit zwei linksbündigen Spalten.

$$P_{r-j} = \begin{cases} 0 & \text{für } r-j \text{ ungerade,} \\ r! (-1)^{(r-j)/2} & \text{für } r-j \text{ gerade.} \end{cases} \quad (1)$$

```
\usepackage{amsmath}
\begin{equation} P_{\{r - j\}} =
\begin{cases}
0 & \text{\texttt{\textsf{für}} \$r - j\$ ungerade,} \\
r! \backslash, (-1)^{\{(r - j)/2\}} & \text{\texttt{\textsf{für}} \$r - j\$ gerade.}
\end{cases}
\end{equation}
```

Bsp.
8-3-1

Man beachte den `\text`-Befehl und den in den Text „eingebetteten Formelmodus“. Mithilfe der Umgebung `aligned` lassen sich weitere `cases`-ähnliche Umgebungen definieren, wie etwa in Beispiel 8-2-20 auf Seite 490.

8.3.2 Die Matrixumgebungen

Die Matrixumgebungen ähneln den `array`-Umgebungen von $\text{\LaTeX}$. Im Unterschied zu diesen besitzen sie jedoch kein Argument, in dem das Format der Spalten bestimmt wird. Stattdessen wird ein Standardformat mit bis zu 10 zentrierten Spalten vorgegeben. Auch die Abstände unterscheiden sich leicht

von den Voreinstellungen der `array`-Umgebung. Das folgende Beispiel veranschaulicht die Matrixumgebungen `matrix`, `pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix` und `Vmatrix`.¹

Bsp.
8-3-2

$$\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \left\{ \begin{array}{cc} 1 & 0 \\ 0 & -1 \end{array} \right\} \\ \left| \begin{array}{cc} a & b \\ c & d \end{array} \right| \left\| \begin{array}{cc} i & 0 \\ 0 & -i \end{array} \right\|$$

```
\usepackage{amsmath}
\begin{gather*}
\begin{matrix}
\begin{matrix} 0 & 1 \end{matrix} \\
\begin{matrix} 0 & -i \end{matrix} \\
\begin{matrix} 0 & -1 \end{matrix} \\
\begin{matrix} 1 & 0 \end{matrix} \\
\begin{matrix} a & b \end{matrix} \\
\begin{matrix} i & 0 \end{matrix}
\end{matrix}
\begin{matrix}
\begin{matrix} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \end{matrix} \\
\begin{matrix} \left\{ \begin{array}{cc} 1 & 0 \\ 0 & -1 \end{array} \right\} \end{matrix} \\
\begin{matrix} \left| \begin{array}{cc} a & b \\ c & d \end{array} \right| \end{matrix} \\
\begin{matrix} \left\| \begin{array}{cc} i & 0 \\ 0 & -i \end{array} \right\| \end{matrix}
\end{matrix}
\end{matrix}
```

Die maximale Anzahl der Spalten in einer Matrixumgebung wird durch den Zähler `MaxMatrixCols` festgelegt, der mithilfe der Standardzählerbefehle von $\LaTeX$ verändert werden kann. Wie in der Standard-`array`-Umgebung wird der Spaltenzwischenraum durch den Wert von `\arraycolsep` bestimmt, jedoch ohne an den Seiten des Arrays Freiraum hinzuzufügen. Mit mehr Spalten hat $\LaTeX$ auch etwas mehr Arbeit und benötigt geringfügig größere Ressourcen. Für die heutzutage üblichen $\TeX$ -Installationen sind diese Einschränkungen jedoch kaum von Bedeutung, so dass man den Wert ohne merkliche Einbußen in der Verarbeitungsgeschwindigkeit auf 20 oder sogar noch höher einstellen kann.

Bsp.
8-3-3

$$\left\| \begin{array}{cccccccccc} a & b & c & d & e & f & g & h & i & \cdots \\ & a & b & c & d & e & f & g & h & \cdots \\ & & a & b & c & d & e & f & g & \cdots \\ & & & a & b & c & d & e & f & \cdots \\ & & & & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots \end{array} \right\|$$

```
\usepackage{amsmath}
\setcounter{MaxMatrixCols}{20}
\begin{Vmatrix}
\, a&b&c&d&e&f&g&h&i \, & \cdots \, \\
& a&b&c&d&e&f&g&h \, & \cdots \, \\
& & a&b&c&d&e&f&g \, & \cdots \, \\
& & & a&b&c&d&e&f \, & \cdots \, \\
& & & & \ddots & \ddots & \cdots & \cdots & \cdots & \cdots
\end{Vmatrix}
```

Dieses Beispiel zeigt auch, wie man den Befehl `\hdotsfor` verwendet, um eine Reihe von Punkten in einer Matrix zu erzeugen, die sich über eine vorgegebene Anzahl von Spalten (hier 4) erstreckt. Der Punktabstand lässt sich über das optionale Argument (hier 2) variieren, das als Multiplikator für den voreingestellten Zwischenraum dient; dieser beträgt 3 mathematische Einheiten (math units, siehe Anhang A.1.5). Der schmale Leerraum und die Klammergruppe `\, \{ \}` am Ende jeder Zeile verschönern lediglich das Layout. Gemeinsam erzeugen sie zwei schmale Leerräume einer Breite von ungefähr 6mu bzw. 1/3em. (Abstände in Formeln werden in Abschnitt 8.7.6 auf Seite 521 ausführlicher besprochen.)

Für die Darstellung einer kleinen Matrix innerhalb von Texten ist die `smallmatrix`-Umgebung am besten geeignet. Man beachte, dass sie den Zei-

¹Man beachte die zuvor erfolgten Warnungen über mögliche Probleme im Zusammenhang mit `matrix` und `pmatrix`.

lenabstand nicht vergrößert, obwohl die Zeile vor der kleinen Matrix Wörter mit Unterlängen enthält.

Um den Einfluss einer Matrix auf die sie umgebenden Zeilen in einem Absatz zu zeigen, steht sie hier: $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ und danach folgt genug Text, dass zumindest eine ganze Zeile unterhalb der Matrix verläuft.

`\usepackage{amsmath}`

Um den Einfluss einer Matrix auf die sie umgebenden Zeilen in einem Absatz zu zeigen, steht sie hier:

```
$ \left( \begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix} \right) $
```

und danach folgt genug Text, dass zumindest eine ganze Zeile unterhalb der Matrix verläuft.

Bsp.
8-3-4

8.3.3 Mehrzeilige Tief- und Hochstellungen

Der Befehl `\substack` wird meistens dazu benutzt, um innerhalb einer Tief- oder Hochstellung mehrere Zeilen zu erzeugen, wobei `\\` als Zeilenbegrenzung dient.

Die `subarray`-Umgebung bietet eine etwas allgemeinere Anordnungsmöglichkeit, in der man die Zeilen auch links- oder rechtsbündig anordnen kann, anstatt zentriert. Sowohl bei der Umgebung wie auch beim `\substack`-Befehl ist jedoch zu beachten, dass sie bei Hoch- oder Tiefstellung in geschweiften Klammern stehen müssen.

$$\sum_{\substack{0 \leq i \leq m \\ 0 \leq j < n}} P(i, j) \quad (1)$$

$$\sum_{\substack{i \in \Lambda \\ 0 \leq i \leq m \\ 0 \leq j < n}} P(i, j) \quad (2)$$

```
\usepackage{amsmath}
\begin{gather}
\sum_{\substack{0 \leq i \leq m \\ 0 < j < n}} P(i, j) \\
\sum_{\subarray{1} i \in \Lambda \\ 0 \leq i \leq m \\ 0 < j < n}} P(i, j)
\end{gather}
```

Bsp.
8-3-5

8.3.4 Kommutative Diagramme

Einige Befehle zum Erzeugen einfacher kommutativer Diagramme, die auf Arrays basieren, sind in einem separaten Paket namens `amscd` verfügbar. Es bietet einige nützliche Kurzformen für beschriftete Pfeile und andere Konnektoren. Es ist jedoch nicht sehr vielseitig – die Konnektoren stehen z.B. nur in horizontaler und vertikaler Form zur Verfügung.

Zwar ließe sich die `picture`-Umgebung für komplexere kommutative Diagramme verwenden, aber für die meisten ernsthaften Arbeiten auf diesem Gebiet benötigt man doch eines der umfangreicheren Pakete. Dazu gehören das `XY-pic`-System von Kristoffer Rose (siehe [58, Kap. 5]) und seine Erweiterung [11] von Michael Barr, das `diagram`-System [22, 23] von Francis Borceux und das `kuvio`-Paket [156] von Anders Svensson.

In der CD-Umgebung erhält man durch die Zeichenfolgen `@>>>`, `@<<<`, `@VVV` und `@AAA` nach rechts, links, unten bzw. oben weisende Pfeile.¹

¹Auf Tastaturen ohne die Zeichen `<` und `>` können alternativ die Notationen `@)))` und `@(((` genutzt werden.

Die nachfolgenden Beispiele zeigen zudem die Verwendung des Befehls `\DeclareMathOperator` (siehe Abschnitt 8.6.2).

Bsp.
8-3-6

$$\begin{array}{ccccc} \mathrm{cov}(L) & \longrightarrow & \mathrm{non}(K) & \longrightarrow & \mathrm{cf}(K) \\ \downarrow & & \uparrow & & \uparrow \\ \mathrm{add}(L) & \longrightarrow & \mathrm{add}(K) & \longrightarrow & \mathrm{cov}(K) \end{array}$$

```
\usepackage{amsmath,amscd}
\DeclareMathOperator\add{add}
\DeclareMathOperator\cf{cf}
\DeclareMathOperator\cov{cov}
\DeclareMathOperator\non{non}
\[\begin{CD}
\mathrm{cov}(L) @>>> \mathrm{non}(K) @>>> \mathrm{cf}(K) \\
@VVV @AAA @AAA \\
\mathrm{add}(L) @>>> \mathrm{add}(K) @>>> \mathrm{cov}(K)
\end{CD}\]
```

Die Pfeile werden nach folgendem Verfahren beschriftet: Bei horizontalen Pfeilen wird das Material zwischen dem ersten und zweiten $\rightarrow$ - oder $\leftarrow$ -Symbol als Hochstellung, und jenes zwischen dem zweiten und dritten als Tiefstellung gesetzt. Entsprechend wird das Material zwischen dem ersten und zweiten bzw. zweiten und dritten A oder V vertikaler Pfeile als links- bzw. rechtsseitige Beschriftung gesetzt; dieses Format wird im nächsten Beispiel benutzt, um den Operator `\End P` rechts neben dem Pfeil zu positionieren.

Die Zeichenfolgen `@=` und `@|` erzeugen horizontale bzw. vertikale Doppellinien. Anstelle eines sichtbaren Pfeils lässt sich auch, soweit erforderlich, (mit der Notation `@.`) auch ein „Null-Pfeil“ setzen um ein Array auszufüllen.

Bsp.
8-3-7

$$\begin{array}{ccc} S^{W_\Lambda} \otimes T & \xrightarrow{j} & T \\ \downarrow & & \downarrow \text{End } P \\ (S \otimes T)/I & \equiv & (Z \otimes T)/J \end{array}$$

```
\usepackage{amsmath,amscd}
\DeclareMathOperator{\End}{End}
\[\begin{CD}
S^{\mathrm{W}_{\Lambda}} \otimes T @>j>> T \\
@VVV @VV\mathrm{End } P\downarrow V \\
(S \otimes T)/I @= (Z \otimes T)/J
\end{CD}\]
```

In Standard- $\mathrm{\LaTeX}$ lässt sich ein ähnliches Layout erzeugen, das jedoch nicht annähernd so gut aussieht:

Bsp.
8-3-8

$$\begin{array}{ccc} S^{W_\Lambda} \otimes T & \xrightarrow{j} & T \\ \downarrow & & \downarrow \text{End } P \\ (S \otimes T)/I & = & (Z \otimes T)/J \end{array}$$

```
\[\begin{array}{ccc}
S^{\mathrm{W}_{\Lambda}} \otimes T & \xrightarrow{j} & T \\
\Big\downarrow & & \Big\downarrow \text{End } P \\
(S \otimes T)/I & = & (Z \otimes T)/J
\end{array}\]
```

Das Beispiel zeigt deutlich, wie viel bessere Ergebnisse das `amscd`-Paket erzeugt: Die Notation ist sehr viel einfacher und das Paket setzt z.B. längere horizontale Pfeile und verteilt die Abstände zwischen den Elementen des Diagramms passender. Mithilfe der spezialisierteren Pakete kann man sogar noch ansprechendere Ergebnisse erzielen.

8.3.5 delarray – Begrenzungszeichen für Arrays

In diesem Abschnitt wird eine nützliche allgemeine Erweiterung für das array-Paket (vgl. Abschnitt 5.2 auf Seite 252) beschrieben, mit deren Hilfe man öffnende und schließende erweiterbare Begrenzungssymbole (vgl. Abschnitt 8.5.3) um eine mathematische array-Umgebung setzen kann. Das Paket delarray wurde von David Carlisle geschrieben und sein Nutzen wird anhand des nächsten, etwas seltsam aussehenden Beispiels veranschaulicht. (Man beachte, dass dieses Paket unabhängig vom amsmath-Paket eingesetzt werden kann, wobei es, falls erforderlich, automatisch das array-Paket lädt.)

$$\left(\begin{array}{cc} X & Y \end{array} \right) \left[\begin{array}{cc} A & B \\ C & D \end{array} \right] \left(\begin{array}{c} L \\ M \end{array} \right)$$

```
\usepackage{delarray}
\left[
\begin{array}[t]{t} ( {cc} ) X & Y \end{array}
\begin{array}[t]{t} [ {cc} ] A & B \\\ C & D \end{array}
\begin{array}[b]{b} \lgroup{cc}\rgroup L \\\ M \end{array}
\right]
```

Bsp.
8-3-9

Die Begrenzungszeichen werden links und rechts der „Präambeldekларation“ (hier {cc}) platziert. Es muss sich um Begrenzungszeichen handeln, die in Tabelle 8.3 auf Seite 511 aufgeführt sind.

Das vorige Beispiel zeigt auch die nützlichste Funktion dieses Paketes: Es verwendet die optionalen Argumente [t] und [b], die nicht in den Matrixumgebungen des amsmath-Paketes verfügbar sind. Sie zeigen, dass die delarray-Syntax anders wirkt als das Umschließen der array-Umgebung mit \left und \right, wodurch sowohl die Begrenzungszeichen als auch das Array selbst angehoben werden.

8.4 Komplexere Gebilde und Beschriftungen

Dieser Abschnitt erläutert einige Befehle, die eine Vielzahl mittelgroßer mathematischer Gebilde erzeugen, einschließlich beschrifteter Symbole und bruchähnlicher Objekte.

8.4.1 Beschriftete Pfeile

Die Befehle \xleftarrow und \xrightarrow erzeugen horizontale Beziehungspfeile ähnlich denjenigen für kommutative Diagramme in Abschnitt 8.3.4; sie können ober- und/oder unterhalb beschriftet werden, wobei sich die Länge der Pfeile automatisch nach dem Text richtet. Diese Pfeile sind normalerweise nur in einer Größe verfügbar. Dadurch eignen sie sich wahrscheinlich nicht für beispielsweise Brüche, Tief- oder Hochstellungen.

Die Beschriftungen unter- und oberhalb der Pfeile werden über ein optionales und ein obligatorisches Argument des Befehls festgelegt.

$$0 \underset{\zeta}{\leftarrow} F \times \Delta(n-1) \xrightarrow{\partial_0 \alpha(b)} E^{\partial_0 b}$$

```
\usepackage{amsmath}
\left[
0 \xleftarrow{[\zeta]{}} F \times \Delta (n - 1)
\xrightarrow{\{\partial_{\text{partial\_0}} \alpha(b)\}} E^{\{\partial_{\text{partial\_0}} b\}}
\right]
```

Bsp.
8-4-1

8.4.2 Kettenbrüche

Der Befehl `\cfrac` erzeugt Reihen von Brüchen, die als „Kettenbrüche“ bekannt sind. Per Voreinstellung sind alle Zähler zentriert; mithilfe der optionalen Argumente `[l]` oder `[r]` lassen sie sich nach links oder rechts ausrichten.

Bsp.
8-4-2

$$\sqrt{2} + \frac{1}{\sqrt{3} + \frac{1}{\sqrt{4} + \frac{1}{\sqrt{5} + \frac{1}{\sqrt{6} + \dots}}}}$$

```
\usepackage{amsmath}
\begin{equation*}
\cfrac {1}{\sqrt{2}} +
\cfrac {1}{\sqrt{3}} +
\cfrac {1}{\sqrt{4}} +
\cfrac[r] {1}{\sqrt{5}} +
\cfrac[l] {1}{\sqrt{6}} + \dotsb \}}\}
\end{equation*}
```

8.4.3 Eingerahmte Formeln

Der Befehl `\boxed` setzt einen Rahmen um sein Argument; er funktioniert wie der Befehl `\fbox`, nur dass sein Inhalt im Formelmodus erscheint (vgl. die in Abschnitt 10.1 beschriebenen Befehle).

Bsp.
8-4-3

$$\boxed{W_t - F \subseteq V(P_i) \subseteq W_t} \quad (1)$$

```
\usepackage{amsmath}
\begin{equation}
\boxed { W_t - F \subseteq V(P_i) \subseteq W_t }
\end{equation}
```

8.4.4 Grenzpositionen

Tief- und Hochstellungen an Integralen, Summen oder anderen Operatoren können entweder direkt über und unter den mathematischen Operatoren positioniert werden oder an den normalen Stellen für Tief-/Hochstellungen rechts der Operatoren. Man sagt, sie „zeigen Grenzen“, wenn das hoch- bzw. tiefgestellte Material (d.h. „die Grenzen“) an hervorgehobener Stelle über und unter und nicht neben dem jeweiligen Symbol oder Operatornamen platziert wird. Typischerweise werden im laufenden Text „keine Grenzen gezeigt“ (um zu vermeiden, dass die Zeilen auseinander gezogen werden); in abgesetzten Formeln hängt ihre Position vom jeweiligen Operator ab. Das nächste Beispiel veranschaulicht die in $\text{\LaTeX}$ voreingestellten Positionen.

Bsp.
8-4-4

$$\sum_{i=1}^n \int_0^\infty \lim_{n \rightarrow 0}$$

Text: $\sum_{i=1}^n, \int_0^\infty, \lim_{n \rightarrow 0}.$

```
\[
\sum_{i=1}^n \quad \int_0^\infty \quad \lim_{n \to 0}
\]
```

Text: $\sum_{i=1}^n$, $\int_0^\infty$, $\lim_{n \to 0}$.

Die Platzierung von Tief- und Hochstellungen an Integralen, Summen und anderen Operatoren ist häufig eine Frage der jeweiligen Layoutrichtlinien einer Zeitschrift. Um dieser Tatsache gerecht zu werden, bietet das `amsmath`-Paket eine umfangreiche Liste von Optionen für ihre Positionierung. In der folgenden Zusammenfassung bezeichnet *Voreinstellung* das Ver-

halten des `amsmath`-Paketes in Kombination mit einer $\text{\LaTeX}$ -Standardklasse, ohne eine der genannten Optionen. In Verbindung mit einer der $\mathcal{AMS}\text{-}\text{\LaTeX}$ -Dokumentenklassen sind unter Umständen andere Voreinstellungen aktiv.

`intlimits, nointlimits` Nur für abgesetzte Formeln; platziert Hoch- und Tiefstellungen an integralartigen Symbolen ober- und unterhalb bzw. seitlich (Voreinstellung).

`sumlimits, nosumlimits` Nur für abgesetzte Formeln; platziert Hoch- und Tiefstellungen an summenartigen Symbolen (auch „große Operatoren“ genannt) ober- und unterhalb (Voreinstellung) bzw. seitlich. Diese Option betrifft auch weitere große Operatoren – $\prod$, $\coprod$, $\otimes$, $\oplus$ usw. – jedoch nicht die verschiedenen Integrale.

`namelimits, nonamelimits` Wie `sumlimits` oder `nosumlimits`, jedoch für bestimmte „Operatornamen“ wie z.B. `det`, `inf`, `lim`, und `max`, `min`, deren Tiefstellungen normalerweise direkt unterhalb erscheinen, zumindest in abgesetzten Formeln.

Die Positionierung an einzelnen Symbolen oder Namen lässt sich auch direkt steuern, und zwar mithilfe eines der folgenden $\text{\TeX}$ -Basisbefehle nach dem Symbol oder Operatornamen: `\limits`, `\nolimits` oder `\displaylimits`. Der letzte Befehl bestimmt, dass der Operator nur dann „Grenzen zeigt“, wenn ein Formelstil für abgesetzte Formeln verwendet wird. Er ist die Voreinstellung für alle Symbole der Klasse `Operator`¹ und für `\mathop`-Konstruktionen. Wenn ein Operator außerhalb einer abgesetzten Formel „Grenzen zeigen“ soll, muss dies einzeln mit dem `\limits`-Befehl deklariert werden. Vergleicht man das nächste Beispiel mit Beispiel 8-4-4, so fällt auf, dass einige Befehle keine Wirkung zeigen, da sie lediglich die Voreinstellung bestätigen.

$\sum_{i=1}^n \int_0^\infty \lim_{n \rightarrow 0}$	$\left[\begin{array}{l} \sum \nolimits_{i=1}^n \quad \int \limits_0^\infty \\ \quad \quad \lim \displaylimits_{n \rightarrow 0} \end{array} \right]$	$\left[\begin{array}{l} \sum \nolimits_{i=1}^n \quad \int \limits_0^\infty \\ \quad \quad \lim \displaylimits_{n \rightarrow 0} \end{array} \right]$
Text: $\sum_{i=1}^n, \int_0^\infty, \lim_{n \rightarrow 0}$.	Text: $\sum \nolimits_{i=1}^n$, $\int \limits_0^\infty$, $\lim \displaylimits_{n \rightarrow 0}$.	

Bsp.
8-4-5

8.4.5 Mehrfachintegrale

Die Befehle `\iint`, `\iiint` und `\iiiint` erzeugen Mehrfachintegrale mit ausgeglichenen Zwischenräumen, sowohl im normalen Text als auch in abgesetzten Formeln. Der Befehl `\idotsint` erzeugt zwei Integralzeichen, die durch Punkte getrennt sind. Das folgende Beispiel zeigt außerdem, wie der Befehl `\limits` die Voreinstellung für die Integralkonstruktionen umgeht und die Grenze V unterhalb des Symbols platziert.

¹In Abschnitt 8.9.1 auf Seite 540 werden die verschiedenen Klassen mathematischer Symbole besprochen.

$\int\!\!\int_V \mu(v,w) \, du \, dv$	<code>\usepackage{amsmath}</code>
$\iiint_V \mu(u,v,w) \, du \, dv \, dw$	<code>\begin{gather*}</code>
	<code>\iint \quad \limits_{-V} \mu(v,w)</code>
	<code>\,,du \,\,,dv \qquad \qquad \backslash\!</code>
$\iiint_V \mu(t,u,v,w) \, dt \, du \, dv \, dw$	<code>\iiint \quad \limits_{-V} \mu(u,v,w)</code>
	<code>\,,du \,\,,dv \,\,,dw \qquad \qquad \backslash\!</code>
	<code>\iiiiint \quad \limits_{-V} \mu(t,u,v,w)</code>
	<code>\,,dt \,\,,du \,\,,dv \,\,,dw \qquad \qquad \backslash\!</code>
$\int \cdots \int_V \mu(z_1,\dots,z_k) \, d\mathbf{z}$	<code>\idotsint \limits_{-V} \mu(z_1, \dots, z_k) \,\,,\mathbf{dz}</code>
	<code>\end{gather*}</code>

8.4.6 Modulo-Operationen

Die Befehle `\mod`, `\bmod`, `\pmod` und `\pod` dienen im `amsmath`-Paket dazu, spezielle Abstandskonventionen der „mod“-Notation für Äquivalenzklassen ganzer Zahlen zu realisieren. Zwei dieser Befehle, `\mod` und `\pod`, sind Varianten von `\pmod`, die von einigen Autoren bevorzugt werden. Der Befehl `\mod` setzt keine Klammern, während `\pod` die Klammern erzeugt, dafür aber den Text „mod“ unterdrückt. Durch `amsmath` werden die Abstände von `\pmod` in eingebetteten (nicht abgesetzten) Formeln verkleinert.

```

\usepackage{amsmath}

\begin{align*}
u &\equiv v + 1 \mod n^2 & u &\equiv v + 1 \pmod{n^2} \\
u &\equiv v + 1 \bmod n^2 & u &\equiv v + 1 \bmod{n^2} \\
u &= v + 1 \pmod{n^2} & u &= v + 1 \pmod{n^2} \\
u &= v + 1 \pod{n^2} & u &= v + 1 \pod{n^2}
\end{align*}

Eingebettetes Layout: $ u = v + 1 \pmod{n^2} $

\begin{gather*}
(m \bmod n) = k^2 \, \, , \, \, ; \, \, \quad
x \equiv y \pmod b \, \, , \, \, ; \, \, \quad
x \equiv y \bmod c \, \, , \, \, ; \, \, \quad
x \equiv y \pod d \, \, , \, \, .
\end{gather*}

```

Eingebettetes Layout: $u = v + 1 \pmod{n^2}$

$$\begin{aligned} (m \bmod n) &= k^2; & x &\equiv y \pmod{b}; & x &\equiv y \pmod{b \wedge c}; & x &\equiv y \pmod{b \vee c}; \\ x &\equiv y \pmod{c}; & x &\equiv y \pmod{d}. & x &\equiv y \pmod{c \wedge d}; & x &\equiv y \pmod{c \vee d}; \end{aligned}$$

8.4.7 Brüche und ähnliche Gebilde

Zusätzlich zu dem gängigen Befehl `\frac` stellt das `amsmath`-Paket die Befehle `\dfrac` und `\tfrac` als praktische Abkürzungen für `{\displaystyle\frac ...}` und `{\textstyle\frac ...}` zur Verfügung. (Die unterschiedlichen mathematischen Formelstile werden ausführlicher in Abschnitt 8.7.1 auf Seite 515 besprochen.)

	(1)	
$\frac{1}{k} \log_2 c(f)$	$\frac{1}{k} \log_2 c(f)$	$\frac{1}{k} \log_2 c(f)$
$\sqrt{\frac{1}{k} \log_2 c(f)}$	$\sqrt{\frac{1}{k} \log_2 c(f)}$	$\sqrt{\frac{1}{k} \log_2 c(f)}$

Text: $\sqrt{\frac{1}{k} \log_2 c(f)}$

```

\usepackage{amsmath}
\begin{equation} \frac{1}{k} \log_2 c(f)
\quad \frac{1}{k} \log_2 c(f)
\end{equation}
Text: $ \sqrt{\frac{1}{k} \log_2 c(f) } \quad
\sqrt{\frac{1}{k} \log_2 c(f) } \backslash, $.

```

Für Binomialkoeffizienten wie $\binom{n}{k}$ kann man die entsprechenden Befehle `\binom`, `\dbinom` und `\tbinom` verwenden.

$$\binom{k}{2} 2^{k-1} + \binom{k-1}{2} 2^{k-2}$$
 (1)

Text: $\binom{k}{2} 2^{k-1} + \binom{k-1}{2} 2^{k-2}$.

```

\usepackage{amsmath}
\begin{equation}
\binom{k}{2} 2^{k-1}
+ \tbinom{k-1}{2} 2^{k-2}
\end{equation}
Text: $ \binom{k}{2} 2^{k-1}
+ \dbinom{k-1}{2} 2^{k-2} $.
  
```

Bsp.
8-4-9

`\genfrac{lbgrenz}{rbgrenz}{stärke}{stil}{zähler}{nenner}`

Alle diese `\binom`- und `\frac`-Befehle sind Sonderformen des allgemeineren Befehls für Brüche, `\genfrac`, der über sechs Argumente verfügt: Bei den ersten beiden Argumenten *lbgrenz* und *rbgrenz* handelt es sich um den linken bzw. rechten Begrenzer. Sie müssen entweder beide leer oder beide gefüllt sein. Will man nur ein Begrenzungszeichen setzen, muss man auf der „leeren“ Seite einen Punkt „.“ einfügen. Das dritte Argument *stärke* überschreibt die voreingestellte Linienstärke des Bruchstrichs. Der Befehl `\binom` z.B. verwendet hier eine Stärke von 0pt für eine unsichtbare Linie. Bleibt dieses Argument leer, so hat die Linienstärke den Wert, der in den Schrifteinstellungen für den Mathematiksatz vorgegeben ist. Die Beispiele in diesem Kapitel verwenden die in Tabelle 8.2 auf der nächsten Seite aufgeführten Voreinstellungen für die verschiedenen Stile (vgl. auch Abschnitt 8.7.1).

Mithilfe des vierten Argumentes *stil* kann man den automatisch gewählten Formelstil, der auch die Schriftgrößen bestimmt, außer Kraft setzen. Er kann als Wert eine ganze Zahl von 0-3 annehmen, die für `\displaystyle`, `\textstyle`, `\scriptstyle` bzw. `\scriptscriptstyle` steht. Wenn dieses Argument leer bleibt, wird der Stil nach den normalen Regeln für Brüche ausgewählt (wie in Tabelle 8.5 auf Seite 515 beschrieben). Die letzten beiden Argumente sind einfach der Zähler (*zähler*) und der Nenner (*nenner*). Zur Veranschaulichung hier mögliche Definitionen für `\frac`, `\tfrac` und `\binom`:

```

\newcommand\frac [2]{\genfrac {}{}{}{}{#1}{#2}}
\newcommand\tfrac [2]{\genfrac {}{}{}{1}{#1}{#2}}
\newcommand\binom [2]{\genfrac {}{}{}{0pt}{#1}{#2}}
  
```

Wenn man natürlich eine besonders komplexe Notation (wie z.B. eine mit `\genfrac` realisierte) in seinem Dokument wiederholt verwenden möchte, tut man sich selbst (und dem Lektor) einen Gefallen, wenn man mithilfe des Befehls `\newcommand`, wie im vorigen Beispiel, einen sprechenden Namen als Abkürzung für diese Notation definiert.

Die ursprünglichen $\TeX$ -Befehle für Brüche, Binomialkoeffizienten und ähnliche Gebilde `\over`, `\overwithdelims`, `\atop`, `\atopwithdelims`, `\above` und `\abovewithdelims` (die Standard- $\LaTeX$ von $\TeX$ geerbt hat), erzeugen in Kombination mit dem `amsmath`-Paket Fehlermeldungen.

Formelstil	Voreingestellte Stärke (ungefähr)
text/display	0.40pt
script	0.34pt
scriptscript	0.24pt

Tabelle 8.2: Voreingestellte Linienstärke der verschiedenen Formelstile

8.4.8 Punkte als Akzente

Die Befehle `\dot` und `\ddot` für mathematische Akzente werden ergänzt durch `\ddd\dot` und `\dddd\dot`, die dreifache bzw. vierfache Punkte als Akzente erzeugen.

Bsp.
8-4-10

$\dot{S}$ $\ddot{P}$ $\dddot{Q}$ $\dddd{R}$ `\usepackage{amsmath}`
`$ \dot{S} \quad \ddot{P} \quad \dddot{Q} \quad \dddd{R} $`

Wenn man eigene mathematische Akzente benutzen möchte, sollte man vielleicht das `accents`-Paket von Javier Bezos verwenden. Es hält, neben weiteren Funktionen, Methoden zum Definieren von Mehrfachakzenten (siehe `\accentset` im Beispiel) und von unterhalb stehenden Akzenten (`\underaccent`, `\undertilde`) bereit. Es lässt sich mit dem `amsmath`-Paket kombinieren. Weitere Details sind in [20] zu finden.

Bsp.
8-4-11

$\overset{*}{X}$ $\overset{\hat{}}{h}$ $\underset{\diamond}{M}$ $\underset{\sim}{C}$ $\underset{\sim}{M}$ $\underset{\sim}{ABC}$ `\usepackage{accents}`
`\[ \accentset{ast}{X} \quad \hat{\accentset{star}{h}} \quad \underaccent{diamond}{mathcal{M}} \quad \underaccent{sim}{C} \quad \underaccent{sim}{M} \quad \underaccent{sim}{ABC} ]`

8.4.9 amsxtra – Akzente als hochgestellte Zeichen

Neben einigen weniger wichtigen Funktionalitäten stellt dieses Paket eine Sammlung einfacher Befehle bereit, mit denen Akzente als hochgestellte Zeichen an einer Unterformel platziert werden können:

Bsp.
8-4-12

$(xyz)^{\cdots}$ $(xyz)^{\ddot{\cdot}}$ $(xyz)^{\dot{\cdot}}$ `\usepackage{amsxtra}`
 $(xyz)^{\smile}$ $(xyz)^{\vee}$ `$(xyz)\spdddot$ \quad $(xyz)\spddot$ \quad $(xyz)\spdot$ \\\`
 $(xyz)^{\sim}$ $(xyz)^{\sim}$ `$(xyz)\spbreve$ \quad $(xyz)\spcheck$ \\\`
 $(xyz)^{\sim}$ $(xyz)^{\sim}$ `$(xyz)\sphat$ \quad $(xyz)\sptilde$`

8.4.10 Zusätzliche Beschriftungen

Standard- $\text{\LaTeX}$ verfügt über den Befehl `\stackrel`, mit dem ein hochgestelltes Zeichen über einem Relationssymbol platziert werden kann. Das `amsmath`-Paket stellt zusätzlich die Befehle `\overset` und `\underset` bereit. Mit ihrer Hilfe lässt sich Material nicht nur über oder unter Relationssymbolen, sondern auch über oder unter beliebigen regulären Symbolen (Klasse `Ordinary`) oder binären Operatoren platzieren. Sie werden genauso wie die Grenzen über oder unter einem Summenzeichen gesetzt.

Der Befehl `\sideset` ergänzt die anderen durch seine besondere Funktion: Er fügt Operatorsymbolen wie $\sum$ oder $\prod$ zusätzlich zu den „normalen“ Grenzen (die ober- und unterhalb stehen) noch weitere Beschriftungen hinzu. Sie erscheinen an den Positionen für Tief- und Hochstellungen, sowohl links als auch rechts seitlich der Operatoren.

$$\overset{*}{X} > \underset{*}{X} \iff \sum'_{a,b \in \mathbf{R}^*} \overset{a}{\underset{b}{X}} = X$$

```
\usepackage{amsmath}
\[\overset{*}{X} > \underset{*}{X}
\iff \sideset{}{'}\sum_{a,b \in \mathbf{R}^*}
\overset{a}{\underset{b}{X}} = X \]
```

Bsp.
8-4-13

Dieses recht komplexe Beispiel zeigt, wie man ein Produktzeichen rundum beschriften kann.

$$\prod_{i=1}^n \prod_{j=2}^m \prod_{k>1}^k \mathcal{T}_{i,j}^k$$

```
\usepackage{amsmath}
\[\sideset_{i = 1}^n \sideset_{j = 2}^m \prod_{k > 1}
\mathcal{T}_{i, j}^k \]
```

Bsp.
8-4-14

8.5 Symbole mit variablen Formen

Bei den meisten $\text{\LaTeX}$ -Befehlen für den Formelsatz erwartet man, dass sie ein fest vorgegebenes Symbol produzieren. Tatsächlich gibt es aber eine ganze Reihe von Fällen, in denen die genaue Form gar nicht festgelegt ist (selbst wenn Zeichensatz und Schriftgrad feststehen). Bestimmte $\text{\TeX}$ -Funktionen für den Mathematiksatz können sogar Gebilde erzeugen, die im Prinzip je nach Bedarf beliebig groß werden können.

Eine solche kontextabhängige Anpassungsfähigkeit ist für den Mathematiksatz sehr wichtig und dieser Abschnitt behandelt einige ihrer Aspekte. Soweit nicht anders erwähnt, sind alle in diesem Abschnitt besprochenen Befehle in Standard- $\text{\LaTeX}$ verfügbar.

Ein bekanntes, aber nicht sehr aufregendes Beispiel für diese Anpassungsfähigkeit sind die Zeichen für mathematische Operatoren, wie z.B. `\sum` und `\prod`, die normalerweise aber nur in zwei Größen vorliegen: einer kleineren für Fließtexte und einer größeren für abgesetzte Formeln. Diese Symbole sind in Tabelle 8.25 auf Seite 552 aufgeführt.

8.5.1 Auslassungspunkte ...

Standard- $\text{\LaTeX}$ verfügt über eine Reihe verschiedener Auslassungspunkte für den Mathematiksatz: `\ldots`, `\cdots` usw. Wenn das `amsmath`-Paket geladen ist, sollten die Auslassungspunkte im Formelmodus jedoch fast immer einfach mit `\dots` gesetzt werden.¹

Die vertikale Position der Auslassungspunkte (auf der Grundlinie oder mittig) sowie der umgebende Leerraum werden automatisch gewählt, passend zu dem Symbol, das auf den Befehl `\dots` folgt. Ist das nächste Symbol z.B. ein Pluszeichen, so werden die Punkte dazu zentriert, ist es ein Komma, erscheinen sie auf der Grundlinie. Sie bestehen immer aus drei Punkten, jedoch

¹Die Befehle `\dots` und `\ldots` können auch im Textmodus genutzt werden, wo beide immer einfache Auslassungspunkte für Text erzeugen.

mit unterschiedlichem Zwischenraum. Diese Voreinstellungen des `amsmath`-Paketes können in einer Klassendatei geändert werden, sofern andere Konventionen gelten.

Bsp.
8-5-1

Eine Reihe $H_1, H_2, \dots, H_n$, eine Summe $H_1 + H_2 + \dots + H_n$, ein orthogonales Produkt $H_1 \times H_2 \times \dots \times H_n$.

`\usepackage{amsmath}`

Eine Reihe `$H_1, H_2, \dots, H_n$`, eine Summe `$H_1 + H_2 + \dots + H_n$`, ein orthogonales Produkt `$H_1 \times H_2 \times \dots \times H_n$`.

Wenn die Punkte am Ende einer Formel stehen, gibt das nächste Objekt (also `\end`, `\`) oder `$`) keinen Aufschluss darüber, wie die Punkte positioniert werden sollen. In diesem Fall muss man deshalb die Position mit einem der folgenden Befehle selbst festlegen: `\dotsc` für „Punkte bei Kommas“, `\dotsb` für „Punkte bei binären Operatoren“, `\dotsm` für „Multiplikationspunkte“, `\dotsi` für „Punkte bei Integralen“ oder sogar `\dotso` für „keinen der genannten Fälle“. Diese Befehle sollten allerdings wirklich nur an diesen besonderen Stellen eingesetzt werden. In allen anderen Fällen ist der Befehl `\dots` vorzuziehen.

Im nächsten Beispiel werden die Auslassungspunkte im ersten Fall auf der Grundlinie und an den anderen Stellen vertikal zentriert gesetzt, mit jeweils passenden Abständen um die Punkte.

Bsp.
8-5-2

Eine Reihe $H_1, H_2, \dots$, eine Summe $H_1 + H_2 + \dots$, ein orthogonales Produkt $H_1 \times H_2 \times \dots$, und ein uneigentliches Integral:

`\usepackage{amsmath}`

Eine Reihe `$H_1, H_2, \dotsc\,$`, eine Summe `$H_1 + H_2 + \dotsb\,$`, ein orthogonales Produkt `$H_1 \times H_2 \times \dotsm\,$`, und ein uneigentliches Integral: `\[ \int_{H_1} \int_{H_2} \dotsi \; {-\Gamma} \, d\Theta \]`

$$\int_{H_1} \int_{H_2} \dots -\Gamma d\Theta$$

Die Symbole und Abstände, die in den verschiedenen Situationen durch den `\dots`-Befehl erzeugt werden, lassen sich durch Undefinieren der Befehle `\dotsc`, `\dotsb`, `\dotsm` und `\dotsi` anpassen. Dies geschieht normalerweise in einer Klassendatei. Man könnte z.B. entscheiden, in manchen Fällen nur zwei Punkte zu verwenden.

8.5.2 Horizontale Erweiterungen

Im Prinzip lässt sich jedes Akzentzeichen für den Mathematiksatz so einrichten, dass es die passende Glyphie aus dem Angebot an verschiedenen Breiten erzeugt, sofern diese in den verfügbaren Fonts enthalten sind. Standard- $\text{\LaTeX}$ kennt jedoch nur zwei derartige Befehle: `\widehat` und `\widetilde`.

In diesem Abschnitt werden einige Befehle beschrieben, die Konstruktionen erzeugen, welche diesen erweiterbaren Akzenten ähneln. Sie erzeugen alle zusammengesetzte Zeichen der mathematischen Klasse Ordinary (reguläre Zeichen, vgl. Abschnitt 8.9.1 auf Seite 540) und sind im nächsten Beispiel dargestellt.

$$\widehat{\psi_\delta(t)E_th} = \widehat{\psi_\delta(t)E_th}$$

$$\overline{\psi_\delta(t)E_th} = \underline{\psi_\delta(t)E_th}$$

Ändern sich nicht bei Stilwechsel:

$$\overbrace{\psi_\delta(t)E_th} = \underbrace{\psi_\delta(t)E_th}$$

Benötigen `amsmath` für Stilwechsel:

$$\overrightarrow{\psi_\delta(t)E_th} = \overleftarrow{\psi_\delta(t)E_th}$$

Benötigen `amsmath`:

$$\overrightarrow{\psi_\delta(t)E_th} = \overleftarrow{\psi_\delta(t)E_th}$$

$$\overleftrightarrow{\psi_\delta(t)E_th} = \overleftrightarrow{\psi_\delta(t)E_th}$$

```
\usepackage{amsmath}
\begin{align*}
\widehat{\psi_\delta(t) E_t h}
&= \widetilde{\psi_\delta(t) E_t h} \quad \backslash\backslash
\overline{\psi_\delta(t) E_t h}
&= \underline{\psi_\delta(t) E_t h} \quad \backslash\backslash
\intertext{Ändern sich nicht bei Stilwechsel:}
\overbrace{\psi_\delta(t) E_t h}
&= \underbrace{\psi_\delta(t) E_t h} \quad \backslash\backslash
\intertext{Benötigen \textsf{amsmath} für}
&\hspace{10em}\text{Stilwechsel:}
\overrightarrow{\psi_\delta(t) E_t h}
&= \overleftarrow{\psi_\delta(t) E_t h} \quad \backslash\backslash
\intertext{Benötigen \textsf{amsmath}:}
\underrightarrow{\psi_\delta(t) E_t h}
&= \underleftarrow{\psi_\delta(t) E_t h} \quad \backslash\backslash
\overleftrightharpoonup{\psi_\delta(t) E_t h}
&= \underleftrightharpoonup{\psi_\delta(t) E_t h}
\end{align*}
```

Bsp.
8-5-3

Die Einzelheiten bezüglich Verfügbarkeit und Eigenschaften dieser Befehle sind leider recht komplex. Sie sind jedoch im Beispiel zusammengefasst. „Stilwechsel“ bedeutet hier, dass die benutzten Zeichen sich mit dem verwendeten Formelstil ändern, so dass sie z.B. in Brüchen oder Tief- bzw. Hochstellungen korrekt erscheinen (siehe Abschnitt 8.7.1 auf Seite 515). Diejenigen, die den Stil nicht verändern, eignen sich nur für die oberste Ebene abgesetzter Formeln.

Ein weiteres horizontal erweiterbares Element in $\text{\LaTeX}$ ist der Balken des Wurzelzeichens; diese Funktion wird am Ende des nächsten Unterabschnitts beschrieben.

8.5.3 Vertikale Erweiterungen

Es gibt viel mehr Möglichkeiten für vertikale Erweiterungen. Alle in Tabelle 8.26 auf Seite 553 abgebildeten Zeichen, sowie noch einige andere, sind potentiell erweiterbar. Die vollständige Liste ist in Tabelle 8.3 dargestellt. Diese Zeichen werden nur in bestimmten Zusammenhängen erweiterbar. Dazu müssen sie alle auf einer Konstruktion der folgenden Form basieren:¹

`\left <ext-open> <Unterformel> \right <ext-close>`

Hinter `<ext-open>` und `<ext-close>` kann sich jedes Symbol aus Tabelle 8.3 verbergen (außer `\sqrt`sign), sowie weitere Zeichen, wenn zusätzliche Pakete geladen sind. Sie müssen dazu als erweiterbare Symbole eingerichtet sein. Die Methoden hierzu sind in $\text{\LaTeX}2_\epsilon$ *font selection* [109] beschrieben, das zu jeder $\text{\LaTeX}$ -Distribution gehört. Es muss also ein Symbol verfügbar sein,

¹Wenn $\text{\LaTeX}$ das Programm eTeX verwendet, lassen sich diese erweiterbaren Symbole auch mit `\middle` kombinieren.

$\left(\right)$	$\{ \}$	$\ \ $	<code>\lVert \rVert</code>
$\langle \rangle$	$\{ \}$	$\ $	<code>\lvert \rvert</code>
$\left(\right)$	$[]$	$ $	
$\int \int$	$[]$	$ $	<code>\vert</code>
$\Downarrow$	$[]$	$ $	<code>\arrowvert</code>
$\Uparrow$	$[]$	$ $	<code>\bracevert</code>
$\Updownarrow$	$[]$	$ $	<code>\Arrowvert</code>
$\downarrow$	$/$	$ $	<code>\lvert</code>
$\uparrow$	$\backslash$	$ $	<code>\Vert</code>
$\updownarrow$	$.$	$\sqrt{\quad}$	<code>\sqrt{\quad}</code>

Blau gedruckte Zeichen benötigen entweder das `amsmath`-Paket oder, bei zusätzlichem `(StM)`-Zeichen, das `stmaryrd`-Paket.

Ein Punkt (.) selbst ist kein erweiterbares Zeichen, aber er kann verwendet werden, um einen „unsichtbaren“ Begrenzer zu erzeugen.

Das \sqrtsign-Zeichen lässt sich nicht mit \left, \right oder \middle kombinieren.

Synonyme: `[ \lbrack, [ \rbrack, ] { \lbrace, \{ } \rbrace, \}` `| \vert, | || \Vert, \lvert`

Tabelle 8.3: Vertikal erweiterbare Symbole

das für die Abwesenheit der tatsächlichen Glyphe steht. Als Zeichen für dieses Symbol, das auch hin und wieder als *Null Delimiter* bezeichnet wird, wurde der Punkt (.) ausgewählt. Die Größe der tatsächlichen Glyphen, mit denen die erweiterbaren Symbole gesetzt werden, orientiert sich an der vertikalen Größe (Höhe und Tiefe) der gesetzten *Unterformel*, die zwischen ihnen liegt. Wie das genau vor sich geht und welche Parameter beteiligt sind, wird in Kapitel 17 und Anhang G (Regel 19) des Buches *The T_EXbook* [87] erläutert. Man kann auch bestimmte Größen für diese Symbole festlegen (siehe Abschnitt 8.7.3 auf Seite 517).

Das Wurzelzeichen `\sqrt` ist noch verblüffender: Es wächst sowohl vertikal als auch horizontal um sich an die Größe seines Argumentes anzupassen. In `LATEX` wird es normalerweise über den Befehl `\sqrt` aufgerufen, der in

Abschnitt 8.7.4 auf Seite 517 näher erläutert wird. Im folgenden Beispiel wird `\sqrtsign` direkt verwendet:

$$\sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + x}}}}}}$$

```
\[
\sqrtsign{1 + \sqrtsign{1 +
\sqrtsign{1 + \sqrtsign{1 +
\sqrtsign{1 + \sqrtsign{1 + x}}}}}}
\]
```

Bsp.
8-5-4

8.6 Text in Formeln

8.6.1 Der `\text`-Befehl

Fontwechselbefehle für Formeln, wie z.B. `\mathrm`, sind nicht dazu gedacht, Text in Formeln zu setzen. Selbst für einzelne Wörter verwendet man hierzu meistens am besten den Befehl `\text`, der dem $\text{\LaTeX}$ -Befehl `\mbox` gleicht, ihm jedoch insofern überlegen ist, dass er sicherstellt, dass der Text im richtigen Schriftgrad gesetzt wird. Als Schrift verwendet er den Font, der außerhalb der Formel aktuell im Gebrauch ist.

```
\usepackage{amsmath}
\begin{gather}
\text{Falls } \Delta_{\text{max rauf}} = \Delta_{\text{min runter}} \\
\text{(für alle rauf und runter) dann} \\
\Delta_{\text{Summe rauf}} = \Delta_{\text{Summe runter}} \quad (1)
\end{gather}
```

Bsp.
8-6-1

8.6.2 Operator- und Funktionsnamen

Die Namen allgemein bekannter mathematischer Funktionen (wie z.B. `log` und `sin`) und Operatoren (wie `max` und `lim`) werden traditionell als Wörter (oder Abkürzungen) in einer aufrechten Serifenschrift formatiert, um sie optisch von den kürzeren Variablennamen zu unterscheiden, die in „Formel-Kursivschrift“ gesetzt werden. Für die häufigsten Funktionsnamen gibt es vordefinierte Befehle, welche die korrekte typographische Umsetzung erzeugen; vgl. Tabelle 8.4 auf der nächsten Seite. Die meisten Funktionen sind bereits in Standard- $\text{\LaTeX}$ verfügbar; die in der Tabelle blau dargestellten benötigen das `amsmath`-Paket. Die mit (ℓ) markierten Funktionen können in abgesetzten Formeln „Grenzen zeigen“ (siehe Abschnitt 8.4.4).

```
\usepackage[fleqn]{amsmath} \setlength{\mathindent}{0pt}
\newcommand{\abs[1]{\lvert#1\rvert}

\begin{gather*}
\lim_{x \rightarrow 0} \frac{\sin^2(x)}{x^2} = 1 \\
\varliminf_{n \rightarrow \infty} |a_{n+1}|/|a_n| = 0 \\
\varinjlim (m_i^\lambda \cdot M)^* \leq \varprojlim_{A/p \rightarrow \lambda(A)} A_p \leq 0
\end{gather*}
```

Bsp.
8-6-2

$\arccos$	<code>\arccos</code>	$\arcsin$	<code>\arcsin</code>	$\arctan$	<code>\arctan</code>
$\arg$	<code>\arg</code>	$\cos$	<code>\cos</code>	$\cosh$	<code>\cosh</code>
$\cot$	<code>\cot</code>	$\coth$	<code>\coth</code>	$\csc$	<code>\csc</code>
$\deg$	<code>\deg</code>	$\det$	<code>\det^(ℓ)</code>	$\dim$	<code>\dim</code>
$\exp$	<code>\exp</code>	$\gcd$	<code>\gcd^(ℓ)</code>	$\hom$	<code>\hom</code>
$\inf$	<code>\inf^(ℓ)</code>	$\operatorname{inj\,lim}$	<code>\injlim^(ℓ)</code>	$\ker$	<code>\ker</code>
$\lg$	<code>\lg</code>	$\lim$	<code>\lim^(ℓ)</code>	$\liminf$	<code>\liminf^(ℓ)</code>
$\limsup$	<code>\limsup^(ℓ)</code>	$\ln$	<code>\ln</code>	$\log$	<code>\log</code>
$\max$	<code>\max^(ℓ)</code>	$\min$	<code>\min^(ℓ)</code>	$\Pr$	<code>\Pr^(ℓ)</code>
$\operatorname{proj\,lim}$	<code>\projlim^(ℓ)</code>	$\sec$	<code>\sec</code>	$\sin$	<code>\sin</code>
$\sinh$	<code>\sinh</code>	$\sup$	<code>\sup^(ℓ)</code>	$\tan$	<code>\tan</code>
$\tanh$	<code>\tanh</code>	$\lim_{\rightarrow}$	<code>\varinjlim^(ℓ)</code>	$\lim$	<code>\varliminf^(ℓ)</code>
$\overline{\lim}$	<code>\varlimsup^(ℓ)</code>	$\lim_{\leftarrow}$	<code>\varprojlim^(ℓ)</code>		

Blaue Funktionen benötigen das amsmath-Paket.

(ℓ) besagt, dass der Operator in abgesetzten Formeln „Grenzen (limits) zeigt“.

Tabelle 8.4: Vordefinierte Operatoren und Funktionen

In der Mathematik werden häufig neue Funktionen dieser Art benötigt. Daher verfügt das amsmath-Paket über einen allgemeinen Mechanismus zum Definieren neuer „Operatorknamen“.

`\DeclareMathOperator*{befehl}{text}` `\operatorname*{text}`

Mithilfe der Deklaration `\DeclareMathOperator` wird ein *befehl* definiert, der einen *text* in der für „Textoperatoren“ passenden Schrift erzeugt. Wenn die neue benannte Funktion ein Operator ist, der in abgesetzten Formeln „Grenzen zeigen“ soll (d.h. Tief- und Hochstellungen für diesen Operator werden an den Positionen für „Grenzen“ über und unter dem Zeichen gesetzt, wie z.B. bei $\lim$, $\sup$ oder $\min$), muss man die Sternform der Deklaration, `\DeclareMathOperator*`, verwenden. Soweit erforderlich, richtet der Befehl `\DeclareMathOperator` zusätzlich zu dem passenden Font auch geeignete Abstände links und rechts des Funktionsnamens ein. So erzeugt er z.B. $A \operatorname{meas} B$ anstelle von $A \operatorname{meas} B$. Das Argument *text* wird in einem „Pseudo-Textmodus“ mit folgenden Eigenschaften gesetzt:

- Das Bindestrichzeichen – erscheint als Textbindestrich (und nicht als Minuszeichen); siehe `\supminus` im nächsten Beispiel.
- Das Sternzeichen * erscheint als erhöhter Text-Stern (nicht vertikal zentriert).
- Ansonsten wird der Text im Formelmodus verarbeitet, so dass Leerzeichen ignoriert werden und man Tief- und Hochstellungen sowie weitere Elemente verwenden kann.

Der verwandte Befehl `\operatorname` (und seine `*`-Form) verwandelt einfach nur sein Argument in einen Funktionsnamen, wie in Beispiel 8-2-12 auf Seite 487. Er ist nützlich für selten verwendete Operatoren.

Das nächste Beispiel zeigt, wie man den Befehl `\meas` für den neuen Funktionsnamen `meas` (Abkürzung für „measure“) sowie die Operatorfunktionen `\esssup` und `\supminus` einrichtet, die beide „Grenzen zeigen“.

$\ f\ _\infty = \operatorname{ess\,sup}_{x \in R^n} f(x) $ $\operatorname{meas}_1 \{u \in R_+^1 : f^*(u) > \alpha\} =$ $\operatorname{ess\,sup}_{x \in R^i} \operatorname{meas}_i \{u \in R^n : f(u) \geq \alpha\}$ $(\forall \alpha \in \operatorname{sup-minus}_{f^*} R_{*+})$	<pre> \usepackage[fleqn]{amsmath} \DeclareMathOperator \meas {meas} \DeclareMathOperator*\esssup {ess \, , sup} \DeclareMathOperator*\supminus{sup - minus*} \newcommand\abs [1]{\lvert#1\rvert} \newcommand\norm[1]{\lVert#1\rVert} \begin{gather*} \norm{f}_\infty = \esssup_{x \in R^n} \abs{f(x)} \\\ \meas_1 \{u \in R_+^1 \colon f^*(u) > \alpha\} = \\\ \quad \esssup_{x \in R^i} \meas_i \{u \in R^n \colon \abs{f(u)} \geq \alpha\} \\\ \quad (\forall \alpha \in \supminus_{f^*} R_{*+}) \end{gather*} </pre>

Bsp.
8-6-3

Leider müssen solche Deklarationen in der Präambel erfolgen, so dass sie nicht zwischendurch geändert werden können. Der Befehl `\DeclareMathOperator` funktioniert sogar nur für Befehlsnamen, die zuvor noch nicht verwendet wurden, so dass man einen existierenden Befehl auch nicht direkt überschreiben kann. Stattdessen muss man zunächst die vorige Definition (in diesem Fall von `\csc`) entfernen, bevor man umdeklariert. Dieses Entfernen geschieht mithilfe von $\text{\TeX}$ Low-Level-Befehlen, da $\text{\LaTeX}$ über keine Methode verfügt, mit denen sich diese Aufgabe erledigen ließe.

$\overline{\lim}_{n \rightarrow \infty} \mathcal{Q}(u_n, u_n - u^\#) \geq \operatorname{cosec}(\mathcal{Q}'(u^\#))$	<pre> \usepackage{amsmath} % Hier wird Low-Level TeX benötigt um die % alte Definition von \csc zu entfernen: \let \csc \relax \DeclareMathOperator\csc{cosec} \newcommand\calQ{\mathcal{Q}} \[\varlimsup_{n \rightarrow \infty} \calQ(u_n, u_n - u^{\{\#\}}) \geq \csc(\calQ'(u^{\{\#\}})) \] </pre>

Bsp.
8-6-4

8.7 Feinabstimmung des Formellayouts

Obwohl $\text{\LaTeX}$ die Elemente einer Formel im Großen und Ganzen passend anordnet, sind manchmal Feinkorrekturen bei der Positionierung erforderlich. Dieser Abschnitt beschreibt, wie man einige der zahlreichen Feinjustierungen am Layout vornimmt, die dem Mathematiksatz den letzten Schliff verleihen. Der Großteil dieses Abschnitts gilt für den gesamten Mathematiksatz in $\text{\LaTeX}$, nur wenige Funktionen stehen erst durch das `amsmath`-Paket zur Verfügung. Diese sind deutlich gekennzeichnet.

<i>Startstil</i>	<i>Hochstellung</i>	<i>Tiefstellung</i>	<i>Zähler</i>	<i>Nenner</i>
D	S	S'	T	T'
D'	S'	S'	T'	T'
T	S	S'	S	S'
T'	S'	S'	S'	S'
S, SS	SS	SS'	SS	SS'
S', SS'	SS'	SS'	SS'	SS'

Tabelle 8.5: Formelstile in Unterformeln

8.7.1 Automatische Skalierung und Abstände

Buchstaben und mathematische Symbole, die in Brüchen, Hoch- oder Tiefstellungen erscheinen, werden normalerweise dichter und in einem kleineren Schriftgrad als dem ausgewählten gesetzt. Insgesamt verfügt $\text{\TeX}$ über acht verschiedene Stile um Formeln darzustellen:

D, D'	<code>\displaystyle</code>	Als eigene Zeilen abgesetzt
T, T'	<code>\textstyle</code>	Im Text eingebettet
S, S'	<code>\scriptstyle</code>	Für Hoch- oder Tiefstellungen
SS, SS'	<code>\scriptscriptstyle</code>	Für alle Hoch- oder Tiefstellungen höherer Ordnung

Bei den Versionen mit Strich (D', T' usw.) handelt es sich um die so genannten *gedrängten* (cramped) Stile, die den normalen Stilen ähneln, außer dass die Hochstellungen etwas niedriger angesetzt werden.

$\text{\TeX}$ verwendet nur drei Formelschriftgrade in den vier Stilen: den Textschriftgrad (der auch in `\displaystyle` benutzt wird), und die Schriftgrade Script und Scriptscript. Der Schriftgrad für jeden Teil einer Formel lässt sich anhand des folgenden Schemas bestimmen.

<i>Ein Symbol im Stil</i>	<i>Wird gesetzt im Schriftgrad</i>	<i>Und erzeugt</i>
D, D', T, T'	Textgröße	(Textgröße)
S, S'	Script-Größe	(Script-Größe)
SS, SS'	Scriptscript-Größe	(Scriptscript-Größe)

In $\text{\LaTeX}$ wird die höchste Ebene einer Formel, die im Fließtext eingebettet ist (innerhalb von zwei $\text{\$}$ -Zeichen oder zwischen `\(...\)`) im Textstil (Stil T) gesetzt. Eine abgesetzte Formel (z.B. eine Formel, die zwischen `\[...\]` steht) wird im Stil für abgesetzte Formeln (Stil D) formatiert. Der Stil einer Unterformel kann dann über die Tabelle 8.5 bestimmt werden, wobei die letzten beiden Spalten die Stile für den Zähler und den Nenner eines Bruches beschreiben.

Die verschiedenen Stile sind im folgenden Beispiel abgebildet.

	<code>\normalsize</code>	%% Verwendeter Stil:	
	<code>\[b</code>	%% D	
	<code>~0</code>	%% S	
	<code>+</code>	%% D	
	<code>\frac{(k + p)</code>	%% T	
	<code>_{{j'}}</code>	%% S'	
	<code>% \displaystyle</code>		
	<code>\pm</code>	%% T [D]	
	<code>\frac{(f + q)</code>	%% S [T]	
	<code>^{{(pk)}</code>	%% SS [S]	
	<code>^y</code>	%% SS	
	<code>_{{j'}}}}</code>	%% SS'	
	<code>{{(h + y)}}}</code>	%% S' [T']	
	<code>{{(1 + q)</code>	%% T'	
	<code>^{{(pk)}}}</code>	%% S'	
	<code>\]</code>		

$$b^0 + \frac{(k+p)_{j'} \pm \frac{(f+q)^{(pk)_{j'}}}{(h+y)}}{(l+q)^{(pk)}}$$

Bsp.
8-7-1

Man kann das Layout dieses Beispiels ändern, indem man für jeden Teil der Formel explizit einen Stil festlegt. Wenn man z.B. das Kommentarzeichen vor `\displaystyle` entfernt, ändern sich einige Stile zu den in Klammern angegebenen. Das Ergebnis sieht dann folgendermaßen aus:

$$b^0 + \frac{(k+p)_{j'} \pm \frac{(f+q)^{(pk)_{j'}}}{(h+y)}}{(l+q)^{(pk)}}$$

Bsp.
8-7-2

In Abschnitt 3.1.4 werden weitere Möglichkeiten beschrieben, wie man den Stil eines einzelnen Symbols ändern kann.

8.7.2 Unterformeln

Während geschweifte Klammernpaare im Text ohne weitere Auswirkungen dazu benutzt werden können, den Geltungsbereich und damit die Auswirkungen einer Deklaration zu begrenzen, bewirken sie im Mathematiksatz *immer* noch etwas anderes. Sie begrenzen eine Unterformel, die dann der äußeren Formel als separat gesetzte Einheit hinzugefügt wird. Im Nebeneffekt werden Unterformeln immer in ihrer natürlichen Breite gesetzt und dehnen sich nicht horizontal, noch schrumpfen sie, wenn $\text{\TeX}$ versucht beim Zeilenumbruch eine Formel in einen Absatz einzupassen.

*Klammernpaare
generieren reguläre
Zeichen*

Wie bereits zuvor gezeigt, wird eine Unterformel in einem Paar geschweif-ter Klammern wie ein einzelnes Symbol (der Klasse `Ordinary`) behandelt. Ein leeres Klammernpaar erzeugt daher ein unsichtbares Zeichen, das die Verteilung der Abstände beeinflussen kann. Genauere Einzelheiten hierzu können in den Kapiteln 17 und 18 und in Anhang G des Buches *The $\text{\TeX}$ book* [87] nachgelesen werden.

Auch die Inhalte von Tief- und Hochstellungen und die Argumente vieler (wenn auch nicht aller) Befehle, wie z.B. `\frac` und `\mathrel`, sind Unterformeln und werden ebenso gesondert behandelt. Ein wichtiges Beispiel für Argumente, die nicht unbedingt als Unterformeln gesetzt werden, sind etwa

diejenigen des Befehls `\bm` (siehe Abschnitt 8.8.2). Wenn eine Gruppe nur den Geltungsbereich einer Deklaration begrenzen soll (d.h. wenn eine eigenständig gesetzte Unterformel falsch wäre), dann sollten die Befehle `\begingroup` und `\endgroup` verwendet werden. Dabei ist zu beachten, dass spezielle Formeldekларationen, wie z.B. Stiländerungen, nicht von `\endgroup` begrenzt werden, sondern bis zum Ende der aktuellen Unterformel oder ihrer expliziten Änderung gelten.

8.7.3 Big-g (große) Begrenzungssymbole

$\text{\LaTeX}$ verfügt über vier Befehle, mit deren Hilfe man die Größe erweiterbarer Begrenzungssymbole steuern kann: `\big`, `\Big`, `\bigg` und `\Bigg`. Sie besitzen ein einziges Argument, das ein erweiterbares Begrenzungssymbol sein *muss*, und erzeugen immer größere Versionen des Symbols von 1,2- bis 3-facher Größe des Basisschriftgrades.

Von jedem der vier Befehle gibt es außerdem drei zusätzliche Varianten, so dass man je vier Größen für folgende Zeichenklassen erhält: öffnende Zeichen (z.B. `\bigl`), Relationszeichen (z.B. `\Bigm`) und schließende Zeichen (z.B. `\Biggr`).¹ Diese 16 Befehle können (und müssen) für alle Symbole verwendet werden, die entweder nach `\left`, `\right` oder (in $\text{\eTeX}$) nach `\middle` stehen können (vgl. Tabelle 8.3 auf Seite 511).

In Standard- $\text{\LaTeX}$ sind die unterschiedlichen Größen dieser Begrenzungssymbole fest eingestellt. Durch das `amsmath`-Paket passen sie sich jedoch, je nach verwendetem Schriftgrad und Formelstil, an die Größe des sie umgebenden Materials an, wie auch im nächsten Beispiel zu sehen ist. Das Gleiche gilt für das `exscale`-Paket (vgl. Abschnitt 7.5.5) oder für jedes Fontpaket, das die `exscale`-Funktionen als Option bereitstellt (d.h. die meisten der Pakete, die in den Abschnitten 7.6 und 7.7 besprochen wurden).

Bsp.
8-7-3

$$\left(\mathbf{E}_y \int_0^{t_\varepsilon} L_{x,y^x(s)} \varphi(x) ds \right)$$

$$\left(\mathbf{E}_y \int_0^{t_\varepsilon} L_{x,y^x(s)} \varphi(x) ds \right)$$

```
\usepackage{amsmath}
\[\biggl(\mathbf{E}_{\!y}\int_0^{t_{\!\varepsilon}}L_{\!x,\,y^x(s)}\varphi(x)\,ds\biggr)\]
\Large
\[\biggl(\mathbf{E}_{\!y}\int_0^{t_{\!\varepsilon}}L_{\!x,\,y^x(s)}\varphi(x)\,ds\biggr)\]
```

8.7.4 Verschieben von Wurzelexponenten

In Standard- $\text{\LaTeX}$ sind Wurzelexponenten nicht immer zufrieden stellend positioniert. Durch das `amsmath`-Paket kann man mithilfe der Befehle `\leftroot` und `\uproot` innerhalb des optionalen Argumentes des Befehls `\sqrt` die Position dieses Exponenten justieren. Enthält ihr Argument eine positive ganze Zahl, dann wird der Exponent nach links bzw. nach oben bewegt, während eine negative ganze Zahl ihn nach rechts bzw. nach unten verschiebt. Die verwendete Maßeinheit sind mathematische Einheiten (siehe Abschnitt 8.7.6), die

¹In Abschnitt 8.9.1 auf Seite 540 sind die verschiedenen mathematischen Symbolklassen beschrieben.

durch ihre geringe Größe eine sehr genaue Positionierung ermöglichen.

$\sqrt[\beta]{k}$	$\sqrt[\beta]{k}$	$\sqrt[\beta]{k}$	<pre> \usepackage{amsmath} \[\sqrt[\beta]{k} \quad \sqrt[\leftroot{2}\uproot{4}]{\beta}{k} \quad \sqrt[\leftroot{1}\uproot{3}]{\beta}{k} \quad \]</pre>

Bsp.
8-7-4

8.7.5 Ghostbusters™: Gestauchte Phantome

Damit Abstände und Ausrichtung „gerade richtig“ werden, ist es häufig am besten, in die Trickkiste der einzigartigen und hoch entwickelten setzerischen Funktionen der T_EX-Basisbefehle zu greifen. Der Zugriff auf diese Fähigkeiten erfolgt über Befehle, die mit `\phantom` und `\smash` verwandt sind. Sie lassen sich sowohl in Formeln als auch in Texten verwenden.

Das umfangreiche Ausrichtungsbeispiel (Beispiel 8-2-10 auf Seite 486) bedient sich z.B. zahlreicher „Phantome“, damit alles genau richtig ausgerichtet ist. Jedes dieser Phantome erzeugt einen unsichtbaren „weißen Kasten“, dessen Größe (Breite und Höhe plus Tiefe) dadurch bestimmt wird, dass der Text in seinem Argument gesetzt und dann vermessen wird.

Der Befehl `\smash` hingegen setzt seinen Inhalt zunächst in einer LR-Box, ignoriert dann aber sowohl die Höhe der Box als auch ihre Tiefe und verhält sich somit, als wären beide Maße gleich null. Der Standard-L^AT_EX-Befehl `\hphantom` ist eine Kombination beider Befehle, der `\smash{\phantom{ein wirklich ruiniertes Phantom!}}` erzeugt: eine unsichtbare Box der Höhe und Tiefe null, aber mit der Breite des Inhalts des Phantom-Argumentes.

Der Befehl `\vphantom` staucht das Phantom auf eine Breite von null zusammen, behält aber seine Höhe und Tiefe bei. Ein Beispiel hierfür ist der Befehl `\mathstrut`, der als „`\vphantom{ }`“ definiert ist. Damit erzeugt er eine Box ohne horizontale Ausdehnung mit der Höhe und Tiefe einer Klammer.

Das `amsmath`-Paket fügt `\smash` ein optionales Argument hinzu, das folgendermaßen verwendet wird: `\smash[t]{...}` ignoriert die Höhe der Boxinhalte, behält jedoch die Tiefe bei, während `\smash[b]{...}` die Tiefe ignoriert aber die Höhe beibehält. In den folgenden vier Zeilen wird lediglich $\sqrt{y}$ unterschiedlich behandelt:

$\sqrt{x} + \sqrt{y} + \sqrt{z}$	<pre> \usepackage{amsmath} \$ \sqrt{x} + \sqrt{y} + \sqrt{z} \$ \par \$ \sqrt{x} + \sqrt{\mathstrut y} + \sqrt{z} \$ \par \$ \sqrt{x} + \sqrt{\smash{y}} + \sqrt{z} \$ \par \$ \sqrt{x} + \sqrt{\smash[b]{y}} + \sqrt{z} \$</pre>

Bsp.
8-7-5

Damit sich die drei Wurzelzeichen angenehm gleichen, scheint man nur die Höhe des y -Zeichen durch eine „unsichtbare Strebe“ (englisch: strut) vergrößern zu müssen – aber das macht alles nur noch schlimmer! Die beste Lösung scheint zu sein, den Fuß des y zu „stauchen“ (aber nicht das ganze Zeichen!).

Ein Wort der Warnung: An einigen wenigen Stellen können Mängel bei der Verarbeitung auf den tiefsten Low-Level-Ebenen von T_EX zu Fehlern bei der Feinjustierung führen. Dieses Risiko betrifft vor allem Formellayouts, in denen folgende Bedingungen zusammentreffen: Eine Unterformel (wie etwa

ein Zähler oder Nenner eines Bruches oder Tief- und Hochstellungen) besteht aus genau einer LR-Box und diese Box besitzt nicht ihre natürliche Breite, wie es bei den komplexeren Formen des Befehls `\makebox`, bei Stauchungen und einigen Phantomen der Fall ist. Dazu folgendes Beispiel:

Bsp. 8-7-6	$\sqrt{\frac{a+b}{x_j}} \quad \sqrt{\frac{a+b}{x_j}} \quad \sqrt{\frac{a+b}{x_j}} \quad \sqrt{\frac{a+b}{x_j+b}}$	<pre> \[\sqrt{\frac{a+b}{x_j}} \quad \sqrt{\frac{a+b}{\smash{x_j}}} \quad \sqrt{\frac{a+b}{\{\}\smash{x_j}}} \quad \sqrt{\frac{a+b}{\smash{x_j+b}}} \quad \] </pre>

Um die Tiefe der Wurzel zu reduzieren, wurde bei der zweiten Wurzel ein `\smash`-Befehl hinzugefügt, jedoch ohne eine Veränderung zu erzielen. Zusammen mit einem leeren Klammernpaar (dritte Wurzel), funktionierte es plötzlich. Andererseits benötigte die vierte Wurzel keine solche Hilfskonstruktion.¹ Aus dem gleichen Grunde benötigt das nächste Beispiel einen `\strut`-Befehl, um den `\smash`-Befehl überhaupt wirksam werden zu lassen. Zusammenfassend lässt sich sagen: Wenn ein `\smash`-Befehl nicht funktioniert, sollte man versuchen, eine leere Unterformel (aus `\}`) vor die einzelne Box zu setzen, damit diese richtig behandelt wird.

Im nächsten Beispiel schiebt der große Bruch in der zweiten Zeile die beiden Teile der Formel visuell zu weit auseinander. Eine Korrektur mit `\smash` erweist sich aber überraschenderweise als ausgesprochen schwierig, weshalb hier die endgültige Korrektur in einigen Einzelschritten erläutert wird. Glücklicherweise sind derartig komplexe Korrekturen nur selten notwendig.

Bsp. 8-7-7	$f_p(x) = \begin{cases} \frac{1}{p} & x = p \\ \frac{(1-x)^{\frac{1}{2}}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases}$	<pre> \usepackage{amsmath} \begin{align*} f_p(x) &= \begin{cases} \frac{1}{p} & x = p \\ \frac{(1-x)^{\frac{1}{2}}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases} \end{align*} </pre>

Versucht man die vertikale Ausdehnung des Exponenten mittels `\smash` zu verstecken, führt dies dazu, dass der Exponent zu tief positioniert wird:

Bsp. 8-7-8	$f_p(x) = \begin{cases} \frac{1}{p} & x = p \\ \frac{(1-x)^{\frac{1}{2}}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases}$	<pre> \usepackage{amsmath} \begin{align*} f_p(x) &= \begin{cases} \frac{1}{p} & x = p \\ \frac{(1-x)^{\smash{\frac{1}{2}}}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases} \end{align*} </pre>

¹Technisch gesehen liegt das daran, dass der Nenner in diesem Fall breiter ist als der Zähler, so dass er von `TeX` nicht noch einmal in eine Box gesetzt wurde, was die Stauchung rückgängig machen würde.

Versucht man dagegen mit `\smash[t]` die vertikale Ausdehnung des gesamten Zählers zu verstecken, zeigt sich überraschenderweise keinerlei Wirkung: Hier schlägt die unglückliche $\text{\TeX}$ -Optimierung zu, die zuvor schon diskutiert wurde.

$$f_p(x) = \begin{cases} \frac{1}{p} & x = p \\ \frac{\frac{(1-x)^{\frac{1}{2}}}{x-\sin(x-p)}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases}$$

```
\usepackage{amsmath}
\begin{align*}
f_p(x) &= \begin{cases}
\frac{1}{p} & x = p \\
\frac{\smash[t]{\frac{(1-x)^{\frac{1}{2}}}{x-\sin(x-p)}}}{\sqrt{1-p} \cos(x-p)} & x \neq p
\end{cases}
\end{align*}
```

Bsp.
8-7-9

Um zu verhindern, dass diese Optimierung zum Tragen kommt, muss man dafür sorgen, dass der Zähler nicht nur aus dem `\smash`-Objekt besteht. Normalerweise kann man in dieser Situation eine leere Klammergruppe verwenden (die ein Ordinary-Atom erzeugt) – wie man allerdings sieht, bringt das in diesem Fall die beiden Zeilen viel zu dicht zusammen:

$$f_p(x) = \begin{cases} \frac{1}{p} & x = p \\ \frac{\frac{(1-x)^{\frac{1}{2}}}{x-\sin(x-p)}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases}$$

```
\usepackage{amsmath}
\begin{align*}
f_p(x) &= \begin{cases}
\frac{1}{p} & x = p \\
\frac{\smash[t]{\frac{(1-x)^{\frac{1}{2}}}{x-\sin(x-p)}}}{\sqrt{1-p} \cos(x-p)} & x \neq p
\end{cases}
\end{align*}
```

Bsp.
8-7-10

Indem man stattdessen ein `\strut` verwendet, gibt man dem Zähler eine gewisse Höhe und verhindert gleichzeitig, dass $\text{\TeX}$ das `\smash` „wegoptimiert“. Versteckt man zusätzlich mit `\smash[b]` die Unterlänge von „ p “, so erhält man zu guter Letzt das gewünschte Ergebnis:

$$f_p(x) = \begin{cases} \frac{1}{p} & x = p \\ \frac{\frac{(1-x)^{\frac{1}{2}}}{x-\sin(x-p)}}{\sqrt{1-p} \cos(x-p)} & x \neq p \end{cases}$$

```
\usepackage{amsmath}
\begin{align*}
f_p(x) &= \begin{cases}
\frac{1}{\smash[b]{p}} & x = p \\
\frac{\strut \smash[t]{\frac{(1-x)^{\frac{1}{2}}}{x-\sin(x-p)}}}{\sqrt{1-p} \cos(x-p)} & x \neq p
\end{cases}
\end{align*}
```

Bsp.
8-7-11

Eine Sammlung weiterer Beispiele veranschaulicht eine andere weit verbreitete Verwendung des Stauchens: Mit einem `\smash`-Befehl, der die Maße

nur teilweise reduziert, erhält man die Möglichkeit, die Höhe der umgebenden Begrenzungszeichen präzise einzustellen. Sie zeigt auch die Probleme, die beim Stauchen auftreten können, da in vielen Fällen die echte Zeilenhöhe bekannt sein muss. Diese wird durch `\vphantom` wiederhergestellt. Im folgenden Beispiel steht `\Hmjd` für ein zusammengesetztes Symbol, das folgendermaßen definiert ist:

```
\newcommand\Hmjd{\widetilde{\mathcal{H}^2}_{MJD}(\chi)}
```

Um die erzielten vertikalen Abstände sichtbar zu machen, wurden einige Linien eingefügt.

Darstellung	Code	Kommentar
$\overline{(\widetilde{\mathcal{H}^2_{MJD}(\chi)})}$	<code>\left(\quad \quad \quad {\Hmjd} \quad \right)</code>	Äußere Klammern zu groß
$\overline{(\widetilde{\mathcal{H}^2_{MJD}(\chi)})}$	<code>\left(\quad \quad \smash{{\Hmjd}} \quad \right)</code>	Äußere Klammern zu klein und Linien zu dicht
$\overline{(\widetilde{\mathcal{H}^2_{MJD}(\chi)})}$	<code>\left(\quad \quad \smash[t]{{\Hmjd}} \quad \right)\vphantom{{\Hmjd}}</code>	Genau richtig!
$\overline{(\widetilde{\mathcal{H}^2_{MJD}(\chi)})}$	<code>\left(\quad \quad \smash[t]{{\Hmjd}} \quad \right)</code>	Sowohl <code>vphantom</code> als auch teilweises Stauchen sind nötig

8.7.6 Horizontale Abstände

Noch genauere, aber auch schwierigere, Feinjustierungen ermöglichen die expliziten Abstandsbefehle in Tabelle 8.6 auf der nächsten Seite. Sowohl die Lang- als auch die Kurzformen dieser Befehle sind robust und lassen sich auch außerhalb von Formeln im normalen Text verwenden. Sie sind mit den schmalen, mittleren und weiten Abständen verwandt, die auf Maschinen verfügbar waren, wie sie Mitte des 20. Jahrhunderts für den Mathematiksatz verwendet wurden.

Die Größe der Abstände, welche diese `\..space`-Befehle hinzufügen, wird durch die aktuellen Werte der drei Parameter `\thinmuskip`, `\medmuskip` und `\thickmuskip` festgelegt. Ihre für `amsmath` voreingestellten Werte sind in der Tabelle aufgeführt. Diese Parameter aus den tiefsten Low-Level-Ebenen von `TEX` benötigen Werte in „mu“ (*math units*, mathematischen Einheiten). Daher müssen sie auch über `TEX`s Low-Level-Anweisungen eingestellt werden (siehe Beispiel 8-9-2 auf Seite 542) und nicht mithilfe von `\setlength` oder ähnlichen Befehlen. Außerdem sollten ihre Werte unter normalen Umständen gar nicht geändert werden, da `TEX` sie intern für den Mathematiksatz verwendet (siehe Tabelle 8.7 auf Seite 541).

 Die Parameterwerte nicht ändern

Eine mathematische Einheit (1mu) entspricht 1/18 em des jeweils aktuellen Schriftgrades für den Mathematiksatz (siehe auch Tabelle A.1 auf Seite 888). Daher variiert der absolute Wert einer mathematischen Einheit mit dem Formelstil, so dass darauf basierende Abstände für jeden Formelstil passend gestaltet sind.

Positive Abstände			Negative Abstände			Wert
Kurz	Abstand	Langform	Kurz	Abstand	Langform	
\,	⇒ ⇐	\thinspace	\!	⇒ ⇐	\negthinspace	3mu
\:	⇒ ⇐	\medspace		⇒ ⇐	\negmedspace	4mu plus 2mu minus 4mu
\;	⇒ ⇐	\thickspace		⇒ ⇐	\negthickspace	5mu plus 5mu
	⇒ ⇐	\enskip				0.5em
	⇒ ⇐	\quad				1em
	⇒ ⇐	\qquad				2em

Anmerkung: Die Spalte „Wert“ wird im Text erklärt.

Tabelle 8.6: Abstandsbefehle in Formeln

Diese mathematischen Einheiten können auch noch allgemeiner eingesetzt werden, um die Abstände in Formeln noch genauer zu justieren. Dies geschieht mithilfe des `amsmath`-Befehls `\mspace`, der dem Befehl `\hspace` entspricht, außer dass er nur in Formeln verwendet werden kann und sein Längenparameter in mathematischen Einheiten angegeben werden muss (z.B. `\mspace{0.5mu}`). Um ein negatives Geviert (`\quad`) in einer Formel zu erhalten, könnte man also `\mspace{-18.0mu}` schreiben. Dadurch erhält man z.B. normalerweise bei einer Tiefstellung höherer Ordnung die Hälfte des Abstandes wie im Basisschriftgrad für Formeln. Der Befehl `\hspace{-1em}` hingegen erzeugt immer den gleichen Abstand, ungeachtet des Formelschriftgrades (wogegen `\text{\hspace{-1em}}` Abstände variabler Größe erzeugt).

8.8 Fonts in Formeln

Bei den meisten Symbolen in einer Formel lässt sich der für die jeweilige Glyphen verwendete Zeichensatz nicht einfach durch eine Fontdeklaration ändern, wie es im normalen Text der Fall ist. In der Tat gibt es im Formelsatz kein Konzept für ein kursives Pluszeichen oder ein Kleinerzeichen in Kapitalchen oder ähnliches.

Eine Ausnahme bilden die lateinischen Buchstaben, deren Darstellung mithilfe von Alphabetbefehlen für Formeln wie z.B. `\mathcal` verändert werden kann. Die Standard- $\text{\LaTeX}$ -Befehle für diesen Zweck wurden in Abschnitt 7.4 besprochen; dieser Abschnitt stellt einige weitere von ihnen vor. Eine weitere Ausnahme findet sich in den fett gedruckten Versionen beliebiger Symbole, die besondere Zeichen mit eigener Bedeutung erzeugen sollen. Dadurch wird die Anzahl der verfügbaren Symbole potentiell verdoppelt, da Fettdruck als erkennbares Attribut für Glyphen nahezu jeder Schriftform dienen kann: je nach Schriftfamilie ist sogar ein „<“ deutlich von einem „<“ zu unterscheiden. Obwohl es den Befehl `\mathbf` gibt, lässt sich das Konzept der Alphabetbefehle nicht auf fett gedruckte Zeichen ausweiten. In Abschnitt 8.8.2 wird hierzu eine bessere Lösung vorgestellt.

Das Erscheinungsbild aller Formeln in einem Dokument lässt sich am besten verändern, indem man alle für die Formeln verwendeten Fonts durch

andere ersetzt. Dies geschieht normalerweise in der Präambel eines Dokumentes, durch Laden eines oder mehrerer geeigneter Pakete, wie sie z.B. in den Abschnitten 7.6 und 7.7 besprochen wurden.

Am Ende dieses Abschnitts werden die Auswirkungen solch umfangreicher Veränderungen, die quasi durch Tastendruck herbeigeführt wurden, anhand einer Formel-Beispielseite demonstriert: In Abschnitt 8.8.3 wird das gleiche Material in sowohl den Computer Modern Math Fonts (der $\LaTeX$ -Voreinstellung) als auch 15 anderen Schriftfamilien für Texte und Formeln gesetzt. Alle verwendeten Schriften sind leicht erhältlich und ungefähr die Hälfte von ihnen ist dabei kostenlos.

8.8.1 Zusätzliche Mathematikfont-Befehle

Durch Laden des Paketes `amsfonts` (oder `amssymb`) erhält man das Euler Fraktur- (`\mathfrak`) und ein Blackboard Bold-Alphabet (`\mathbb`).

Bsp.
8-8-1

$$\forall n \in \mathbb{N} : \mathfrak{M}_n \leq \mathfrak{A}$$

```
\usepackage{amsfonts}
$ \forall n \in \mathbb{N} : \mathfrak{M}_n \leq \mathfrak{A} $
```

Als Beispiel für geringfügige Veränderungen im Mathematiksatz kann jeder, der ein sichtbar anderes Blackboard Bold-Alphabet bevorzugt, dieses aus den Math Pazo Fonts laden. Abschnitt 7.6.3 enthält nähere Informationen zu den Math Pazo Fonts und Abschnitt 7.4.1 erklärt Einzelheiten zu dem Befehl `\DeclareMathAlphabet`. Im nächsten Beispiel wird zunächst das `amsfonts`-Paket geladen und dann die Definition für `\mathbb` überschrieben.

Bsp.
8-8-2

$$\{n, m \in \mathbb{N} \mid \mathfrak{N}_{n,m}\}$$

```
\usepackage{amsfonts}
\DeclareMathAlphabet\mathbb{U}{fplmbb}{m}{n}
$ \lbrace n,m \in \mathbb{N} \mid \mathfrak{N}_{n,m} \rbrace $
```

Dieses Beispiel zeigt, wie man beliebige Fonts seiner $\LaTeX$ -Distribution (sofern sie lateinische Buchstaben in adäquater Position enthalten) als Mathematikalphabet einrichtet, wobei die Argumente des Befehls `\DeclareMathAlphabet` entscheidend sind. Auch wenn es zunächst knifflig scheint, diese korrekt anzugeben, ist es doch nicht so schwierig, wenn man weiß, wo man nachschlagen muss. Geeignete Fonts müssen über eine `.fd`-Datei verfügen – d.h. entsprechend dem Namen der Schriftfamilie nach den Berry-Namenskonventionen (siehe Abschnitt 7.10.2) sollte eine Datei namens `\enc\name\name.fd` existieren. Zum Beispiel hat

der (kommerzielle) Font Lucida Handwriting

den Familiennamen `hlcw`. Er ist in verschiedenen Kodierungen erhältlich, unter anderem in T1, so dass man z.B. nach einer Datei namens `t1hlcw.fd` suchen könnte. In dieser Datei finden sich die verbleibenden Argumente für die Deklaration. Der Font ist nur in der Schriftserie `m` und der Form `it` verfügbar. Alle anderen Schriftformen enthalten Ersetzungen (Abschnitt 7.10.6 enthält nähere Einzelheiten zum Dateiformat für `.fd`-Dateien). Aus all diesen Informationen lässt sich dann der `\mathscr`-Befehl zusammenstellen. Als weitere

Möglichkeit kann man dieses Alphabet auch als Ersatz für den Standardbefehl `\mathcal` verwenden.

$$A_B \neq \mathcal{A}_B \neq \mathcal{A}_B$$

```
\DeclareMathAlphabet\mathscr{T1}{hlcw}{m}{it}
```

```
$A_B \neq \mathscr{A}_B \neq \mathcal{A}_B$
```

Bsp.
8-8-3

Die Existenz der Datei `t1hlcw.fd` (und weiterer Hilfsdateien) auf dem jeweiligen System ist natürlich noch keine Garantie dafür, dass das vorige Beispiel dort auch wirklich funktioniert. Dazu muss man außerdem die entsprechende kommerzielle Schrift installieren. Die meisten modernen $\text{\LaTeX}$ -Installationen enthalten solche Hilfsdateien für verschiedene kommerzielle Schriftsätze, damit man die Fonts sofort verwenden kann, sobald man sie gekauft und installiert hat. Für das vorige Beispiel benötigt man außerdem noch eine Datei namens `hlcw8a.pfb`.

In der Praxis muss man wahrscheinlich gar keine Schriften dazukaufen, da die kostenlos erhältlichen Fonts bereits eine riesige Auswahl bieten. Das Programm `nfssfont.tex` kann bei der Auswahl einer Schrift wertvolle Hilfe leisten, da es Beispiele und Zeichentabellen der installierten Fonts ausgeben kann (siehe Abschnitt 7.5.7).

8.8.2 **bm** – Fette Zeichen

Der Befehl `\mathbf` eignet sich nur für den Fettdruck lateinischer Buchstaben, für alles andere gibt es das **bm**-Paket. Auch wenn das **amsmath**-Paket die Befehle `\boldsymbol` und `\pmb` bereithält, lassen sich die Regeln, wann welcher dieser Befehle zu verwenden ist, und viele der Einschränkungen ihrer Funktion nun einfach umgehen: Man lädt einfach das **bm**-Paket und setzt seine Formeln mithilfe des Befehls `\bm` so fett und schön, wie es die verfügbaren Schriften erlauben.

Das nächste Beispiel zeigt viele Möglichkeiten, wie man die Befehle `\bm` und `\mathbf` einsetzen kann, sowie eine Strategie zum Definieren von Namens Kürzeln für häufig auftretende fette Zeichen. Dazu werden sowohl der Standard- $\text{\LaTeX}$ -Befehl `\newcommand` als auch der Befehl `\bmdefine` des **bm**-Paketes verwendet. Dabei ist zu beachten, dass `\mathbf{xy}` nicht mit `\bm{xy}` identisch ist: Ersteres erzeugt „**xy**“ in fatter Serifenschrift und Letzteres „***xy***“ (d.h. in fatter Formel-Kursivschrift).

```
\usepackage{amsmath,amssymb,bm}
\newcommand\bfB{\mathbf{B}} \newcommand\bfX{\mathbf{x}}
\bmdefine\bpi{\pi} \bmdefine\binfty{\infty}

\section{Die fette Entsprechung}

$$\sum_{j < B} \prod_{\lambda} : \bm{\sum_{x_j}} \prod_{\lambda} \bm{\lambda}$$

\begin{gather}
B_{\infty} + \pi B_1 \sim \bfB_{\binfty} \bm{+} \bpi \bfB_{\bm{1}} \\
\bm{\sim B_{\infty} + \pi B_1} \\
B_{\binfty} + \bpi B_{\bm{1}} \bm{\in} \bm{\biggl\lceil} \bm{\biggr\rceil} \\
(\bfB, \bfX) : \frac{\partial \bfB}{\partial \bfX} \\
\bm{\lnapprox} \bm{1} \bm{\biggr\rceil}
\end{gather}
\end{gather}
```

1 Die fette Entsprechung $\sum_{j < B} \prod_{\lambda} : \sum_{x_j} \prod_{\lambda}$

Bsp.
8-8-4

$$B_{\infty} + \pi B_1 \sim \mathbf{B}_{\infty} + \pi \mathbf{B}_1 \sim B_{\infty} + \pi B_1 \quad (1)$$

$$B_{\infty} + \pi B_1 \in \left\{ (\mathbf{B}, \mathbf{x}) : \frac{\partial \mathbf{B}}{\partial \mathbf{x}} \lesssim 1 \right\} \quad (2)$$

Im vorigen Beispiel tut `bm` sein Bestes, um den Forderungen nach fetten Versionen der einzelnen Symbole und Buchstaben nachzukommen, aber wenn man genauer hinsieht, ist das Ergebnis nicht immer optimal. Die Symbole $\sum$, $\prod$ und $\lesssim$ wurden z.B. alle nach einem Verfahren in fett dargestellt, das als *poor man's bold* („Arme-Leute-Version“ des Fettdrucks) bekannt ist und bei dem die Symbole dreimal, jeweils mit einem leichten Versatz, überdruckt werden. Außerdem sind die geschweiften Klammern gar nicht fett gedruckt. Solche Unzulänglichkeiten lassen sich nicht vermeiden, da für einige Symbole innerhalb der Computer Modern-Mathematikfonts einfach keine fetten Varianten existieren.

Die Situation ändert sich, wenn man das `txfonts`-Paket lädt. Dazu wird die erste Zeile des vorigen Beispiels wie folgt geändert:

```
\usepackage{amsmath,amssymb,txfonts,bm}
```

Die so geladene Schriftfamilie enthält fette Varianten für *alle* Standard- $\LaTeX$ - und `amssymb`-Symbole. Sie erzeugt die folgende Darstellung:

1 Die fette Entsprechung $\sum_{j < B} \prod_{\lambda} : \sum_{x_j} \prod_{\lambda}$

Bsp.
8-8-5

$$B_{\infty} + \pi B_1 \sim \mathbf{B}_{\infty} + \pi \mathbf{B}_1 \sim B_{\infty} + \pi B_1 \quad (1)$$

$$B_{\infty} + \pi B_1 \in \left\{ (\mathbf{B}, \mathbf{x}) : \frac{\partial \mathbf{B}}{\partial \mathbf{x}} \lesssim 1 \right\} \quad (2)$$

Nach welchen genauen Regeln erzeugt der Befehl `\bm` die fetten Varianten der Symbole, die er als Argument erhält? Kurz gesagt: Er macht sich die Tatsache zunutze, dass $\LaTeX$ über ein Formellayout namens **bold** verfügt (auf das man mit `\boldmath` zugreifen kann), mit dem sich eine ganze Formel in fetter Schrift setzen lässt (sofern geeignete fette Schriften verfügbar und eingerichtet sind). Bei jedem Symbol überprüft der `\bm`-Befehl, wie dieses Formellayout das jeweilige Symbol verarbeiten würde. Wenn der für dieses Symbol ausgewählte Font sich von dem des normalen Formellayouts unterscheidet, wird das Symbol in dieser fetten Schrift gesetzt, womit ein perfektes Ergebnis erzielt wird (vorausgesetzt das fette Formellayout wurde richtig eingerichtet). Wenn beide Layouts den gleichen Fonts verwenden, nimmt das Paket an, dass es keine fette Variante gibt und erzeugt daher die „Arme-Leute-Version“ des Fettdrucks (wie zuvor beschrieben).

Bei Begrenzungssymbolen wie etwa `\biggl\lbracket` im Beispiel, ist die Lage noch etwas komplizierter: Ein Begrenzer in $\TeX$ wird typischerweise mit

 Das `bm`-Paket muss nach allen Paketen geladen werden, die Einstellungen für Mathematikfonts ändern!

einer Glyphen gesetzt, die aus einer Reihe verschiedener Größen danach ausgewählt wird, ob sie über die angeforderte Höhe verfügt (siehe Abschnitt 8.5.3 auf Seite 510). Darüber hinaus können diese Glyphen über mehrere Zeichensätze verteilt sein, wobei es für eine bestimmte Größe eine fette Variante geben mag oder auch nicht, so dass der Befehl `\bm` unmöglich zu einem zuverlässigen Schluss gelangen kann, ob das „Poor Man’s Bold“-Verfahren angewandt werden soll. Daher setzt er das Begrenzungszeichen im Grunde in jedem beliebigen Font, den das Layout für fettgedruckte Formeln gerade bietet. Die Computer Modern-Mathematikfonts verfügen nur über fette Varianten der kleinsten Begrenzungszeichen, alle anderen Größen stammen aus Fonts ohne fette Varianten.

$\left\{ \left\{ \left\{ \left\{ \mathcal{Q} \right\} \right\} \right\} \right\}$

```
\usepackage{bm}
$ \bm{\Bigl\lbrace \biggl\lbrace \Bigl\lbrace
    \bigl\lbrace \lbrace
    \mathcal{Q}
    \rangle \bigr\rangle \rangle
    \Bigr\rangle \biggr\rangle \rangle \Bigr\rangle \rangle \rangle $
```

Bsp.
8-8-6

Diese Situation lässt sich durch das `txfonts`-Paket verbessern (wie in Beispiel 8-8-5), oder durch andere Fontsammlungen mit kompletten Sätzen fetter Varianten, wie das hier gezeigte `pxfonts`-Paket:

$\left\{ \left\{ \left\{ \left\{ \left\{ \mathcal{Q} \right\} \right\} \right\} \right\} \right\}$

```
\usepackage{pxfonts,bm}
$ \bm{\Bigl\lbrace \biggl\lbrace \Bigl\lbrace
    \bigl\lbrace \lbrace
    \mathcal{Q}
    \rangle \bigr\rangle \rangle
    \Bigr\rangle \rangle \rangle \biggr\rangle \rangle \rangle \Bigr\rangle $
```

Bsp.
8-8-7

Normalerweise müssen alle Befehle im Argument des Befehls `\bm`, die selbst über Argumente verfügen, vollständig (also Befehl und Argument) innerhalb des `\bm`-Argumentes stehen. Dadurch werden allerdings alle Teile des gesetzten Materials fett gedruckt. Wenn man wirklich einen Befehl so setzen muss, dass nur ein Teil seiner Argumente fett erscheint, hat man mehr Arbeit. Dazu sollte man die betreffenden nicht fett zu druckenden Symbole in einem `\mbox`-Befehl platzieren und das Formellayout innerhalb dieser Box explizit mithilfe des Befehls `\unboldmath` zurücksetzen. $\TeX$ betrachtet einen `\mbox`-Befehl als Symbol der Klasse `Ordinary` (siehe Abschnitt 8.9.1). Damit die Abstände stimmen, muss man ihn daher möglicherweise in einen der Befehle `\mathbin`, `\mathrel` oder `\mathop` einschließen.

$\sqrt[2]{x \times \alpha}$ aber $\sqrt{x \times \alpha}$ oder
ähnlich $\sqrt{x \times \alpha}$

```
\usepackage{amsmath,bm}
$ \bm{\sqrt[2]{x \times \alpha}} $ aber
$ \bm{\sqrt[2]{x \mathbin{\mbox{\unboldmath$ \times $}}
    \alpha}} $ oder ähnlich
$ \bm{\sqrt[2]{\bm{x} \times \bm{\alpha}}} $
```

Bsp.
8-8-8

Zum Glück sind solche Verrenkungen nur selten erforderlich. In den meisten Fällen, an denen Befehle mit Argumenten beteiligt sind, muss nur ein Teil

dieser Argumente fett gedruckt werden. Das lässt sich erreichen, indem man `\bm` innerhalb dieser Argumente verwendet. Für den gängigen Fall fett zu druckender Akzente oder auch `\sqrtsign` (wie im vorigen Beispiel), ist `\bm` speziell programmiert, so dass das Argument des Akzents auch außerhalb des `\bm`-Argumentes stehen darf. Wenn man solche Akzente häufiger einsetzt, ist es jedoch klüger mit `\bmdefine` eine eigene Abkürzung für sie zu definieren, wie im folgenden Beispiel.

Obwohl es so scheint, als sei `\bmdefine\bpi{\pi}` einfach nur eine Kurzform für `\newcommand\bpi{\bm{\pi}}`, ist tatsächlich fast das Gegenteil der Fall: Wird `\bm` verwendet, so definiert es mithilfe des Befehls `\bmdefine` einen neuen verborgenen temporären Befehl und nutzt diesen dann unmittelbar um das fette Symbol zu erzeugen. Mit anderen Worten: `\bmdefine` übernimmt die ganze schwere Arbeit! Wenn man z.B. regelmäßig etwas verwendet, das als `\bm{\alpha}` definiert ist, dann wird bei jedem seiner Vorkommen `\bmdefine` ausgeführt. Wenn man das Ganze aber mit `\bmdefine\balpha{\alpha}` einrichtet, dann führt `\bmdefine` seine zeitaufwendige Verarbeitung nur einmal durch, egal wie häufig `\balpha` vorkommt.

Die Verarbeitung beschleunigen

Bsp.
8-8-9

$$\hat{a} \neq \hat{\hat{a}} \neq \hat{\hat{a}} = \hat{\hat{a}} \neq \hat{\hat{a}}$$

```
\usepackage{bm}
\bmdefine\bhat{\hat}
\[
\hat a \neq \bm{\hat a} \neq \bm{\hat a}
= \bhat a \neq \bm{\widehat a}
\]
```

Dieses Beispiel zeigt auch, dass Akzente variabler Breite (z.B. `\widehat`) eine Unzulänglichkeit mit den Begrenzungszeichen gemeinsam haben: Wenn die Computer Modern-Mathematikfonts verwendet werden, stammen sie aus einem Zeichensatz, für den es keine fette Variante gibt.

Das `bm`-Paket gibt sich viel Mühe, um die richtigen Abstände zwischen Symbolen (sowohl innerhalb als auch außerhalb des `\bm`-Argumentes) herzustellen. Zu diesem Zweck muss `\bm` die Definitionen der Befehle in seinem Argument „untersuchen“, um die richtige mathematische Klasse für die von ihnen erzeugten Symbole zu bestimmen (siehe Abschnitt 8.9.1 auf Seite 540). Diese Untersuchung kann potentiell durch sehr komplexe oder seltsame Konstruktionen durcheinander gebracht werden. Wenn das geschieht, ist es so gut wie sicher, dass $\text{\LaTeX}$ die Verarbeitung mit einer ebenso seltsamen Fehlermeldung abbricht. Idealerweise sollte dieses Problem nicht mit Befehlen von Standard- $\text{\LaTeX}$ oder $\mathcal{A}\mathcal{M}\mathcal{S}\text{\LaTeX}$ auftreten, aber Syntaxanalyse in $\text{\TeX}$ ist sehr schwierig, so dass es immer noch sein kann, dass irgendein seltener Fall übersehen wurde.¹

Handhabung seltsamer Fehlermeldungen

Wenn irgendein Befehl innerhalb von `\bm` einen Fehler erzeugt, kann man diesen *mit all seinen Argumenten* immer in zusätzliche Klammern setzen. Man kann also z.B. `\bm{. . {\cmd . .} . .}` anstelle von `\bm{. . {\cmd . .}}` schreiben. Der `\bm`-Befehl wird nicht versuchen, den Inhalt von Klammern zu

¹Der Autor hatte z.B. beim Schreiben dieses Abschnitts damit zu kämpfen, dass `\bm` versuchte, das Argument des `\hspace`-Befehls zu verarbeiten, anstatt den gewünschten Leerraum zu setzen (dieses Problem ist in Version 1.1a behoben).

parsen. Stattdessen setzt er die gesamte in Klammern stehende Formel mithilfe des `\boldmath`-Befehls. Die so erzeugte, fett gedruckte Unterformel wird dann wie ein „Symbol“ der Klasse `Ordinary` eingefügt. Damit die Abstände um sie herum stimmen, muss man ihre Klasse daher unter Umständen explizit festlegen. Für eine Relation würde man z.B. `\bm{... \mathrel{\cmd{...}} ...}` verwenden (siehe Abschnitt 8.9.1 auf Seite 540).

8.8.3 Verschiedene Einstellungen für Mathematikfonts

In diesem Abschnitt wird der Beispieltex aus Abbildung 8.2 mit verschiedenen Konfigurationen für Mathematik- und Textfonts gesetzt. Abbildung 8.1 zeigt diesen Text in einer Formatierung mit den Computer Modern Mathematik- und Textfonts, entsprechend der Standardeinstellung in $\text{\LaTeX}$.

Die Abbildungen 8.3 bis 8.17 auf den Seiten 530–539 (mit [blauen Legenden](#) zur optischen Trennung von Legende und Beispiel) enthalten den gleichen Beispieltex, wobei jedes Mal andere Unterstützungspakete für Text- und Mathematikfonts geladen werden. So wurde die Ausgabe in Abbildung 8.3 zum Beispiel durch die folgende zusätzliche Zeile in der Präambel erzeugt.

```
\usepackage[boldsans]{ccfonts}
```

Diese Pakete nehmen eigenständig alle erforderlichen Änderungen an den internen Tabellen von $\text{\LaTeX}$ vor. Weitere Konfigurationen und zusätzliche Informationen finden sich in [24].

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) \, dx \, dy \, dz$

und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) \, dw \, dx \, dy \, dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int Q(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta}$$
(1)

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Bsp.
8-8-10

Abbildung 8.1: In Computer Modern gesetzte Beispielseite

```

\documentclass{article}
\pagestyle{empty} \setlength\textwidth{340pt}
\usepackage{amsmath,amssymb}
\newcommand\binom[2]{\genfrac\lbrace\rbrace{0pt}{}{#1}{#2}}
\usepackage{bm}

\begin{document}
\section{Beispielseite für den Mathematiksatz}

Zunächst einige große Operatoren sowohl eingebettet:
$ \iiint\limits_{\{\mathcal{Q}\}} f(x,y,z)\,dx\,dy\,dz $
und $ \prod_{\{\gamma \in \Gamma_{\{\widetilde{C}\}\}} } $
\partial(\widetilde{X}_{\gamma}) $; als auch abgesetzt:
\begin{equation}
\begin{split}
%% Diese Zeile ist absichtlich länger, um die
%% unterschiedlichen Breiten zu zeigen; sie liegt etwas
%% über dem Maß in article/Computer Modern.
\iiint\limits_{\{\mathbf{Q}\}} f(w,x,y,z)\,dw\,dx\,dy\,dz
&\leq
\oint_{\{\bm{\partial Q}\}} f'
\left(
\max \left\{ \frac{w}{r}, \frac{z}{r}, \frac{w^2 + x^2}{r}, \frac{y^2 + z^2}{r}, \frac{w}{r} \oplus \frac{z}{r} \right\}
\right)
&\precapprox
\biguplus_{\{\mathbb{Q}\}} \subset \bar{\mathbf{Q}}
\left[ f^* \right]
\left(
\frac{\left\{ \mathbb{Q}(t) \right\}}{\sqrt{1 - t^2}}
\right)
\right]_{t=\alpha}^{t=\vartheta}
\end{split}
\end{equation}

Für $x$ im offenen Intervall $[-1, 1]$ ist die
unendliche Summe in Gleichung~\eqref{eq:binom1} konvergent; dies
gilt jedoch nicht für das geschlossene Intervall
$[-1, 1]$.
\begin{equation}
(1 - x)^{-k} = \sum_{j=1}^{\infty} (-1)^j \binom{k}{j} x^j
\quad \text{für } k \in \mathbb{N}; k \neq 0.
\label{eq:binom1}
\end{equation}
\end{document}

```

Abbildung 8.2: Quelltext für die Beispielseiten

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_{\mathbb{Q}} f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_{\mathbb{Q}} f(w, x, y, z) dw dx dy dz \leq \oint_{\partial \mathbb{Q}} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{\mathbb{Q} \in \tilde{\mathbb{Q}}} \left[f^* \left(\frac{\int \mathbb{Q}(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta}$$
(1)

Bsp.
8-8-11

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.3: In Concrete Fonts gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_{\mathbb{Q}} f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_{\mathbb{Q}} f(w, x, y, z) dw dx dy dz \leq \oint_{\partial \mathbb{Q}} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{\mathbb{Q} \in \tilde{\mathbb{Q}}} \left[f^* \left(\frac{\int \mathbb{Q}(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta}$$
(1)

Bsp.
8-8-12

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.4: In Concrete und Euler gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint\limits_{\mathcal{Q}} f(x, y, z) dx dy dz$ und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint\limits_{\mathcal{Q}} f(w, x, y, z) dw dx dy dz \leq \oint_{\partial \mathcal{Q}} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}, \frac{\|z\|}{|y^2 + z^2|}, \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right) \quad (1)$$

$$\approx \bigcup_{\mathbb{Q} \in \tilde{\mathbb{Q}}} \left[f^* \left(\frac{\left(\int \mathbb{Q}(t) \right)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\theta}$$

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.5: In Fourier gesetzte Beispielseite

Die Concrete Roman-Textfonts wurden von Donald Knuth gestaltet, Ulrik Vieth entwarf die passenden Mathematikfonts (siehe Abschnitt 7.7.2). Sie werden in Abbildung 8.3 gezeigt, die erzeugt wurde, indem `\usepackage[boldsans]{ccfonts}` in der Präambel des Beispieldokumentes eingefügt wurde. Man beachte, dass die Concrete Fonts keine fetten Varianten enthalten, so dass die Tiefstellung am Integral nach dem „Poor Man’s Bold“-Verfahren gesetzt wurde.

Abbildung 8.4 kombiniert Concrete Roman mit Euler Math (von Hermann Zapf). Diese Kombination wurde mit folgenden Befehlen erzeugt:

```
\usepackage{ccfonts} \usepackage[euler-digits]{eulervm}
```

Sie weist im Fettdruck mathematischer Zeichen keine Mängel auf (vgl. Abschnitt 7.7.10). Man wünscht sich hier wahrscheinlich eine andere Formatierung der Überschriften, da die Voreinstellung (Computer Modern breitfett) nicht sehr gut zu Concrete Roman passt.

In Abbildung 8.5 sieht man eine Kombination aus Utopia und den Fourier Mathematikfonts (von Michel Bovani). Diese Kombination wurde in Abschnitt 7.7.7 besprochen und durch Hinzufügen des Befehls `\usepackage{fourier}` in der Präambel erzielt. Hier weist der Fettdruck in der Tiefstellung erneut Mängel auf, diese sollen jedoch in einer zukünftigen Version der Fonts behoben werden.

Die METAFONT-Versionen der Concrete-Schriften, sowohl Roman als auch Math, sind kostenlos erhältlich. Skalierbare Outline-Versionen können von MicroPress¹ erworben werden. Die Fourier-Lösung ist im Type 1-Format ebenfalls kostenlos erhältlich.

¹<http://www.micropress-inc.com>

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_{\mathcal{Q}} f(x, y, z) dx dy dz$ und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$;
als auch abgesetzt:

$$\begin{aligned} \iiint_{\mathcal{Q}} f(w, x, y, z) dw dx dy dz &\leq \oint_{\partial \mathcal{Q}} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right) \\ &\approx \bigcup_{\mathbb{Q} \in \tilde{\mathbb{Q}}} \left[f^* \left(\frac{\mathbb{Q}(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta} \end{aligned} \quad (1)$$

Bsp.
8-8-14

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.6: In Times und Symbol gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_{\mathcal{Q}} f(x, y, z) dx dy dz$ und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$;
als auch abgesetzt:

$$\begin{aligned} \iiint_{\mathcal{Q}} f(w, x, y, z) dw dx dy dz &\leq \oint_{\partial \mathcal{Q}} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right) \\ &\approx \bigcup_{\mathbb{Q} \in \tilde{\mathbb{Q}}} \left[f^* \left(\frac{\mathbb{Q}(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta} \end{aligned} \quad (1)$$

Bsp.
8-8-15

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.7: In Times und TX Fonts gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$ und $\prod_{\gamma \in \Gamma_C} \partial(\tilde{X}_\gamma)$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{\mathbb{Q} \in \mathbb{Q}} \left[f^* \left(\frac{\int \mathbb{Q}(t)}{\sqrt{1-t^2}} \right) \right]_{t=\alpha}^{t=\beta} \quad (1)$$

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1-x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.8: In Times und TM Math Fonts gesetzte Beispielseite

Diese Doppelseite zeigt drei Mathematikfont-Lösungen für Times Roman als Grundschrift. Da diese heute sehr häufig verwendet wird, existieren dazu zahlreiche Lösungen.

Abbildung 8.6 zeigt eine frei verfügbare Lösung von Alan Jeffrey und anderen (siehe Abschnitt 7.6.2), die mit `\usepackage{mathptmx}` in der Präambel erzeugt wurde. Sie benutzt für die meisten mathematischen Zeichen den Adobe Symbol Font und zeigt die dafür typischen Mängel (einige fette Zeichen für den Mathematiksatz fehlen). Im Gegensatz zu anderen Fontlösungen bietet sie keine eigenen Formen für die erweiterte AMS-Symbolsammlung, sondern nur die Standardformen der Computer Modern.

Abbildung 8.7 zeigt eine weitere freie Umsetzung mittels der von Young Ryu gestalteten TX Fonts. Hier sind alle mathematischen Symbole einschließlich ihrer fetten Varianten verfügbar, die Symbole werden jedoch so dicht aneinander gesetzt, dass sie sich in Formeln manchmal berühren (siehe Abschnitt 7.7.5 für weitere Einzelheiten). Sie wird durch `\usepackage{txfonts}` in der Präambel aktiviert.

Abbildung 8.8 schließlich zeigt die kommerzielle TM Math-Lösung von MicroPress¹ mit deutlich größeren Abständen in Formeln. Sie enthält fette Symbole und eigene Formen für die erweiterte AMS-Symbolsammlung und lässt sich mit `\usepackage{tmmath, tmams}` in der Präambel aktivieren.

Weitere kommerzielle Mathematikfonts im Type 1-Format, zur Kombination mit Times Roman sind MathTime und MathTime Professional (von Michael Spivak), erhältlich über PcTeX^2 .

¹<http://www.micropress-inc.com>

²<http://www.pctex.com>

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{c}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}, \frac{\|z\|}{|y^2 + z^2|}, \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right) \\ \approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int Q(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta} \quad (1)$$

Bsp.
8-8-17

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.9: In Palatino und Math Pazo gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{c}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}, \frac{\|z\|}{|y^2 + z^2|}, \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right) \\ \approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int Q(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta} \quad (1)$$

Bsp.
8-8-18

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.10: In Palatino und PX Fonts gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{z}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}, \frac{\|z\|}{|y^2 + z^2|}, \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \mathbb{Q}} \left[f^* \left(\frac{\lfloor \mathbb{Q}(t) \rfloor}{\sqrt{1-t^2}} \right) \right]_{t=\alpha}^{t=\beta}$$
(1)

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1-x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.11: In Palatino und PA Math gesetzte Beispielseite

Die Schrift Palatino wurde 1948 von Hermann Zapf für den Zeichensatzhersteller Stempel gestaltet und basiert auf typographischen Vorbildern der italienischen Renaissance. Seitdem hat sie sich zu einer der am weitesten verbreiteten Schriften entwickelt und ist heute wahrscheinlich die beliebteste aller existierenden Renaissance-Antiqua-Schriften. Für Palatino als Grundschrift gibt es eine ganze Reihe von Mathematikfont-Lösungen.

Abbildung 8.9 zeigt die kostenlos erhältlichen Math Pazo Fonts (von Diego Puga), die sich mit `\usepackage{mathpazo}` aktivieren lassen. Sie enthalten fettgedruckte Symbole und ein passendes Blackboard Bold-Alphabet, jedoch keine eigens gestalteten Formen für die AMS-Symbole (siehe auch Abschnitt 7.6.3).

Im Gegensatz decken die kostenlosen PX Fonts (von Young Ryu) die komplette Symbolsammlung ab. Sie sind in Abbildung 8.10 zu sehen. Genau wie bei den TX Fonts stehen die Zeichen sehr dicht beieinander (Abschnitt 7.7.6 erläutert nähere Einzelheiten). Diese Lösung kann mit `\usepackage{pxfonts}` aktiviert werden.

Abbildung 8.11 zeigt die kommerzielle Lösung von MicroPress¹. Sie bietet einen ähnlichen Umfang an Symbolen wie die Math Pazo Lösung mit ungefähr der gleichen Laufweite, jedoch mit deutlich anderen Formen. Diese Lösung kann mit `\usepackage{pamath}` aktiviert werden.

¹<http://www.micropress-inc.com>

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$ und

$\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int \mathbb{Q}(t)}{\sqrt{1-t^2}} \right) \right]_{t=\alpha}^{t=\beta} \quad (1)$$

Bsp.
8-8-20

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1-x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.12: In Baskerville gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{C}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int \mathbb{Q}(t)}{\sqrt{1-t^2}} \right) \right]_{t=\alpha}^{t=\beta} \quad (1)$$

Bsp.
8-8-21

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1-x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.13: In Charter gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{y \in \Gamma_{\tilde{\chi}}} \partial(\tilde{X}_y)$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\lfloor Q(t) \rfloor}{\sqrt{1-t^2}} \right) \right]_{t=\alpha}^{t=\beta}$$
(1)

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1-x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.14: In Lucida Bright gesetzte Beispielseite

Abbildung 8.12 nutzt die Schrift Baskerville als Textfont. Diese „Transitional“-Schrift wurde ursprünglich von John Baskerville (1706–1775) gestaltet und kann über viele Schriftenhersteller bezogen werden. Als Mathematikfont wird BA Math von MicroPress¹ verwendet, deren Distribution außerdem eine Variante des hier benutzten Baskerville-Textfonts enthält. Die BA Math Fonts beinhalten fette Symbole, jedoch keine Formen für die erweiterte AMS-Symbolsammlung. Man beachte, dass die einzelnen Symbole zwar nicht sehr groß erscheinen, die abgesetzten Formeln aber trotzdem vertikal mehr Platz benötigen als in den anderen Beispielen. Diese Fonteneinstellungen werden mit `\usepackage{ba}` aktiviert.

Abbildung 8.13 zeigt die kommerziellen CH Math Fonts (ebenfalls von MicroPress¹). Ihre Distribution wurde passend zu den kostenlos erhältlichen Charter Fonts gestaltet (siehe Abschnitt 7.6.1). Die CH Math Fonts enthalten die vollständige Bandbreite mathematischer Symbole einschließlich der AMS-Zusätze. Sie werden durch Hinzufügen der Präambelzeile `\usepackage{chmath, chams}` aktiviert.

Die Schriften Lucida Bright und Lucida New Math sind in Abbildung 8.14 dargestellt. Diese Kombination kommerzieller Text- und Mathematikfonts wurde von Charles Bigelow und Kris Holmes gestaltet und war bei Y&Y² erhältlich. Die Fontlösung verfügt über alle mathematischen Standardsymbole einschließlich der AMS-Erweiterungen und wird durch Laden des `lucida`-Paketes aktiviert. Man sieht, dass die Formeln eine große Laufweite haben, wodurch die Lesbarkeit auf Kosten des Platzes verbessert wird. Die Grundschrift dieses Buches ist Lucida Bright. Für die Beispiele wurde jedoch normalerweise Computer Modern verwendet, damit sie aussehen wie in Standard- $\text{\LaTeX}$.

¹<http://www.micropress-inc.com>

²<http://www.tug.org/yandy>

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{c}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int Q(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta}$$
(1)

Bsp.
8-8-23

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.15: In CM Bright gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_Q f(x, y, z) dx dy dz$

und $\prod_{\gamma \in \Gamma_{\tilde{c}}} \partial(\tilde{X}_{\gamma})$; als auch abgesetzt:

$$\iiint_Q f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right)$$

$$\approx \bigcup_{Q \in \tilde{Q}} \left[f^* \left(\frac{\int Q(t)}{\sqrt{1 - t^2}} \right) \right]_{t=\alpha}^{t=\vartheta}$$
(1)

Bsp.
8-8-24

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.16: In Helvetica Math gesetzte Beispielseite

1 Beispielseite für den Mathematiksatz

Zunächst einige große Operatoren sowohl eingebettet: $\iiint_{\mathbb{Q}} f(x, y, z) dx dy dz$ und

$\prod_{Y \in \tilde{\mathcal{Z}}} \partial(\tilde{X}_Y)$; als auch abgesetzt:

$$\iiint_{\mathbb{Q}} f(w, x, y, z) dw dx dy dz \leq \oint_{\partial Q} f' \left(\max \left\{ \frac{\|w\|}{|w^2 + x^2|}; \frac{\|z\|}{|y^2 + z^2|}; \frac{\|w \oplus z\|}{\|x \oplus y\|} \right\} \right) \quad (1)$$

$$\approx \bigcup_{\mathbb{Q} \in \tilde{\mathcal{Q}}} \left[f^* \left(\frac{\int \mathbb{Q}(t)}{\sqrt{1 - t^2}} \right) \right]_{t=a}^{t=b}$$

Für x im offenen Intervall $] -1, 1[$ ist die unendliche Summe in Gleichung (2) konvergent; dies gilt jedoch nicht für das geschlossene Intervall $[-1, 1]$.

$$(1 - x)^{-k} = 1 + \sum_{j=1}^{\infty} (-1)^j \left\{ \begin{matrix} k \\ j \end{matrix} \right\} x^j \quad \text{für } k \in \mathbb{N}; k \neq 0. \quad (2)$$

Abbildung 8.17: In Informal Math gesetzte Beispielseite

Diese Doppelseite zeigt zwei serifenlose Schriftkombinationen und eine „informelle“ Lösung für den Mathematiksatz. Die serifenlosen Lösungen können in vielen Fällen von Vorteil sein, wie etwa für konventionelle Artikel, Präsentationen (z.B. Folien, Berichte), Online-Dokumentationen oder Zeitschriften. Die informelle Info Math-Lösung sollte hingegen wohl auf Ankündigungen, Flyer und ähnliche Druckwerke beschränkt bleiben.

Abbildung 8.15 zeigt die Computer Modern Bright Fonts (von Walter Schmidt), die auf dem Design der Computer Modern Fonts basieren. Die Lösung bietet die gesamte Bandbreite mathematischer Symbole in normaler und fetter Ausprägung und wird durch Laden des Paketes `cmbright` aktiviert (siehe Abschnitt 7.7.3). Die Fonts sind im METAFONT-Format kostenlos erhältlich; die Type 1-Versionen können von MicroPress¹ erworben werden.

Abbildung 8.16 zeigt eine Kombination von Mathematikfonts für Helvetica (ursprünglich gestaltet von Max Miedinger). Die HV Math Fonts wurden bei MicroPress¹ entworfen und beinhalten einen vollständigen Satz mathematischer Symbole. Diese Lösung wird durch Laden der Pakete `hvmath` und `hvams` (für die AMS-Symbolsammlung) aktiviert. Während die Type 1 Fonts nur kommerziell vertrieben werden, ist eine Bitmap-Version mit einer Auflösung von 300dpi kostenlos bei MicroPress erhältlich.

Und schließlich zeigt Abbildung 8.17 die Informal Math-Lösung, die ebenfalls von MicroPress¹ angeboten wird. Die Gestaltung der Fonts ist andeutungsweise der Tekton-Familie von Adobe nachempfunden. Diese Lösung wird durch Laden des `infomath`-Paketes aktiviert. Dabei ist zu beachten, dass die Textfonts nur in der OT1-Kodierung erhältlich sind und die AMS-Symbolsammlung nicht unterstützen.

¹<http://www.micropress-inc.com>

8.9 Symbole in Formeln

Die Tabellen am Ende dieses Abschnitts machen die riesige Auswahl mathematischer Symbole der $\mathcal{AMS}$ -Fontpakete bekannt, gemeinsam mit den Befehlen für jedes einzelne Symbol. Sie enthalten auch die zusätzlichen Symbole des St Mary Road Fonts, der von Alan Jeffrey und Jeremy Gibbons gestaltet wurde. Dieser Font stellt eine Erweiterung der $\mathcal{AMS}$ -Symbolfont-Sammlungen dar. Das entsprechende Paket `stmaryrd` sollte normalerweise zusätzlich zum `amssymb`-Paket geladen werden, jedoch immer danach. Es stellt zusätzliche Symbole für Bereiche wie funktionale Programmierung, Prozess-Algebra, Domänen-Theorie, lineare Logik und viele mehr zur Verfügung. Ein wahrer Schatz an Informationen und eine noch größere Auswahl an Symbolen finden sich in der *Comprehensive L^AT_EX Symbol List* von Scott Pakin [136].

Die Tabellen zeigen auch, welche Pakete für welche Symbolbefehle zusätzlich geladen werden müssen. Sie sind folgendermaßen organisiert: Symbole mit schwarzen Befehlsnamen sind ohne zusätzliche Pakete in Standard- $\text{\LaTeX}$ verfügbar; Symbole in **blau** erfordern entweder das `amsmath`-, das `amssymb`- oder das `stmaryrd`-Paket, wie in den Tabellenfußnoten erläutert. Soweit erforderlich werden die Symbole durch Kennzeichen weiter klassifiziert: Eine *(STM)*-Markierung kennzeichnet ein Symbol des `stmaryrd`-Paketes, sofern die jeweilige Tabelle auch Symbole aus anderen Paketen enthält; ein *(kernel)* bedeutet, dass dieses Symbol in Standard- $\text{\LaTeX}$ nur als Kombination von zwei oder mehr Glyphen verfügbar ist, während das genannte Paket eine einzelne Glyphe dafür enthält; und *(var)* markiert „Alphabetzeichen/-symbole“ (des Typs `\mathalpha`; siehe Tabelle 7.30 auf Seite 447), die im Geltungsbereich eines Alphabetbefehls ihr Aussehen verändern (siehe Abschnitt 7.4).

8.9.1 Mathematik-Symbolklassen

Die Symbole werden in erster Linie nach ihrer „mathematischen Klasse“ unterteilt, die gelegentlich auch „mathematischer Symboltyp“ genannt wird. Diese Klassifizierung entspricht der Symbolbedeutung im typischen technisch-mathematischen Umfeld, aber für den Mathematiksatz ist sie insofern von Bedeutung, dass sie sich auf das Layout einer Formel auswirkt. So stellt das Formatierungsprogramm für den Mathematiksatz in $\text{\TeX}$ die horizontalen Abstände an den Seiten jedes Symbols danach ein, welcher mathematischen Klasse es angehört. Außerdem gibt es noch einige feinere Unterscheidungen, wie etwa zwischen Akzenten und einfachen Symbolen und beim Verteilen der riesigen Liste von Relationszeichen auf mehrere Tabellen.

Die Einstellungen für den Mathematiksatz weisen jedes Symbol einer der folgenden Klassen zu: Ordinary (Ord, reguläre Zeichen), Operator (Op, Operatoren), Binary (Bin, binäre Operatoren), Relation (Rel, Relationszeichen), Opening (Open, öffnende Zeichen), Closing (Close, schließende Zeichen) oder Punctuation (Punct, Interpunktion).

Symbole lassen sich nach dem in Abschnitt 7.10.7 auf Seite 445 beschriebenen Verfahren einer der oben genannten Klassen hinzufügen. Außerdem bilden bestimmte Unterformeln – vor allem Brüche, sowie mit `\left` und `\right` erzeugte Elemente – eine Klasse namens `Inner` (innere Elemente). Sie kann explizit über den Befehl `\mathinner` zugewiesen werden.

		Rechtes Objekt							
		Ord	Op	Bin	Rel	Open	Close	Punct	Inner
Linkes Objekt	Ord	0	1	(2)	(3)	0	0	0	(1)
	Op	1	1	*	(3)	0	0	0	(1)
	Bin	(2)	(2)	*	*	(2)	*	*	(2)
	Rel	(3)	(3)	*	0	(3)	0	0	(3)
	Open	0	0	*	0	0	0	0	0
	Close	0	1	(2)	(3)	0	0	0	(1)
	Punct	(1)	(1)	*	(1)	(1)	(1)	(1)	(1)
	Inner	(1)	1	(2)	(3)	(1)	0	(1)	(1)

0 = kein Abstand, 1 = `\thinmuskip`, 2 = `\medmuskip`, 3 = `\thickmuskip`, * = unmöglich
 Einträge in (blau) werden in „scriptstyle“-Formelstilen nicht hinzugefügt (vgl. Abschnitte 8.7.1 und 8.7.6).

Tabelle 8.7: Abstände zwischen Symbolen

Die Zuordnung der Symbole zu Klassen kann mit den folgenden Befehlen explizit geändert werden: `\mathord`, `\mathop`, `\mathbin`, `\mathrel`, `\mathopen`, `\mathclose` und `\mathpunct`, wodurch sich unter anderem die umgebenden Abstände verändern. Im nächsten Beispiel werden die Zeichen `\#` und `\top` (die beide per Voreinstellung der Klasse `Ord` zugewiesen sind) in ein `Rel`- und ein `Op`-Zeichen verwandelt.

Bsp.
8-9-1

$a \# \top_x^\alpha x_b^\alpha$	<code>\usepackage[fleqn]{amsmath}</code>
$a \# \top_x^\alpha x_b^\alpha$	<code>\[a \# \top_x^\alpha x_b^\alpha \]</code>
$a \# \top_x^\alpha x_b^\alpha$	<code>\[a \mathrel{\#} \mathop{\top}_x^\alpha x_b^\alpha \]</code>

`TeX` gestaltet die Zwischenräume in Formeln einfach, indem es die Klasse jedes Formelobjektes bestimmt und dann zwischen den benachbarten Objekten die in Tabelle 8.7 festgelegten Abstände einfügt. Diese Tabelle ist leider fest in die Mathematik-Routinen von `TeX` integriert, so dass sie sich nicht mithilfe von Makropaketen ändern lässt.¹ Die Tabellenwerte 0, 1, 2 und 3 stehen für keinen, einen schmalen (`\,`), einen mittleren (`\:`) bzw. einen weiten (`\;`) Abstand. Die genauen Maße der Abstände sind in Abschnitt 8.7.6 auf Seite 521 aufgeführt.

Ein Binärzeichen wird in ein reguläres Zeichen (Ordinary) umgewandelt, wenn davor und dahinter Zeichen stehen, die von ihrer Art her nicht mit einer binären Operation vereinbar sind. Darum sind einige Einträge der Tabelle mit einem Stern markiert, um zu zeigen, dass es sich um unmögliche Kombinationen handelt. So ergibt z.B. `\$+x\$` die Zeichenfolge `+x` (ein „unäres Plus“) und nicht `+ x`, das sich wiederum mit `\${}\+x\$` erzeugen lässt.

In Tabelle 8.7 bedeuten blaue Einträge, dass die entsprechenden Abstände in den Formelstilen `Script` und `Scriptscript` nicht eingefügt werden.

Man betrachte die folgende Formel als Beispiel für die Umsetzung dieser Regeln (die voreingestellten Werte wurden absichtlich geändert, damit die

¹Auch wenn ein paar der Tabelleneinträge in Frage gestellt werden können, führt sie doch insgesamt zu einem ansprechenden Ergebnis.

eingefügten Abstände deutlicher sichtbar sind):

$$a - b = -\max\{x, y\}$$

\thinmuskip=10mu \medmuskip=17mu \thickmuskip=30mu
\[a - b = -\max \{ x , y \} \]

Bsp.
8-9-2

TeX identifiziert die Objekte als Ord, Bin, Ord usw. und fügt dann die folgenden Abstände ein:

$$a - b = -\max\{x, y\}$$

Ord \: Bin \: Ord \; Rel \; Ord \, Op Open Ord Punct \, Ord Close

Das Minuszeichen vor `\max` wird in ein reguläres Zeichen (Ordinary) umgewandelt, da auf ein Relationszeichen kein binäres Zeichen folgen kann.

Tabelle 8.7 zeigt, dass zwischen einer „`\left...\right`“-Konstruktion, bei der die gesamte von dieser Konstruktion begrenzte Unterformel zu einem einzigen Objekt der Klasse Inner wird (siehe Abschnitt 8.5.3 auf Seite 510), und Befehlen wie `\Bigl` und `\Bigr`, die einzelne Symbole der Klassen Opening bzw. Closing erzeugen, ein Unterschied¹ besteht. Auch wenn sie möglicherweise zu Begrenzungszeichen der gleichen vertikalen Größe führen, können sich, je nach den benachbarten Zeichen in der Formel, unterschiedliche horizontale Zwischenräume ergeben. Zwischen einem Zeichen der Klasse Ordinary und einem darauf folgenden der Klasse Opening wird z.B. kein Abstand eingefügt, zwischen der Klasse Ordinary gefolgt von der Klasse Inner jedoch ein schmaler Abstand. Die Abstände in der Unterformel innerhalb der „`\left...\right`“-Konstruktion gestalten sich wie erwartet, mit einem Opening-Symbol am Anfang und einem Closing-Symbol am Ende. Im nächsten Beispiel werden die Unterschiede ebenfalls durch vergrößerte Abstände verdeutlicht.

$$a\left(\sum x\right) \neq a\left(\sum x\right)$$

\thinmuskip=10mu \medmuskip=17mu \thickmuskip=30mu
\[a \Bigl(\sum x \Bigr) \neq a \left(\sum x \right) \]

Bsp.
8-9-3

Alles in allem reicht es nicht aus, ein Symbol in den folgenden Tabellen nachzuschlagen. Man sollte auch prüfen, ob es der gewünschten Symbolklasse angehört, damit es bei seiner Verwendung die richtigen Abstände erhält. In Beispiel 8-9-4 auf Seite 545 wird gezeigt, wie man neue Symbole definiert, die sich nur in ihrer Klasse von bereits existierenden unterscheiden.

8.9.2 Buchstaben, Ziffern und andere reguläre Zeichen

Die akzentlosen lateinischen ASCII-Buchstaben und die Arabischen Ziffern (siehe Tabelle 8.8) werden alle unter dem Namen „Alphabetzeichen“ zusammengefasst. Die für sie verwendete Schrift kann variieren: In mathematischen Formeln ist für lateinische Buchstaben Kursivschrift voreingestellt, während arabische Ziffern in aufrechter Schriftform mit Serifen gesetzt werden. Alle Alphabetzeichen gehören zur Klasse der regulären Zeichen (Ordinary).

¹Ein weiterer wichtiger Unterschied liegt darin, dass das Material innerhalb einer „`\left...\right`“-Konstruktion als separate Unterformel verarbeitet wird (siehe Abschnitt 8.7.2 auf Seite 516).

$$A B C D E F G H I J K L M N O P Q R S T U V W X Y Z$$

$$a b c d e f g h i j k l m n o p q r s t u v w x y z$$

$$0 1 2 3 4 5 6 7 8 9$$

Tabelle 8.8: Symbole der Klasse `\mathord` (lateinische Buchstaben und arabische Ziffern)

Anders als lateinische Buchstaben, haben die griechischen Buchstaben für mathematische Formeln nicht mehr viel mit den Glyphen zum Setzen normaler griechischer Texte gemein. Ein interessanter Zufall im 18. Jahrhundert führte dazu, dass die vorherrschende europäische Tradition des Mathematiksatzes für griechische Kleinbuchstaben in mathematischen Formeln Kursivschrift vorsieht, während für griechische Großbuchstaben die aufrechte Serifenschrift vorgegeben ist. (Andere Disziplinen, wie z.B. die Physik und die Chemie, folgen etwas anderen Traditionen.)

Die griechischen Großbuchstaben in den ersten Zeilen von Tabelle 8.9 gehören ebenfalls zu den Alphabetzeichen, deren Schrift variiert, wobei eine aufrechte Serifenschrift voreingestellt ist. Diejenigen griechischen Buchstaben, deren Form lateinischen Buchstaben entspricht (z.B. *A* und *Alpha*, *B* und *Beta*, *K* und *Kappa*, *O* und *Omicron*), sind nicht in der Tabelle aufgeführt. Ebenso fehlt das Omikron in der Liste der griechischen Kleinbuchstaben, da es genauso aussieht wie das lateinische *o*. In der Praxis bedeutet dies, dass griechische Buchstaben mit lateinischen Doppelgängern nicht in mathematischen Formeln verwendet werden.

In Tabelle 8.10 auf der nächsten Seite sind weitere buchstabenförmige Zeichen der Klasse `Ordinary` aufgelistet. Bei den ersten vier handelt es sich um hebräische Buchstaben. Tabelle 8.11 enthält die verbleibenden regulären Symbole einschließlich einiger gängiger Interpunktionszeichen. Diese verhalten sich wie Buchstaben und Ziffern, erhalten also niemals einen zusätzlichen Abstand.

Δ <code>\Delta</code> ^(var)	Γ <code>\Gamma</code> ^(var)	Λ <code>\Lambda</code> ^(var)	Ω <code>\Omega</code> ^(var)	Φ <code>\Phi</code> ^(var)
Π <code>\Pi</code> ^(var)	Ψ <code>\Psi</code> ^(var)	Σ <code>\Sigma</code> ^(var)	Θ <code>\Theta</code> ^(var)	Υ <code>\Upsilon</code> ^(var)
Ξ <code>\Xi</code> ^(var)	α <code>\alpha</code>	β <code>\beta</code>	χ <code>\chi</code>	δ <code>\delta</code>
F <code>\digamma</code>	ϵ <code>\epsilon</code>	η <code>\eta</code>	γ <code>\gamma</code>	ι <code>\iota</code>
κ <code>\kappa</code>	λ <code>\lambda</code>	μ <code>\mu</code>	ν <code>\nu</code>	ω <code>\omega</code>
ϕ <code>\phi</code>	π <code>\pi</code>	ψ <code>\psi</code>	ρ <code>\rho</code>	σ <code>\sigma</code>
τ <code>\tau</code>	θ <code>\theta</code>	υ <code>\upsilon</code>	ε <code>\varepsilon</code>	$\varkappa$ <code>\varkappa</code>
φ <code>\varphi</code>	ϖ <code>\varpi</code>	ϱ <code>\varrho</code>	ς <code>\varsigma</code>	ϑ <code>\vartheta</code>
ξ <code>\xi</code>	ζ <code>\zeta</code>			

Blaue Symbole erfordern das `amssymb`-Paket. ^(var) kennzeichnet ein variables Alphabetzeichen.

Tabelle 8.9: Symbole der Klasse `\mathord` (griechische)

$\aleph$	<code>\aleph</code>	$\beth$	<code>\beth</code>	$\daleth$	<code>\daleth</code>	$\gimel$	<code>\gimel</code>
$\$$	<code>\\$</code>	$\Im$	<code>\Im</code>	$\Re$	<code>\Re</code>	$\Bbbk$	<code>\Bbbk</code>
$\mathbb{R}$	<code>\circledR</code>	$\mathbb{S}$	<code>\circledS</code>	$\complement$	<code>\complement</code>	ℓ	<code>\ell</code>
$\eth$	<code>\eth</code>	$\Finv$	<code>\Finv</code>	$\Game$	<code>\Game</code>	$\hbar$	<code>\hbar</code> ^(kernel)
$\hslash$	<code>\hslash</code>	$\imath$	<code>\imath</code>	$\jmath$	<code>\jmath</code>	$\mathdollar$	<code>\mathdollar</code>
$\mathparagraph$	<code>\mathparagraph</code>	$\mathsection$	<code>\mathsection</code>	$\mathsterling$	<code>\mathsterling</code>	$\mho$	<code>\mho</code>
$\P$	<code>\P</code>	∂	<code>\partial</code>	$\pounds$	<code>\pounds</code>	$\S$	<code>\S</code>
$\wp$	<code>\wp</code>	$\yen$	<code>\yen</code>				

Blaue Symbole erfordern das amssymb-Paket.

Synonyme: $\$$ `\mathdollar`, $\$$ `\$` $\mathparagraph$ `\mathparagraph`, $\P$ `\P` $\mathsection$ `\mathsection`, $\S$

$\mathsterling$ `\mathsterling`, $\pounds$ `\pounds`

Tabelle 8.10: Symbole der Klasse `\mathord` (buchstabenförmige)

!	!	.	.	/	/
?	?	@	@		
#	<code>\#</code>	%	<code>\%</code>	&	<code>\&</code>
-	<code>_</code>		<code>\ </code>	$\angle$	<code>\angle</code> ^(kernel)
	<code>\Arrowvert</code>		<code>\arrowvert</code>	$\backprime$	<code>\backprime</code>
\	<code>\backslash</code>	★	<code>\bigstar</code>	◆	<code>\blacklozenge</code>
■	<code>\blacksquare</code>	▲	<code>\blacktriangle</code>	▼	<code>\blacktriangledown</code>
⊥	<code>\bot</code>	,	<code>\bracevert</code>	✓	<code>\checkmark</code>
♣	<code>\clubsuit</code>	©	<code>\copyright</code>	\	<code>\diagdown</code>
/	<code>\diagup</code>	◇	<code>\diamondsuit</code>	∅	<code>\emptyset</code>
∃	<code>\exists</code>	♭	<code>\flat</code>	∀	<code>\forall</code>
♥	<code>\heartsuit</code>	∞	<code>\infty</code>	⚡	<code>\lightning</code> ^(StM)
¬	<code>\lnot</code>	◇	<code>\lozenge</code>	⌘	<code>\maltese</code>
∠	<code>\measuredangle</code>	∇	<code>\nabla</code>	ℵ	<code>\natural</code>
¬	<code>\neg</code>	∄	<code>\nexists</code>	′	<code>\prime</code>
♯	<code>\sharp</code>	♠	<code>\spadesuit</code>	∠	<code>\sphericalangle</code>
□	<code>\square</code>	√	<code>\surd</code>	⊤	<code>\top</code>
△	<code>\triangle</code>	▽	<code>\triangledown</code>	©	<code>\varcopyright</code> ^(StM)
∅	<code>\varnothing</code>		<code>\Vert</code>		<code>\vert</code>

Blaue Symbole erfordern das amssymb-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das stmaryrd-Paket.

Man beachte, dass Frage- und Ausrufungszeichen sowie der Punkt in Formeln nicht als Interpunktionszeichen behandelt werden.

Synonyme: $\neg$ `\lnot`, $\neg$ `\neg` $\vert$ `\vert`, $\vert$ `\vert` $\Vert$ `\Vert`, $\Vert$ `\Vert`

Tabelle 8.11: Symbole der Klasse `\mathord` (verschiedene)

$\acute{x}$	<code>\acute{x}</code>	$\bar{x}$	<code>\bar{x}</code>	$\breve{x}$	<code>\breve{x}</code>	$\check{x}$	<code>\check{x}</code>
$\dddot{x}$	<code>\dddot{x}</code>	$\ddot{x}$	<code>\ddot{x}</code>	$\ddot{x}$	<code>\ddot{x}</code>	$\dot{x}$	<code>\dot{x}</code>
$\grave{x}$	<code>\grave{x}</code>	$\hat{x}$	<code>\hat{x}</code>	$\mathring{x}$	<code>\mathring{x}</code>	$\tilde{x}$	<code>\tilde{x}</code>
$\vec{x}$	<code>\vec{x}</code>	$\widehat{xyz}$	<code>\widehat{xyz}</code>	$\widetilde{xyz}$	<code>\widetilde{xyz}</code>		

Blaue Akzente erfordern das amsmath-Paket.

Die letzten beiden Akzente sind in verschiedenen Breiten verfügbar, wobei automatisch die größte passende verwendet wird.

Tabelle 8.12: Akzente, die `\mathord`-Unterformeln erzeugen

Die Zeichen der Tabelle 8.11 werden häufig fälschlicherweise direkt als binäre Operatoren oder Relationszeichen benutzt, anstatt dazu einen entsprechend definierten Befehl für mathematische Symbole zu verwenden. Wenn man also Befehle wie `\#`, `\square` oder `\&` verwendet, sollte man sorgfältig darauf achten, dass die Symbolzwischenräume stimmen, oder noch besser einen eigenen Symbolbefehl definieren.

Bsp.
8-9-4

$a \neg b$ $x \square y + z$

$a \neg b$ $x \square y + z$

$a \neg b$ $x \square y + z$

```
\usepackage[fleqn]{amsmath} \usepackage{amssymb}
\DeclareMathSymbol\neg {\mathbin}{symbols}{"3A}
\DeclareMathSymbol\square{\mathrel}{AMSA}{"03}
\[ a \neg b \qquad x \square y + z \]
\[ a \mathbin{\neg} b \qquad x \mathrel{\square} y + z \]
\[ a \neg b \qquad b \qquad x \rsquare y + z \]
```

Die Deklaration `\DeclareMathSymbol` wurde in Abschnitt 7.10.7 erläutert. Die richtigen Werte für ihre Argumente lassen sich sehr einfach finden, indem man sich die Definitionen in der Datei `amssymb.sty` oder `fontmath.ltx` (für die Standardsymbole) anschaut. Für das Beispiel wurden die Definitionen für `\neg` und `\square` herausgesucht, beide `\mathord`-Befehle ersetzt und die Symbole schließlich mit einem neuen Namen versehen.

8.9.3 Mathematische Akzente

Die Akzentbefehle für Formeln sind in Tabelle 8.12 aufgeführt. Die meisten von ihnen sind bereits in Standard- $\text{\LaTeX}$ definiert. In Abschnitt 8.4.8 wurde erläutert, wie man zusätzliche Akzentbefehle definiert, und Abschnitt 8.5.2 enthält nähere Informationen zu erweiterbaren Akzenten. Wenn man ein Symbol mit einem Formelakzent versieht, erhält man immer ein reguläres Symbol (Ordinary). Man kann also nicht ohne weiteres nur mit einem Akzent neue Symbole der Klassen Binary oder Relation erzeugen.

Bsp.
8-9-5

$a = b$ aber $a \cong b$ ist nicht $a \cong b$

```
\usepackage{amstext}
\[ a = b \text{ aber } a \tilde{=} b
\text{ ist nicht } a \mathrel{\tilde{=}} b \]
```

In Abschnitt 8.4.10 wurden weitere Möglichkeiten gezeigt, wie man Symbole über Relationszeichen positioniert. Wenn man ein i oder ein j mit einem Akzent versehen möchte, verwendet man dazu am besten die Varianten ohne Punkt, `\imath` und `\jmath`, also z.B. `\hat{\jmath}` um $\hat{j}$ zu erhalten.

*	*	+	+	−	−
∏	<code>\amalg</code>	*	<code>\ast</code>	⌀	<code>\baro^(StM)</code>
⌒	<code>\barwedge</code>	∥	<code>\bbslash^(StM)</code>	▽	<code>\bigtriangledown</code>
△	<code>\bigtriangleup</code>	⊓	<code>\Cap</code>	∩	<code>\cap</code>
⊔	<code>\Cup</code>	∪	<code>\cup</code>	⋈	<code>\curlyvee</code>
⋈	<code>\curlywedge</code>	†	<code>\dag</code>	†	<code>\dagger</code>
‡	<code>\ddag</code>	‡	<code>\ddagger</code>	◇	<code>\diamond</code>
÷	<code>\div</code>	*	<code>\divideontimes</code>	⋈	<code>\dotplus</code>
⊔	<code>\doublecap</code>	⊔	<code>\doublecup</code>	∥	<code>\fatbslash^(StM)</code>
⋈	<code>\fatsemi^(StM)</code>	//	<code>\fatslash^(StM)</code>	>	<code>\gtrdot</code>
⋈	<code>\intercal</code>		<code>\interleave^(StM)</code>	∧	<code>\land</code>
⌊	<code>\lbag^(StM)</code>	◁	<code>\leftslice^(StM)</code>	∧	<code>\leftthreetimes</code>
≤	<code>\lessdot</code>	∨	<code>\lor</code>	⋈	<code>\ltimes</code>
⋈	<code>\merge^(StM)</code>	⊖	<code>\minuso^(StM)</code>	⋈	<code>\moo^(StM)</code>
⋈	<code>\mp</code>	⊕	<code>\nplus^(StM)</code>	±	<code>\pm</code>
∫	<code>\rbag^(StM)</code>	▷	<code>\rightslice^(StM)</code>	⋈	<code>\rightthreetimes</code>
⋈	<code>\rtimes</code>	\	<code>\setminus</code>	⋈	<code>\smallsetminus</code>
□	<code>\sqcap</code>	⊔	<code>\sqcup</code>	//	<code>\sslash^(StM)</code>
*	<code>\star</code>	⊔	<code>\talloblong^(StM)</code>	×	<code>\times</code>
◁	<code>\triangleleft</code>	▷	<code>\triangleright</code>	⊕	<code>\uplus</code>
▽	<code>\varbigtriangledown^(StM)</code>	△	<code>\varbigtriangleup^(StM)</code>	⋈	<code>\varcurlyvee^(StM)</code>
⋈	<code>\varcurlywedge^(StM)</code>	×	<code>\vartimes^(StM)</code>	∨	<code>\vee</code>
∨	<code>\veebar</code>	∧	<code>\wedge</code>	⌊	<code>\wr</code>
⋈	<code>\Ydown^(StM)</code>	◁	<code>\Yleft^(StM)</code>	⋈	<code>\Yright^(StM)</code>
⋈	<code>\Yup^(StM)</code>				

Blaue Symbole erfordern das `amssymb`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryrd`-Paket.

Die nach links und nach rechts weisenden Dreiecke sind auch als Relationszeichen verfügbar.

Das Paket `stmaryrd` verwandelt die binären Zeichen `\bigtriangleup` und `\bigtriangledown` verwirrenderweise in Zeichen der Klasse `Operator`, so dass nur ihre Synonyme `\varbigtriangleup` und `\varbigtriangledown` als binäre Operatoren verbleiben.

Synonyme: `∧ \land, \wedge` `∨ \lor, \vee` `⊔ \doublecup, \Cup` `⊓ \doublecap, \Cap`
`*` `\ast, *` `† \dag, \dagger` `‡ \ddag, \ddagger`

Tabelle 8.13: Symbole der Klasse `\mathbin` (verschiedene)

⊠	<code>\boxast^(StM)</code>	⊠	<code>\boxbar^(StM)</code>	⊠	<code>\boxbox^(StM)</code>
⊠	<code>\boxbslash^(StM)</code>	⊠	<code>\boxcircle^(StM)</code>	⊠	<code>\boxdot</code>
⊠	<code>\boxempty^(StM)</code>	⊠	<code>\boxminus</code>	⊠	<code>\boxplus</code>
⊠	<code>\boxslash^(StM)</code>	⊠	<code>\boxtimes</code>	⊠	<code>\oblong^(StM)</code>

Alle Symbole erfordern entweder das `amssymb`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryrd`-Paket.

Tabelle 8.14: Symbole der Klasse `\mathbin` (Kästen)

○	<code>\bigcirc</code>	●	<code>\bullet</code>	·	<code>\cdot</code>
▪	<code>\centerdot</code>	○	<code>\circ</code>	⊗	<code>\circledast</code>
⊙	<code>\circledcirc</code>	⊖	<code>\circleddash</code>	⊗	<code>\oast^(StM)</code>
⊖	<code>\obar^(StM)</code>	⊗	<code>\obslash^(StM)</code>	⊙	<code>\ocircle^(StM)</code>
⋅	<code>\odot</code>	⊗	<code>\ogreaterthan^(StM)</code>	⊗	<code>\olessthan^(StM)</code>
⊖	<code>\ominus</code>	⊕	<code>\oplus</code>	⊗	<code>\oslash</code>
⊗	<code>\otimes</code>	⊗	<code>\ovee^(StM)</code>	⊗	<code>\owedge^(StM)</code>
○	<code>\varbigcirc^(StM)</code>	⊗	<code>\varoast^(StM)</code>	⊖	<code>\varobar^(StM)</code>
⊗	<code>\varobslash^(StM)</code>	⊙	<code>\varocircle^(StM)</code>	⋅	<code>\varodot^(StM)</code>
⊗	<code>\varogreaterthan^(StM)</code>	⊗	<code>\varolessthan^(StM)</code>	⊖	<code>\varominus^(StM)</code>
⊕	<code>\varoplus^(StM)</code>	⊗	<code>\varoslash^(StM)</code>	⊗	<code>\varotimes^(StM)</code>
⊗	<code>\varovee^(StM)</code>	⊗	<code>\varowedge^(StM)</code>		

Blaue Symbole erfordern das `amssymb`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryrd`-Paket.

Die Option `heavycircles` des `stmaryrd`-Paketes betrifft alle Befehle, die mit `\var` beginnen sowie ihre normalen Varianten.

Synonyme: ⊗ `\oast`, `\circledast` ⊙ `\ocircle`, `\circledcirc`

Tabelle 8.15: Symbole der Klasse `\mathbin` (Kreise)

8.9.4 Binäre Operatorzeichen

Bei den Operatoren der Klasse Binary kann man unter mehr als 100 Symbolen wählen. Die meisten dieser binären Zeichen sind in Tabelle 8.13 auf der gegenüberliegenden Seite zu sehen. Einige von ihnen sind unter anderem Namen auch als Relationssymbole verfügbar.

Das Paket `amssymb` enthält ein paar Kasten-Symbole für binäre Operatoren; das `stmaryrd`-Paket fügt noch viele weitere hinzu. Sie sind in Tabelle 8.14 aufgelistet.

Das `stmaryrd`-Paket kann mit der Option `heavycircles` geladen werden. Dadurch tauscht jeder Kreissymbol-Befehl aus Tabelle 8.15, der mit `\var` beginnt, seine Definition mit dem entsprechenden Befehl ohne „`var`“; also wird z.B. das Symbol `\varodot` zu `\odot` und umgekehrt.

8.9.5 Relationssymbole

Die Klasse der binären Relationszeichen ist noch größer als die der Operatoren der Klasse Binary. Die Liste beginnt mit Gleichheits- und Ordnungszeichen (Tabelle 8.16 auf der nächsten Seite). Man kann jedes Symbol der Klasse Relation durchstreichen, indem man ihm den Befehl `\not` voranstellt. Das negierte Symbol stellt das Komplement (oder die Negation) der Relation dar.

Bsp.
8-9-6

$u \not\prec v$ oder $a \notin \mathbf{A}$ $\$ u \not< v \$$ oder $\$ a \not\in \mathbf{A} \$$

Insbesondere bei größeren Symbolen führt diese generische Methode der Negation eines Relationszeichens nicht immer zu einem ansprechenden Ergebnis, da der Schrägstrich stets die gleiche Größe, Position und Lage aufweist.

$<$	$<$	$=$	$=$	$>$	$>$	$\approx$	<code>\approx</code>
$\approx$	<code>\approxeq</code>	$\asymp$	<code>\asymp</code>	$\backsimeq$	<code>\backsimeq</code>	$\backsimeq$	<code>\backsimeq</code>
$\bumpeq$	<code>\Bumpeq</code>	$\bumpeq$	<code>\bumpeq</code>	$\circeq$	<code>\circeq</code>	$\cong$	<code>\cong</code>
$\curlyeqprec$	<code>\curlyeqprec</code>	$\curlyeqsucc$	<code>\curlyeqsucc</code>	$\doteq$	<code>\doteq</code>	$\doteq$	<code>\doteq</code>
$\doteqdot$	<code>\doteqdot</code>	$\eqcirc$	<code>\eqcirc</code>	$\eqsim$	<code>\eqsim</code>	$\eqslantgtr$	<code>\eqslantgtr</code>
$\eqslantless$	<code>\eqslantless</code>	$\equiv$	<code>\equiv</code>	$\fallingdotseq$	<code>\fallingdotseq</code>	$\ge$	<code>\ge</code>
$\geq$	<code>\geq</code>	$\geqq$	<code>\geqq</code>	$\geqslant$	<code>\geqslant</code>	$\gg$	<code>\gg</code>
$\ggg$	<code>\ggg</code>	$\gggtr$	<code>\gggtr</code>	$\gtrapprox$	<code>\gtrapprox</code>	$\gtreqless$	<code>\gtreqless</code>
$\gtreqqless$	<code>\gtreqqless</code>	$\gtrless$	<code>\gtrless</code>	$\gtrsim$	<code>\gtrsim</code>	$\leftrightharpoonup^{(StM)}$	<code>\leftrightharpoonup^{(StM)}</code>
$\leq$	<code>\leq</code>	$\leqq$	<code>\leqq</code>	$\leqslant$	<code>\leqslant</code>	$\lessapprox$	<code>\lessapprox</code>
$\lesseqgtr$	<code>\lesseqgtr</code>	$\lesseqqgtr$	<code>\lesseqqgtr</code>	$\lessgtr$	<code>\lessgtr</code>	$\lesssim$	<code>\lesssim</code>
$\ll$	<code>\ll</code>	$\lll$	<code>\lll</code>	$\lllless$	<code>\lllless</code>	$\lllless$	<code>\lllless</code>
$\prec$	<code>\prec</code>	$\precapprox$	<code>\precapprox</code>	$\preccurlyeq$	<code>\preccurlyeq</code>	$\preceq$	<code>\preceq</code>
$\precsim$	<code>\precsim</code>	$\risingdotseq$	<code>\risingdotseq</code>	$\sim$	<code>\sim</code>	$\simeq$	<code>\simeq</code>
$\succ$	<code>\succ</code>	$\succapprox$	<code>\succapprox</code>	$\succcurlyeq$	<code>\succcurlyeq</code>	$\succeq$	<code>\succeq</code>
$\succsim$	<code>\succsim</code>	$\thickapprox$	<code>\thickapprox</code>	$\thicksim$	<code>\thicksim</code>	$\triangleq$	<code>\triangleq</code>

Blaue Symbole erfordern das `amssymb`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryrd`-Paket.

Synonyme: $\leq$ `\le`, `\leq` $\geq$ `\ge`, `\geq` $\doteq$ `\Doteq`, `\doteqdot` $\lllless$, `\lll`
 $\ggg$ `\gggtr`, `\ggg`

Tabelle 8.16: Symbole der Klasse `\mathrel` (Gleichheit und Ordnung)

$\gtrapprox$	<code>\gnapprox</code>	$\gneq$	<code>\gneq</code>	$\gneqq$	<code>\gneqq</code>	$\gnsim$	<code>\gnsim</code>
$\gvertneqq$	<code>\gvertneqq</code>	$\lnapprox$	<code>\lnapprox</code>	$\lneq$	<code>\lneq</code>	$\lneqq$	<code>\lneqq</code>
$\lnsim$	<code>\lnsim</code>	$\lvertneqq$	<code>\lvertneqq</code>	$\ncong$	<code>\ncong</code>	$\ne$	<code>\ne</code>
$\neq$	<code>\neq</code>	$\ngeq$	<code>\ngeq</code>	$\ngeqq$	<code>\ngeqq</code>	$\ngeqslant$	<code>\ngeqslant</code>
$\ngtr$	<code>\ngtr</code>	$\nleq$	<code>\nleq</code>	$\nleqq$	<code>\nleqq</code>	$\nleqslant$	<code>\nleqslant</code>
$\nless$	<code>\nless</code>	$\nprec$	<code>\nprec</code>	$\npreceq$	<code>\npreceq</code>	$\nsim$	<code>\nsim</code>
$\nsucc$	<code>\nsucc</code>	$\nsucceq$	<code>\nsucceq</code>	$\precnapprox$	<code>\precnapprox</code>	$\precneqq$	<code>\precneqq</code>
$\precnnsim$	<code>\precnnsim</code>	$\succnapprox$	<code>\succnapprox</code>	$\succneqq$	<code>\succneqq</code>	$\succnnsim$	<code>\succnnsim</code>

Blaue Symbole erfordern das `amssymb`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryrd`-Paket.

Synonyme: $\neq$ `\ne`, `\neq`

Tabelle 8.17: Symbole der Klasse `\mathrel` (Gleichheit und Ordnung – negiert)

$\blacktriangleleft$ <code>\blacktriangleleft</code>	$\blacktriangleright$ <code>\blacktriangleright</code>	$\in$ <code>\in</code>
$\in$ <code>\inplus^(StM)</code>	$\ni$ <code>\ni</code>	$\niplus$ <code>\niplus^(StM)</code>
$\ntrianglelefteqslant$ <code>\ntrianglelefteqslant^(StM)</code>	$\ntrianglerighteqslant$ <code>\ntrianglerighteqslant^(StM)</code>	$\owns$ <code>\owns</code>
$\sqsubset$ <code>\sqsubset</code>	$\sqsubseteq$ <code>\sqsubseteq</code>	$\sqsupset$ <code>\sqsupset</code>
$\sqsupseteq$ <code>\sqsupseteq</code>	$\Subset$ <code>\Subset</code>	$\subset$ <code>\subset</code>
$\subseteq$ <code>\subseteq</code>	$\subseteqq$ <code>\subseteqq</code>	$\subsetplus$ <code>\subsetplus^(StM)</code>
$\subseteqq$ <code>\subseteqq^(StM)</code>	$\Supset$ <code>\Supset</code>	$\supset$ <code>\supset</code>
$\supseteq$ <code>\supseteq</code>	$\supseteqq$ <code>\supseteqq</code>	$\supsetplus$ <code>\supsetplus^(StM)</code>
$\supseteqq$ <code>\supseteqq^(StM)</code>	$\trianglelefteq$ <code>\trianglelefteq</code>	$\trianglelefteqslant$ <code>\trianglelefteqslant^(StM)</code>
$\trianglerighteq$ <code>\trianglerighteq</code>	$\trianglerighteqslant$ <code>\trianglerighteqslant^(StM)</code>	$\vartriangle$ <code>\vartriangle</code>
$\vartriangleleft$ <code>\vartriangleleft</code>	$\vartriangleright$ <code>\vartriangleright</code>	

Blaue Symbole erfordern das `amssymb`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryrd`-Paket.

Synonyme: `\owns`, `\ni`

Tabelle 8.18: Symbole der Klasse `\mathrel` (Mengenoperatoren)

$\notin$ <code>\notin</code>	$\nsubseteq$ <code>\nsubseteq</code>	$\nsubseteqq$ <code>\nsubseteqq</code>
$\nsupseteq$ <code>\nsupseteq</code>	$\nsupseteqq$ <code>\nsupseteqq</code>	$\ntriangleleft$ <code>\ntriangleleft</code>
$\ntrianglelefteq$ <code>\ntrianglelefteq</code>	$\ntriangleright$ <code>\ntriangleright</code>	$\ntrianglerighteq$ <code>\ntrianglerighteq</code>
$\subsetneq$ <code>\subsetneq</code>	$\subsetneqq$ <code>\subsetneqq</code>	$\supsetneq$ <code>\supsetneq</code>
$\supsetneqq$ <code>\supsetneqq</code>	$\varsubsetneq$ <code>\varsubsetneq</code>	$\varsubsetneqq$ <code>\varsubsetneqq</code>
$\varsupsetneq$ <code>\varsupsetneq</code>	$\varsupsetneqq$ <code>\varsupsetneqq</code>	

Blaue Symbole erfordern das `amssymb`-Paket.

Tabelle 8.19: Symbole der Klasse `\mathrel` (Mengenoperatoren – negiert)

Daher sind auch einige eigens gestaltete „negierte Symbole“ verfügbar (siehe Tabelle 8.17 auf der vorherigen Seite). Wenn man die Wahl hat, sollte man normalerweise den eigens gestalteten Glyphen den Vorzug geben. Warum wird durch einen Vergleich der Symbole im folgenden Beispiel deutlich.

Bsp.
8-9-7

$\nleq$ $\nleqq$ $\not\leq$
 $\nleq$ $\nleqq$ $\approx$

`\usepackage{amssymb}`

`\$ \not\leq \ \ \not\succeq \ \ \not\sim \$ \ \par`
`\$ \nleq \ \ \nsucceq \ \ \nsim \ \ \$`

Als nächstes folgen die Symbole der Klasse `Relation`, die als Mengenoperatoren dienen, sowie ihre Negationen (siehe die Tabellen 8.18 und 8.19). Ihnen folgen pfeilförmige Relationszeichen (in den Tabellen 8.20 und 8.21). Einige erweiterbare Pfeilkonstruktionen, die zusammengesetzte Relationszeichen erzeugen, wurden in Abschnitt 8.5.2 auf Seite 509 beschrieben.

	<code>\circlearrowleft</code>		<code>\circlearrowright</code>
	<code>\curlyveedownarrow^(StM)</code>		<code>\curlyveeuparrow^(StM)</code>
	<code>\curlywedgedownarrow^(StM)</code>		<code>\curlywedgeuparrow^(StM)</code>
	<code>\curvearrowleft</code>		<code>\curvearrowright</code>
$\dashrightarrow$	<code>\dashrightarrow</code>	$\dashleftarrow$	<code>\dashleftarrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\Downarrow$	<code>\Downarrow</code>
$\downarrow$	<code>\downarrow</code>	$\Downarrow$	<code>\downdownarrows</code>
$\downharpoonright$	<code>\downharpoonright</code>	$\leftarrow$	<code>\gets</code>
$\hookleftarrow$	<code>\hookleftarrow</code>	$\hookrightarrow$	<code>\hookrightarrow</code>
$\Leftarrow$	<code>\Leftarrow</code>	$\leftarrow$	<code>\leftarrow</code>
$\leftarrowtail$	<code>\leftarrowtail</code>	$\leftarrow$	<code>\leftarrowtriangle^(StM)</code>
$\leftrightharpoonup$	<code>\leftrightharpoonup^(StM)</code>	$\leftharpoonup$	<code>\leftharpoonup</code>
$\leftharpoonup$	<code>\leftharpoonup</code>	$\Leftarrow$	<code>\leftleftarrows</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\leftrightarrow$	<code>\leftrightarrow</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Leftrightarrow$	<code>\leftrightharpoons</code>
$\Leftrightarrow$	<code>\Leftrightarrow</code>	$\Leftarrow$	<code>\Lleftarrow</code>
$\Longleftarrow$	<code>\Longleftarrow</code>	$\longleftarrow$	<code>\longleftarrow</code>
$\Longleftrightarrow$	<code>\Longleftrightarrow</code>	$\longleftrightarrow$	<code>\longleftrightarrow</code>
$\Longmapsto$	<code>\Longmapsto^(StM)</code>	$\longmapsto$	<code>\longmapsto^(StM)</code>
$\Longrightarrow$	<code>\Longrightarrow</code>	$\longrightarrow$	<code>\longrightarrow</code>
$\looparrowleft$	<code>\looparrowleft</code>	$\looparrowright$	<code>\looparrowright</code>
$\lsh$	<code>\lsh</code>	$\Leftarrow$	<code>\Mapsfrom^(StM)</code>
$\mapsto$	<code>\mapsto^(StM)</code>	$\mapsto$	<code>\Mapsto^(StM)</code>
$\mapsto$	<code>\mapsto</code>	$\multimap$	<code>\multimap</code>
$\nearrow$	<code>\nearrow</code>	$\nearrow$	<code>\nnearrow^(StM)</code>
$\nwarrow$	<code>\nwarrow^(StM)</code>	$\nwarrow$	<code>\nwarrow</code>
$\restriction$	<code>\restriction</code>	$\Rightarrow$	<code>\Rightarrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\rightarrowtail$	<code>\rightarrowtail</code>
$\rightarrowtriangle$	<code>\rightarrowtriangle^(StM)</code>	$\rightarrow$	<code>\rightharpoonup</code>
$\rightarrow$	<code>\rightarrow</code>	$\rightleftarrows$	<code>\rightleftarrows</code>
$\rightarrow$	<code>\rightarrow</code>	$\Rightarrow$	<code>\Rightarrow</code>
$\rightarrow$	<code>\rightarrow</code>	$\Rightarrow$	<code>\Rightarrow</code>
$\rightsquigarrow$	<code>\rightsquigarrow</code>	$\searrow$	<code>\searrow</code>
$\Rsh$	<code>\Rsh</code>	$\searrow$	<code>\shortleftarrow^(StM)</code>
$\shortdownarrow$	<code>\shortdownarrow^(StM)</code>	$\leftarrow$	<code>\shortleftarrow^(StM)</code>
$\rightarrow$	<code>\shortrightarrow^(StM)</code>	$\uparrow$	<code>\shortuparrow^(StM)</code>
$\searrow$	<code>\searrow^(StM)</code>	$\swarrow$	<code>\swarrow^(StM)</code>
$\swarrow$	<code>\swarrow</code>	$\rightarrow$	<code>\rightarrow</code>
$\twoheadleftarrow$	<code>\twoheadleftarrow</code>	$\twoheadrightarrow$	<code>\twoheadrightarrow</code>
$\Uparrow$	<code>\Uparrow</code>	$\uparrow$	<code>\uparrow</code>
$\Updownarrow$	<code>\Updownarrow</code>	$\updownarrow$	<code>\updownarrow</code>
$\upharpoonleft$	<code>\upharpoonleft</code>	$\upharpoonright$	<code>\upharpoonright</code>
$\Uparrow$	<code>\Uparrow</code>		

Blaue Symbole erfordern das amssymb-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das stmaryrd-Paket.

Synonyme: $\leftarrow$ `\gets`, `\leftarrow` $\rightarrow$ `\rightarrow`, `\rightarrow` $\restriction$ `\upharpoonright`
 $\dashrightarrow$ `\dashrightarrow`, `\dashrightarrow`

Tabelle 8.20: Symbole der Klasse `\mathord` (Pfeile)

$\nleftarrow$	<code>\nLeftarrow</code>	$\nrightarrow$	<code>\nRightarrow</code>	$\Leftrightarrow$	<code>\nLeftrightarrow</code>
$\leftrightsquigarrow$	<code>\nletrightarrow</code>	$\rightsquigarrow$	<code>\nRrightarrow</code>	$\rightharpoonup$	<code>\nrightharpoonup</code>

Blaue Symbole erfordern das amssymb-Paket.

Tabelle 8.21: Symbole der Klasse `\mathord` (Pfeile – negiert)
$$\begin{array}{llll} / \backslash \text{Arrownot}^{(StM)} & / \backslash \text{arrownot}^{(StM)} & \hookleftarrow \backslash \text{hook} & / \backslash \text{Longarrownot}^{(StM)} \\ / \backslash \text{longarrownot}^{(StM)} & | \backslash \text{Mapsfromchar}^{(StM)} & | \backslash \text{mapsfromchar}^{(StM)} & | \backslash \text{Mapstochar}^{(StM)} \\ | \backslash \text{mapstochar} & / \backslash \text{not} & \rhook & \end{array}$$

Blaue Symbole erfordern das stmaryrd-Paket.

Diese Symbole sind zur Kombination mit Pfeilen gedacht; z.B. ergibt `\longarrownot\longleftarrow` das Zeichen $\nleftarrow$.

Mit `\joinrel` lassen sich Relationszeichen „zusammenkleben“; z.B. ergibt `\hook\joinrel\longrightarrow` das Zeichen $\hookrightarrow$. Die Abmessungen dieser Symbole machen sie für andere Zwecke ungeeignet.

Tabelle 8.22: Symbole der Klasse `\mathrel` (Negationen und Pfeilerweiterungen)

:	\backepsilon	∴	\because	∕	\between
⋈	\bowtie	⊣	\dashv	⌢	\frown
	\mid	⊨	\models	↑	\nmid
⋈	\nshortmid	⋈	\nshortparallel	⋈	\nVDash
⋈	\nvDash	⋈	\nvdash		\parallel
⋈	\pitchfork	∝	\propto	⊥	\perp
⋈	\smallfrown	⋈	\smallsmile	⋈	\shortparallel
∝	\varpropto	⋈	\Vdash	∴	\therefore
	\Vdash	⋈	\vdash	⋈	\vdash

Blaue Relationssymbole erfordern das amssymb-Paket.

Da \therefore ein Relationszeichen ist, hat es andere Abstände als vielleicht im Allgemeinen erwartet.

Tabelle 8.23: Symbole der Klasse `\mathrel` (verschiedene)

Zusätzlich zu dem Befehl `\not`, der zur Negation allgemeiner Relationszeichen verwendet wird, wurden eigens weitere Bausteine gestaltet, um pfeilförmige Zeichen zu negieren oder zu erweitern. Sie sind in Tabelle 8.22 zusammengestellt.

Bsp.
8-9-8

 $\longleftrightarrow$

←

`\usepackage{stmaryrd}`

 $\$ \rightarrowtail \longleftrightarrow \quad \nrightarrow \hookleftarrow \$$

In Tabelle 8.23 findet sich abschließend noch eine Sammlung verschiedenster Relationszeichen.

, , $\cdots$ `\cdots` ... `\hdots` ... `\ldots` ... `\mathellipsis`
 ; ; : `\colon` $\ddots$ `\ddots` $\vdots$ `\vdots`

Blaue Interpunktionszeichen erfordern das `amsmath`-Paket.

Die logischen `amsmath`-Befehle, mit denen man normalerweise auf `\cdots` und `\ldots` zugreift, sind in Abschnitt 8.5.1 beschrieben.

Der `\colon`-Befehl wird in `amsmath` umdefiniert, so dass er sich nicht mehr als allgemeines Interpunktionszeichen eignet.

Synonyme: ... `\hdots`, `\ldots` ... `\mathellipsis`, `\ldots`

Tabelle 8.24: Symbole der Klassen `\mathpunct`, `\mathord`, `\mathinner` (Interpunktion)

8.9.6 Interpunktion

Die Zeichen der Klasse Punctuation sind in Tabelle 8.24 dargestellt, zusammen mit einigen anderen, den Interpunktionen ähnlichen Zeichen. Man beachte, dass einige der typischen Interpunktionszeichen (und zwar „. ! ?“) nicht der Symbolklasse Punctuation, sondern Ordinary (reguläre Zeichen) zugeordnet sind. Dadurch verhalten sich diese Symbole, insbesondere ! und ?, beim üblichen Gebrauch möglicherweise anders als erwartet. Einige der hier aufgeführten Punktsymbole gehören zur Klasse Inner; Abschnitt 8.5.1 auf Seite 508 bietet nähere Informationen dazu, wie man Punkte als mathematische Auslassungszeichen verwendet.

$\int$ <code>\int</code> <code>\int</code>	$\oint$ <code>\oint</code> <code>\oint</code>	$\square\square$ <code>\bigbox</code> ^(StM)
$\bigcap$ <code>\bigcap</code> <code>\bigcap</code>	$\bigcup$ <code>\bigcup</code> <code>\bigcup</code>	$\bigvee$ <code>\bigcurlyvee</code> ^(StM)
$\bigwedge$ <code>\bigcurlywedge</code> ^(StM)	$\big $ <code>\biginterleave</code> ^(StM)	$\bigoplus$ <code>\bignplus</code> ^(StM)
$\odot$ <code>\bigodot</code>	$\bigoplus$ <code>\bigoplus</code>	$\bigotimes$ <code>\bigotimes</code>
$\big $ <code>\bigparallel</code> ^(StM)	$\bigsqcap$ <code>\bigsqcap</code> ^(StM)	$\bigsqcup$ <code>\bigsqcup</code>
$\bigtriangledown$ <code>\bigtriangledown</code> ^(StM)	$\bigtriangleup$ <code>\bigtriangleup</code> ^(StM)	$\biguplus$ <code>\biguplus</code>
$\bigvee$ <code>\bigvee</code>	$\bigwedge$ <code>\bigwedge</code>	$\coprod$ <code>\coprod</code>
$\prod$ <code>\prod</code> <code>\prod</code>	$\int\int$ <code>\smallint</code>	$\sum$ <code>\sum</code>

Blaue Operatorsymbole erfordern das `stmaryd`-Paket.

Das Paket `stmaryd` verwandelt die binären Zeichen `\bigtriangleup` und `\bigtriangledown` verwirrenderweise in Zeichen der Klasse Operator, es gibt jedoch Alternativbefehle für die binären Operatoren.

Man beachte, dass `\smallint` gleich groß bleibt.

Tabelle 8.25: Symbole der Klasse `\mathop`

Das Zeichen `:` erzeugt einen Doppelpunkt der Klasse `Relation` und kein Interpunktionszeichen. Als Alternative verfügt Standard-`LATEX` über den Befehl `\colon` für das Zeichen der Klasse `Punctuation`. Leider nimmt das `amsmath`-Paket ziemlich unglückliche, deutliche Änderungen an den Abständen für den Befehl `\colon` vor, so dass er sich nur noch für ein spezielles Layout in Konstruktionen wie $f\colon A\rightarrow B$ eignet, wo er $f\colon A\rightarrow B$ erzeugt. Daher empfiehlt es sich für den einfachen Doppelpunkt in Formeln immer `\mathpunct{:}` zu verwenden.

8.9.7 Operatorzeichen

Die Zeichen der Klasse Operator liegen normalerweise in zwei Größen vor, für die Verwendung in eingebetteten und in abgesetzten Formeln. Die meisten von ihnen sind mit ähnlichen binären Operatoren verwandt. Ob ein Operatorzeichen in abgesetzten Formeln Grenzen zeigt, hängt von verschiedenen Faktoren ab (siehe Abschnitt 8.4.4). Die verfügbare Sammlung ist in Tabelle 8.25 auf der vorherigen Seite abgebildet.

8.9.8 Öffnende und schließende Zeichen

Die Paare der erweiterbaren Begrenzungszeichen erzeugen Symbole der Klassen Opening bzw. Closing, wenn sie allein verwendet werden (also nicht nach einem `\left`, `\right` oder `\middle`-Befehl stehen). Sie sind in Tabelle 8.26 aufgelistet. Abschnitt 8.5.3 auf Seite 510 enthält weitere Informationen zu den erweiterbaren Symbolen.

Um die Schreibweise vertikaler Striche flexibler zu gestalten, werden im `amsmath`-Paket zwei neue Paare erweiterbarer Begrenzungszeichen definiert: `\lvert`, `\rvert`, `\lVert` und `\rVert`. Diese Befehle sind mit den Standard- $\text{\LaTeX}$ -Befehlen `\angle` und `\rangle` vergleichbar.

$\left[\right]$	$\{ \}$	$\ \ $	<code>\lVert \rVert</code>
$\left[\right]$	$\{ \}$	$\ $	<code>\lvert \rvert</code>
$\left[\right]$	$()$	$()$	<code>\lgroup \rgroup</code>
$\left[\right]$	$\langle \rangle$	$\int \int$	<code>\lmoustache \rmoustache</code>
$\left[\right]$			<code>\llbracket \rrbracket</code> (<i>StM</i>)

Blaue Symbole erfordern das `amsmath`-Paket oder, wenn sie mit ^(StM) gekennzeichnet sind, das `stmaryd`-Paket.

Synonyme: `[ \lbrack, [ \rbrack, ] { \lbrace, \{ } \rbrace, \}`

Tabelle 8.26: Symbolpaare der Klassen `\mathopen` und `\mathclose` (erweiterbar)

$\llcorner$	$\lrcorner$	$\llcorner$	$\lrcorner$	$\ulcorner$	$\urcorner$	
$\lceil$	$\rceil$	$\lceil$	$\rceil$	$\lceil$	$\rceil$	
$\lfloor$	$\rfloor$	$\lfloor$	$\rfloor$	$\lfloor$	$\rfloor$	
$\llcorner$	$\lrcorner$	$\llcorner$	$\lrcorner$	$\ulcorner$	$\urcorner$	

Symbole der ersten Zeile erfordern das amssymb-Paket, alle anderen das stmaryrd-Paket. Sie sind nicht erweiterbar.

Tabelle 8.27: Symbolpaare der Klassen `\mathopen` und `\mathclose` (nicht erweiterbar)

Die Pakete `amssymb` und `stmaryrd` fügen noch eine Zusammenstellung nicht erweiterbarer Symbolpaare der Klassen `Opening` und `Closing` hinzu, die in Tabelle 8.27 aufgelistet sind.

KAPITEL 9

L^AT_EX in einem mehrsprachigen Umfeld

9.1 T _E X und nicht englische Sprachen	555
9.2 Die babel-Benutzerschnittstelle	560
9.3 Sprachspezifische Benutzerbefehle	566
9.4 Unterstützung nicht lateinischer Alphabete.	585
9.5 Anpassen des babel-Systems	596
9.6 Andere Ansätze	607

Dieses Kapitel beginnt mit einer kurzen Erläuterung, welche technischen Probleme zu bewältigen sind, damit L^AT_EX Texte nicht englischer Sprachen verarbeiten kann. Der Großteil des verbleibenden Kapitels behandelt dann das babel-System, mit dessen Hilfe sich Texte verschiedenster Sprachen komfortabel verarbeiten lassen. Dabei liegt ein besonderes Augenmerk auf französischen, deutschen, russischen, griechischen und hebräischen Texten, da sich anhand dieser Sprachen sehr gut die unterschiedlichen Problematiken darstellen lassen, mit denen man beim Setzen von Texten in einer nicht englischen Umgebung konfrontiert wird. In Abschnitt 9.5 wird der Aufbau der Sprachdefinitionsdateien des babel-Systems für die verschiedenen Sprachoptionen erläutert. Abschließend folgen noch einige Hinweise zum Umgang mit anderen Sprachen wie z.B. Arabisch und Chinesisch, die nicht von babel unterstützt werden.

9.1 T_EX und nicht englische Sprachen

Aufgrund seiner Beliebtheit im akademischen Bereich hat T_EX schnell weltweit Fuß gefasst und wird heute nicht nur für Sprachen verwendet, die auf dem lateinischen Alphabet beruhen, sondern auch für nicht lateinische Schriften, wie etwa Russisch, Griechisch, Arabisch, Persisch, Hebräisch, Thai, Vietnamesisch und einige indische Sprachen. Es existieren außerdem Lösungen für die

chinesische, japanische und koreanische Sprache, die auf Kanji basierende, ideographische Schriften verwenden.

Mit Einführung der 8-Bit-Versionen von T_EX und METAFONT, die Donald Knuth offiziell im März 1990 veröffentlichte, wurden zum ersten Mal einfache Lösungen zur Unterstützung anderer Sprachen ermöglicht. Diese Versionen allein können jedoch auch nicht all die Probleme lösen, die auftreten, wenn man eine komfortable L^AT_EX-Umgebung anstrebt, mit der sich verschiedene und/oder nicht englische Sprachen verarbeiten lassen.

Um dieses Ziel zu erreichen, müssten T_EX und seine Partnerprogramme von Grund auf international aufgesetzt werden. Dabei sind die folgenden Punkte zu berücksichtigen:

1. Programme müssen auf die entsprechende(n) Sprache(n) anpassbar sein:
 - Unterstützen des Setzens in unterschiedlichen Schreibrichtungen. Diese Funktion wird von mehreren Programmen angeboten (z.B. eT_EX und Omega) [27, 82].
 - Erstellen passender Zeichensätze mit den landesüblichen Symbolen [140].
 - Definieren von Standards für Fontkodierungen.
 - Generieren von Trennmustern für den Silbentrennungsalgorithmus.
2. Übersetzen der sprachspezifischen Zeichenfolgen, Erstellen landesspezifischer Layouts für Standarddokumente und Programmieren sprachspezifischer Regeln zum automatischen Setzen der Texte mit T_EX [131].
3. Unterstützen der Verarbeitung mehrsprachiger Dokumente (mit mehr als einer Sprache innerhalb eines Dokumentes) und der Arbeit im internationalen Umfeld (eine Sprache je Dokument mit Auswahl unter mehreren Möglichkeiten). So sollte sich z.B. die Sortierung von Stichwörtern (im Index) und Literaturverweisen am Alphabet und der Sortierfolge der jeweiligen Sprache orientieren (vgl. die Erläuterungen zu xindy in Abschnitt 11.3).

Gleichzeitig sollte es möglich sein, Dokumente komfortabel in verschiedenen Eingabekodierungen zu bearbeiten, zu betrachten und auszudrucken, wobei L^AT_EX die so erzeugten Dokumente ohne Probleme verarbeiten sollte. Es gibt jedoch nahezu genauso viele unterschiedliche Eingabekodierungen wie Sprachen (IBM-PCs z.B. verfügen über Dutzende verschiedener Kodiertabellen). Zusätzlich existieren noch unterschiedliche nationale und internationale Standards, wie z.B. die ISO 8859-x-Serie [72]. Darum sollte man auch der Kompatibilität und der Portabilität einige Aufmerksamkeit widmen. Wenn ein Dokument in verschiedensten Umgebungen reproduzierbar sein soll, gewinnt die Standardisierung an Bedeutung. So konnte man z.B. eine Zeit lang 8-Bit-kodierte Dokumente nicht per E-Mail versenden, weil einige Mail-Gateways das Bit höherer Ordnung einfach unterschlugen, so dass sich die Dokumente nicht mehr verarbeiten ließen. Dieses Problem ist mittlerweile mehr oder weniger behoben, da fast alle Mail-Versender sich nach dem MIME-Standard (Multipart Internet Mail Extensions) richten, bei dem die jeweilige Kodierung (z.B. ISO 8859-x) explizit im Header der E-Mail angegeben wird. Es bleibt

jedoch dabei, dass man wissen muss, mit welcher Kodierung ein Dokument erstellt wurde. L^AT_EX bewältigt diese Aufgabe mithilfe des `inputenc`-Paketes, das in Abschnitt 7.11.3 auf Seite 456 beschrieben wurde.

Die Probleme der Dokumentenkodierung werden erst dann endgültig behoben sein, wenn neue Standards gelten, mit deren Hilfe sich nicht nur alphabetorientierte Sprachen, sondern auch ideographische Schriften wie Chinesisch, Japanisch und Koreanisch kodieren lassen. Zur Darstellung der Zeichen dieser Schriften reichen 8 Bit natürlich bei weitem nicht aus. Daher wurden bereits Mehrbyte-Standards für die elektronische Kodierung entwickelt, insbesondere „16-Bit“-Unicode, das eine Untermenge des Mehrbyte-Standards ISO 10646 [69, 70] bildet. Unicode wird höchstwahrscheinlich sehr bald die Standardkodierung der meisten Betriebssysteme sein. Darüber hinaus ist es auch das Herzstück der XML-Metasprache [26], auf der alle in letzter Zeit entwickelten Internet-Markup-Sprachen basieren. Dadurch lässt sich die Integrität elektronischer Dokumente sowohl hinsichtlich ihres Aufbaus als auch ihres Inhaltes in vollem Umfang garantieren. L^AT_EX unterstützt mit der Option `utf8` für das `inputenc`-Paket eine eingeschränkte Version der UTF-8-Darstellung von Unicode. Diese Option wurde in Abschnitt 7.5.2 besprochen.

Auf dem Treffen in Portland, Oregon, im Jahre 1992 richtete das Technical Council der TUG unter dem Namen Technical Working Group on Multiple Language Coordination (TWGMLC) eine Arbeitsgruppe zur Koordinierung der verschiedenen Sprachen ein, die von Yannis Haralambous geleitet wird. Diese Gruppe erhielt den Auftrag, die Standardisierung und Entwicklung von sprachspezifischer T_EX-Software zu fördern und zu koordinieren. Sie hatte das Ziel, für jede Sprache oder Sprachgruppe ein Paket herauszubringen, mit dem sich das Setzen von Dokumenten erleichtern ließ. Ein solches Paket sollte Einzelheiten über Zeichensätze, Eingabekonventionen und Trennmuster enthalten, sowie eine L^AT_EX-Optionsdatei, die mit dem `babel`-Konzept kompatibel ist, und unter Umständen einen Präprozessor; und natürlich sollte eine Dokumentation in Englisch und in der Zielsprache vorhanden sein.

9.1.1 Sprachspezifische Aspekte der Formatierung

Um die Formatierung von Dokumenten in nicht englischen Sprachen zu unterstützen, muss man einige Aspekte berücksichtigen.

An allererster Stelle steht hier, dass in verschiedenen Sprachen die Silbentrennung unterschiedlich geregelt ist. Dieser Tatsache wird T_EX gerecht, indem es die Verwendung mehrerer Trennmuster unterstützt. In manchen Sprachen verändern sich jedoch bestimmte Buchstabenkombinationen an den Trennstellen. Diese Funktion wird von T_EX nicht automatisch unterstützt.

Manche Sprachen benötigen spezielle Zeichensätze zur korrekten Formatierung ihrer Texte. Dabei kann es sich um zusätzliche „Akzentbuchstaben“ handeln (wie es bei vielen europäischen Sprachen der Fall ist) oder gar um ein völlig anderes Alphabet (wie z.B. bei Sprachen, die das kyrillische oder griechische Alphabet verwenden). Bei außereuropäischen Sprachen kann zudem die Schreibrichtung abweichen (wie z.B. von rechts nach links bei arabischen und hebräischen Texten) oder es können so viele Zeichen erforderlich sein, dass die T_EX-Standardmechanismen damit nicht mehr zurecht kommen (wie im Falle der Kanji-Schrift).

Ein eher „unterschwelliges“ Problem zeigt sich angesichts der Standarddokumentenklassen, die Bestandteil jeder L^AT_EX-Distribution sind: Sie wurden für den angloamerikanischen Raum gestaltet. Ein besonderes Beispiel, bei dem Konflikte mit anderen Sprachen auftreten, ist der Kapitelanfang. In manchen Sprachen reicht es nicht, einfach nur das Wort „Chapter“ zu übersetzen. Zusätzlich erscheinen die Elemente der Bezeichnung in anderer Reihenfolge und auch die Nummerierung muss, einzig aufgrund der grammatischen Regeln, geändert werden. Wo der englische Leser den Text „Chapter 1“ erwartet, möchte der französische Leser „1^{er} Chapitre“ sehen.

9.1.2 Kulturspezifische Aspekte der Formatierung

Noch unangenehmer für die Unterstützung verschiedener Sprachen ist die Tatsache, dass es, selbst in Ländern mit gleicher Sprache, unterschiedliche Formatierungsregeln gibt. So unterscheiden sich z.B. die Regeln für die Silbentrennung in britischem und amerikanischem Englisch. Englische Begriffe können von Land zu Land unterschiedlich zu übersetzen sein, wie es z.B. für das in Deutschland gesprochene Deutsch und das in Österreich gesprochene (und geschriebene) Deutsch gilt.

Und auch die typographischen Regeln können sich von Land zu Land unterscheiden. Es gibt keine weltweiten Standards zur Formatierung verschachtelter Listen – ganz im Gegenteil, sie können sich je nach Sprache, nach Land und sogar nach Verlag unterscheiden. Diese Gesichtspunkte führen zu der fließenden Grenze zwischen sprachlichen und kulturellen Aspekten der Formatierung. Diese Grenze lässt sich nicht eindeutig bestimmen. Wenn man jedoch Lösungen für die Formatierung von Dokumenten einer bestimmten Sprache entwickelt, muss man diesen Unterschied berücksichtigen. Die sprachspezifischen Aspekte lassen sich auf einer allgemeinen Ebene unterstützen, kulturellen Aspekten dagegen wird man meistens besser (und einfacher) durch spezielle Dokumentenklassen gerecht.

9.1.3 Babel – L^AT_EX spricht mehrere Sprachen

Die L^AT_EX-Distribution enthält ein paar Standarddokumentenklassen, die von den meisten Anwendern benutzt werden. Diese Klassen (`article`, `report`, `book` und `letter`) haben einen gewissen amerikanischen Charakter, der nicht jedem zusagt. Außerdem werden sprachspezifische Zeichenfolgen, wie etwa „Chapter“ und „Table of Contents“ entsprechend der Voreinstellungen immer in englischer Sprache ausgegeben (Tabelle 9.2 auf Seite 561 enthält eine Liste von Befehlen mit sprachspezifischen Texten).

Das `babel`-Paket von Johannes Braams [25] bietet eine Reihe von Optionen, mit deren Hilfe man die Sprache oder Sprachen wählen kann, in denen ein Dokument gesetzt wird. Das Paket hat die folgenden Eigenschaften:

- Es können gleichzeitig mehrere Sprachen verwendet werden.
- Die Trennmuster, die geladen werden, wenn INIT_EX das L^AT_EX-Format erzeugt, lassen sich dynamisch über eine externe Datei definieren.

<i>Sprache</i>	<i>Option</i>	<i>Sprache</i>	<i>Option</i>
Bahasa	bahasa	Kroatisch	croatian
Baskisch	basque	Latein	latin
Bretonisch	breton	Niederländisch	dutch, afrikaans
Bulgarisch	bulgarian	Niedersorbisch	lowersorbian
Dänisch	danish	Nordsamisch	samin
Deutsch	german (<i>germanb</i>), ngerman, austrian, naustrian	Norwegisch	norsk, nynorsk
Englisch	english, USenglish, (<i>american, canadian</i>), UKenglish (<i>british</i>), australian (<i>newzealand</i>)	Obersorbisch	uppersorbian
Esperanto	esperanto	Polnisch	polish
Estnisch	estonian	Portugiesisch	portuges (<i>portuguese</i>), brazilian (<i>brazil</i>)
Finnisch	finnish	Rumänisch	romanian
Französisch	french (<i>frenchb, francais</i> , <i>acadian, canadien</i>)	Russisch	russian
Galizisch	galician	Schottisch-Gälisch	scottish
Griechisch	greek, polutonikogreek	Schwedisch	swedish
Hebräisch	hebrew	Serbisch	serbian
Interlingua	interlingua	Slowakisch	slovak
Irisch-Gälisch	irish	Slowenisch	slovene
Isländisch	icelandic	Spanisch	spanish
Italienisch	italian	Tschechisch	czech
Katalanisch	catalan	Türkisch	turkish
		Ukrainisch	ukrainian
		Ungarisch	magyar (<i>hungarian</i>)
		Walisisch	welsh

Optionen in Klammern sind Alternativbezeichnungen für die vorhergehende Option.

Optionen für die gleiche Sprache unterscheiden sich typischerweise in ihren Regeln für die Silbentrennung, in Datumskonventionen oder sprachspezifischen Zeichenfolgen.

Die Option english kombiniert amerikanische Trennmuster mit einem britischen Datumsformat.

Tabelle 9.1: Vom babel-System unterstützte Sprachoptionen

- Für mehr als 40 Sprachen werden übersetzte sprachspezifische Zeichenfolgen und Befehle bereitgestellt, um die Texteingabe zu erleichtern (siehe Tabelle 9.1).

Im nächsten Abschnitt folgt eine Beschreibung der Benutzerschnittstelle des babel-Systems. Danach werden die zusätzlichen Befehle für verschiedene Sprachen behandelt, sowie die Unterstützung für das Setzen von Sprachen, die nicht das lateinische Alphabet nutzen. Abschließend wird besprochen, wie man babel an seine eigenen Bedürfnisse anpassen kann, und wie die *Sprachdefinitionsdateien* (.ldf), welche die sprachspezifischen Befehle für babel einrichten, im Einzelnen aufgebaut sind. In allen Abschnitten veranschaulichen ausführliche Beispiele den Umgang mit den verschiedenen Sprachen, die von babel unterstützt werden.

9.2 Die babel-Benutzerschnittstelle

Beim Laden des babel-Paketes sollte jede Sprache, die in dem betreffenden Dokument vorkommt, als Option deklariert werden. Alternativ dazu lassen sich die Sprachen, weil sie eine generelle Eigenschaft des Dokumentes darstellen, auch als *globale Optionen* des Befehls `\documentclass` festlegen. Dadurch werden sie jedem Paket zur Verfügung gestellt, das mit den Spracheinstellungen des Dokumentes sein Verhalten ändert. Die zurzeit unterstützten Optionen sind in Tabelle 9.1 auf der vorherigen Seite aufgezählt. Die folgende Deklaration z.B. bereitet das Setzen eines Dokumentes in den Sprachen Deutsch (Option `ngerman` für die neuen Regeln der Silbentrennung) und Italienisch (Option `italian`) vor:

```
\usepackage[ngerman,italian]{babel}
```

Die letzte Sprache in der Befehlszeile des `\usepackage`-Befehls wird am Anfang des Dokumentes verwendet. Das Dokument im vorigen Beispiel beginnt mit den italienischen Einstellungen für sprachspezifische Texte, Trennmuster und gegebenenfalls einige sprachspezifische Befehle (wie etwa das Datum). Sie gelten bis zu dem Punkt, an dem eine andere Sprache ausgewählt wird. Die Trennmuster werden allerdings nur aktiviert, wenn sie beim Erzeugen des L^AT_EX-Formats durch INIT_EX für die jeweilige Sprache geladen wurden (vgl. die Erläuterungen auf Seite 596).

Wenn man `ngerman` und `italian` als globale Optionen verwendet, erhalten auch andere Pakete diese Information. Durch die folgenden Codezeilen z.B. kann das Paket `varioref` (das in Abschnitt 2.4.2 auf Seite 72 beschrieben wurde) die Optionen des `\documentclass`-Befehls wahrnehmen und nutzen:

```
\documentclass[ngerman,italian]{article}
\usepackage{babel}
\usepackage{varioref}
```

Wer ein Dokument mit mehr als einer Sprache erstellt und eigene sprachspezifische Zeichenfolgen für die `varioref`-Befehle definieren möchte, sollte dies nach den in Abschnitt 9.5 auf Seite 596 beschriebenen Methoden tun und nicht mithilfe der in Abschnitt 2.4.2 besprochenen.

Die meisten in L^AT_EX verfügbaren Dokumentenklassen definieren eine Reihe von Befehlen zum Speichern von sprachspezifischen Texten. Tabelle 9.2 auf der nächsten Seite zeigt eine Übersicht über diese Befehle und ihre voreingestellten Textinhalte.

*Sprachspezifische
Zeichenfolgen*

9.2.1 Einstellen oder Abfragen der aktuellen Sprache

Innerhalb eines Dokumentes kann man die aktuelle Sprache auf verschiedene Weise ändern. So lassen sich z.B. alle sprachabhängigen Einstellungen einschließlich der Übersetzungen generierter Texte, wie etwa „Chapter“, die Formatierungskonventionen und die Konfiguration der Kurzformen ändern. Oder man behält die Übersetzungen bei und ändert alles andere (wenn man z.B. nur kurze Passagen des Haupttextes in einer anderen Sprache setzt). Und schließlich kann man auch einfach nur die Regeln für die Silbentrennung ändern.

<i>Befehl</i>	<i>Voreinstellung</i>	<i>Befehl</i>	<i>Voreinstellung</i>
<code>\abstractname</code>	Abstract	<code>\indexname</code>	Index
<code>\alsoname</code>	see also	<code>\listfigurename</code>	List of Figures
<code>\appendixname</code>	Appendix	<code>\listtablename</code>	List of Tables
<code>\bibname</code>	Bibliography	<code>\pagename</code>	Page
<code>\ccname</code>	cc	<code>\partname</code>	Part
<code>\chaptername</code>	Chapter	<code>\prefacename</code>	Preface
<code>\contentsname</code>	Contents	<code>\proofname</code>	Proof
<code>\enclname</code>	encl	<code>\refname</code>	References
<code>\figurename</code>	Figure	<code>\seename</code>	see
<code>\glossaryname</code>	Glossary	<code>\tablename</code>	Table
<code>\headtoname</code>	To (letter class)		

Tabelle 9.2: Sprachspezifische Texte in babel (Englische Voreinstellungen)

`\selectlanguage{sprache}`

Alle sprachspezifischen Einstellungen lassen sich in einem Schritt mithilfe des Befehls `\selectlanguage` ändern. Wenn man z.B. ins Deutsche wechseln möchte, würde man den Befehl `\selectlanguage{ngerman}` verwenden. Auf die gleiche Weise kann man auch auf andere Sprachen umschalten. Dazu muss, wie bereits erklärt, jede der gewählten Sprachen zuvor in der Präambel als Sprachoption deklariert worden sein. Der Befehl `\selectlanguage` ruft die Makros der Sprachdefinitionsdatei auf (siehe Abschnitt 9.5) und aktiviert die spezifischen Definitionen für die besagte Sprache. Außerdem aktualisiert er die Einstellung des TeX-Basisbefehls `\language`, der für die Silbentrennung benutzt wird.

`\begin{otherlanguage}{sprache}`

Die Umgebung `otherlanguage` funktioniert genauso wie die Deklaration `\selectlanguage`, mit dem Unterschied, dass der Sprachwechsel nur innerhalb der Umgebung gilt. Zum Kombinieren wechselnder Schreibrichtungen von links nach rechts und von rechts nach links ist diese Umgebung eine zwingende Voraussetzung. Das Argument *sprache* gibt die Sprache an, zu der man umschalten möchte.

`\foreignlanguage{sprache}{phrase}`
`\begin{otherlanguage*}{sprache} text \end{otherlanguage*}`

Der Befehl `\foreignlanguage` formatiert den Inhalt des Argumentes *phrase* nach den Regeln der festgelegten *sprache*. Er wechselt zur Silbentrennung der Sprache und aktiviert ihre Spezialdefinitionen; die Datumskonventionen

und die sprachspezifischen Texte werden *nicht* geändert. Die entsprechende Umgebung ist `otherlanguage*`.

Die festen Bezeichnungen für Dokumentenelemente sind von der jeweiligen Sprache abhängig: in Deutsch erhalten wir “Literatur” oder “Kapitel”. In English we get “References” or “Chapter”. Voici en français : “Références” ou “Chapitre”. But in short phrases “Références” does not change!

```
\usepackage[english,french,ngerman]{babel}
\raggedright

Die festen Bezeichnungen für Dokumentenelemente
sind von der jeweiligen Sprache abhängig:
in Deutsch erhalten wir “\refname” oder
“\chaptername”. \par
\selectlanguage{english} In English we get
“\refname” or “\chaptername”. \par
\begin{otherlanguage}{french} Voici en
fran\c cais: “\refname” ou “\chaptername”.
\foreignlanguage{english}{But in short phrases
“\refname” does not change!}
\end{otherlanguage}
```

Bsp.
9-2-1

```
\begin{hyphenrules}{sprache}
```

Für den Inhalt der Umgebung `hyphenrules` ändern sich *nur* die Regeln für die Silbentrennung entsprechend der angegebenen *sprache*; `\language` und alle anderen Einstellungen werden beibehalten. Wenn im Format keine Trennmuster für die *sprache* geladen wurden, hat die Umgebung keinerlei Auswirkungen.

Mithilfe dieser Umgebung kann man auch jegliche Silbentrennung vermeiden, sofern in der Datei `language.dat` die „Sprache“ `nohyphenation` definiert ist. Dies geschieht, wie in Abschnitt 9.5.1 auf Seite 596 erklärt, durch Laden der Datei `zerohyph.tex`.

Dieser Text zeigt Auswirkungen der Silbentrennung.
Dieser Text zeigt Auswirkungen der Silbentrennung.

```
\usepackage[ngerman]{babel}
\begin{minipage}{4cm}
Dieser Text zeigt Auswirkungen der Silbentrennung.\par
\begin{hyphenrules}{nohyphenation}
Dieser Text zeigt Auswirkungen der Silbentrennung.
\end{hyphenrules}
\end{minipage}
```

Bsp.
9-2-2

Man beachte, dass dieses Verfahren auch funktioniert, wenn die „Sprache“ `nohyphenation` dem `babel`-Paket nicht als Option mitgegeben wurde.

Beim Einsatz mehrerer Sprachen muss man manchmal wissen, welche Sprache an einer bestimmten Position im Dokument gerade aktiv ist. Das lässt sich durch Aufrufen von `\iflanguage` überprüfen:

```
\iflanguage{sprache}{wahr-anweisung}{falsch-anweisung}
```

Zunächst wird anhand des ersten Argumentes (*sprache*) überprüft, ob die angegebene Sprache überhaupt für `babel` deklariert wurde. Ist sie bekannt, so vergleicht der Befehl sie mit der aktuellen Sprache. Wenn beide übereinstimmen, werden die Befehle der *wahr-anweisung* ausgeführt. Ansonsten werden die Befehle im dritten Argument, der *falsch-anweisung*, ausgeführt.

Dieser Schritt vergleicht eigentlich die `\l@{sprache}`-Befehle miteinander, die auf die Trennmuster für die beiden Sprachen verweisen (siehe Abschnitt 9.5.1 auf Seite 596). Daher werden zwei „Sprachen“ dann als identisch angesehen, wenn sie die gleichen Muster verwenden (zum Beispiel Dialekte¹ einer Sprache, wie etwa `austrian`), insbesondere Sprachen, für die keine Muster geladen sind.

Bsp.
9-2-3

Deutsch und Österreichisch verwenden die gleichen und Englisch und Österreichisch verschiedene Trennmuster.

```
\usepackage[english,naustrian,ngerman]{babel}
Deutsch und Österreichisch verwenden
\iflanguage{naustrian}{die gleichen}{verschiedene} und
\foreignlanguage{english}{Englisch und Österreichisch
\iflanguage{naustrian}{die gleichen}{verschiedene}}
Trennmuster.
```

`\language`

Der Befehl `\language` gibt den Namen der aktuellen Sprache aus.

Bsp.
9-2-4

- (1) Die Sprache ist english.
- (2) Die Sprache ist german.
- (3) Die Sprache ist french.
- (4) Die Sprache ist english.
- (5) Pas en français.
- (6) Die Sprache ist german.

```
\usepackage[german,french,english]{babel}
\par(1) Die Sprache ist \language.
\par(2) \selectlanguage{german}%
        Die Sprache ist \language.
\par(3) \begin{otherlanguage}{french}
        Die Sprache ist \language.
        \end{otherlanguage}
\par(4) \foreignlanguage{english}{%
        Die Sprache ist \language.}
\par(5) \iflanguage{french}{En fran\c cais.}
        {Pas en fran\c cais.}
\par(6) Die Sprache ist \language.
```

9.2.2 Vom Umgang mit Kurzformen

Autoren, die nicht in englischer Sprache schreiben, müssen die Buchstaben für die Sprache ihres Dokumentes oftmals auf recht umständliche Weise eingeben. Sie benötigen relativ häufig Buchstaben mit Akzenten ober- oder unterhalb – manchmal sogar mit mehreren gleichzeitig. Wenn man solche Glyphen erzeugen muss und nicht mit einer 8-Bit- sondern nur mit einer 7-Bit-Kodierung arbeiten kann, ist eine Vereinfachung für die Eingabe der Zeichenbefehle höchst willkommen. Darum (und aus weiteren Gründen, die später noch besprochen werden) unterstützt babel das Konzept von „Kurzformen“. Eine „Kurzform“ besteht aus einer Eingabe von ein oder zwei Zeichen, wobei das erste Zeichen, das so genannte „Kurzformzeichen“, die Kurzform einleitet. Bei einer Kurzform, die aus zwei Zeichen besteht, legt das zweite Zeichen das Verhalten der Kurzform fest.

Babel kennt drei Arten von Kurzformen: „system“- , „sprach“- und „anwender“-spezifische. Eine systemspezifische Kurzform kann durch eine

¹Nur hinsichtlich ihrer babel-Umsetzung! Dort werden aus rein T_EXnischen Gründen einige Sprachen als „Dialekte“ anderer Sprachen behandelt; das ist in keiner Weise als Diskriminierung gedacht.

andere für die jeweilige Sprache außer Kraft gesetzt werden; eine sprachspezifische Kurzform wiederum kann durch eine vom Anwender definierte Kurzform übergangen werden.

Befehle für Kurzformen auf Dokumentenebene

Dieser Abschnitt beschäftigt sich mit Kurzformbefehlen, die im Dokument verwendet werden können, sowie verschiedenen Gesichtspunkten des Kurzformkonzeptes. Kurzformen auf Sprach- oder Systemebene werden in Sprachdefinitionsdateien deklariert (siehe Abschnitt 9.5 auf Seite 596).

```
\usesshorthands{zeichen}
```

Der Befehl `\usesshorthands` leitet die Definition von anwenderspezifischen Kurzformen ein. Das Argument *zeichen* ist das Zeichen, mit dem die Kurzform beginnt.

```
\defineshorthand{kurzform}{langform}
```

Mit dem Befehl `\defineshorthand` wird eine Kurzform definiert. Sein erstes Argument, *kurzform*, besteht aus ein oder zwei Zeichen, und das zweite Argument, *langform*, ist der Code, den die Kurzform erzeugen soll.

```
\aliasshorthand{zeichen1}{zeichen2}
```

Mithilfe des Befehls `\aliasshorthand` kann man das Zeichen *zeichen2* als Alternative für ein existierendes Kurzformzeichen *zeichen1* festlegen. Wenn man z.B. anstelle von " lieber das Zeichen | verwendet, kann man dies über den Befehl `\aliasshorthand{"}{|}` angeben.

Im nächsten Beispiel werden obige Deklarationen benutzt um einige Kurzformen zur englischen Sprachoption hinzuzufügen.

```
\usepackage[english]{babel}      \usesshorthands{"}
\defineshorthand{"a"}{"{a}}      \defineshorthand{"i"}{"{\i}}
\aliasshorthand{""}{|}
```

Use and effect of "a: ä
and "i: ï.

Use and effect of |a: ä
and |i: ï.

Use and effect of \verb="a=: "a and \verb="i=: "i.

Use and effect of \verb|=a=: |a and \verb|=i=: |i.

Bsp.
9-2-5

```
\languageshorthands{sprache}
```

Mithilfe des Befehls `\languageshorthands` kann man auf die Kurzformen der als Argument angegebenen *sprache* umschalten. Dabei muss die *sprache* für `babel` im aktuellen Dokument deklariert sein. Beim Wechsel von einer Sprache zur anderen geben die Sprachdefinitionsdateien normalerweise diesen Befehl für die entsprechende Sprache aus. Die Datei `frenchb.1df` z.B. enthält z.B. die Deklaration `\languageshorthands{french}`.

Hin und wieder muss man die Kurzformfunktion eines bestimmten Zeichens vorübergehend abschalten, da es in seiner ursprünglichen Bedeutung benötigt wird. Dafür existieren zwei Deklarationen.

<code>\shorthandon{zeichen}</code>	<code>\shorthandoff{zeichen}</code>

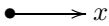
Der Befehl `\shorthandoff` setzt den `\catcode` für jedes Zeichen seines Argumentes *zeichen* auf „other“ (Sonstige, 12). Der Befehl `\shorthandon` hingegen setzt den `\catcode` für die Zeichen seines Argumentes *zeichen* auf „active“ (13). Beide Befehle wirken sich nur auf „bekannte“ Kurzzeichen aus. Ist ein Zeichen nicht als Kurzzeichen bekannt, so bleibt seine Kategorie unverändert.

Die Sprachdefinitionsdatei `german.ldf` kennt z.B. die beiden Befehle `\mdqoff` und `\mdqon`, die das Kurzformverhalten des Zeichens " aus- bzw. einschalten. Sie sind folgendermaßen definiert:

```
\newcommand\mdqon{\shorthandon{"}}
\newcommand\mdqoff{\shorthandoff{"}}
```

Die Sprachdefinitionsdatei für die französische Sprache (`frenchb.ldf`) aktiviert die „doppelten“ Satzzeichen „?“, „!“, „:“ und „;“. Dieses Verhalten lässt sich abschalten, indem man die Zeichen dem Befehl `\shorthandoff` im Argument übergibt. Dieser Schritt ist bei manchen Paketen erforderlich, in denen diese Zeichen eine besondere Bedeutung erhalten. Im nächsten Beispiel, in Kombination mit dem `xy`-Paket, werden die Zeichen „;“ und „?“ als Kurzzeichen deaktiviert, da sie in der `xy`-Umgebung des Paketes [58, Kapitel 5] eine besondere Funktion ausüben.

Voici un exemple avec *xy-pic* :



Quelle belle flèche !

Bsp.
9-2-6

```
\usepackage{xy} \usepackage[french]{babel}
Voici un exemple avec \emph{xy-pic}:
\[ \shorthandoff{?;}
\begin{xy} (0,0)*{\bullet}, (0,0) ; (10,0),
**\dir {-} ?>* \dir {>}, (12,0)*{x}, \end{xy}
\]
Quelle belle flèche !
```

9.2.3 Sprachattribute

Hin und wieder müssen sprachspezifische Lösungen an unterschiedliche Bedingungen angepasst werden. In diesem Fall lassen sich Attribute für die jeweilige Sprache definieren. Bei der Unterstützung zum Setzen lateinischer Texte finden sich zwei solche Beispiele. Bei Auswahl des Attributes `medieval` werden die Bezeichnungen bestimmter Dokumentenelemente anders geschrieben und die Buchstaben „u“ und „v“ werden zu einem Klein- und Großbuchstabenpaar zusammengefasst. Das Attribut `withprosodicmarks` findet in Grammatiken, Wörterbüchern, Lehrtexten und ähnlichen Werken Verwendung, in denen Betonungszeichen für eine vollständige Beschreibung von Wörtern oder Versen unerlässlich sind. Es stellt besondere Kurzformen für die Akzente Brevis und Macron bereit, die in Kombination mit anderen Paketen zu Konflikten führen können.

<code>\languageattribute{sprache}{sprachattr}</code>

Der Befehl `\languageattribute` legt fest, welche Attribute für die jeweilige Sprache gelten. Er muss in der Dokumentenpräambel nach dem Befehl

`\usepackage[...]{babel}` stehen, der das `babel`-Paket lädt. Der Befehl besitzt zwei Argumente: Das Argument *sprache* enthält den Namen der Sprache und *sprachattr* enthält eine durch Kommas unterteilte Liste von Attributen für diese Sprache. Der Befehl überprüft, ob die gegebene Sprache im aktuellen Dokument bekannt ist und ob die Attribute für diese Sprache zur Verfügung stehen.

So verfügt `babel` z.B. über zwei Varianten für die griechische Sprache: `monotoniko` (nur ein Akzent), die Voreinstellung, und `polutoniko` (mit verschiedenen Akzenten). Die Variante `polutoniko` wird in der Dokumentenpräambel mit dem Befehl `\languageattribute` ausgewählt. Die nächsten beiden Beispiele machen den Unterschied deutlich.

Das griechische Wort für „Index“ ist Εἰσαγωγή.

```
\usepackage[greek,ngerman]{babel}
Das griechische Wort für "‘Index'" ist
\begin{otherlanguage}{greek}\indexname\end{otherlanguage}.
```

Bsp.
9-2-7

Mit dem Attribut `polutoniko` erhält man ein anderes Ergebnis:

Das griechische Wort für „Index“ ist Εἰσαγωγή.

```
\usepackage[greek,ngerman]{babel}
\languageattribute{greek}{polutoniko}
Das griechische Wort für "‘Index'" ist
\begin{otherlanguage}{greek}\indexname\end{otherlanguage}.
```

Bsp.
9-2-8

9.3 Sprachspezifische Benutzerbefehle

Dieser Abschnitt bietet einen allgemeinen Überblick darüber, welche Funktionen üblicherweise von den verschiedenen Sprachoptionen angeboten werden. Dazu gehören Übersetzungen für sprachspezifische Texte sowie eine Übersicht über typische Kurzformen zur Erleichterung der Eingabe oder zur Erfüllung besonderer Formatierungsaufgaben. Einige Sprachoptionen definieren überdies zusätzliche Befehle für spezielle Datums- oder Zahlenformate. Des Weiteren werden Layoutanpassungen besprochen, wie sie für französische und hebräische Texte erfolgen, sowie Schnittstellen zur Handhabung unterschiedlicher Schriften (z.B. der lateinischen und der kyrillischen) im gleichen Dokument.

9.3.1 Übersetzungen

Wie bereits zuvor besprochen, stellt `babel` Übersetzungen für die Bezeichnungen von Dokumentelementen zur Verfügung, die L^AT_EX in seinen Dokumentenklassen verwendet. Die englischen Versionen dieser Texte sind in Tabelle 9.2 auf Seite 561 aufgeführt. Die Tabelle 9.3 auf Seite 567 zeigt die entsprechenden Übersetzungen für eine Reihe weiterer Sprachen, von denen eine nicht auf der lateinischen Schrift basiert.

Will man einzelne Übersetzungen für eine Sprache ändern, geschieht dies mit dem `\addto`-Befehl. Die zu ändernden sprachspezifischen Texte werden

Befehl	Deutsch	Französisch	Polnisch	Russisch
<code>\abstractname</code>	Zusammenfassung	Résumé	Streszczenie	Аннотация
<code>\alsoname</code>	siehe auch	<i>voir aussi</i>	Porównaj także	см. также
<code>\appendixname</code>	Anhang	Annexe	Dodatek	Приложение
<code>\bibname</code>	Literaturverzeichnis	Bibliographie	Bibliografia	Литература
<code>\ccname</code>	Verteiler	Copie à	Kopie:	исх.
<code>\chaptername</code>	Kapitel	Chapitre	Rozdział	Глава
<code>\contentsname</code>	Inhaltsverzeichnis	Table des matières	Spis treści	Содержание
<code>\enclname</code>	Anlage(n)	P. J.	Załącznik	вкл.
<code>\figurename</code>	Abbildung	FIG.	Rysunek	Рис.
<code>\glossaryname</code>	Glossar	Glossaire	Glossary	Glossary
<code>\headtoname</code>	An		Do	вх.
<code>\indexname</code>	Index	Index	Indeks	Предметный указатель
<code>\listfigurename</code>	Abbildungsverzeichnis	Table des figures	Spis rysunków	Список иллюстраций
<code>\listtablename</code>	Tabellenverzeichnis	Liste des tableaux	Spis tablic	Список таблиц
<code>\pagename</code>	Seite	page	Strona	с.
<code>\partname</code>	Teil	Deuxième partie	Część	Часть
<code>\prefacename</code>	Vorwort	Préface	Przedmowa	Предисловие
<code>\proofname</code>	Beweis	Démonstration	Dowód	Доказательство
<code>\refname</code>	Literatur	Références	Literatura	Список литературы
<code>\seename</code>	siehe	<i>voir</i>	Porównaj	см.
<code>\tablename</code>	Tabelle	TAB.	Tablica	Таблица

Bsp.
9-3-1

Im Französischen gibt `\partname` auch die Nummern der Teile als Wörter aus, wie z.B. „*Première, Deuxième, ...*“

Tabelle 9.3: Sprachspezifische Texte in babel (Deutsch, Französisch, Polnisch, Russisch)

damit dem Parameter `\captions{sprache}` hinzugefügt, siehe auch Seite 605.

`\tablename` erzeugt jetzt im Deutschen das Wort *Tafel*, aber im Österreichischen immer noch *Tabelle*.

```

\usepackage[austrian,ngerman]{babel}
\addto\captionsngerman{\renewcommand\tablename{Tafel}}
\verb=\tablename= erzeugt jetzt im Deutschen das Wort
\tablename{}, aber im \selectlanguage{austrian}%
Österreichischen immer noch \tablename.

```

Bsp.
9-3-2

Abgesehen von den übersetzten Texten in Tabelle 9.3 verfügen die Sprachdefinitionsdateien, wie im nächsten Beispiel gezeigt wird, normalerweise auch über alternative Versionen des Befehls `\today`.

Das Datum „29. Februar 2004“, wird in Amerika „February 29, 2004“ in England „29th February 2004“ und in Bulgarien „29 февруари 2004 г.“ geschrieben. In Katalonien schreibt man „29 de febrer de 2004“.

Bsp.
9-3-3

```

\usepackage[catalan,bulgarian,
             american,british,ngerman]{babel}
\raggedright
Das Datum "\today", wird in Amerika
"{\selectlanguage{american}\today}" in England
"{\selectlanguage{british}\today}" und in Bulgarien
"{\selectlanguage{bulgarian}\today}" geschrieben. In
Katalonien schreibt man
"{\selectlanguage{catalan}\today}".

```

9.3.2 Verfügbare Kurzformen

Viele Sprachdefinitionsdateien stellen Kurzformen zur Verfügung. Einige sollen die Eingabe erleichtern, andere hingegen dienen dem Erzeugen recht aufwendiger Spezialeffekte. Viele Anwender sind sich vielleicht gar nicht bewusst, dass L^AT_EX selbst auch eine Kurzform definiert (die allerdings nicht so genannt wird) und die sie sicherlich recht häufig verwenden: Es ist das Tilde-Zeichen (~), mit dem ein „geschützter“ Leerschritt eingegeben wird.

Viele Kurzformdefinitionen beziehen sich auf „Akzentbuchstaben“. Sie wurden zu jener Zeit erfunden, als T_EX weder 8-Bit-Eingaben noch 8-Bit-Trennmuster unterstützte. Sofern echte 8-Bit-Trennmuster zur Verfügung stehen, sind diese normalerweise zu bevorzugen. Mithilfe des `inputenc`-Paketes lässt sich dazu die passende Eingabekodierung auswählen (siehe Abschnitt 7.1.2 auf Seite 339). Sind jedoch bei Trennung direkt an einem Akzentbuchstaben besondere Verarbeitungsschritte erforderlich (wie etwa bei der Silbentrennung im Niederländischen), so kommt man nicht mehr ohne Kurzformen aus.¹

Anführungszeichen

Das beliebteste Kurzzeichen ist das Anführungszeichen ("). Es wird als solches in folgenden Sprachen verwendet: Baskisch, Bulgarisch, Dänisch, Deutsch, Estnisch, Finnisch, Galizisch, Isländisch, Italienisch, Katalanisch, Latein, Niederländisch, Norwegisch, Obersorbisch, Polnisch, Portugiesisch, Russisch, Schwedisch, Serbisch, Slowenisch, Spanisch und Ukrainisch. Es würde zu weit führen, alle Fälle zu beschreiben, in denen das Anführungszeichen als Kurzzeichen verwendet wird. Stattdessen empfiehlt es sich, die Einzelheiten zu den gewünschten Sprachen in der Dokumentation des `babel`-Paketes nachzulesen. An dieser Stelle lässt sich nur sagen, dass man die Verwendung der Zeichen verschiedenen Kategorien zuordnen kann, die alle im Folgenden anhand einiger Beispiele beschrieben werden.

Einfügen von Akzentbuchstaben Bei verschiedenen Sprachen wurden Kurzzeichen eingeführt um die Eingabe von Akzentbuchstaben zu erleichtern. Mit der Verbreitung der 8-Bit-Eingabe- und Ausgabekodierungen scheint dieser Bedarf nicht mehr zu bestehen, was jedoch nicht immer zutrifft. Im Niederländischen z.B. muss ein Akzent entfernt werden, wenn an dem entsprechenden Akzentbuchstaben eine Trennung erfolgt.

Den Koning van Hispaniën heb ik altijd ge-eerd! Den Koning van Hispaniën heb ik altijd geëerd!

```
\usepackage[dutch]{babel}
```

Den Koning van Hispani"en heb ik altijd ge"eerd! Den Koning van Hispani"en heb ik altijd ge"eerd!

Bsp.
9-3-4

Einfügen von Sonderzeichen Im Katalanischen benötigt man eine besondere Glyphe, das „Doppel-l“, zum korrekten Formatieren von Texten [168].

¹Das trifft nur zu, wenn T_EX als Formatierungsprogramm im Hintergrund eingesetzt wird. Omega z.B. verfügt über zusätzliche Funktionen, die solche Fälle automatisch verarbeiten können.

Im Falle einer Trennung fällt der Punkt zwischen den beiden „l“ weg.

Bsp.
9-3-5

Das „Doppel-l“ kommt in Wörtern wie *intelligència*, *il·lusió* vor.

```
\usepackage[catalan,ngerman]{babel}
```

Das "‘Doppel-l’" kommt in Wörtern wie `\foreignlanguage{catalan}{inte"lig\‘encia, i"lusi\‘o}` vor.

Dieses Zeichen lässt sich auch durch die Befehle `\lgem` und `\Lgem` erzeugen, oder mit den Kombinationen `\l.` und `\L.`, sobald die Option `catalan` ausgewählt ist.

Einfügen von besonderen Anführungszeichen $\LaTeX$ unterstützt entsprechend seiner Voreinstellung einzelne und doppelte Anführungsstriche: ‘Zitat’ und “Zitat”. Diese Versionen eignen sich nur für wenige europäische Sprachen, denn viele von ihnen haben ihre eigenen Richtlinien und benötigen meistens andere Zeichen. Die setzerische Tradition der Niederlande erfordert z.B., dass die öffnenden Anführungszeichen an der Grundlinie stehen, im Deutschen sind die schließenden Anführungszeichen umgekehrt und in Frankreich werden Guillemets (doppelte spitze Klammern) verwendet. Auch im Isländischen finden sich die Guillemets, allerdings seitenverkehrt, also mit den Spitzen nach „innen“ anstatt nach „außen“ (eine Schreibweise, die manchmal auch in der deutschen Typographie vorkommt).

Bsp.
9-3-6

Englische “Zitate” haben andere Anführungszeichen als niederländische „Zitate” oder deutsche „Zitate“ oder französische «Zitate».

```
\usepackage[dutch,french,ngerman]{babel}
```

Englische ‘‘Zitate’’ haben andere Anführungszeichen als `\selectlanguage{dutch}`niederländische “‘Zitate”’ oder `\selectlanguage{ngerman}`deutsche "‘Zitate”’ oder `\selectlanguage{french}`französische \og Zitate\fg.

Die Fontkodierung T1 enthält zwar Guillemets (siehe Tabelle 7.32 auf Seite 462), in französischen Texten werden sie jedoch mithilfe der Befehle `\og` und `\fg` erzeugt. Diese Befehle setzen die Guillemets nicht nur, sondern erzeugen auch die passenden Abstände zum eingeschlossenen Text.

Einfügen bei Sonderfällen der Silbentrennung In einigen Sprachen gelten besondere Regeln, wenn Zeilen an bestimmten Buchstaben umbrochen werden. So wurde z.B. in der alten deutschen Rechtschreibung die Buchstabenkombination `. . ck . .` als `. . k-k . .` getrennt und ein dreifaches `f` in einem zusammengesetzten Wort wurde normalerweise zu `ff` zusammengefasst, bei Trennungen jedoch wieder um das dritte `f` erweitert, wie im nächsten Beispiel zu sehen ist.

Bsp.
9-3-7

Brote bak-
ken

Farbstoff-
fabrik

```
\usepackage[german]{babel}
```

```
\fbox{\parbox[t]{1,5cm}{Brote ba"cken}} \quad
\fbox{\parbox[t]{1,5cm}{Farbsto"ffabrik}}
```

Einfügen besonderer Hinweise zu Trennungen Einige Kurzformen lassen sich verwenden, um $\LaTeX$ auf Besonderheiten bei der Kopplung von Buchstaben hinzuweisen. In manchen Sprachen muss man $\LaTeX$ z.B. am Setzen einer Ligatur hindern, etwa bei zusammengesetzten Wörtern. Das lässt sich mithilfe einer kleinen Unterschneidung erreichen, die zwischen den

Buchstaben, die normalerweise eine Ligatur bilden, eingefügt wird. Das entsprechende Kurzzeichen " | ist in vielen Sprachdefinitionen verfügbar.

```
\usepackage[ngerman]{babel}
```

Das deutsche Wort „Auflage“ sollte nicht so, sondern als »Auflage« gesetzt werden.

Das deutsche Wort "‘Auflage’" sollte nicht so, sondern als ">Auf"|lage"< gesetzt werden.

Bsp.
9-3-8

Ein weiteres beliebtes Kurzzeichen ist "-", das (genau wie \-) darauf hinweist, dass an dieser Stelle getrennt werden darf, ohne dass dadurch jedoch Trennungen im übrigen Wort (Silbenrätsel) unterdrückt werden:

```
\usepackage[dutch]{babel}
```

```
\fbox{\parbox[t]{1cm}{letter}-greepraadsel}}
\fbox{\parbox[t]{1cm}{letter}\-greepraadsel}}
\fbox{\parbox[t]{1cm}{lettergreepraadsel}}}
```

Bsp.
9-3-9

letter- greep- raadsel	letter- greepraadsel	lettergreepraadsel

Außerdem gibt es noch die folgenden Kurzformen: "" gleicht "-", ohne jedoch ein Divis-Zeichen auszugeben; "=" fügt einen expliziten Trennstrich ein, an dem umbrochen werden darf, während die verbundenen Wörter weiterhin separat getrennt werden können; und "~" fügt einen geschützten Bindestrich ein, an dem nicht umbrochen werden darf. Das nächste Beispiel demonstriert die Auswirkungen dieser Kurzzeichen anhand eines einzelnen Wortes.

1.	Gutenberg-Universität	Gutenberg-Universität
2.	GutenbergUniversität	Gutenberg-Universität
3.	GutenbergUniversität	GutenbergUniversität
4.	Gutenberg-Universi-tät	Gutenberg-Universität
5.	Gutenberg-Universität	Gutenberg-Universität

```
\usepackage[german]{babel}
```

```
\newcommand\present[1]{%
  \fbox{\parbox[t]{31mm}{#1}}
  \fbox{\parbox[t]{16mm}{#1}}
  \par}
```

1. \present{Gutenberg-Universit"at}
2. \present{Gutenberg"-Universit"at}
3. \present{Gutenberg""Universit"at}
4. \present{Gutenberg"=Universit"at}
5. \present{Gutenberg"~Universit"at}

Bsp.
9-3-10

Die Tilde

In den Sprachen Baskisch, Estnisch, Galizisch, Griechisch und Spanisch wird die Tilde nicht für ein geschütztes Leerzeichen, sondern für andere Zwecke verwendet.

- In der estnischen Typographie muss der Tilde-Akzent etwas niedriger gesetzt werden, als dies normalerweise durch L^AT_EX geschieht.
- Im griechischen Textsatz mit verschiedenen Akzenten muss die Tilde wie ein normaler Buchstabe behandelt werden, damit die Ligaturen der griechischen Zeichensätze fehlerfrei funktionieren.
- Im Baskischen, Galizischen und Spanischen kommt die Tilde in den Kurzzeichen ~n (ñ), ~N (Ñ) und ~- (besonderer Bindestrich) vor. Die Konstruktion ~- erzeugt (genau wie --- und ----) einen Bindestrich, an dem kein

Zeilenumbruch erfolgen darf. Wenn der Tilde irgendein anderes Zeichen folgt, behält sie ihre eigentliche Funktion als „geschützter Leerschritt“ (der zum Überlaufen der ersten Zeile im nächsten Beispiel führt). Wenn ein solcher Leerschritt vor einem „n“ benötigt wird, kann man dazu eine leere Gruppe einfügen (wie in der zweiten Zeile des Beispiels).

Bsp.
9-3-11

La eñe está presente en \alph y \Alph.
Como en castellano no se usan números
romanos en minúscula, \roman se redefi-
ne para que los dé en versalitas.

```
\usepackage[spanish,activeacute]{babel}
La e~ne est'a presente en
\verb|\alph|~y~\verb|\Alph|. Como en
castellano~{}no se usan n'umeros romanos en
min'uscula, \verb|\roman| se redefine para
que los d'e en versalitas.
```

Doppelpunkt, Semikolon, Ausrufezeichen und Fragezeichen

In den Sprachen Bretonisch, Französisch, Russisch und Ukrainisch dienen diese vier Zeichen als Kurzformen, mit deren Hilfe die korrekte Umsetzung typographischer Konventionen erleichtert wird. In der türkischen Typographie wird diese Funktion nur bei Doppelpunkt und Semikolon benötigt. Gemäß den jeweils gültigen Konventionen sollten diese Zeichen mit einem kleinen Abstand zum vorhergehenden Zeichen gesetzt werden.

Bsp.
9-3-12

En français on doit mettre un « petit
espace » devant la ponctuation double :
comme cela ! Wie hier zu sehen: Das gilt
nicht für die deutsche Sprache!

```
\usepackage[ngerman,french]{babel}
En fran\c{c}ais on doit mettre un \og petit
espace\fg\ devant la ponctuation double: comme
cela! \selectlanguage{ngerman}Wie hier zu
sehen: Das gilt nicht für die deutsche Sprache!
```

Der Abstand wird entsprechend der Voreinstellungen automatisch gesetzt. Diese Einstellungen lassen sich jedoch in einer Konfigurationsdatei ändern. Die Verwendung des Doppelpunktes als Kurzzeichen kann zu Konflikten mit anderen Paketen oder im Dokument eingebundenen PostScript-Dateien führen. In diesen Fällen muss man das Kurzzeichen gegebenenfalls (zeitweise) mithilfe des Befehls `\shorthandoff` deaktivieren, wie es in Beispiel 9-2-6 auf Seite 565 erklärt wurde.

Der Gravis-Akzent

Die Lösungen für die katalanische und die ungarische Sprache ermöglichen, den Gravis-Akzent (´) als Kurzzeichen zu verwenden.

- Im Katalanischen wird das Gravis-Zeichen nur unterstützt, wenn man babel mit der Option `activegrave` lädt. Das Kurzzeichen erleichtert die Eingabe von Akzentbuchstaben, welche die Silbentrennung nicht unterdrücken. Es lässt sich mit den Buchstaben a, e, o und A, E, O kombinieren.

Bsp.
9-3-13

„Pàgina, Apèndix, Pròleg“
sind katalanische Entsprechungen für „Seite, Anhang und Vorwort“.

```
\usepackage[catalan,activegrave,ngerman]{babel}
"'\foreignlanguage{catalan}{P'agina, Ap'endix, Pr'oleg}"
sind katalanische Entsprechungen für "'Seite, Anhang und
Vorwort"'.

```

- Im Ungarischen lässt sich dieses Kurzzeichen mit den groß- und kleingeschriebenen Versionen der Buchstaben c, d, g, l, n, s, t und z kombinieren. Es markiert bestimmte Stellen, an denen getrennt werden darf, um dort ein besonderes Verfahren für die korrekte Silbentrennung einzuleiten.

loccsan	locs- csan	eddzünk	edz- dzünk
poggyász	pogy- gyász	Kodály	Kodály-
mennyei	meny- nyi	vissza	visz- sza
pottyan	poty- tyan	rizzsel	rizs- zsel

```
\usepackage[hungarian]{babel}
\newcommand\present[1]{%
  \fbox{\parbox[t]{20mm}{#1}}
  \fbox{\parbox[t]{8,5mm}{#1}}\quad}

\present{lo'ccsan}
\present{e'ddz\"unk} \par
\present{po'ggy'asz}
\present{Kod\'a'llya}\par
\present{me'nnyei}
\present{vi'ssza} \par
\present{po'ttyan}
\present{ri'zzsel}
```

Bsp.
9-3-14

Der Akut-Akzent

Die Lösungen für die Sprachen Galizisch, Katalanisch und Spanisch ermöglichen, den Akut-Akzent (') als Kurzzeichen zu verwenden.

- Zum Formatieren katalanischer Texte lässt sich das Kurzzeichen mit den Vokalen (a, e, i, o, u), sowohl in Groß- als auch in Kleinschreibung, kombinieren. Es fügt den Akzent hinzu, ohne die Silbentrennung zu unterdrücken.
- In galizischen Texten funktioniert das Zeichen analog zum Katalanischen, nur dass außerdem die Kombination 'n das Zeichen ñ erzeugt.

„Páxina, Capítulo, Apéndice“
sind galizische Entsprechungen für
„Seite, Kapitel und Anhang“.

```
\usepackage[galician,activeacute,ngerman]{babel}
"\foreignlanguage{galician}{P'axina, Cap'itulo,
Ap'endice}" sind galizische Entsprechungen für
"Seite, Kapitel und Anhang".
```

Bsp.
9-3-15

- Bei der Unterstützung von Texten in spanischer Sprache bietet das Kurzzeichen die gleichen Funktionen wie im Katalanischen und Galizischen.

Die beschriebenen Funktionen werden durch die Option `activeacute` aktiviert. Diese Wahlmöglichkeit wird der Tatsache gerecht, dass der Akut-Akzent in L^AT_EX noch andere Funktionen hat, die nicht mehr zur Verfügung stehen, sobald das Zeichen als Kurzzeichen dient.

Der Zirkumflex

Die Lösungen für die Sprachen Esperanto und Latein ermöglichen, den Zirkumflex-Akzent (^) als Kurzzeichen zu verwenden.

- Zum Setzen von Texten in Esperanto sind zwei Akzente erforderlich: der Zirkumflex- und der Brevis-Akzent. Der Zirkumflex erscheint über den Buchstaben c, g, h, j und s, der Brevis über dem Buchstaben u. Beide

lassen sich mit Groß- und Kleinbuchstaben kombinieren. Der Zirkumflex ist als Kurzzeichen definiert, das die Silbentrennung bewahrt und den Zirkumflex-Akzent über dem Buchstaben „h“ etwas tiefer setzt (ĥ). In Kombination mit dem Buchstaben u setzt das Kurzzeichen einen Brevis-Akzent (^u wird zu ŭ); zusammen mit dem vertikalen Strich fügt es einen expliziten Trennstrich ein, der die Silbentrennung im übrigen Wort nicht unterdrückt.

Bsp.
9-3-16

„Paĝo, Ĉapitro, Citaĵoj“ sind Esperanto-Entsprechungen für „Seite, Kapitel und Quellen/Zitate“.

```
\usepackage[esperanto,ngerman]{babel}
"\foreignlanguage{esperanto}{Paĝo, Ĉapitro,
Citaĵoj}" sind Esperanto="Entsprechungen für
"Seite, Kapitel und Quellen/Zitate".
```

- Wird ein lateinischer Text gesetzt und das Attribut `withprosodicmarks` ist ausgewählt, dann erzeugt der Zirkumflex als Kurzzeichen einen Brevis-Akzent über kleingeschriebenen Vokalen (außer über den Ligaturen æ und œ). Die Silbentrennung bleibt davon unbeeinträchtigt.

Bsp.
9-3-17

```
\usepackage[latin]{babel} \languageattribute{latin}{withprosodicmarks}
ă ě ĭ ŏ ŭ \ProsodicMarksOn ^a ^e ^i ^o ^u
```

Das Gleichheitszeichen

Die Lösungen für die Sprachen Latein (mit ausgewähltem `withprosodicmarks`-Attribut) und Türkisch erlauben es, das Gleichheitszeichen (=) als Kurzzeichen zu verwenden.

- Wird ein lateinischer Text gesetzt und das Attribut `withprosodicmarks` ist ausgewählt, dann erzeugt das Gleichheitszeichen als Kurzzeichen einen Macron-Akzent über kleingeschriebenen Vokalen (außer über den Ligaturen æ und œ). Die Silbentrennung wird nicht beeinträchtigt.

Bsp.
9-3-18

```
\usepackage[latin]{babel} \languageattribute{latin}{withprosodicmarks}
ā ē ī ō ū \ProsodicMarksOn =a =e =i =o =u
```

- Entsprechend der Regeln zum Formatieren türkischer Texte muss das Gleichheitszeichen mit einem kleinen Abstand zum vorhergehenden Zeichen gesetzt werden. Das wird automatisch erreicht, indem man es als Kurzzeichen definiert, das ein vorausgehendes Leerzeichen durch einen kleinen Abstand ersetzt.

Bsp.
9-3-19

```
a =b \usepackage[english,turkish]{babel}
a=b \selectlanguage{english} a =b \par \selectlanguage{turkish} a =b
```

Die Verwendung des Gleichheitszeichen als Kurzformzeichen ist problematisch, da viele Pakete dann nicht mehr einwandfrei funktionieren, einschließlich der Einbindung von PostScript-Graphiken. In diesen Fällen muss das Kurzzeichen mit dem Befehl `\shorthandoff` abgeschaltet werden.

Die Größer- und Kleinerzeichen

Die Lösung für die spanische Sprache erlaubt, die Größer- und Kleinerzeichen (< und >) als Kurzzeichen für eine besondere Zitatumgebung zu verwenden. Diese Umgebung erzeugt unterschiedliche Anführungszeichen, wenn sie in sich selbst eingebettet wird. Sie unterstützt maximal drei Ebenen verschachtelter Zitate. Außerdem erzeugt sie automatisch zusätzliche „fortsetzende“ Anführungszeichen, wenn *innerhalb* eines Zitates ein neuer Absatz beginnt.

La regla es: «dentro de las comillas latinas se usan las “inglesas y dentro de éstas las ‘sencillas’”.

»Las comillas de seguir son como las de cerrar.»

```
\usepackage[spanish]{babel}
```

La regla es: <<dentro de las comillas latinas se usan las <<inglesas y dentro de éstas las <<sencillas>>>>.

Las comillas de seguir son como las de cerrar.>>

Bsp.
9-3-20

Man beachte, dass in Kurzzeichen umgewandelte Zeichen nicht mehr vom Ligaturmechanismus der Zeichensätze verarbeitet werden. In Zeichensätzen der T1-Kodierung sind beispielsweise zwei aufeinander folgende „Kleinerzeichen“ als Ligatur definiert, die Guillemets erzeugt. Die inneren Zitate im vorigen Beispiel zeigen deutlich, dass keine solche Ligatur erscheint.

Der Punkt

Die Lösung für die spanische Sprache erlaubt außerdem, den Punkt (.) im Formelmodus als Kurzzeichen zu verwenden. Er steuert, ob Dezimalzahlen mit einem Komma (`\decimalcomma`) oder einem Punkt (`\decimalpoint`) als Dezimalzeichen geschrieben werden.

1000,10

1000.10

```
\usepackage[spanish]{babel}
```

```
\decimalcomma $1000.10$ \par \decimalpoint $1000.10$
```

Bsp.
9-3-21

9.3.3 Sprachspezifische Befehle

Neben den bereits besprochenen Übersetzungen und Kurzformen verfügen einige Sprachdefinitionsdateien noch über zusätzliche Befehle. Manche davon dienen der Arbeitserleichterung beim Erstellen von regelkonform gesetzten Texten. Andere bieten zusätzliche Funktionen, die von Standard-L^AT_EX normalerweise nicht zur Verfügung gestellt werden. Einige dieser Befehle werden in diesem Abschnitt beschrieben.

Datumsformate

In einigen Sprachen gibt es mehrere Konventionen Datumsangaben zu formatieren. In diesen Fällen helfen zusätzliche Befehle dabei, ein Datum unterschiedlich auszugeben. In Bulgarien etwa ist es üblich, Monate in Ziffern aus römischen Großbuchstaben anzugeben; dazu steht der Befehle `\todayRoman` zur Verfügung.

29 февруари 2004 г.
29. II. 2004 г.

```
\usepackage[bulgarian]{babel}
```

```
\today \par \todayRoman
```

Bsp.
9-3-22

Für Esperanto stellen die Befehle `\hodiau` und `\hodiaun` zwei leicht voneinander abweichende Möglichkeiten der Datumsformatierung bereit.

Bsp.
9-3-23

29-a de februaro, 2004	
la 29-a de februaro, 2004	<code>\usepackage[esperanto]{babel}</code>
la 29-an de februaro, 2004	<code>\today \par \hodiau \par \hodiaun</code>

In griechischen Dokumenten lässt sich das Datum auch in griechischen anstelle der arabischen Zahlen darstellen. Dies geschieht mithilfe des Befehls `\Grtoday`.

Bsp.
9-3-24

29 Φεβρουαρίου 2004	<code>\usepackage[greek]{babel}</code>
ΚΘ' Φεβρουαρίου ,ΒΔ'	<code>\today \par \Grtoday</code>

Die Lösung zum Setzen hebräischer Texte verfügt über den Befehl `\hebdate`, der ein beliebiges gregorianisches Datum im Format „Tag, Monat, Jahr“ in ein gregorianisches Datum in hebräischer Sprache umwandelt. Er ersetzt den herkömmlichen L^AT_EX-Befehl `\today`. Zum Erzeugen „gewöhnlicher“ hebräischer Datumsangaben benötigt man das Paket `hebc` mit dem Befehl `\Hebrewtoday`. *Außerhalb* der hebräischen Umgebung erzeugt er ein hebräisches Datum in englischer Sprache.

Bsp.
9-3-25

2004 29 בפברואר	<code>\usepackage[english,hebrew]{babel}</code>
ז' באדר, תשס"ד	<code>\usepackage{hebc}</code>
29th February 2004: Adar 7, 5764	<code>\hebday \par \Hebrewtoday \par</code>
8 בנובמבר 1997	<code>\selectlanguage{english} \today: \Hebrewtoday</code>
	<code>\selectlanguage{hebrew} \hebdate{8}{11}{1997}</code>

Die Lösung für die ungarische Sprache erzeugt mithilfe des Befehls `\ontoday` ein Datumsformat für Redewendungen wie „am 10. Februar“.

Für die ober- und niedersorbische Sprache gibt es zwei unterschiedliche Sätze von Monatsnamen. Die Lösungen für diese Sprachen sind auf ein „modernes“ Datumsformat voreingestellt, können jedoch auch Datumsangaben „im alten Stil“ erzeugen. Der „alte Stil“ wird für das Niedersorbische mit dem Befehl `\olddatelsorb` aufgerufen; `\newdatelsorb` wechselt (zurück) zur modernen Form. Wie im folgenden Beispiel zu sehen ist, existieren für das Obersorbische entsprechende Befehle.

Bsp.
9-3-26

29. februara 2004	<code>\usepackage[usorbian,lsorbian]{babel}</code>
29. małego rožka 2004	<code>\newdatelsorb \today \par</code>
29. februara 2004	<code>\olddatelsorb \today \par</code>
29. małego rožka 2004	<code>\newdateusorb \today \par</code>
	<code>\olddateusorb \today</code>

In schwedischen Dokumenten werden Datumsangaben üblicherweise nur als Zahlen wiedergegeben. Sie können in zwei Formaten erscheinen: `YYYY-MM-DD` und `DD/MM YYYY`. Der Befehl `\datesymd` ändert die Definition des Befehls `\today` so, dass er ein Datum in der ersten numerischen Schreibweise

ausgibt, der Befehl `\datesdmy` definiert `\today` so um, dass er die zweite Schreibweise erzeugt.

Voreinstellung: 29 februari 2004
`\datesymd` ergibt: 2004-02-29
`\datesdmy` ergibt: 29/2 2004

```
\usepackage[swedish]{babel}
```

Voreinstellung: `\today\`
`\verb|\datesymd|` ergibt: `\datesymd \today \`
`\verb|\datesdmy|` ergibt: `\datesdmy \today`

Bsp.
9-3-27

Nummerierung

Für manche Sprachen gibt es zusätzliche Befehle für eine Nummerierung mit Buchstaben. L^AT_EX stellt zu diesem Zweck die Befehle `\alph` und `\Alph` zur Verfügung. Für Esperanto existieren die Befehle `\esper` und `\Esper`. Die Lösung für die griechische Sprache definiert `\alph` und `\Alph` so um, dass sie griechische Buchstaben erzeugen, für die bulgarische Sprache sind es entsprechend kyrillische Buchstaben. Die Lösungen für die russische und die ukrainische Sprache enthalten die Befehle `\asbuk` und `\Asbuk` als Alternativen zu den L^AT_EX-Befehlen.

In hebräischen Texten erzeugt der Befehl `\alph` Buchstabenfolgen nach dem Schema der „Gimatria“. Da es keine Großbuchstaben gibt, setzt `\Alph` die gleichen Buchstabenfolgen, jedoch mit einem vorangestellten Apostroph. Zusätzlich ergibt der Befehl `\Alphfinal` hebräische Buchstaben als Endbuchstaben mit Apostroph. Diese werden für hebräische Jahresbezeichnungen benötigt. In Tabelle 9.4 auf der gegenüberliegenden Seite werden die verschiedenen Nummerierungen miteinander verglichen.

In französischen Texten werden Zahlen nach anderen Regeln gesetzt als in englischen Texten. Anstelle des Kommas als Tausendertrennzeichen wird ein kleines Leerzeichen benutzt. Diese Art der Darstellung erreicht man mithilfe des Befehls `\nombre`. Er lässt sich auch *außerhalb* der französischen Sprachumgebung nutzen, wo er Nummern dann nach den englischen Regeln formatiert. Der Befehl `\nombre` verfügt über ein optionales Argument, mit dem sich ein anderes Dezimalzeichen (als das in `\decimalsep` voreingestellt) festlegen lässt. Diese Funktion kann sich in Kombination mit dem `dcolum`-Paket, in dem man das optionale Argument für eine korrekte Ausrichtung benötigt (siehe Abschnitt 5.7.2), als hilfreich erweisen.

```
\usepackage[english,french]{babel} \usepackage{dcolum}
% an explizitem ‘,’ ausrichten
% aber \decimalsep ausgeben :
\newcolumntype{d}{D{,}{\decimalsep}{-1}}

\begin{tabular}{|d|} \hline
12,34567 \\
12,345 67 \\
12,345 67 \\
9 876 543,21 \\
\hline
\end{tabular}

\par\vspace{1cm} \selectlanguage{english} % Sprachwechsel
\begin{tabular}{|d|} \hline
\nombre[,]{12,34567} \\\nombre[,]{9876543,21} \\\hline
\end{tabular}
```

12,34567
12,345 67
12,345 67
9 876 543,21

12.345,67
9,876,543.21

Bsp.
9-3-29

	Voreinstellung		Esperanto	Griechisch	Russisch	Bulgarisch	Hebräisch
Wert	\alph\Alpha	\esper\Esper	\alph\Alpha	\asbuk\Asbuk	\alph\Alpha	\alph\Alpha\Alphfinal	
1	a A	a A	α' A'	a A	a A	א א'	א
2	b B	b B	β' B'	б Б	б Б	ב ב'	ב
3	c C	c C	γ' Γ'	в В	в В	ג ג'	ג
4	d D	ĉ Ĉ	δ' Δ'	г Г	г Г	ד ד'	ד
5	e E	d D	ε' Ε'	д Д	д Д	ה ה'	ה
6	f F	e E	Ϝ' ϝ'	e E	e E	ו ו'	ו
7	g G	f F	ζ' Ζ'	ж Ж	ж Ж	ז ז'	ז
8	h H	g G	η' Η'	з З	з З	ח ח'	ח
9	i I	ĝ Ĝ	θ' Θ'	и И	и И	ט ט'	ט
10	j J	h H	ι' Ι'	к К	к К	י י'	י
11	k K	ĥ Ĥ	κ' Κ'	л Л	л Л	יא יא'	יא
12	l L	i I	λβ' ΙΒ'	м М	м М	יב יב'	יב
13	m M	j J	ιγ' ΙΓ'	н Н	н Н	יג יג'	יג
14	n N	ĵ Ĵ	ιδ' ΙΔ'	о О	о О	יד יד'	יד
15	o O	k K	ιε' ΙΕ'	п П	п П	טו טו'	טו
16	p P	l L	ιϜ' Ιϝ'	p P	p P	טז טז'	טז
17	q Q	m M	ιζ' ΙΖ'	с С	с С	יז יז'	יז
18	r R	n N	ιη' ΙΗ'	т Т	т Т	יח יח'	יח
19	s S	o O	ιθ' ΙΘ'	y Y	y Y	יט יט'	יט
20	t T	p P	κ' Κ'	φ Φ	φ Φ	כ כ'	כ
21	u U	s S	κα' ΚΑ'	x X	x X	כא כא'	כא
22	v V	ŝ Ŝ	κβ' ΚΒ'	ц Ц	ц Ц	כב כב'	כב
23	w W	t T	κγ' ΚΓ'	ч Ч	ч Ч	כג כג'	כג
24	x X	u U	κδ' ΚΔ'	ш Ш	ш Ш	כד כד'	כד
25	y Y	ŭ Ŭ	κε' ΚΕ'	щ Щ	щ Щ	כה כה'	כה
26	z Z	v V	κϜ' Κϝ'	э Э	ю Ю	כו כו'	כו
27	- -	z Z	κζ' ΚΖ'	ю Ю	я Я	כז כז'	כז
28	- -	- -	κη' ΚΗ'	я Я	- -	כח כח'	כח
29	- -	- -	κθ' ΚΘ'	- -	- -	כט כט'	כט
30	- -	- -	λ' Λ'	- -	- -	ל ל'	ל
40	- -	- -	μ' Μ'	- -	- -	מ מ'	מ
50	- -	- -	ν' Ν'	- -	- -	נ נ'	נ
100	- -	- -	ρ' Ρ'	- -	- -	ק ק'	ק
250	- -	- -	σν' Σ'Ν'	- -	- -	רנ רנ'	רנ
500	- -	- -	φ' Φ'	- -	- -	תק תק'	תק

Bsp.
9-3-28

Tabelle 9.4: Darstellen von Nummern durch Buchstaben

Im Griechischen existiert eine alternative Art Zahlen darzustellen. Sie basiert darauf, dass man Zahlenbereiche durch Buchstaben wiedergibt. Dieses System wurde gegen Ende des 19. und Anfang des 20. Jahrhunderts in amtlichen Veröffentlichungen benutzt. Heutzutage verwenden es die meisten Griechen für kleine Zahlen. Das Wissen, wie man Zahlen über 20 oder 30 schreibt, ist nicht sehr weit verbreitet und wird hauptsächlich von der östlich-orthodoxen Kirche und Gelehrten angewandt. Sie benutzen diesen Ansatz um Zahlen bis 999999 nach folgendem System zu notieren:

- Es lassen sich nur Zahlen größer null schreiben.

- Für die Einer von 1 bis einschließlich 9 werden die Buchstaben Alpha, Beta, Gamma, Delta, Epsilon, Stigma, Zeta, Eta und Theta verwendet, gefolgt von der so genannten „numerischen Markierung“, die dem mathematischen Ableitungssymbol ($\backslash prime$) ähnelt. Da der Buchstabe Stigma nicht in jedem Zeichensatz verfügbar ist, wird er alternativ häufig durch die ersten zwei Buchstaben seines Namens ersetzt. Die babel-Lösung erzeugt den Buchstaben Stigma und nicht das Digramm Sigma Tau.
- Für die Zehner 10 bis einschließlich 90 werden die Buchstaben Iota, Kappa, Lambda, My, Ny, Xi, Omikron, Pi und Qoppa verwendet, wiederum gefolgt von der numerischen Markierung. Das Qoppa in griechischen Zahlen hat eine deutliche Zickzackform, in der es sich vom normalen Qoppa unterscheidet, das eher dem lateinischen „q“ ähnelt.
- Die Hunderter 100 bis einschließlich 900 werden durch die Buchstaben Rho, Sigma, Tau, Ypsilon, Phi, Chi, Psi, Omega und Sampi dargestellt, wiederum gefolgt von einer numerischen Markierung.
- Nach diesen Regeln lässt sich jede Zahl zwischen 1 und 999 als Buchstabengruppe aus Hunderter, Zehner und Einer, gefolgt von *einer* numerischen Markierung, darstellen.
- Bei den Zahlen von 1000 bis einschließlich 999000 werden die Vielfachen der Tausender entsprechend durch die zuvor genannten Buchstaben ausgedrückt, denen in diesem Falle eine numerische Markierung vorangeht. Diese Markierung ist um 180 Grad gedreht und wird *unter* der Grundlinie positioniert. Wie im nächsten Beispiel zu sehen, werden in Kombinationen aus zwei Buchstabengruppen *beide* numerischen Markierungen verwendet.

123456 in griechischer Schreibweise: ρ,χ,γυυτ´

987654 in griechischer Schreibweise: ρ,λ,Π,ZXNΔ´

```
\usepackage[ngerman,greek]{babel}
\newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
123456 \de{in griechischer Schreibweise:}
\greeknumeral{123456} \par
987654 \de{in griechischer Schreibweise:}
\Greeknnumeral{987654}
```

Bsp.
9-3-30

Im alten Griechenland gab es noch ein weiteres Zahlensystem, das dem römischen insofern ähnelt, dass es für gewisse Zahlen Buchstaben verwendet. Mehrfach vorhandene Buchstaben bedeuten ein Mehrfaches der Grundzahl; der Buchstabe I z.B. bedeutet 1, also steht III für 3. Das athenische Zahlensystem verwendete die folgenden Grundziffern:

- I steht für die Zahl eins (1).
- II steht für die Zahl fünf (5).
- Δ steht für die Zahl zehn (10).
- H steht für die Zahl einhundert (100).
- X steht für die Zahl eintausend (1000).
- M steht für die Zahl zehntausend (10000).

Darüber hinaus bedeuten die Buchstaben Δ, H, X und M das Fünffache ihres ursprünglichen Wertes, wenn sie unter dem Buchstaben II stehen: Das Symbol

☐ steht also für die Zahl 5000, das Symbol ☐ für die Zahl 50. Man beachte, dass dieses Zahlensystem keine negativen Zahlen und kein Symbol für die Zahl Null kennt.

Das athenische Zahlensystem sowie einige weitere werden in einem Artikel in der Enzyklopädie *Δομή* beschrieben (Band 2, 7. Auflage, Seite 280, Athen, 2. Oktober 1975). Es wird über den Befehl `\athnum` des Paketes `athnum` unterstützt, das zum `babel`-System gehört.

Bsp.
9-3-31

6284 in griechischer Schreibweise: ☐XHH☐ΔΔΔIII

```
\usepackage[ngerman,greek]{babel}
\usepackage{athnum}
\newcommand\de[1]{\foreignlanguage{ngerman}{#1}}
6284 \de{in griechischer Schreibweise:} \athnum{6284}
```

In isländischen Dokumenten müssen Zahlen nach den isländischen Regeln gesetzt werden. Zu diesem Zweck steht der Befehl `\tala` zur Verfügung. Wie der Befehl `\nombre` verfügt er über ein optionales Argument zum Austausch des Dezimalzeichens, beispielsweise bei Verwendung in Kombination mit dem `dcolumn`-Paket.

Bsp.
9-3-32

3 141,592 653
3,141.592,653

3,14
123,456 7
9 876,543

```
\usepackage[english,icelandic]{babel}
\usepackage{dcolumn} \newcolumntype{d}{D{,}{\decimalsep}{-1}}
\tala{3141,592653} \par
\foreignlanguage{english}{\tala{3141,592653}}\par \bigskip
\begin{tabular}{|d|} \hline
3,14 \\\tala[,]{123,4567} \\\tala[,]{9876,543} \\\hline
\end{tabular}
```

Verschiedene Extras

Im Französischen ist es üblich, Familiennamen in Kapitälchen *ohne* Trennung zu setzen. Dies geschieht mithilfe des Befehls `\bsc` (boxed small caps, Kapitälchen in einer Box). Abkürzungen des französischen Wortes „numéro“ sollten nach bestimmten Regeln gesetzt werden. Diese wurden in den Befehlen `\no` und `\No` umgesetzt. Und schließlich gibt es für bestimmte Aufzählungen in französischen Texten die Befehle `\primo`, `\secundo`, `\tertio` und `\quarto`.

... für französische
Texte

Bsp.
9-3-33

Leslie LAMPORT N° 9 1° 3°

```
\usepackage[french]{babel}
Leslie~\bsc{Lamport} \quad \No9 \ \primo \ \tertio
```

Im Italienischen ist es üblich, den Artikel und das darauf folgende Substantiv zusammen zu schreiben, wie etwa in „nell'altezza“. Für die Silbentrennung bei solchen Konstruktionen wird das Zeichen ' so eingestellt, dass es sich wie ein normaler Buchstabe verhält.

... für italienische
Texte

Im Ungarischen kann der bestimmte Artikel je nach Kontext „a“ oder „az“ lauten. Besonders bei Quer- und Quellenverweisen ist nicht immer von vornherein zu erkennen, welche Form gerade verwendet werden muss. Die Lösung für die ungarische Sprache enthält Befehle, welche die Fälle, in denen dem Artikel ein „z“ angefügt werden muss, erkennen können. Sie verfügen alle über ein Argument, dessen Text analysiert wird, um zu bestimmen, welche Form des bestimmten Artikels vor diesem Text gesetzt werden muss.

... für ungarische
Texte

$\overbrace{\text{LATEX}}$		$\overbrace{\text{Serbisch}}$		$\overbrace{\text{Russisch}}$	
<code>\tan</code>	tan	<code>\tg</code>	tg	<code>\tg</code>	tg
<code>\cot</code>	cot	<code>\ctg</code>	ctg	<code>\ctg</code>	ctg
<code>\sinh</code>	sinh	<code>\sh</code>	sh	<code>\sh</code>	sh
<code>\cosh</code>	cosh	<code>\ch</code>	ch	<code>\ch</code>	ch
<code>\tanh</code>	tanh	<code>\th</code>	th	<code>\th</code>	th
<code>\coth</code>	coth	<code>\cth</code>	cth	<code>\cth</code>	cth
<code>\csc</code>	csc			<code>\cosec</code>	cosec
		<code>\arsh</code>	arsh		
		<code>\arch</code>	arch		
<code>\arctan</code>	arctan	<code>\arctg</code>	arctg	<code>\arctg</code>	arctg
		<code>\arcctg</code>	arcctg	<code>\arcctg</code>	arcctg

Bsp.

9-3-34

Man beachte, dass die geänderte Definition von `\th` zu Konflikten mit seiner Standardeigenschaft als LICR-Befehl für *p* (Thorn) führt, so dass `babel` diese Änderung auf den Formelmodus in kyrillischen Texten beschränkt.

Tabelle 9.5: Mathematische Operatoren in osteuropäischen Sprachen

`\az{text}` `\Az{text}`

Diese Befehle geben den Artikel und das Argument aus. Ist das Argument ein Stern (wie in `\az*`), so wird nur der Artikel gesetzt. Die Form `\Az` ist für den Satzanfang gedacht.

`\aref{text}` `\Aref{text}` `\apageref{text}` `\Apageref{text}`

Die ersten beiden Befehle sollten anstelle von `a(z)~\ref{label}` verwendet werden. Wird auf eine Formel verwiesen, so kann das Argument in runden anstatt in geschweiften Klammern stehen. Bei Seitenverweisen sollte man den Befehl `\apageref` (bzw. `\Apageref`) verwenden, damit L^AT_EX automatisch den passenden bestimmten Artikel erzeugen kann.

`\acite{text}` `\Acite{text}`

Für Quellenverweise sollte der Befehl `\acite` verwendet werden. Sein Argument kann eine Liste von Quellenverweisen enthalten, von denen dann der erste bestimmt, welche Form des Artikels gesetzt wird.

... und speziell für Formeln

In Osteuropa haben einige mathematische Operatoren in Formeln ein anderes Erscheinungsbild als in der „westlichen Welt“. Tabelle 9.5 zeigt die entsprechenden Befehle für die verschiedenen Sprachen. Die Befehle für die russische Sprache gelten auch für Bulgarisch und Ukrainisch. Das in der `babel`-Distribution enthaltene Paket `grmath` definiert diese Operatoren so um, dass sie Abkürzungen ihrer griechischen Bezeichnungen erzeugen. Dieses Paket kann nur zusammen mit der `babel`-Option `greek` genutzt werden.

9.3.4 Erwägungen zum Layout

Einige der Dateien zur Sprachunterstützung im `babel`-Paket enthalten Befehle, die das Layout eines Dokumentes automatisch verändern können. Einige än-

dern nur die Gestaltung der Abstände nach Interpunktionszeichen, oder sorgen dafür, dass der erste Absatz nach einer Abschnittsüberschrift eingerückt wird. Andere haben viel weiter reichende Auswirkungen.

In dem Buch *The T_EXbook* [87, S. 72–74] wird das Konzept zusätzlicher Leerräume nach Interpunktionszeichen besprochen. In gut formatierten Texten müssen sich die Abstände zwischen Sätzen beim Dehnen und Stauchen (für den Randausgleich) etwas anders verhalten als die Wortzwischenräume. Dieses Verfahren wird jedoch nicht immer als hilfreich angesehen, so dass die entsprechende Funktion in folgenden Sprachen mithilfe des Befehls `\frenchspacing` abgeschaltet wird: Bretonisch, Bulgarisch, Dänisch, Deutsch, Estnisch, Finnisch, Französisch, Norwegisch, Russisch, Spanisch, Tschechisch, Türkisch und Ukrainisch.

*Abstände nach
Interpunktionszeichen*

Ein weiterer Gestaltungsaspekt der meisten L^AT_EX-Klassen ist die Unterdrückung des Absatzeinzuges für den ersten Absatz nach einer Abschnittsüberschrift. Auch dieses Verfahren passt nicht immer. Die Lösungen für die französische, die serbokroatische und die spanische Sprache sind so abgewandelt, dass *alle* Absätze eingezogen werden. Dieses Verhalten lässt sich durch Laden des Paketes `indentfirst` für alle Dokumente einstellen.

*Absatzeinzug nach
Überschriften*

Die Lösung für französische Texte (und die für bretonische Texte, welche aus der französischen Lösung abgeleitet wurde) ist noch weiter an die in Frankreich üblichen Formatierungsregeln angepasst. Sie ändert allgemein die Formatierung von Listen durch Reduzierung der in ihnen enthaltenen vertikalen Weißräume. In der `itemize`-Umgebung entfernt sie alle vertikalen Abstände zwischen den Listenelementen und ändert deren Erscheinungsbild, indem sie „•“ durch „–“ ersetzt.

Listenlayout

Etwas Text mit einer Liste.

- Punkt 1
- Punkt 2

Und noch etwas Text danach.

Un peu de texte précédant la liste.
– premier élément ;
– deuxième élément.
Et quelques mots suivant la liste.

```
\usepackage[french,english]{babel}
\begin{minipage}[t]{3cm}
  Etwas Text mit einer Liste.
  \begin{itemize} \item Punkt 1
    \item Punkt 2 \end{itemize}
  Und noch etwas Text danach.
\end{minipage}
\quad \selectlanguage{french}
\begin{minipage}[t]{3cm}
  Un peu de texte précédant la liste.
  \begin{itemize} \item premier élément;
    \item deuxième élément. \end{itemize}
  Et quelques mots suivant la liste.
\end{minipage}
```

Bsp.
9-3-35

<code>\FrenchLayout</code>	<code>\StandardLayout</code>

Für Dokumente, die in mehreren Sprachen erstellt werden, bietet die Lösung für französische Texte die Möglichkeit, alle Listen einheitlich entweder im „französischen Layout“ oder im „L^AT_EX-Layout“ zu formatieren. Dazu verwendet man den Befehl `\FrenchLayout` bzw. `\StandardLayout` in der Dokumentenpräambel. Wenn ein Dokument allerdings nicht mit den Standard-L^AT_EX-Dokumentenklassen formatiert wird, oder wenn man Erweiterungspakete wie `paralist` benutzt, können solche Layoutänderungen zu überraschenden und

unerwünschten Effekten führen. In diesen Fällen empfiehlt sich die Verwendung des Befehls `\StandardLayout`.

`\AddThinSpaceBeforeFootnotes` `\FrenchFootnotes`

Fußnotenlayout

In der französischen Setzertradition werden Fußnoten anders gehandhabt als in der angloamerikanischen. Im Fließtext soll vor dem Fußnotenzeichen ein kleiner Leerraum eingefügt werden. Dieses Verhalten ist optional und wird über den Befehl `\AddThinSpaceBeforeFootnotes` in der Dokumentenpräambel ausgewählt. Auch der Fußnotentext kann nach französischen Regeln gesetzt werden, und zwar mithilfe des Befehls `\FrenchFootnotes`.

Un peu de texte ^a.

^aavec une note de
bas de page

Un peu de texte ^a.

^a. avec une note
de bas de page

```
\usepackage[french,english]{babel}
\AddThinSpaceBeforeFootnotes
\begin{minipage}{70pt} Un peu de texte\footnote{avec
    une note de bas de page}. \end{minipage}
\quad
\FrenchFootnotes
\begin{minipage}{70pt} Un peu de texte\footnote{avec
    une note de bas de page}. \end{minipage}
```

Bsp.
9-3-36

Legendenlayout

Als letzte Layoutänderung ersetzt die `babel`-Lösung für französische Texte den Doppelpunkt in Tabellen- und Abbildungslegenden durch einen Strich in Halbgeviertbreite, wenn eine der Dokumentenklassen von Standard-`LATEX` benutzt wird.

Umdefinierte
interne Befehle für 
magyar

Die Lösung für ungarische Dokumente geht sogar noch einen Schritt weiter: Sie definiert einige interne `LATEX`-Befehle so um, dass sie korrekte Legenden für Abbildungen und Tabellen erzeugen. Nach derselben Methode verändert sie auch das Layout von Abschnittsüberschriften. Auch einige der Befehle, auf denen die theoremähnlichen Umgebungen beruhen, werden verändert. Wie bereits erwähnt, können solche Abwandlungen unerwartete und sogar unerwünschte Auswirkungen haben, daher sollte man damit vorsichtig sein.

Schreibweise von rechts nach links

Zur Unterstützung hebräischer Texte sind noch drastischere Änderungen erforderlich, da die hebräische Sprache von rechts nach links geschrieben wird. Daher benötigt man in diesem Falle eine Erweiterung zu `TEX` (etwa `eTEX` mit einem `LATEX`-Format), um hebräische Dokumente korrekt setzen zu können.

9.3.5 Sprachoptionen und Fontkodierungen

Wie bereits einige der früheren Beispiele zeigten, reicht es für manche Sprachen nicht aus, einfach nur z.B. einige Texte zu übersetzen und besondere Ansprüche der Silbentrennung zu unterstützen. Viele Sprachen benötigen Zeichen, die nicht in der `LATEX-T1`-Kodierung enthalten sind. Für einige dieser Sprachen fehlen nur wenige Zeichen, die aus dem vorhandenen Zeichenvorrat zusammengesetzt werden können, andere Sprachen werden normalerweise gar nicht in lateinischer Schrift geschrieben. Einige von ihnen werden vom `babel`-System unterstützt.

Erweiterungen für die Kodierungen OT1 und T1

Für manche Sprachen fehlen nur einzelne Zeichen in der OT1-Kodierung, und manchmal auch in der T1-Kodierung. Wenn sich die fehlenden Zeichen aus dem vorhandenen Zeichenvorrat zusammensetzen lassen, kann man dieses Problem relativ einfach beheben. Das gilt z.B. für Altisländisch. Es benötigt einige Zeichen, die durch eine Kombination vorhandener Glyphen mit dem „Ogonek“-Akzent dargestellt werden können. Sie lassen sich über die im nächsten Beispiel verwendeten Kurzzeichen aufrufen. Man beachte, dass jedes dieser Kurzzeichen aus " und einem 8-Bit-Zeichen besteht, so dass man das inputenc-Paket benötigt.

Bsp.
9-3-37

o Q ó Ó ę E é Ę aber: "é "É

```
\usepackage[icelandic]{babel}
\usepackage[T1]{fontenc} \usepackage[latin1]{inputenc}
"o "Q "ó "Ó "e "E "é "Ę aber: "\'e "\'E
```

Altisländisch ist vielleicht keine alltägliche Sprache, Polnisch hingegen schon. Auch für diese Sprache fehlen der OT1-Kodierung einige Zeichen (man beachte, dass sie in T1 jedoch alle enthalten sind). Auch hier *können* die fehlenden Zeichen zusammengesetzt werden und Kurzformen erleichtern ihre Eingabe. Wie im folgenden Beispiel dargestellt, werden zwei Arten der Eingabe für die Buchstaben „z mit Punkt“ und „z mit Akzent“ unterstützt. Diese Dualität hat geschichtliche Hintergründe.

Bsp.
9-3-38

ą Ą ć Ć ę Ę ł ń Ó ó Ś ś Ż ż Ż ż "x "X
"r "R ż Ż Ż Ż

```
\usepackage[polish]{babel}
"a "A "c "C "e "E "l "ł "ń "N "o "O "s "S \par
\polishrz "r "R "z "Z "x "X \par
\polishzx "r "R "z "Z "x "X \par
```

Alle diese Kurzformen wurden entwickelt, als 7-Bit-Fontkodierungen noch die Norm waren und die Erstellung einer Glyphen wie „Ą“ einiger interner Makros bedurfte (wenn sie überhaupt möglich war). Die heutigen 8-Bit-Kodierungen benötigen diese Kurzformen nicht mehr. Bei T1-kodierten Fonts z.B. kann man stattdessen auf Standard-Eingabemethoden zurückgreifen.

Bsp.
9-3-39

ą Ą ć Ć ę Ę ł ń Ó ó Ś ś Ż ż Ż

```
\usepackage[T1]{fontenc}
\k a \k A \'c \'C \k e \k E \l{} \'n \'N \'o \'O \'s \'S
\.z \.Z \'z \'Z
```

Umschalten zwischen Fontkodierungen

Wenn das Zusammensetzen einiger zusätzlicher Zeichen zum Setzen von Texten in einer bestimmten Sprache nicht mehr ausreicht, muss man auf eine andere Fontkodierung wechseln. Dieser Abschnitt beschreibt die entsprechenden babel-Befehle und -Dateien für die Sprachunterstützung. Man beachte, dass diese Befehle normalerweise in der babel-Benutzerschnittstelle „verborgen“ sind.

```
\latinencoding \cyrillicencoding \hebrewencoding
```

Das babel-Paket verwendet den Befehl \latinencoding um die verwendete lateinische Kodierung eines Dokumentes (OT1 oder T1) zu speichern. Zum

Bestimmen der benutzten Kodierung testet `babel` während der Ausführung von `\begin{document}`, ob diese T1 ist; ist dies nicht der Fall, geht es (vielleicht fälschlicherweise) von der OT1-Kodierung aus.

Auf dem kyrillischen Alphabet basierende Sprachen definieren den Befehl `\cyrillicencoding` um den Namen der kyrillischen Kodierung zu speichern. Für die hebräische Fontkodierung gibt es entsprechend den Befehl `\hebrewencoding`. Als dieses Buch geschrieben wurde, gab es keinen Befehl `\greekencoding`, da `babel` nur eine einzige Kodierung für die griechische Sprache (LGR) unterstützte.

`\textlatin{text}`

Dieser Befehl setzt sein Argument in einem Font mit lateinischer Kodierung, unabhängig davon, auf welcher Kodierung der umgebende Text basiert.

`\textcyrillic{text}`

Dieser Befehl wird nur bei Verwendung der Optionen `bulgarian`, `russian` oder `ukrainian` definiert. Er setzt sein Argument in einem Font mit der kyrillischen Kodierung, die in dem Befehl `\cyrillicencoding` gespeichert ist.

`\textgreek{text}` `\textol{text}`

Diese Befehle werden von der Sprachoption `greek` definiert. Beide setzen ihr Argument in einem Font mit der griechischen Kodierung. Der Befehl `\textol` verwendet eine Umrissschrift (Outline).

Zu diesen `\text...`-Befehlen gibt es auch die deklarativen Varianten `\latintext`, `\greektext`, `\outlfamily` und `\cyrillictext`.

Umschalten zwischen Schreibrichtungen

Um das Setzen hebräischer Texte zu unterstützen, muss auch die Schreibrichtung geändert werden. Zu diesem Zweck wurden eine Reihe von Befehlen mit unterschiedlichen Namen definiert.

`\sethebrew` `\unsethebrew`

Der Befehl `\sethebrew` schaltet die Schreibrichtung auf „von rechts nach links“ um, wechselt die Fontkodierung zu einer hebräischen Kodierung *und* verschiebt den „Punkt des Schreibbeginns“ an den rechten Rand. Der Befehl `\unsethebrew` schaltet die Schreibrichtung auf „von links nach rechts“ um, wechselt die Fontkodierung zurück zu derjenigen, die vor `\sethebrew` gültig war, *und* verschiebt den „Punkt des Schreibbeginns“ an den linken Rand.

`\R{text}` `\L{text}`

Die Befehle `\R` und `\L` sollten benutzt werden, um kleine Textpassagen in hebräischer Sprache in den Fließtext einzubetten. Ihre Verwendung wird im

nächsten Beispiel demonstriert. Man beachte die Position des zweiten Textes in hebräischen Zeichen.

```
\usepackage[X2,T1]{fontenc} \usepackage[greek,russian,hebrew,ngerman]{babel}
Text in Deutsch, \R{in Hebräisch,} \textgreek{in Griechisch,}
\textcyrillic{in Kyrillisch,}
\sethebrew erneut in Hebräisch,\unsethebrew{} wieder in Deutsch.
```

Bsp.
9-3-40

Text in Deutsch, מ ווערן געשריבן in Griechisch, ἰσχυρῶς ἡ ἀληθεία
מישן, wieder in Deutsch. ועמך מ ווערן געשריבן

9.4 Unterstützung nicht lateinischer Alphabete

Die babel-Distribution unterstützt drei nicht lateinische Alphabete: das kyrillische, das griechische und das hebräische. Sie werden in den folgenden Abschnitten besprochen.

9.4.1 Das kyrillische Alphabet

Das kyrillische Alphabet wird in einigen slawischen Sprachen Osteuropas und in Dutzenden weiterer Sprachen auf dem Gebiet der früheren Sowjetunion verwendet. Vladimir Volovich und Werner Lemberg haben in Zusammenarbeit mit dem L^AT_EX-Team eine grundlegende Unterstützung für die kyrillische Sprache in L^AT_EX integriert. Dieser Abschnitt befasst sich mit den kyrillischen Zeichensätzen und der Kodierungsschnittstelle sowie ihrer Integration in babel.

Geschichtlich gesehen bietet die American Mathematical Society [14] schon seit längerer Zeit Unterstützung für die russische Sprache in T_EX an. Das AMS-System bedient sich der wncyr-Fonts und basiert auf einer Transliterationstabelle, die ursprünglich für Namen russischer Journale und Titel russischer Artikel in der Zeitschrift *Mathematical Reviews* entwickelt wurde. In dieser Zeitschrift legt die AMS Wert darauf, dass die elektronischen Dateien entweder den russischen Text in russischen Zeichen erzeugen, oder seine Transliteration in englischen Zeichen ohne jede Mehrdeutigkeit.

Mit der zunehmenden Verbreitung von T_EX in Russland wuchs jedoch auch der Bedarf an geeigneter Unterstützung für die russische Sprache (sowie später für andere Sprachen, die das kyrillische Alphabet verwenden). Im Laufe der Jahre wurden sowohl zahlreiche 7- und 8-Bit-Kodierungen als auch viele Fontkodierungen entwickelt. Das kyrillische System ist so gestaltet, dass es jede 8-Bit-Eingabekodierung unterstützt und all diese Kodierungen einigen wenigen kyrillischen Zeichensätzen zuordnen kann, von denen jeder wiederum eine Reihe von Sprachen unterstützt.

Fonts und Fontkodierungen

Aus Gründen der Kompatibilität stehen für neue Glyphen im Wesentlichen nur die oberen 128 Zeichen eines 8-Bit-Zeichensatzes für T_EX zur Verfügung. Im 20. Jahrhundert bedienten sich die Sprachen, die auf dem kyrillischen Alphabet basieren, jedoch weit mehr als 128 verschiedener Glyphen. Daher wurden vier „kyrillische Fontkodierungen“ entwickelt [17]. Drei von ihnen – T2A,

T2B und T2C – entsprechen den L^AT_EX-Anforderungen für T*-Kodierungen und lassen sich daher in mehrsprachigen Dokumenten mit anderen Sprachen kombinieren.¹

Die Arbeit an den T2*-Kodierungen wurde von Alexander Berdnikov in Zusammenarbeit mit Mikhail Kolodin und Andrew Janishevsky durchgeführt. Vladimir Volovich nahm die Integration in L^AT_EX vor.

Es gibt darüber hinaus noch zwei weitere L^AT_EX-Fontkodierungen für die kyrillische Sprache: zum einen die 7-Bit-Kodierung OT2 der American Mathematical Society für kurze kyrillische Texte und zum anderen die 8-Bit-Kodierung LCY, die zu den T*-Kodierungen von L^AT_EX inkompatibel ist und sich daher nicht eignet, um mehrsprachige Dokumente damit zu setzen. Die OT2-Kodierung ist so gestaltet, dass ein und dieselbe Quelldatei Texte in kyrillischen Zeichen oder als entsprechende Transliteration erzeugen kann.

Kyrillische Computer Modern Fonts

Die in L^AT_EX voreingestellte Schriftfamilie sind die Computer Modern Fonts von Donald Knuth in ihrer 7- oder 8-Bit-Variante, als OT1-kodierte CM Fonts bzw. T1-kodierte EC Fonts. Olga Lapko und Andrey Khodulev entwickelten die LH Fonts, deren Glyphen zur Computer Modern Fontfamilie passen und alle kyrillischen Fontkodierungen abdecken. Sie verfügen über die gleichen Schriftformen und -grade wie ihre lateinische Entsprechung, die EC Fonts. Diese Schriften sind bei CTAN im Verzeichnis `fonts/cyrillic/lh` zu finden. Die Installationsanleitung ist in der Datei `INSTALL` der Distribution enthalten.²

Mit der `ruhyphen`-Distribution (`language/hyphenation/ruhyphen`) ist bei CTAN eine Sammlung von Trennmustern für die russische Sprache verfügbar, welche die T2*-Kodierungen sowie andere beliebte Fontkodierungen für russische Texte unterstützt, einschließlich der internen Kodierung von Omega. Die Muster für andere kyrillische Sprachen sollten an die T2*-Kodierungen angepasst werden.

Kyrillische Sprache in Dokumenten

Die Unterstützung für die kyrillische Sprache in L^AT_EX basiert auf den Standardpaketen `fontenc` und `inputenc` sowie auf dem `babel`-Paket. Man kann beispielsweise folgende Anweisungen in die Dokumentenpräambel aufnehmen:

```
\usepackage[T2A]{fontenc}    \usepackage[koi8-r]{inputenc}
\usepackage[russian]{babel}
```

Die Eingabekodierung `koi8-r` (KOI8 optimiert für die russische Sprache) lässt sich durch eine der nachfolgenden kyrillischen Eingabekodierungen ersetzen.

¹Bei der vierten kyrillischen Kodierung, X2, sind alle 256 Zeichenpositionen mit kyrillische Glyphen besetzt, so dass sie sich nur für besondere, rein kyrillische Anwendungen eignet. Sie wird hier nicht näher besprochen.

²Es können auch andere Schriften, einschließlich Type 1-Fonts, verwendet werden, sofern ihre T_EX-Fontkodierung zu den T2*-Kodierungen kompatibel ist. Besonders die CM-Super-Schriften decken die gesamte Bandbreite der kyrillischen Kodierungen ab. Nähere Erläuterungen hierzu befinden sich in Abschnitt 7.5.1 auf Seite 365.

cp855 MS-DOS-Standardcodepage für Kyrillisch.

cp866 MS-DOS-Standardcodepage für die russische Sprache. Es gibt mehrere Varianten, die sich in den Zeichenpositionen 242–254 unterscheiden: cp866av (Alternative für Kyrillisch), cp866mav (Modifizierte Alternative Variante), cp866nav (Neue Alternative Variante) und cp866tat (für die tatarische Sprache).

cp1251 MS Windows-Standardcodepage für Kyrillisch.

koi8-r Kyrillische Standardcodepage, die in vielen UNIX-ähnlichen Systemen zur Unterstützung der russischen Sprache eingesetzt wird. Varianten für die ukrainische Sprache sind koi8-u und koi8-ru. Eine ECMA-Variante (ISO-IR 111 ECMA) ist isoir111.

iso88595 ISO-Standard ISO 8859-5 (auch bekannt als ISO-IR 144).

maccyr Apple Macintosh-Codepage für Kyrillisch (auch bekannt als Microsoft cp10007) und macukr, die Apple Macintosh-Codepage für die ukrainische Sprache.

ctt, dbk, mnk, mos, ncc Mongolische Codepage-Varianten.

Nicht alle diese Codepage-Varianten gehören zur Standarddistribution von `inputenc`, einige von ihnen sind nur separat erhältlich.

Wenn in einem Dokument mehr als eine Eingabekodierung verwendet wird, kann man mit dem Befehl `\inputencoding` zwischen ihnen umschalten. Die automatische Groß- oder Kleinschreibung lässt sich über zwei L^AT_EX-Standardbefehle erzeugen, und zwar `\MakeUppercase` bzw. `\MakeLowercase`. Die T_EX-Low-Level-Befehle `\uppercase` und `\lowercase` sollten nie in L^AT_EX verwendet werden und funktionieren nicht in kyrillischen Texten.

Im letzten Präambelbeispiel wurde die zu verwendende Fontkodierung (T2A) explizit deklariert. In mehrsprachigen Dokumenten sollten *alle* benötigten Kodierungen mithilfe des Befehls `\usepackage[...]{fontenc}` aufgelistet werden. Man kann zwar mit dem Befehl `\fontencoding` von einer Fontkodierung auf die andere umschalten, es empfiehlt sich jedoch solche Änderungen mithilfe einer höher angesiedelten Schnittstelle wie z.B. dem Befehl `\selectlanguage` vorzunehmen. Sofern man `babel` verwendet, kann man etwa Folgendes deklarieren:

```
\usepackage[koi8-r]{inputenc} \usepackage[russian]{babel}
```

Das `babel`-System wählt dann automatisch die voreingestellte Fontkodierung für die russische Sprache (T2A), soweit diese verfügbar ist. Tabelle 9.6 auf der nächsten Seite zeigt den Aufbau der T2A-Kodierung.

Fontkodierungen für kyrillische Sprachen

Die kyrillischen Fontkodierungen unterstützen die nachfolgend aufgelisteten Sprachen. Man beachte, dass sich einige Sprachen, wie z.B. Bulgarisch und Russisch, mit mehr als einer Kodierung richtig setzen lassen.

	‘0	‘1	‘2	‘3	‘4	‘5	‘6	‘7	
‘00x	˘	˙	ˆ	˜	¨	˘	˚	ˇ	"0x
‘01x	˘	˘	˙	˙	˙	I	⟨	⟩	
‘02x	“	”	ˆ	˜	˘	–	—		"1x
‘03x	o	ı	j	ff	fi	fl	ffi	ffl	
‘04x	„	!	"	#	\$	%	&	'	"2x
‘05x	(	)	*	+	,	-	.	/	
‘06x	0	1	2	3	4	5	6	7	"3x
‘07x	8	9	:	;	<	=	>	?	
‘10x	@	A	B	C	D	E	F	G	"4x
‘11x	H	I	J	K	L	M	N	O	
‘12x	P	Q	R	S	T	U	V	W	"5x
‘13x	X	Y	Z	[	\	]	^	_	
‘14x	‘	a	b	c	d	e	f	g	"6x
‘15x	h	i	j	k	l	m	n	o	
‘16x	p	q	r	s	t	u	v	w	"7x
‘17x	x	y	z	{		}	~	-	
‘20x	Г	Г	Т	Т	Н	Ж	З	Л	"8x
‘21x	Ї	Қ	К	К	Æ	Ң	Н	С	
‘22x	Ө	Ң	Ү	Ү	Ү	Х	Ц	Ч	"9x
‘23x	Ч	Є	Ә	Н	Ё	№	Ѡ	§	
‘24x	г	г	ђ	ђ	h	ж	з	л	"Ax
‘25x	і	қ	к	к	æ	ң	н	с	
‘26x	ө	с	ү	ү	ү	х	ц	ч	"Bx
‘27x	ч	є	ә	н	ё	„	«	»	
‘30x	А	Б	В	Г	Д	Е	Ж	З	"Cx
‘31x	И	Й	К	Л	М	Н	О	П	
‘32x	Р	С	Т	У	Ф	Х	Ц	Ч	"Dx
‘33x	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	
‘34x	а	б	в	г	д	е	ж	з	"Ex
‘35x	и	й	к	л	м	н	о	п	
‘36x	р	с	т	у	ф	х	ц	ч	"Fx
‘37x	ш	щ	ъ	ы	ь	э	ю	я	
	"8	"9	"A	"B	"C	"D	"E	"F	

Blau dargestellte Zeichen müssen in jeder Textkodierung (an ihren angegebenen Positionen) vorhanden sein, da sie transparent durch T_EX hindurchgereicht werden.

Tabelle 9.6: Glyphentabelle eines T2A-kodierten Fonts (larm1000)

- T2A: Abasinisch, Awarisch, Agulisch, Adygeisch, Aserbaidshanisch, Altaisch, Balkarisch, Baschkirisch, Bulgarisch, Burjatisch, Weißrussisch, Gagausisch, Darginisch, Dunganisch, Inguschisch, Kabardino-Tscherkessisch, Kasachisch, Kalmückisch, Karakalpakisch, Karatschaisch, Karelisch, Kirgisisch, Komi-Syrjänisch, Komi-Permjakisch, Kumükisch, Lakkisch, Lesgisch, Mazedonisch, Berg-Mari, Wiesen-Mari, Moldauisch, Mongolisch, Mokscha-Mordwinisch, Ersä-Mordwinisch, Nogaisch, Orotschisch, Ossetisch, Russisch, Rutulisch, Serbisch, Tabassaranisch, Tadschikisch, Tatarisch, Tatisch, Teleutisch, Tofalarisch, Tuwinisch, Turkmenisch, Udmurtisch, Usbekisch, Ukrainisch, Mittel-Ob-Chantisch, Surgut-Chantisch, Romani (Gipsi), Tschetschenisch, Tschuwaschisch, Krimtatarisch
- T2B: Abasinisch, Awarisch, Agulisch, Adygeisch, Aleutisch, Altaisch, Balkarisch, Weißrussisch, Bulgarisch, Burjatisch, Gagausisch, Darginisch, Dolganisch, Dunganisch, Inguschisch, Itelmenisch, Kabardino-Tscherkessisch, Kalmückisch, Karakalpakisch, Karatschaisch, Karelisch, Ketisch, Kirgisisch, Komi-Syrjänisch, Komi-Permjakisch, Korjakisch, Kumükisch, Kurdisch, Lakkisch, Lesgisch, Mansisch, Wiesen-Mari, Moldauisch, Mongolisch, Mokscha-Mordwinisch, Ersä-Mordwinisch, Nanaisch, Nganasanisch, Negidalisch, Nenzisch, Niwchisch, Nogaisch, Orotschisch, Russisch, Rutulisch, Selkupisch, Tabassaranisch, Tadschikisch, Tatarisch, Tatisch, Teleutisch, Tofalarisch, Tuwinisch, Turkmenisch, Udeheisch, Uigurisch, Ultschaisch, Chakassisch, Wach-Chantisch, Kasym-Chantisch, Mittel-Ob-Chantisch, Surgut-Chantisch, Schuryschkary-Chantisch, Romani (Gipsi), Tschetschenisch, Tschuktschisch, Schorisch, Ewenkisch, Ewenisch, Enezisch, Eskimo, Jukagirisch, Krimtatarisch, Jakutisch
- T2C: Abchasisch, Bulgarisch, Gagausisch, Karelisch, Komi-Syrjänisch, Komi-Permjakisch, Kumükisch, Mansisch, Moldauisch, Mokscha-Mordwinisch, Ersä-Mordwinisch, Nanaisch, Oroksch (Uilta), Negidalisch, Nogaisch, Orotschisch, Russisch, Samisch, Altbulgarisch, Altrussisch, Tatisch, Teleutisch, Mittel-Ob-Chantisch, Surgut-Chantisch, Ewenkisch, Krimtatarisch

Die grundlegende L^AT_EX-Distribution enthält bereits alle Kodierungs- und Fontdefinitionsdateien für kyrillische Texte. Das `babel`-Paket unterstützt die Sprachen Bulgarisch, Russisch und Ukrainisch. In Kombination mit den Fontdateien, die extra installiert werden müssen, kann L^AT_EX mithilfe dieses Paketes alle Sprachen unterstützen, die auf dem kyrillischen Alphabet basieren.

MakeIndex und BiBTeX

Angesichts der Tatsache, dass die Programme *MakeIndex* und *BiBTeX* von sich aus nicht mit 8-Bit-Eingabekodierungen umgehen können, enthält die T2-Sammlung einige Hilfsmittel, die eine korrekte Verarbeitung von kyrillischen 8-Bit-Daten durch diese Programme ermöglichen.

Das Programm `rumakeindex` unterstützt *MakeIndex* bei der Erstellung von Stichwortverzeichnissen mit kyrillischen Buchstaben, indem es für eine korrekte Sortierung sorgt. Es benötigt zusätzlich das Programm `sed`.¹ Wenn man einen Index mit kyrillischen Buchstaben erstellt, sollte man anstelle von *MakeIndex* dieses Hilfsprogramm verwenden. Man beachte, dass das `rumakeindex`-Script unter UNIX die `koi8-r`-Kodierung verwendet, während die entsprechende Batch-Datei `rumkidxd.bat` für MS-DOS die `cp866`-Kodierung und die Batch-Datei `rumkidxw.bat` für MS Windows die `cp1251`-Kodierung benutzt. Wenn man eine andere Kodierung benötigt, müssen die jeweiligen Dateien entsprechend angepasst werden. Als Alternative kann man auch das Programm `xindy` in Betracht ziehen, ein neueres Programm zum Generieren von Stichwortverzeichnissen, das in Abschnitt 11.3 beschrieben wird.

Das Programm `rubibtex` unterstützt L^AT_EX bei der Erstellung von Literaturverweisen mit kyrillischen Buchstaben, die den Verweisschlüsseln entsprechen, sofern eine L^AT_EX-Literaturdatenbank verwendet wird. In diesem Falle sollte man außerdem das Programm `citehack` aus der T2-Sammlung installieren. Die installierte Version des L^AT_EX-Programms sollte überdies fähig sein, 8-Bit-Daten zu verarbeiten (wie z.B. das in Abschnitt 13.1.1 beschriebene Programm L^AT_EX8). Auch das `rubibtex`-Script und die entsprechenden Batch-Dateien benötigen das `sed`-Programm, welches bereits zuvor im Falle von *MakeIndex* beschrieben wurde.

Man beachte, dass das `rubibtex`-Script unter UNIX die `koi8-r`-Kodierung verwendet, während die entsprechende Batch-Datei `rubibtex.bat` für MS-DOS die `cp866`-Kodierung benutzt. Wenn man eine andere Kodierung benötigt, sollten die jeweiligen Dateien entsprechend angepasst werden.

9.4.2 Das griechische Alphabet

Die griechische Sprache wird von `babel` in zwei Varianten unterstützt: in der Variante `monotoniko` mit einem Akzent (der Voreinstellung), die in Griechenland heutzutage meistens für Belange der alltäglichen Kommunikation genutzt wird, und der Variante `polutoniko` mit mehreren Akzenten, die, wie in Abschnitt 9.2.3 beschrieben, als Attribut angegeben werden muss.

Die erste griechische Schriftfamilie für T_EX wurde Mitte der 1980er Jahre von Silvio Levy [116] gestaltet. Andere Entwickler verbesserten oder erweiterten diese Fonts oder erschufen ihre eigenen griechischen Schriften.

Die Lösung für die griechische Sprache in `babel` basiert auf der Zusammenarbeit von Claudio Beccari mit Apostolos Syropoulos, der die griechische `cb`-Schriftfamilie [12] entwickelt hat. In ihrer Abhandlung besprechen die Autoren recht ausführlich frühere Versuche, die griechische Sprache in T_EX zu unterstützen. Die Quellen der `cb`-Fonts sind bei CTAN im Verzeichnis `languages/greek/cb` oder auf der T_EX Live-CD im Verzeichnis `texmf/fonts/source/public/cbgreek` verfügbar. Die zu dieser Schriftfamilie passenden Muster für die Silbentrennung finden sich in der Datei `grhyph.tex` oder `grphyph.tex` im gleichen Verzeichnis bei CTAN bzw. unter `texmf/tex/generic/hyphen` auf der T_EX Live-CD.

¹Dieses ist Bestandteil jedes UNIX-Systems und wird von GNU für Microsoft Betriebssysteme angeboten (z.B. unter <http://www.simtel.net>).

Die cb-Fonts verwenden die LGR-Fontkodierung. Als dieses Buch geschrieben wurde, befand sich eine zu den L^AT_EX-Standards kompatible Fontkodierung in Arbeit. Nach ihrer Fertigstellung wird sie als T7-Kodierung zur Verfügung stehen. Tabelle 9.7 auf der nächsten Seite zeigt den Aufbau der gesamten LGR-Kodierung.

Man kann griechische Texte entsprechend der Transliterationstabelle 9.8 auf Seite 593 mit lateinischen Buchstaben eingeben. Die Tabelle zeigt, dass es für das lateinische „v“-Zeichen in der griechischen Transkription keine direkte Entsprechung gibt. Es wird stattdessen verwendet, um anzuzeigen, dass man *kein* End-Sigma setzen möchte. Die Kombination „sv“ erzeugt beispielsweise auch dann ein normales Sigma, wenn sie am Ende eines Wortes steht.

Die babel-Option für die griechische Sprache verwendet in ihrer Voreinstellung die monotoniko-Variante der griechischen Schrift. Mit dem Sprachattribut polutoniko für die Option greek lässt sich die Variante mit mehreren Akzenten aufrufen:

```
\usepackage[greek]{babel}
\languageattribute{greek}{polutoniko}
```

Für beide Modi wurden einige selten verwendete Zeichen so definiert, dass sie sich wie Buchstaben verhalten (`\catcode 11`). In der monotoniko-Variante handelt es sich um die Zeichen ’ und ". In der polutoniko-Variante verhalten sich auch die Zeichen <, >, ~, ‘ und | wie Buchstaben. Das ist erforderlich, damit die LGR-Kodierung für die zahlreichen Ligaturen mit diesen Zeichen die richtigen Glyphen darstellen kann (vgl. Tabelle 9.9 auf Seite 593). In Tabelle 9.10 sind die verfügbaren Kombinationen von Akzent- und Spirituszeichen aufgeführt.

9.4.3 Das hebräische Alphabet

Die erste Lösung für hebräische Texte, die in die babel-Distribution aufgenommen wurde, ist eine Entwicklung von Boris Lavva und Alon Ziv, die auf früheren Arbeiten basiert, mit denen das Setzen hebräischer Texte in L^AT_EX 2.09 und T_EX-_X_LT unterstützt wurde. Diese Unterstützung wurde von den beiden Autoren in Zusammenarbeit mit Rama Porrat weiterentwickelt. Als dieses Buch entstand, rief Tzafir Cohen gerade ein SourceForge-Projekt namens „ivritex“ (<http://ivritex.sf.net>) ins Leben, das die Bemühungen auf eine noch breitere Basis stellen soll.

Die aktuelle Lösung für hebräische Texte basiert auf Fonts der Hebrew University of Jerusalem. Diese Zeichensätze verfügen über eine besondere 7-Bit-Kodierung, für welche die Kodierung „Local Hebrew“ (LHE) entwickelt wurde. Abbildung 9.1 auf Seite 594 wurde mithilfe des Jerusalem-Fonts gesetzt. In Tabelle 9.11 auf Seite 595 ist die Kodierung der Fonts dargestellt. Die babel-Lösung verwendet den Jerusalem-Font als reguläre Schrift, Old Jaffa als Kursivschrift und den Dead Sea-Font zum Setzen fetter Buchstaben. Als serifenlose Schrift wird der Tel Aviv-Font benutzt, der gleichzeitig als Ersatz für eine Schreibmaschinenschrift dient.

	‘0	‘1	‘2	‘3	‘4	‘5	‘6	‘7	
‘00x	—	ˆ	⏏	⏐	⏑	⏒	⏓	⏔	“0x
‘01x	ı	Aı	Hı	Ωı	A	Ŧ	α	ü	
‘02x	/	\	ı	ŧ	˘	ŧ	⏑	⏒	“1x
‘03x	€	‰	ə	⏑	˘	˘	˘	—	
‘04x	˘	!	˘	˘	˘	‰	˘	˘	“2x
‘05x	(	)	*	+	,	-	.	/	
‘06x	0	1	2	3	4	5	6	7	“3x
‘07x	8	9	:	˘	<	=	>	;	
‘10x	˘	A	B	˘	Δ	E	Φ	Γ	“4x
‘11x	H	I	Θ	K	Λ	M	N	O	
‘12x	Π	X	P	Σ	T	Υ	˘	Ω	“5x
‘13x	Ξ	Ψ	Z	[	˘	]	˘	˘	
‘14x	`	α	β	ς	δ	ε	φ	γ	“6x
‘15x	η	ι	θ	κ	λ	μ	ν	ο	
‘16x	π	χ	ρ	ς	τ	υ		ω	“7x
‘17x	ξ	ψ	ζ	«	.	»	~	—	
‘20x	à	á	â	ã	ä	å	æ	ç	“8x
‘21x	á	â	ã	ä	å	æ	ç	ç	
‘22x	ã	ä	å	ƒ	ç	ç	ç	˘	“9x
‘23x	ĥ	ĥ	ĥ		ĥ	ĥ	ĥ		
‘24x	ĥ	ĥ	ĥ	ĥ	ĥ	ĥ	ĥ	ĥ	“Ax
‘25x	ĥ	ĥ	ĥ	ĥ	ĥ	ĥ	ĥ	ĥ	
‘26x	ò	ó	ô	õ	ö	ö	ö	ö	“Bx
‘27x	ó	ô	õ	õ	ö	ö	ö	ö	
‘30x	õ	õ	õ	F	ö	ö	ö		“Cx
‘31x	ì	í	î	ï	ù	ú	û	ü	
‘32x	í	î	ï	ì	ú	û	ü	ü	“Dx
‘33x	ï	ï	ï	Ï	ü	ü	ü	Ŧ	
‘34x	è	é	ê	ë	ò	ó	ô	õ	“Ex
‘35x	é	ê	ë	ë	ó	ô	ô	õ	
‘36x	ï	ï	ï	Ï	ü	ü	ü	ü	“Fx
‘37x	ø	η	φ	ř	ř		˘	˘	
	“8	“9	“A	“B	“C	“D	“E	“F	

Die *blau* dargestellten Zeichen müssen eigentlich in jeder L^AT_EX-Textkodierung ASCII-Zeichen sein (vgl. Tabelle 9.6 auf Seite 588), da sie transparent durch T_EX hindurchge-
reicht werden. Dies trifft bei der LGR-Kodierung für A-Z und a-z nicht zu, so dass sie in
mehrsprachigen Dokumenten zu Problemen führen kann.

Tabelle 9.7: Glyphentabelle eines LGR-kodierten Fonts (grmn1000)

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
α	β	γ	δ	ε	φ	χ	η	ι	θ	κ	λ	μ	ν	ξ	ο	π	ρ	σ	τ	υ	ω	ξ	ψ	ζ	
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Α	Β	Γ	Δ	Ε	Φ	Χ	Η	Θ	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Ω	Ξ	Ψ	Ζ		

Bsp.
9-4-1

Tabelle 9.8: Griechische Transliteration mit lateinischen Buchstaben (LGR)

	Eingabe	Ergebnis	Beispiel
Akut	'a 'e 'h 'i 'o 'u 'w	ά έ ή ί ό ύ ώ	g'ata γάτα
Trema	"i "u "I "U	ϊ ü Ĩ Ÿ	qa"ide'uh c χαΐδεύης
Behaucht	<a <e <h <i <o <r <u <w	ǎ ǣ ħ ǫ ǫ ǫ ǫ	<'otan όταν
Unbehaut	>a >e >h >i >o >r >u >w	ǎ ǣ ħ ǫ ǫ ǫ ǫ	>'aneu άνευ
Gravis	'a 'e 'h 'i 'o 'u 'w	à è ħ ò ù ò	dad'i δαδì
Zirkumflex	~a ~h ~i ~u ~w	ǣ ħ ĩ ũ ǫ	ful~hc φυλής
Punkt darunter	a h w 'w 'w >'w >'w <'w <'w	ǣ ħ ǫ ǫ ǫ ǫ ǫ ǫ ǫ	

Bsp.
9-4-2

Tabelle 9.9: LGR-Ligaturen, die Glyphen mit einem Akzent erzeugen

Eingabe	Ergebnis
>'a >'e >'h >'i >'o >'u >'w	ǎ ǣ ħ ĩ ǫ ũ ǫ
>'A >'E >'H >'I >'O >'U >'W	ǎ ǣ ħ ĩ ǫ ũ ǫ
>'a >'e >'h >'i >'o >'u >'w	ǎ ǣ ħ ĩ ǫ ũ ǫ
>'A >'E >'H >'I >'O >'U >'W	ǎ ǣ ħ ĩ ǫ ũ ǫ
<'a <'e <'h <'i <'o <'u <'w	ǎ ǣ ħ ĩ ǫ ũ ǫ
<'A <'E <'H <'I <'O <'U <'W	ǎ ǣ ħ ĩ ǫ ũ ǫ
<'a <'e <'h <'i <'o <'u <'w	ǎ ǣ ħ ĩ ǫ ũ ǫ
<'A <'E <'H <'I <'O <'U <'W	ǎ ǣ ħ ĩ ǫ ũ ǫ
>~a >~h >~i >~u >~w	ǎ ħ ĩ ũ ǫ
>~A >~H >~I >~U >~W	ǎ ħ ĩ ǫ ũ ǫ
<~a <~h <~i <~u <~w	ǎ ħ ĩ ũ ǫ
<~A <~H <~I <~U <~W	ǎ ħ ĩ ǫ ũ ǫ
'"i '"i '"u '"u	ı ı ı ı

Bsp.
9-4-3

Tabelle 9.10: Verfügbare Kombinationen aus Spiritus- und Akzentzeichen

Als Alternative zu den genannten werden noch zwei weitere Fonts unterstützt, die zwar urheberrechtlich geschützt, aber bei CTAN frei erhältlich sind: Bei Hclassic handelt es sich um einen Font in „modernisiertem klassischem Hebräisch“ und Hcaption ist seine schräg gestellte Version. Außerdem gibt es noch drei Shalom-Fonts: ShalomScript10 (handgeschriebene hebräische Buchstaben), ShalomStick10 (serifenlose Buchstaben) und ShalomOldStyle10 (Buchstaben im alten Stil).

כותרת של מאמר שנכתב באמצעות L^AT_EX בעברית

רמה פורת

2 בפברואר 2002

1 זאת ההתחלה

אפשר להתחיל בעברית ואחר כך לעבור ללועזית and continue with English
אפשר גם לכלול נוסחאות $83 + a^2 = 102$ ולחזור לעברית.
text

1.1 תת סעיף בעבודה

* המצב לא טוב אבל יש תקווה לעתיד.

* יש לפגוע שולחן עם המון מטבעות. לכל מטבע צד לבן וצד שחור. עיניך
קשורות. אתה יודע ש 10 מטבעות הן עם הצד הלבן למעלה, והשאר עם
הצד השחור למעלה. עליך לחלק את המטבעות לשתי קבוצות, כך שבכל
קבוצה יהיה אותו מספר של לבנים (כלומר צד לבן למעלה).

לא מצאת את הפיתרון? - rama@huji.ac.il

Last section 2

אפשר להשתמש בכל האפשרויות של התוכנה בגירסתה הלועזית.

Bsp.
9-4-4

Abbildung 9.1: Ein hebräisch-englisches Dokument

Als weitere Schriftfamilie sind die Frank Ruehl-Fonts erhältlich, die über senkrechte, breitfette und schräg gestellte Formen verfügen. Die für Überschriften und Hervorhebungen gestaltete Carmel-Schriftfamilie bietet senkrechte und schräg gestellte Formen. Die Redis-Familie umfasst senkrechte, schräg gestellte und breitfette Schriftformen. Alle unterstützten Schriftfamilien lassen sich über Befehle des Paketes `hebfont` auswählen. Diese Befehle sind in Tabelle 9.12 auf der gegenüberliegenden Seite aufgelistet.

Die Unterstützung für die hebräische Sprache verfügt auch über einige wenige Eingabekodierungen. Sie sind nicht Bestandteil der `inputenc`-Distribution.

si960 Diese hebräische 7-Bit-Kodierung verwendet die für ASCII-Zeichen reservierten Positionen 32–127. Sie ist auch als „oldcode“ bekannt und entspricht dem israelischen Standard SI-960.

8859-8 Diese gemischte 8-Bit-Kodierung aus hebräischen und lateinischen Zeichen ist auch als „newcode“ bekannt. Sie entspricht der Standardkodierung ISO 8859-8.

	'0	'1	'2	'3	'4	'5	'6	'7	
'02x	א	ב	ג	ד	ה	ו	ז	ח	"1x
'03x	ט	י	כ	ל	מ	נ	ס	ע	
'04x	פ	צ	ק	ר	ש	ת			"2x
'05x	(	)			,		.	/	
'06x	.	"	'	:	\	-	~	^	"3x
'07x	~	~	:	:	~	.	~	?	
'10x	"	.	"	'	:	\	-	~	"4x
'11x	~	~	~	.	"	'	:	\	
'12x	-	'	~	~	~	:	~	.	"5x
'13x	.	.	#	/	\				
'14x	א	ב	ג	ד	ה	ו	ז	ח	"6x
'15x	ט	י	כ	ל	מ	נ	ס	ע	
'16x	פ	צ	ק	ר	ש	ת			"7x
'17x	ר	ש	ת						
	"8	"9	"A	"B	"C	"D	"E	"F	

Tabelle 9.11: Glyphentabelle eines LHE-kodierten Fonts (shold10)

Befehl	Entspricht der Deklaration	Fontfamilie	Beispiel
<code>\textjm</code>	<code>\rmfamily</code>	Jerusalem Font	מפומפ
<code>\textds</code>	<code>\bfseries</code>	Dead Sea Font	מפומפ
<code>\textoj</code>	<code>\itshape</code>	Old Jaffa Font	מפומפ
<code>\textta</code>	<code>\sffamily</code>	Tel Aviv Font	מפומפ
	<code>\ttfamily</code>		
<code>\textcrml</code>	<code>\fontfamily{crml}</code>	Carmel Fonts	מפומפ
<code>\textfr</code>	<code>\fontfamily{fr}</code>	Frank Ruehl Fonts	מפומפ
<code>\textredis</code>	<code>\fontfamily{redis}</code>	Redis Fonts	פ פ מ פ מ פ
<code>\textclas</code>	<code>\fontfamily{clas}</code>	Classic Fonts	מפומפ
<code>\textshold</code>	<code>\fontfamily{shold}</code>	Shalom Old Style Font	מפומפ
<code>\textshscr</code>	<code>\fontfamily{shscr}</code>	Shalom Script Font	מפומפ
<code>\textshstk</code>	<code>\fontfamily{schstk}</code>	Shalom Stick Font	מפומפ

Bsp.
9-4-5

Tabelle 9.12: Fontwechselbefehle für hebräische Texte

cp862 Diese IBM-Codepage wird zurzeit von MS-DOS auf IBM-kompatiblen PCs verwendet. Sie ist auch als „pccode“ bekannt.

cp1255 Die MS Windows-Codepage 1255 (Hebräisch) gleicht dem Standard ISO 8859-8. Neben hebräischen Buchstaben weist diese Kodierung auch Vokale und Punktierungen (Nikud) auf.

9.5 Anpassen des babel-Systems

Dieser Abschnitt erklärt einige der Befehle, die der Kern des babel-Paketes für die Programmierung von Sprachdefinitionsdateien zur Verfügung stellt. Diese Dateien werden normalerweise geladen, wenn eine Sprachoption angefordert wird. In Abschnitt 9.5.3 folgt ein genauerer Blick auf die Vorlagendatei `language.skeleton`, die sich als Basis für die Unterstützung weiterer Sprachen nutzen lässt.

Sprachdefinitionsdateien (Dateierweiterung `.ldf`) müssen eine Reihe von Konventionen beachten, da sie den gemeinsam genutzten Programmcode der Datei `babel.def` ergänzen, mit dessen Hilfe das babel-System sprachspezifische Texte erzeugt. Ebenso gelten gewisse Regeln für die Umschaltung zwischen verschiedenen Sprachen, wie babel sie ermöglicht. Es gelten folgende Voraussetzungen:

- Jede Sprachdefinitionsdatei `<sprache>.ldf` muss fünf Makros definieren, die dann im weiteren Verlauf zum Aktivieren und Deaktivieren der sprachspezifischen Definitionen dienen. Bei diesen Makros handelt es sich um: `\<sprache>hyphenmins`, `\<sprache>captions`, `\<sprache>date`, `\<sprache>extras` und `\<sprache>noextras`, wobei `<sprache>` entweder der Name der Sprachdefinitionsdatei oder einer babel-Paketooption ist. Die Makros und ihre Funktionen werden im Folgenden noch beschrieben.
- Wird eine Sprachdefinitionsdatei geladen, kann sie `\l@<sprache>` als Variante (*Dialekt*) von `\language0` definieren, sofern `\l@<sprache>` nicht bereits definiert ist.
- Sprachdefinitionsdateien müssen so programmiert sein, dass sie nicht nur in der Präambel, sondern auch während der laufenden Verarbeitung eines Dokumentes gelesen werden können.

9.5.1 Silbentrennung für verschiedene Sprachen

Seit T_EX in der Version 3.0 lassen sich Trennmuster für verschiedene Sprachen gemeinsam verwenden. Diese Muster müssen irgendwie verwaltet werden. Insbesondere Plain-T_EX-Benutzer müssen wissen, für welche Sprachen Trennmuster geladen wurden und welchem Wert des Befehls `\language` sie entsprechen. Das babel-Paket abstrahiert von dieser Low-Level-Schnittstelle und verwaltet die entsprechenden Informationen in einer externen Datei namens `language.dat`. Darin wird festgehalten, für welche Sprachen Trennmuster vorliegen *und* in welchen Dateien diese gespeichert sind. Diese Konfigurationsdatei wird verarbeitet¹, wenn INIT_EX ein neues L^AT_EX-Format generiert. Eine solche Datei kann folgendermaßen aussehen:

```
%%% Filename      : language.dat
%%% Description  : Instruct iniTeX which pattern files to load.
```

¹Man sollte darauf achten, dass die T_EX-Installation stets nur eine solche Datei aufweist, da ansonsten nicht immer eindeutig zu klären ist, welche von ihnen für die Formaterstellung herangezogen wird. Bei der Produktion des Buches waren die Autoren recht verblüfft, als INIT_EX sich der Systemkonfigurationsdatei bediente und nicht der eigens erstellten, die alle Muster für die Beispiele enthielt.

english	ushyph.tex	% amerikanisches Englisch
=USenglish		
=american		
russian	ruhyph.tex	% Russisch
french	frhyph.tex	frhyphx.tex % Französisch
=patois		
=français		
UKenglish	gbhyph.tex	% britisches Englisch
=british		
%german	dehyph.tex	% Deutsch, alte Rechtschreibung (nicht geladen)
ngerman	dehyphn.tex	% Deutsch, neue Rechtschreibung
%dutch	nehyp96.tex	% Niederländisch (nicht geladen)
dumylang	dumyhyph.tex	% Zum Testen neuer Sprachen
nohyphenation	zerohyph.tex	% Sprache ohne Trennmuster

Die Konfigurationsdatei `language.dat` kann leere Zeilen und Kommentare enthalten, sowie Zeilen, die mit einem Gleichheitszeichen (=) beginnen. Solche Zeilen weisen $\LaTeX$ darauf hin, dass das gerade verarbeitete Trennmuster auch unter einem anderen Namen verwendet wird. Das erste Element jeder Zeile gibt den Namen der Sprache an. Ihm folgt der Name der Datei, welche die Trennmuster enthält. Ein optionaler dritter Eintrag verweist auf eine Datei mit Ausnahmeregelungen für die Silbentrennung, sofern diese in einer gesonderten Datei gespeichert sind (z.B. `frhyphx.tex` im vorigen Beispiel).

Für jede Sprache in der Datei `language.dat` wird im $\LaTeX$ -Format ein Befehl `\l@<sprache>` definiert (also `\l@english` usw.). Bei der Verarbeitung des Dokumentes mithilfe eines solchen Formates überprüft `babel` für jede Sprache, ob der Befehl `\l@<sprache>` definiert ist und lädt, sofern dies zutrifft, die zugehörigen Trennmuster. Andernfalls werden die Muster für die voreingestellte Sprache 0 geladen (die als erste von $\INITEX$ geladen wurde). Aus Gründen der Kompatibilität sollte diese Sprache Trennmuster für amerikanisches Englisch enthalten.

```

initex latex.ltx
This is TeX, Version 3.14159 (Web2C 7.3.3.1) (INITEX)
(/tex/texmf/tex/latex/base/latex.ltx
....
24 hyphenation exceptions
Hyphenation trie of length 33878 has 835 ops out of 1501
  2 for language 5
 207 for language 4
 224 for language 3
  86 for language 2
 135 for language 1
 181 for language 0
No pages of output.
```

Entsprechend der Definitionen in der Datei `language.dat` werden sieben „Sprachen“ in das Format geladen: `english` (0), `russian` (1), `french` (2), `UKenglish` (3), `german` (4), `dumylang` (5) und `nohyphenation` (6; nicht gelistet da ohne Trennmuster). Diese Zeichenfolgen, bzw. ihre Entsprechungen

mit vorangestelltem Gleichheitszeichen (=) in der Datei `language.dat`, verwendet `babel` um eine Sprache zu identifizieren.

Wenn die Datei `language.dat` während des L^AT_EX-Laufs nicht gelesen werden kann, versucht `babel` stattdessen die voreingestellte Datei für die Silbentrennung, `hyphen.tex`, zu verwenden. Der Anwender erhält in diesem Falle eine entsprechende Meldung.

9.5.2 Die Paketdatei

Um die Funktionen von L^AT_EX besser nutzen zu können, enthält das `babel`-Paket eine Paketdatei namens `babel.sty`. Diese Datei wird mit dem `\usepackage`-Befehl geladen und definiert alle von `babel` unterstützten Sprachoptionen (siehe Tabelle 9.1 auf Seite 559). Sie regelt außerdem einige Kompatibilitätsprobleme mit anderen Paketen. Mithilfe der Konfigurationsdatei `bblopts.cfg`, die am Ende von `babel.sty` gelesen wird, kann man `babel` an lokale Erfordernisse anpassen.

Neben den in Tabelle 9.1 auf Seite 559 aufgeführten Sprachoptionen deklariert `babel` vorab einige Optionen, die sich auf das Verhalten der Sprachdefinitionsdateien auswirken können. Die Optionen `activeacute` und `activegrave` bewirken beispielsweise gemäß der Voreinstellungen zunächst nichts. Im Katalanischen (`catalan.ldf`) aktivieren sie jedoch die Akzente Akut und Gravis, sofern die entsprechenden Optionen angegeben sind.

Eine dritte Option, `KeepShorthandsActive`, weist `babel` an, die Kurzzeichen nach der Verarbeitung der Paketdatei nicht zu deaktivieren. Man beachte, dass dies *nicht* das voreingestellte Verhalten ist, da es zu Problemen mit anderen Paketen führen kann. Trotzdem kann sich diese Option als hilfreich erweisen, etwa, wenn man in der Präambel eines Dokumentes Kurzzeichen verwenden muss.

Kurzformen
normalerweise erst
nach der Präambel
aktiv

9.5.3 Der Aufbau der babel-Sprachdefinitionsdatei

Die `babel`-Distribution enthält die Datei `language.skeleton`, die sich gut als Grundgerüst zum Entwickeln einer eigenen Sprachdatei für eine neue Sprache eignet. Sie ist ein praktisches Beispiel für die richtige Verwendung der zentralen `babel`-Befehle. Die Datei wird im Folgenden gezeigt und die darin verwendeten Befehle werden in der Reihenfolge ihres Erscheinens erläutert.

In der Datei `language.skeleton` kommt immer wieder die Zeichenfolge „(language)“ vor. Diese sollte durch den Namen derjenigen Sprache ersetzt werden, welche durch die Datei unterstützt wird. Verfügt diese Sprache über einen Dialekt, der etwas andere Anforderungen stellt, so lassen sich diese ebenfalls erfüllen. In diesem Fall muss die Zeichenfolge „(dialect)“ durch den Namen des Dialekts ersetzt werden. Wird kein Dialekt unterstützt, dann sollte man die entsprechenden Zeilen aus der Datei entfernen.

Copyright und
Einleitung

Die Datei beginnt mit einem Copyright-Vermerk und Lizenzdaten.

```
1 % \iffalse meta-comment
2 %
3 % Copyright 1989-2003 Johannes L. Braams and any individual authors
4 % listed elsewhere in this file. All rights reserved.
5 %
6 % This file is part of the Babel system release 3.7.
```

```

7 % -----
8 %
9 % This work may be distributed and/or modified under the
10 % conditions of the LaTeX Project Public License, either version 1.3
11 % of this license or (at your option) any later version.
12 % The latest version of this license is in
13 % http://www.latex-project.org/lppl.txt
14 % and version 1.3 or later is part of all distributions of LaTeX
15 % version 2003/12/01 or later.
16 %
17 % This work has the LPPL maintenance status "maintained".
18 %
19 % This Current Maintainer of this work is Johannes Braams.
20 %
21 % \fi
22 % \Checksum{0}
23 %%% docstring = " This file can act as a template for
24 %%% people who want to provide extra
25 %%% languages to be included in the babel
26 %%% distribution.
27 %

```

Danach folgen Angaben zur Datei und der unterstützten Sprache.

Festlegen der Sprache

```

28 %<dt>
29 % \iffalse
30 % Tell the \LaTeX\ system who we are and write an entry in the
31 % transcript file.
32 \ProvidesFile{<language>.dtx}
33 %</dt>
34 %<code>\ProvidesLanguage{<language>}
35 %\fi
36 %\ProvidesFile{<language>.dtx}
37 [2003/03/18 v1.5 <Language> support from the babel system]

```

`\ProvidesLanguage{<name>}[<versionsdaten>]`

Der Befehl `\ProvidesLanguage` (Zeile 34) dient der Identifikation der Sprachdefinitionsdatei. Er verwendet die gleiche Syntax wie `\ProvidesPackage`. Die Datei `welsh.ldf` z.B. enthält die folgende Deklaration:

```
\ProvidesLanguage{welsh}
```

Mithilfe des optionalen Argumentes *versionsdaten* kann man angeben, dass mindestens diese Version von babel benötigt wird.

Im nächsten Abschnitt wird dann ein Dokumentationstreiber eingerichtet, um zu ermöglichen, dass die Datei selbst mithilfe des doc-Paketes gesetzt werden kann. Nähere Einzelheiten dazu sind in Kapitel 14 beschrieben.

*Ein Dokumentations-
treiber*

```

38 %\iffalse
39 %% Babel package for LaTeX version 2e
40 %% Copyright (C) 1989 -- 2003
41 %% by Johannes Braams, TeXnief
42 %
43 %% Please report errors to: J.L. Braams
44 %% babel@braams.cistron.nl
45 %
46 % This file is part of the babel system, it provides the source code for
47 % the <Language> language definition file.
48 %<filedriver>
49 \documentclass{ltxdoc}
50 \newcommand*{\TeXhax}{\TeX hax}
51 \newcommand*{\babel}{\textsf{babel}}

```

```

52 \newcommand*{\langvar}{${\langle \mathit lang \rangle}$}
53 \newcommand*{\note}[1]{ }
54 \newcommand*{\Lopt}[1]{\textsf{#1}}
55 \newcommand*{\file}[1]{\texttt{#1}}
56 \begin{document}
57 \DocInput{<language>.dtx}
58 \end{document}
59 %</filedriver>
60 %\fi
61 %% \GetFileInfo{<language>.dtx}
62 %

```

Dokumentation und Initialisierung

Der nächste Abschnitt beginnt mit der Beschreibung der Funktionen der Sprachdefinitionsdatei. Mithilfe der in Kapitel 14 beschriebenen Methoden wird der Programmcode dokumentiert und eine kurze Bedienungsanleitung hinzugefügt.

```

63 % \changes{v1.1}{1994/02/27}{Rearranged the file a little}
64 % \changes{v1.2}{1994/06/04}{Update for \LaTeXe}
65 % \changes{v1.3}{1995/05/13}{Update for \babel\ release 3.5}
66 % \changes{v1.4}{1996/10/30}{Update for \babel\ release 3.6}
67 % \changes{v1.5}{1997/03/18}{Update for \babel\ release 3.7}
68 %
69 % \section{The <Language> language}
70 %
71 % The file \file{\filename}\footnote{The file described in this
72 % section has version number \fileversion\ and was last revised on
73 % \filedate.} defines all the language definition macros for the
74 % <Language> language.
75 %
76 % \StopEventually{ }
77 %
78 % The macro \LdfInit takes care of preventing that this file is
79 % loaded more than once, checking the category code of the
80 % \texttt{@} sign, etc.
81 % \begin{macrocode}
82 %<*code>
83 \LdfInit{<language>}{captions<language>}
84 % \end{macrocode}
85 %

```

\LdfInit

Das Makro \LdfInit (Zeile 83) führt eine Reihe von Standardüberprüfungen durch, die am Anfang einer Sprachdefinitionsdatei erfolgen müssen. Es kontrolliert z.B., welcher Kategorie das @-Zeichen zugeordnet ist, und verhindert, dass die .ldf-Datei zweimal verarbeitet wird.

Definieren der Sprache und ihrer Dialekte

```

86 % When this file is read as an option, i.e. by the \usepackage|
87 % command, \texttt{<language>} could be an ‘unknown’ language in
88 % which case we have to make it known. So we check for the
89 % existence of |\l@<language>| to see whether we have to do
90 % something here.
91 %
92 % \begin{macrocode}
93 \ifx\undefined\l@<language>
94 \nopatterns{<Language>}
95 \adddialect\l@<language>0\fi
96 % \end{macrocode}
97 % For the <Dialect> version of these definitions we just add a
98 % ‘‘dialect’’. Also, the macros |\captions<dialect>| and
99 % |\extras<dialect>| are |\let| to their \texttt{<language>}
100 % counterparts when these parts are defined.
101 % \begin{macrocode}
102 \adddialect\l@<dialect>\l@<language>

```

```

103 % \end{macrocode}
104 % The next step consists of defining commands to switch to (and
105 % from) the <Language> language.
106 %

```

```
\adddialect{\l@variante}{\l@sprache}
```

Der Befehl `\adddialect` fügt den Namen einer Sprachvariante (Dialekt) `\l@variante` hinzu, für welche die bereits definierten Silbentrennungsmuster der Sprache *sprache* gelten.¹ Um mehrere Varianten zu einer Sprache aufzulisten, kann man diesen Abschnitt beliebig oft wiederholen.

Die Bezeichnung „Dialekt“ hat sich so eingebürgert, auch wenn sie den Sachverhalt nicht wirklich trifft. Die Begriffe *sprache* und *variante* in *babel* enthalten keinerlei Abstufung und bedeuten keinesfalls, dass eine der Sprachen eine Hauptsprache darstellt. Die Vorstellung von „Dialekten“ ist einfach nur sehr anschaulich, wenn man davon spricht, dass sich mehrere Sprachen die gleichen Muster für die Silbentrennung teilen. Außerdem gilt: Wurden beim Erzeugen des $\LaTeX$ -Formats vorab keine Trennmuster für die jeweilige *sprache* geladen, dann definiert *babel* diese Sprache normalerweise als „Dialekt“ der voreingestellten Sprache `\language0`.

Die erste der beiden folgenden Befehlszeilen besagt beispielsweise, dass man für die österreichische Sprache die deutschen Trennmuster verwenden kann, wie sie in der Datei `german.ldf` definiert sind. Die zweite Zeile bedeutet, dass Nynorsk die gleichen Trennmuster verwendet, wie Norwegisch (aus der Datei `norsk.ldf`).

```

\adddialect{\l@austrian}{\l@german}
\adddialect{\l@nynorsk}{\l@norsk}

```

Das folgende Beispiel zeigt, wie man mithilfe des Dialektmechanismus Sprachvarianten erhält, die andere Namen für Gliederungselemente oder andere Datumsformate verwenden.

	<code>\usepackage[dutch,afrikaans,norsk,nynorsk,ngerman]{babel}</code>	
Dialektvarianten:	Dialektvarianten:	<code>\par</code>
Norwegisch: Bibliografi	<code>\selectlanguage{norsk}</code> Norwegisch:	<code>\bibname \par</code>
Nynorsk: Litteratur	<code>\selectlanguage{nynorsk}</code> Nynorsk:	<code>\bibname \par</code>
Niederländisch: 29 februari 2004	<code>\selectlanguage{dutch}</code> Niederländisch:	<code>\today \par</code>
Afrikaans: 29 Februarie 2004	<code>\selectlanguage{afrikaans}</code> Afrikaans:	<code>\today</code>

Bsp.
9-5-1

Im nächsten Teil wird besprochen, wie man, soweit erforderlich, Sprachattribute am besten einrichtet.

Definieren von Sprachattributen

```

107 % Now we declare the |<attrib>| language attribute.
108 % \begin{macrocode}
109 \bbl@declare@ttribute{<language>}{<attrib>}{%
110 % \end{macrocode}
111 % This code adds the expansion of |\extras<attrib><language>| to
112 % |\extras<language>|.

```

¹Beim Laden der Trennmuster durch $\INIT_{\text{E}}\text{X}$ verwendet *babel* den Befehl `\addlanguage` um die verschiedenen in der Datei `language.dat` aufgeführten Sprachen zu deklarieren (vgl. Abschnitt 9.5.1).

```

113 % \begin{macrocode}
114 \expandafter\addto\expandafter\extras<language>
115 \expandafter{\extras<attrib><language>}%
116 \let\captions<language>\captions<attrib><language>
117 }
118 % \end{macrocode}
119 %

```

`\bbl@declare@ttribute{sprache}{attr}{code}`

Dieser Befehl (in Zeile 109) legt fest, dass für das Attribut *attr* in der Sprache *sprache* die Befehle *code* ausgeführt werden. In der Datei `greek.ldf` wird z.B. das Attribut `polutoniko` für die griechische Sprache definiert:

```
\bbl@declare@ttribute{greek}{polutoniko}{...}
```

Wenn man Griechisch mit der Option `polutonikogreek` lädt (was gleichbedeutend mit dem Setzen des Attributes `polutoniko` ist), werden griechische Texte mit mehreren Akzenten gesetzt (entsprechend der Anweisungen im dritten Argument).

Um mehrere Attribute für die aktuelle Sprache zu definieren, kann man diesen Abschnitt beliebig oft wiederholen.

Justieren der Trennmuster

Nun geht es um die Mindestanzahl an Zeichen, die links und rechts der Trennpositionen verbleiben müssen.

```

120 % \begin{macro}{\<language>hyphenmins}
121 % This macro is used to store the correct values of the hyphenation
122 % parameters |\lefthyphenmin| and |\righthyphenmin|.
123 % \begin{macrocode}
124 \providehyphenmins{<language>}{\tw@\thr@@}
125 % \end{macrocode}
126 % \end{macro}
127 %

```

`\providehyphenmins{sprache}{minimum} \<sprache>hyphenmins`

Der Befehl `\providehyphenmins` (Zeile 124) initialisiert den Parameter `\<sprache>hyphenmins`, sofern dieser nicht aus irgendeinem Grund bereits definiert ist. Dieser Parameter legt *Voreinstellung* für die Trennparameter `\lefthyphenmin` (Mindestanzahl an Zeichen links der ersten Trennposition) und `\righthyphenmin` (Mindestanzahl rechts der Trennposition) für die jeweilige *sprache* fest. Das `babel`-Paket erkennt, ob eine Trennmuster-Datei explizit Werte für `\lefthyphenmin` und `\righthyphenmin` festlegt und definiert in diesem Fall `\<sprache>hyphenmins` automatisch. In diesem Fall hat die Deklaration `\providehyphenmins` keine Auswirkungen.

Die `babel`-Syntax geht auf die Zeit zurück, als noch jedes Bit zählte, und ist somit speicheroptimiert. Darum besteht das Argument *minimum* einfach nur aus zwei Ziffern, in der Annahme, dass man niemals ein Minimum benötigt, das größer ist als neun. Trifft diese Annahme nicht zu, dann muss man die Werte innerhalb von *minimum* in geschweifte Klammern setzen. Zum Beispiel würde `\providehyphenmins{german}{10}{5}` bedeuten, dass vor einem Trennstrich mindestens zehn und danach mindestens fünf Buchstaben stehen müssten (so dass eine Trennung so gut wie unmöglich wäre).

Wer die Einstellungen ohne Rücksicht auf existierende Angaben explizit überschreiben möchte, kann selbst einen Wert für den Parameter `\<sprache>hyphenmins` direkt festlegen. Durch die Zeile

```
\def\germanhyphenmins{43}
```

kommt keine Trennposition in Frage, an der weniger als vier Buchstaben vor und weniger als drei nach dem Trennstrich stehen. Somit werden niemals Wörter mit weniger als sieben Zeichen getrennt.

Trennmuster werden immer im Bezug auf bestimmte Werte für diese Parameter erstellt. Eine Reduzierung dieser Werte unter diejenigen, die beim Erzeugen der Trennmuster verwendet wurden, führt einfach zu Fehlern bei der Silbentrennung. Man kann jedoch höhere Werte benutzen und vermindert so lediglich die Anzahl der möglichen Trennpositionen.

Als nächstes werden die Übersetzungen der sprachspezifischen Zeichenfolgen eingerichtet.

*Übersetzungen für
sprachspezifische
Zeichenfolgen*

```
128 % \begin{macro}{\captions<language>}}
129 %   The macro |\captions<language>| defines all strings used in the
130 %   four standard documentclasses provided with \LaTeX.
131 % \begin{macrocode}
132 \def\captions<language>{}
133 % \end{macrocode}
134 % \end{macro}
135 %
136 % \begin{macro}{\captions<dialect>}}
137 % \begin{macrocode}
138 \let\captions<dialect>\captions<language>
139 % \end{macrocode}
140 % \end{macro}
141 %
```

`\captions<sprache>{ersatztext-definitionen}`

Das Makro `\captions<sprache>` (Zeile 132) definiert die Makros, welche die Übersetzungen der sprachspezifischen Zeichenfolgen enthalten, die $\LaTeX$ für die Sprache `<sprache>` verwendet. Es muss für jeden einzurichtenden Dialekt gesondert aufgeführt werden. Wenn ein Dialekt auf die gleiche Übersetzung zugreift, lässt sich dies mit dem Befehl `\let` (wie in Zeile 138) festlegen. Andernfalls muss eine vollständige Definition erfolgen.

```
142 % \begin{macro}{\date<language>}}
143 %   The macro |\date<language>| redefines the command |\today| to
144 %   produce <Language> dates.
145 % \begin{macrocode}
146 \def\date<language>{%
147 }
148 % \end{macrocode}
149 % \end{macro}
150 %
151 % \begin{macro}{\date<dialect>}}
152 %   The macro |\date<dialect>| redefines the command |\today| to
153 %   produce <Dialect> dates.
154 % \begin{macrocode}
155 \def\date<dialect>{%
156 }
157 % \end{macrocode}
158 % \end{macro}
159 %
```

`\date<sprache>\{datums-definition}`

Das Makro `\date<sprache>` (Zeile 146) legt den Text für den `\today`-Befehl der Sprache `<sprache>` fest, die in der `.ldf`-Datei definiert wird.

Bereitstellen zusätzlicher Funktionen

Einige Sprachen (oder Dialekte) erfordern noch zusätzliche Definitionen. Diese folgen im nächsten Abschnitt.

```

160 % \begin{macro}\{extras<language>}
161 % \begin{macro}\{noextras<language>}
162 %   The macro |\extras<language>| will perform all the extra
163 %   definitions needed for the <Language> language. The macro
164 %   |\noextras<language>| is used to cancel the actions of
165 %   |\extras<language>|. For the moment these macros are empty but
166 %   they are defined for compatibility with the other
167 %   language definition files.
168 %
169 %   \begin{macrocode}
170 \addto\extras<language>{}
171 \addto\noextras<language>{}
172 %   \end{macrocode}
173 % \end{macro}
174 % \end{macro}
175 %
176 % \begin{macro}\{extras<dialect>}
177 % \begin{macro}\{noextras<dialect>}
178 %   Also for the ‘<dialect>’ variant no extra definitions are
179 %   needed at the moment.
180 %   \begin{macrocode}
181 \let\extras<dialect>\extras<language>
182 \let\noextras<dialect>\noextras<language>
183 %   \end{macrocode}
184 % \end{macro}
185 % \end{macro}
186 %
```

`\extras<sprache>\{zusätzliche definitionen}`

Das Makro `\extras<sprache>` (Zeile 170) enthält alle zusätzlich erforderlichen Definitionen für die Sprache `<sprache>`, die in der `.ldf`-Datei definiert wird. Dabei kann es sich z.B. um das An- oder Abschalten von Kurzformen handeln oder um das Aktivieren bestimmter Zeichen, das Einstellen des „French Spacing“, das Positionieren von Umlauten usw.

`\noextras<sprache>\{zusätzliche definitionen umkehren}`

Damit man zwischen zwei beliebigen Sprachen wechseln kann, muss T_EX zunächst wieder in einen eindeutigen Zustand zurückkehren können, insbesondere hinsichtlich der Definitionen, die durch den Befehl `\extras<sprache>` vorgenommen wurden. Das Makro `\noextras<sprache>` (Zeile 171) muss daher Befehle enthalten, die alle diese Definitionen umkehren und L^AT_EX so wieder in einen definierten Zustand zurückversetzen.

Aufräumen und Beenden

Die Datei endet mit den folgenden Programmzeilen:

```

187 %   The macro |\ldf@finish| takes care of looking for a
188 %   configuration file, setting the main language to be switched on
189 %   at |\begin{document}| and resetting the category code of
190 %   \texttt{@} to its original value.
191 %   \begin{macrocode}
192 \ldf@finish{<language>}
193 %</code>
194 %   \end{macrocode}
```

```

195 %
196 % \Finale
197 %\endinput

```

```
\ldf@finish{sprache}
```

Das Makro `\ldf@finish` (Zeile 192) führt eine Reihe von Aufgaben aus, die am Ende jeder `.ldf`-Datei anfallen. Das Argument *sprache* ist der Name der Sprache, der in der Sprachdefinitionsdatei definiert wurde. Zunächst überprüft das Makro, ob es auf dem System eine Datei namens *sprache.cfg* gibt, also eine Datei, die genauso heißt wie die Sprachdefinitionsdatei, jedoch mit der Erweiterung `.cfg`. Mithilfe dieser Datei kann man eine Sprachdefinitionsdatei um systemspezifische Aktionen ergänzen, wie etwa das Hinzufügen von Zeichenfolgen zu dem Befehl `\captions{sprache}`, um auch lokale Dokumentenklassen zu unterstützen, oder das Aktivieren oder Deaktivieren von Kurzformen für Akut- oder Gravis-Akzente. Insbesondere die *babel*-Distribution für die französische Sprache von Daniel Flipo verfügt über eine Datei namens *frenchb.cfg*, die einige (auskommentierte) ergänzende Definitionen zum Setzen französischer Texte enthält. Diese können vom Anwender (durch Entfernen des Kommentars) aktiviert werden, sofern sie nützlich erscheinen. Darüber hinaus setzt das Makro die Kategorie des `@`-Zeichens zurück und bereitet die Sprache vor, die am Anfang des Dokumentes aktiviert werden soll.

Zusätzliche Definitionen für die babel-Datenstrukturen

In einigen Zeilen (114, 170, 171) wurde der Befehl `\addto` verwendet, um eine der Datenstrukturen von *babel*, die Übersetzungen oder Befehle für eine bestimmte Sprache enthalten, zu erweitern.

```
\addto\csname{code}
```

Dieser Befehl erweitert die Definition des Befehls `\csname` um die im Argument *code* angegebenen $\TeX$ -Anweisungen. Dazu muss der Befehl `\csname` nicht zuvor bereits definiert worden sein. Die folgenden Zeilen sind beispielsweise der Datei *russianb.ldf* entnommen, in der die Befehle `\captionsrussian`, `\extrarussian` und `\noextrarussian` um zusätzliche Anweisungen erweitert werden.

```

\addto\captionsrussian{%
  \def\prefacename{%
    {\cyr\CYRP\cyrr\cyre\cyrd\cyri\cyrs\cyr\l\cyro\cyrv\cyri\cyre}}%
  ...
}
\addto\extrarussian{\cyrillictext}
\addto\noextrarussian{\latintext}
\initiate@active@char{"}
\addto\extrarussian{\languageshorthands{russian}}
\addto\extrarussian{\bbl@activate{"}}
\addto\noextrarussian{\bbl@deactivate{"}}

```


Befehle für Kurzformen auf Sprachenebene

Sprach- und systemspezifische Kurzformen werden in der Sprachdefinitionsdatei eingerichtet. Im vorigen Abschnitt wurde dieser Vorgang ansatzweise anhand eines Beispiels beleuchtet. In diesem Abschnitt folgt eine Beschreibung aller Befehle und Deklarationen, die sich für diesen Zweck verwenden lassen.

```
\initiate@active@char{zeichen}
```

Mithilfe dieses Makros kann man das angegebene *zeichen* in einer Sprachdefinitionsdatei in ein „Kurzzeichen“ umwandeln. Wurde das betreffende Zeichen bereits zuvor als Kurzzeichen definiert, so hat das Makro keine Auswirkungen. Andernfalls wird der Befehl `\normal@char{zeichen}` so definiert, dass er das angegebene *zeichen* in seinem „normalen Zustand“ erzeugt, und das aktive Zeichen wird so definiert, dass es per Voreinstellung den Befehl `\normal@char{zeichen}` aufruft. Anschließend kann seine Definition mithilfe des Befehls `\bbl@activate{zeichen}` dahingehend geändert werden, dass es zu `\active@char{zeichen}` expandiert. Ist ein Zeichen erst einmal aktiviert, so bleibt es in diesem Zustand, bis man es wieder deaktiviert oder das Ende des Dokumentes erreicht ist. Seine Definition lässt sich während der Formatierung eines Dokumentes jederzeit ändern.

Einige Sprachdefinitionsdateien aktivieren beispielsweise die Anführungszeichen mit dem folgenden Befehl:

```
\initiate@active@char{"}
```

Für die französische Sprache definiert die Konfigurationsdatei `frenchb.cfg` zwei Zeichen umfassende Kurzformen:

```
\initiate@active@char{<<} \initiate@active@char{>>}
```

```
\bbl@activate{zeichen} \bbl@deactivate{zeichen}
```

Der Befehl `\bbl@activate` „schaltet“ das aktive Verhalten des angegebenen *zeichens* „ein“, indem er dessen Definition dahingehend ändert, dass es den Befehl `\active@char{zeichen}` ausführt (anstelle des Befehls `\normal@char{zeichen}`). Der Befehl `\bbl@deactivate` hingegen lässt das aktive *zeichen* wieder den Befehl `\normal@char{zeichen}` ausführen. Die Kategorie (`\catcode`) des Zeichens wird von den Befehlen dabei nicht verändert – es bleibt weiterhin aktiv.

```
\textormath{text-code}{math-code}
```

Dieses Makro wird der Tatsache gerecht, dass manche in Sprachdefinitionsdateien deklarierte Kurzformen sowohl in Texten als auch in Formeln verwendbar sein müssen. Mit seiner Hilfe kann man unterschiedliche Anweisungen für den Textmodus (*text-code*) und den Formelmodus (*math-code*) festlegen. Wie

bereits auf Seite 458 erklärt, können Befehle, die für den Text- bzw. Formelmodus bestimmt sind, unerwünschte Nebeneffekte haben, so dass man sehr vorsichtig mit diesem Makro umgehen sollte.

```
\allowhyphens    \bbl@allowhyphens
```

Wenn $\LaTeX$ ein Wort nicht vernünftig trennen kann – beispielsweise, weil es sich um ein zusammengesetztes Wort handelt oder weil das Wort Akzentbuchstaben enthält, die mit dem Basisbefehl `\accent` erzeugt wurden – benötigt es etwas Unterstützung. Dazu lässt man $\LaTeX$ annehmen, dass es mit zwei Wörtern zu tun hat, die lediglich als ein Wort auf der Seite erscheinen. Zu diesem Zweck verfügt `babel` über den Befehl `\allowhyphens`, der einen unsichtbaren horizontalen Abstand einfügt, sofern die aktuelle Kodierung nicht T1 ist.¹ Möchte man diese kleine „Hilfe“ unbedingt einfügen, so kann man den Befehl `\bbl@allowhyphens` verwenden. Der unsichtbare Abstand lässt $\LaTeX$ annehmen, dass es sich um zwei Wörter handelt, die einzeln getrennt werden können.

```
\declare@shorthand{name}{kurzform}{exec}
```

Das Makro `\declare@shorthand` definiert Kurzformen um die Texteingabe für die jeweilige Sprache zu erleichtern. Das erste Argument, *name*, gibt den Namen der Sammlung von Kurzformen an, zu der die Definition gehört. Das zweite Argument, *kurzform*, besteht aus einem oder mehreren Zeichen für die zu definierende Kurzform. Das dritte Argument, *exec*, enthält die Befehle, die ausgeführt werden sollen, wenn die Kurzform im Dokument auftaucht. Es folgen ein paar Beispiele aus verschiedenen Sprachdefinitionsdateien.

```
\declare@shorthand{dutch}{\textormath{\ij}}{\ddot y}
\declare@shorthand{german}{a}{\textormath{"a}\allowhyphens}{\ddot a}
\declare@shorthand{french}{;}{...}
\declare@shorthand{system}{;}{\string;}
```

Die letzten beiden Anweisungen finden sich in der Datei `frenchb.ldf`. Die Erste von ihnen behandelt die Fälle, in denen das Zeichen `;` aktiv ist und ihr drittes Argument enthält Programmcode, der dafür sorgt, dass vor „hohen“ Satzzeichen (`;`, `:`, `!` und `?`) ein schmaler Zwischenraum eingefügt wird. Der letzte Befehl kümmert sich um die Fälle, in denen die französischen Regeln der Zeichensetzung deaktiviert sind (man beachte, dass diese vier Satzzeichen in der Datei `frenchb.ldf` zu aktiven Zeichen gemacht werden).

9.6 Andere Ansätze

Im Allgemeinen leistet das `babel`-Paket bei der Übersetzung der Namen für die Dokumentenelemente und bei der Vereinfachung der Texteingabe gute Arbeit. Für manche Sprachen haben einzelne Personen oder lokale Benutzergruppen

¹Im Gegensatz zur OT1-Kodierung enthält T1 echte Glyphen für die meisten Akzentbuchstaben, so dass der `\accent`-Basisbefehl so gut wie nie zum Einsatz kommt.

jedoch Pakete und T_EX-Versionen entwickelt, die eine tiefer gehende Anpassung des Systems an die Erfordernisse einer bestimmten Sprache vornehmen, insbesondere durch eine bessere Integration der typographischen Gepflogenheiten der Zielsprache.

Ein Beispiel dafür ist das Paket `french` [53, 68], das von Bernard Gaulle entwickelt wurde. Es gibt auch besonders angepasste Versionen von (L^A)T_EX (z.B. für Polnisch und Tschechisch, die von den T_EX-Benutzergruppen GUST bzw. C_STUG verbreitet werden).

9.6.1 Komplexere Sprachen

In der Welt nicht lateinischer Alphabete erreicht man eine neue Stufe an Komplexität, wenn man Sprachen wie Arabisch oder Hebräisch [142] verarbeiten möchte. Sie werden nicht nur von rechts nach links geschrieben, sondern im Arabischen ändert sich zudem die Form eines Buchstabens je nach seiner Position im Wort.

Bei CTAN sind verschiedene Systeme zum Setzen hebräischer Texte verfügbar (`language/hebrew`). Speziell `babel` bietet eine von Boris Lavva entwickelte Schnittstelle für die hebräische Sprache. Für arabische Texte gibt es das ArabT_EX-System [104] von Klaus Lagally. Dieses Paket erweitert (L^A)T_EX mithilfe einer ASCII-Transliteration um die Fähigkeit, arabische Schrift zu erzeugen (CTAN: `nonfree/language/arabtex`).

Serguei Dachian, Arnak Dalalyan und Vardan Hakobian haben eine Lösung für die armenische Sprache entwickelt (CTAN: `language/armtex`).

Die Lösungen für die Sprachen des indischen Subkontinents stützen sich größtenteils auf die Arbeit von Frans Velthuis. Kürzlich entwickelte insbesondere Anshuman Pandey Pakete für Bengali (das `bengali`-Paket und entsprechende Fonts bei CTAN: `language/bengali/pandey`), Sanskrit (das `devnag`-Paket von Anshuman Pandey bei CTAN: `language/devanagari/velthuis`) und Gurmukhi (CTAN: `language/gurmukhi/pandey`).

Mit dem Paket `manjutex` von Oliver Corff und Dorjpalam Dorj kann man Sprachen verarbeiten, die auf der Mandschuschrift (Mongolisch) basieren (CTAN: `language/manju/manjutex`).

Äthiopisch wird in `babel` durch das Paket `ethiop` und die zugehörigen Fonts (CTAN: `language/ethiopia/ethiop`) unterstützt. Sie stammen von Berhanu Beyene, Manfred Kudlek, Olaf Kummer und Jochen Metzinger.

Für Chinesisch, Japanisch und Koreanisch (die so genannten CJK-Schriften) kann man das `cjk`-Paket von Werner Lemberg [115] verwenden, das auch Fonts und nützliche Hilfsprogramme enthält (CTAN: `language/chinese/CJK`).

9.6.2 Omega

Wenn man darüber spricht, Dokumente in anderen Sprachen zu setzen, darf auch Omega [140] nicht fehlen, eine T_EX-Erweiterung, die von Yannis Haralambous und John Plaice entwickelt wurde. Das erklärte Ziel von Omega besteht darin, durch weitreichende Änderungen an T_EX (dem eigentlichen Programm) dessen Fähigkeiten zum Setzen von Texten in unterschiedlichsten Sprachen

deutlich zu verbessern. Mithilfe der folgenden Funktionen bietet es möglicherweise schon jetzt wesentlich einfachere Lösungen auf den von **babel** behandelten Gebieten:

- Omega kann Textdateien beliebiger Kodierung lesen (8-Bit, 16-Bit und mehr).
- Omega setzt Kurzformen intern um, indem es erkannte Sequenzen eingetragener Zeichen festgelegten Transformationen unterzieht.
- Omega verfügt über wesentlich flexiblere interne Strukturen zum Umgang mit großen Zeichenvorräten und umfangreichen Fonts als $\TeX$.
- Omega unterstützt viele verschiedene Arten von Schriften und alle heutzutage verwendeten Schreibrichtungen.

Diese Verbesserungen der Verarbeitungsmuster von $\TeX$ werden das Setzen von Texten in vielen verschiedenen Sprachen erleichtern: Arabisch, Bantu, Baskisch, Georgisch, Hindi, Khmer, Chinesisch, Cree oder Mongolisch – und alle im gleichen Dokument! Seit Ende 2003 besteht die Hoffnung, dass auch $\LaTeX$ bald so verbessert wird, dass es diese neuen Funktionen unterstützt und so zu einem echten mehrsprachigen $\LaTeX$ -System wird.

KAPITEL 10

Graphikgenerierung und -bearbeitung

10.1 Portable Graphiken und Zierrahmen	613
10.2 Geräteabhängige Graphikunterstützung	631
10.3 Manipulieren von graphischen Objekten	646
10.4 Seitenbeschreibungssprachen – PostScript, PDF und SVG. . . .	652

Unter den heutigen Satzsystemen für automatischen Computersatz besitzt $\text{\TeX}$ wahrscheinlich den besten Algorithmus, um Absätze zu formatieren und aus ihnen Seiten zu erstellen. Aber in dieser Zeit, in welcher der Informationsaustausch ständig zunimmt, beschränken sich die meisten Veröffentlichungen nicht mehr auf reine Texte – die Bedeutung von Graphiken ist stark angestiegen.

$\text{\TeX}$ befasst sich nicht selbst mit dieser Problematik, da es sich ausschließlich auf die Positionierung von (schwarzen) Boxen auf einer Seite konzentriert. Knuth hat jedoch eine Schnittstelle vorgesehen, um „Funktionen“, die in der Basissprache nicht verfügbar sind, über den `\special`-Befehl zu implementieren. Dieser Befehl hat keinen Einfluss auf die Formatierung der endgültigen Seite, doch er veranlasst $\text{\TeX}$, den Inhalt des Argumentes des `\special`-Befehls wortwörtlich an der gegenwärtigen Position in der `.dvi`-Datei zu platzieren.¹ Es ist dann Aufgabe des `dvi`-Treibers, die so erhaltenen Daten zu interpretieren und entsprechend das endgültige Erscheinungsbild zu erstellen (siehe auch [146]).

Der *L $\text{\TeX}$ Graphics Companion* [58, Kapitel 1] beschreibt ausführlich verschiedene Methoden und Techniken, die zur Erzeugung von Graphiken mit $\text{\TeX}$ verwendet werden können. Die folgende Liste gibt einen kurzen Überblick

¹Es gibt Situationen, in denen der `\special`-Befehl die Formatierung beeinflusst, da er zusätzliche Umbruchstellen erzeugen kann und $\text{\LaTeX}$ möglicherweise daran hindert, vorangegangenen Leerraum als solchen zu erkennen.

über die unterschiedlichen Ansätze. Interessierte Leser seien für weitere Einzelheiten auf das genannte Buch verwiesen.

1. ASCII-Zeichnungen, etwa erstellt durch PCTeX , das eine vollständige Sprache zur Erstellung von Zeichnungen zur Verfügung stellt. Kurven werden hierbei durch Kombination zahlreicher kleiner Punkte erzeugt.
2. Zeichensätze mit Bildelementen wie etwa $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ s `picture`-Umgebung. Das von Kristoffer Rose entwickelte `xypic`-System [58, Kapitel 5] verwendet z.B. spezielle Zeichensätze, um Diagramme zu setzen.
3. Bildmakropakete, die hauptsächlich auf der `picture`-Umgebung basieren oder auf $\text{T}_{\text{E}}\text{X}$ s Linienbefehlen. Unter anderem existieren Pakete zum Zeichnen von Feynman-Diagrammen, chemischen Formeln [58, Abschnitt 6.2] und Balken- und Baumdiagrammen (siehe Abschnitt 10.1.6).
4. Bildzeichensätze, in denen jedes Zeichen, das gesetzt werden soll, einer eventuell sehr großen „Drucktype“ in einem Zeichensatz entspricht. Zum Erstellen der Bilder kann man `METAFONT` oder `MetaPost` verwenden [58, Kapitel 3] oder auf bereits existierende Bitmaps zurückgreifen, die man direkt in eine `.pk`-Datei umwandeln kann [58, Abschnitt 1.3].
5. Halbton-Zeichensätze mit kleinen Flächen, die aus verschiedenen Graustufen bestehen. Zum Erstellen von Bildern können sie auf die herkömmliche $\text{T}_{\text{E}}\text{X}$ -Weise miteinander kombiniert werden [39, 97].
6. Graphiken, die mithilfe des `\special`-Befehls eingefügt werden können. Dieser Ansatz ist per Definition geräteabhängig, da er auf den Möglichkeiten des `dvi`-Treibers und des Ausgabegerätes aufsetzt. Das Paket `graphics`, das in Abschnitt 10.2 beschrieben wird, bietet Unterstützung auf höherer Ebene (durch Verkapselung der geräteabhängigen Aspekte) an. Da es inzwischen sehr viele kostengünstige PostScript-Drucker und Previewer gibt, erfreut sich dieser Ansatz großer Beliebtheit. Andere High-Level-Systeme, die $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ und PostScript zusammen verwenden, sind z.B. `psfrag` und `pstricks` [58, Kapitel 4].

Dieses Kapitel befasst sich mit den Verfahren zur Erzeugung portabler Graphiken (insbesondere jener, die auf Punkt 3 basieren) sowie auf die Schnittstelle zur Unterstützung von geräteabhängigen Graphiken (Punkt 6).

Der erste Abschnitt konzentriert sich vornehmlich auf die systemeigenen Zeichenwerkzeuge von $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. Es wird gezeigt, wie Zierrahmen erstellt werden können. Diese können nützlich sein, um wichtige Passagen hervorzuheben. Anschließend werden die Pakete `epic` und `eepic` erläutert, durch welche die `picture`-Umgebung um eine Reihe von neuen Befehlen erweitert wird. Sie werden im Detail besprochen, ergänzt durch Beispiele, die zeigen, wie sie in der Praxis verwendet werden.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ verfügt über eine allgemeine, treiberunabhängige Schnittstelle, um externe Graphiken zu importieren und $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ -Boxen zu skalieren und zu drehen.¹ Abschnitt 10.2 befasst sich mit der Einbindung von Graphikdateien.

¹Es gibt auch ein allgemeines Paket für Farbe; weitere Einzelheiten hierzu siehe das *$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ -Manual* [106] oder den *$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ Graphics Companion* [58].

Hierfür bietet $\LaTeX$ sowohl eine einfache Schnittstelle (das `graphics`-Paket, siehe Abschnitt 10.2.2), die mit diversen Befehlen zur Skalierung und zur Rotation kombiniert werden kann, als auch eine etwas komplexere Schnittstelle (`graphicx`, siehe Abschnitt 10.2.3), die eine Reihe eigener, leistungsfähiger Optionen zur Bildbearbeitung bereitstellt. Skalierung und Rotation sind das Thema in Abschnitt 10.3.

Im letzten Abschnitt werden die wichtigsten Seitenbeschreibungssprachen angeschnitten (PostScript, PDF, SVG). Außerdem wird kurz auf `dvips` eingegangen, ein häufig verwendetes Programm zur Übersetzung von `dvi` nach PostScript, und `pspicture` beschrieben. Letzteres ist eine Erweiterung von $\LaTeX$ s `picture`-Umgebung, die PostScript-Zeichenbefehle verwendet, die direkt auf den `dvips`-Treiber zugreifen.

10.1 Portable Graphiken und Zierrahmen

In $\LaTeX$ bezieht sich der Begriff „Portable Graphiken“ hauptsächlich auf Graphiken, die aus Boxen, Linien und Zeichen bestehen. Einen kurzen Überblick über die $\LaTeX$ -Boxen gibt Anhang A.2. An dieser Stelle werden zunächst Pakete vorgestellt, die Erweiterungen zu den herkömmlichen $\LaTeX$ -Boxen darstellen. Später befasst sich dieser Abschnitt mit Strichgraphiken.

10.1.1 boxedminipage – Gerahmte Boxen

Die Umgebung `boxedminipage`, die im Paket `boxedminipage` (von Mario Wolczko) definiert ist, verhält sich wie die Standardumgebung `minipage`, mit dem Unterschied, dass das Ergebnis von einem Rahmen umgeben ist, so als ob es in einer `\fbox` platziert worden wäre. Die Dicke der Linien und ihr Abstand zum Text wird durch die Stilparameter `\fboxrule` bzw. `\fboxsep` festgelegt. Im Gegensatz zu einer Konstruktion mit `\fbox` können innerhalb der Umgebung auch Verbatim-Befehle verwendet werden.

Dies ist ein Beispiel für eine kleine gerahmte `minipage`-Umgebung, die eine Fußnote^a und einen `\verb`-Befehl besitzt.

^aSehr einfaches Beispiel

```
\usepackage{boxedminipage}
\begin{boxedminipage}[t]{5cm}
  Dies ist ein Beispiel für eine kleine
  gerahmte minipage-Umgebung, die eine
  Fußnote\footnote{Sehr einfaches Beispiel}
  und einen \verb=\verb=-Befehl besitzt.
\end{boxedminipage}
```

Bsp.
10-1-1

10.1.2 shadow – Schattierte Boxen

Das Paket `shadow` (von Mauro Orlandini) definiert den Befehl `\shabox`, der dem $\LaTeX$ -Befehl `\fbox` ähnelt. Im Unterschied zu diesem wird beim `\shabox`-Befehl unterhalb und rechts von der Box ein „Schatten“ hinzugefügt.

Drei Parameter steuern das Aussehen der Box (die Standardwerte sind in Klammern angegeben): `\sboxrule` definiert die Dicke der Linien für den Rahmen (0.4pt); `\sboxsep` definiert den Abstand vom Rahmen zum Text (10pt)

und `\sdim` gibt die Breite der Schattierung an (4pt).

Ein ganzer Absatz kann hervorgehoben werden, indem er in einer Absatzbox gesetzt wird, die in einer `shabox` eingebettet wurde.

```
\usepackage{shadow}
\setlength\sdim{10pt}

\shabox{\parbox{6cm}{Ein ganzer
Absatz kann hervorgehoben werden,
indem er in einer Absatzbox gesetzt
wird, die in einer \texttt{shabox}
eingebettet wurde.}}
```

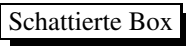
Bsp.
10-1-2

10.1.3 fancybox – Zierrahmen

Timothy Van Zandt entwickelte für sein `seminar`-Paket, mit welchem Folien erstellt werden, das `fancybox`-Paket. Es führt verschiedene neue Befehle zur Umrahmung von Texten, Bildern usw. in $\text{\LaTeX}$ ein. Dieser Abschnitt gibt lediglich einen Überblick über ein paar der grundlegenden Befehle. Weitere Informationen kann der interessierte Leser der Dokumentation entnehmen, die mit dem oben genannten `seminar`-Paket mitgeliefert wird.

Das Paket enthält vier Varianten für den Befehl `\fbox`. Wie beim `\fbox`-Befehl wird der Abstand des Rahmens zur Box durch den Längenparameter `\fboxsep` (voreingestellt in $\text{\LaTeX}$ ist 3pt) angegeben. Weitere Parameter, mit denen diese Boxen manipuliert werden können, sind unten aufgeführt.

Der Befehl `\shadowbox` fügt eine Schattierung der Breite `\shadowsize` ein (voreingestellt auf 4pt). Die Box wird unten an der Schattierung ausgerichtet, wodurch sich dieser Befehl weniger für die Einbettung im laufenden Text eignet als der zuvor besprochene Befehl `\shabox`. Man beachte die unterschiedlichen Voreinstellungen für die Abstände.

X  Y  Z

```
\usepackage{fancybox}
\usepackage{shadow}

X \shadowbox{Schattierte Box}
Y \shabox{Schattierte Box} Z
```

Bsp.
10-1-3

Der Befehl `\ovalbox` generiert einen Rahmen mit abgerundeten Ecken. Die Breite des Rahmens ist identisch mit der, die für Standardbildelemente erzeugt wird, wenn die Deklaration `\thinlines` verwendet wird. Der Befehl `\Ovalbox` ist ähnlich, wobei die Dicke der Linien denen der Deklaration `\thicklines` entspricht. Der Durchmesser der Eckbögen wird mit der Deklaration `\cornersize` festgelegt. Die Form `\cornersize{num}` stellt diesen auf $num \times \text{Minimum (Breite der Box, Höhe der Box)}$ ein, während die Form `\cornersize*{länge}` ihn auf die Länge *länge* setzt. Voreinstellung ist `\cornersize{0.5}`.

Dies ist eine ovale Box

Dies ist eine ovale Box

Dies ist eine
Ovalbox

```
\usepackage{fancybox}

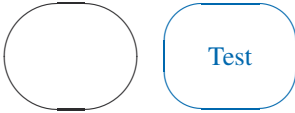
\centering

\ovalbox{Dies ist eine ovale Box}
\cornersize{1} \ovalbox{Dies ist eine ovale Box}
\\[8pt]
\setlength\fboxsep{6pt} \cornersize*{7mm}
\Ovalbox{\shortstack{Dies ist eine\\Ovalbox}}
```

Bsp.
10-1-4

Als Alternative zu L^AT_EXs Bildbefehl `\oval` bietet das Paket `\fancyoval` an. Während `\oval` stets den Durchmesser der Eckbögen so groß wie möglich macht, verwendet `\fancyoval` die Deklaration `\cornersize`, um den Durchmesser zu bestimmen.

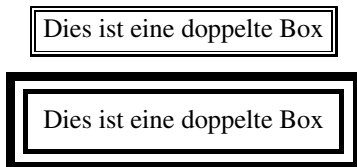
Bsp.
10-1-5



```
\usepackage{fancybox,color}
\cornersize{0.7}
\begin{picture}(110,40)
  \put(25,20){\oval(50,40)}
  \color{blue}
  \put(85,20){\makebox(0,0){Test}}
  \put(85,20){\fancyoval(50,40)}
\end{picture}
```

Schließlich hält das Paket den Befehl `\doublebox` bereit, der zwei rechteckige Rahmen generiert. Ihre Breite, ihr Verhältnis zueinander sowie das zum Text sind Bruchteile des `\fboxrule`-Parameterwertes: Die Breite des inneren Rahmens beträgt 0,75 von `\fboxrule` und die des äußeren Rahmens 1,5 von `\fboxrule`. Der Abstand zwischen den beiden Rahmen ist 1,5 von `\fboxrule` plus 0,5pt.

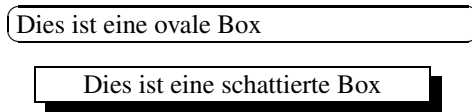
Bsp.
10-1-6



```
\usepackage{fancybox}
\centering
\doublebox{Dies ist eine doppelte Box} \\\[5pt]
\setlength{\fboxsep}{6pt} % default 3pt
\setlength{\fboxrule}{2pt}
\doublebox{Dies ist eine doppelte Box}
```

Im Gegensatz zu `\framebox` und `\makebox` hat keiner der obigen Befehle optionale Argumente. Dieselbe Funktionalität erhält man, indem man im Argument dieser Rahmenbefehle `\makebox` verwendet.

Bsp.
10-1-7



```
\usepackage{fancybox}
\centering
\cornersize{0.8}
\ovalbox{\makebox[6cm]{1}
  {Dies ist eine ovale Box}} \\\[8pt]
\shadowbox{\makebox[5cm]
  {Dies ist eine schattierte Box}}
```

Für einige Dokumentenarten, wie Folien, wäre es schön, wenn die Seiten gerahmt werden könnten – d.h. wenn Befehle, wie jene aus diesem Kapitel, in der Definition des Seitenstils verwendet werden könnten. Diese Möglichkeit wird durch das Paket `fancybox` über die Deklaration `\fancypage{innen}{außen}` geboten. Die vollständige Seite (vor dem Hinzufügen von Kopf- und Fußzeilen) wird gerahmt (entsprechend hat sie die Breite `\textwidth` und die Höhe `\textheight`) und dann in Form eines Argumentes an den Code übergeben, der in *innen* angegeben ist. Anschließend werden Kopf- und Fußzeilen hinzugefügt, wobei die neue Breite der Seite verwendet wird, sofern diese von *innen* geändert wurde. Das Ergebnis wird als

Argument an den Code in *außen* übergeben, der wiederum ein Argument erwartet. Im einfachsten Fall könnte man also einen dieser Rahmenbefehle aus diesem Kapitel verwenden oder sogar eins der Argumente leer lassen. Das nächste Beispiel zeigt eine Anwendung, in der die Argumente auch einige Parametereinstellungen enthalten, welche die Form der hinzugefügten Rahmen beeinflussen.

6 1 EIN TEST

Text für diese Seite, der immer und immer wieder verwendet wird.

1 Ein Test

Text für diese Seite, der immer und immer wieder verwendet wird.

```

\setlength\textwidth{160pt}
\setlength\textheight{7\baselineskip}
\pagestyle{headings}

\usepackage{fancybox}
\fancypage
{\setlength\fbboxsep{10pt}\ovalbox}
{\setlength{\fbboxsep}{8pt}}%
\setlength{\shadowsize}{8pt}}%
\shadowbox}

\newcommand\sample{ Text für diese Seite,
der immer und immer wieder verwendet wird.}
\sample \section{Ein Test} \sample

```

Bsp.
10-1-8

*Falsch positionierte
Kolumnentitel*

Man beachte, dass die Position der Kolumnentitel automatisch so korrigiert wurde, dass diese in die erweiterte Textbreite innerhalb des Rahmens passen. Diese Korrektur funktioniert jedoch nur für Standardseitenstile. Bei Verwendung von fancyhdr werden z.B. die endgültigen Kopf- und Fußzeilen manchmal zu schmal, da dieses Paket seine eigene Methode zum Erzeugen dieser Objekte verwendet.

ABC
XYZ

Text für diese Seite, der immer und immer wieder verwendet wird.

6

ABC
XYZ

1 Ein Test

Text für diese Seite, der immer und immer wieder verwendet wird.

7

```

\usepackage{fancyhdr}
\pagestyle{fancy}
\cfoot{\thepage}
\lhead{ABC} \rhead{XYZ}
% Kommentar in nächster Zeile
% entfernen für korrekte
% Ausrichtung des Kopfes:
% \fancyhfoffset[R]{20.8pt}
\usepackage{fancybox}
% \sample wie zuvor

\fancypage
{\setlength\fbboxsep{10pt}}%
\ovalbox}
{\setlength{\fbboxsep}{8pt}}%
\setlength{\shadowsize}{8pt}}%
\shadowbox}

\sample
\section{Ein Test} \sample

```

Bsp.
10-1-9

Im Fall von fancyhdr kann das Problem behoben werden, indem man mit `\fancyhfoffset` einen zusätzlichen Versatz hinzufügt. Der Wert 20.8pt

wurde von Hand berechnet als zweimal der Abstand zwischen Text und Rahmen (10pt) und der Breite der Rahmenlinie (0.4pt).

Die Deklaration `\fancypage` wird, beginnend mit der aktuellen Seite, auf alle Seiten angewendet, bis die nächste `\fancypage`-Deklaration im Dokument erscheint. Wenn nur auf der aktuellen Seite Rahmen eingefügt werden sollen, muss stattdessen `\thisfancypage` verwendet werden. „Aktuell“ bedeutet in diesem Kontext die zu erstellende Seite an der Stelle, an der L^AT_EX auf die Deklaration stößt, auch wenn diese Stelle im Dokument später auf einer anderen Seite dargestellt wird. In dieser Hinsicht verhält sich die Deklaration also genauso wie `\pagestyle`. Wenn dadurch Probleme entstehen, muss die Deklaration entweder an einer früheren oder späteren Stelle im Dokument eingefügt werden, oder vor der Deklaration muss der Befehl `\pagebreak` eingefügt werden. Dies verhindert, dass L^AT_EX zu weit vorausschaut.

Einschränkungen

Das andere potentielle Problem, das mit den Befehlen `\thisfancypage` und `\fancypage` auftreten kann, ist, dass sie die Ausgaberroutine von L^AT_EX ändern. Damit können sie normalerweise nicht mit anderen Paketen zusammenarbeiten, die das Gleiche tun (ein Beispiel hierfür ist `fancyhdr`, auch wenn bei vorsichtiger Anwendung beide durchaus koexistieren können). Fehlerhaft konstruierter Code in den Argumenten kann schwerwiegende Fehler verursachen – sie führen zu nicht aussagekräftigen Fehlermeldungen.

```
\fancyput*(x,y){horizontal-material}
```

Eine flexiblere Möglichkeit, auf jeder Seite Objekte an einer festen Position hinzuzufügen, bietet die Deklaration `\fancyput`. Sie hat eine ähnliche Syntax wie L^AT_EXs `\put`-Befehl, benötigt aber Maße für die *x*- und *y*-Koordinaten. Der Ursprung (0pt, 0pt) liegt jeweils einen Zoll vom oberen und linken Rand des Papiers. Um ein Objekt zwei Zoll vom linken und drei Zoll vom oberen Rand zu positionieren, muss man also Folgendes angeben: (1in, -2in).

Text für diese Seite, der immer und immer wieder verwendet wird.

1 Ein Test **DRAFT**

Text für diese Seite, der immer und immer wieder verwendet wird.

```
\usepackage{color,fancybox}
\fancyput(2in,-1.2in)
{\Huge\bfseries
 \textcolor{blue}{DRAFT}}
% \sample wie zuvor
\sample
\section{Ein Test} \sample
```

Bsp.
10-1-10

Die Variante `\thisfancyput` wirkt sich nur auf die aktuelle Seite aus, analog zu `\thisfancypage`. Bei Verwendung der Sternform (bei beiden Befehlen) wird der neue Text nicht ersetzt, sondern an den bestehenden Text angehängt, der vorher mit `\fancyput` oder `\thisfancyput` eingefügt worden ist.

Das Paket stellt auch Boxversionen der abgesetzten Standardumgebungen von L^AT_EX bereit. Die Größe der resultierenden Box wird durch die längste Zeile festgelegt. Alle Umgebungen unterstützen ein optionales Argument, das die Positionierung der Box im Verhältnis zu den Objekten auf der Zeile angibt; diese kann *t* für Ausrichtung oben oder *b* für Ausrichtung unten sein, standardmäßig wird die Box jedoch zentriert.

Abgesetzte Umgebungen in Boxen

Die Umgebungen `Bcenter`, `Bflushleft` und `Bflushright` erzeugen eine Box mit zentriertem (`centered`), linksbündigem (`flushleft`) oder rechtsbündigem Inhalt (`flushright`). Das Beispiel zeigt sie alle in Aktion. Man beachte die Verwendung des Befehls `\vspace`, der sicherstellt, dass die äußere `Bflushleft`-Box unten ausgerichtet wird. Zum Vergleich siehe auch die in Abschnitt A.2.2 auf Seite 896 besprochenen Beispiele.

```

\usepackage{fancybox}
\newcommand\HR{\rule{.5em}{0.4pt}}

\HR\begin{Bflushleft}[b]
\begin{Bflushleft}[t] A A A\\ A A A\\ A A
\end{Bflushleft} \HR
\begin{Bflushright}[t] B\\ B B B\\ B B B\\ B B
\end{Bflushright} \par\vspace{0pt}
\end{Bflushleft} \HR
\begin{Bcenter} C C C\\ C\\ C C\end{Bcenter} \HR

```

A A A _ B
A A A B B B
A A B B B C C C
_ B B _ C _
C C

Bsp.
10-1-11

`Bitemize`, `Benumerate` und `Bdescription` erzeugen Boxversionen der Umgebungen `itemize`, `enumerate` bzw. `description`. Intern wird $\LaTeX$ s `tabular`-Umgebung verwendet, was bedeutet, dass vertikal ausgerichtetes Material, wie `\vspace`, nicht funktioniert. Stattdessen verwendet der Befehl `\item` ein optionales Argument (mithilfe von runden Klammern!), um zusätzlichen Weißraum vor dem Listenpunkt einzufügen. Seine Verwendung wird im nächsten Beispiel demonstriert.

Für Mathematik-Anwendungen erzeugt `Beqnarray` eine gerahmte Umgebung, die der von `eqnarray` erzeugten ähnelt, wobei die Gleichungsnummer stets auf der rechten Seite steht. `Beqnarray*` wirkt ähnlich wie `eqnarray*`, doch die entstandene Box ist gerade groß genug, damit alle Gleichungen hineinpassen. Ein optionales Argument für die Position wird nicht unterstützt.

Test: • Erster Listenpunkt.

• Zweiter
über zwei Zeilen.

• Dritter mit extra Leerraum.

• Und wieder ohne.

```
\usepackage{fancybox}
```

```

Test: \fbox{\begin{Bitemize}[t]
\item Erster Listenpunkt.
\item Zweiter\\ über zwei Zeilen.
\item(2pt) Dritter mit extra Leerraum.
\item Und wieder ohne.
\end{Bitemize}}

```

```
\par\bigskip
```

```

Test: \fbox{\begin{Beqnarray}
& y & = & x^2 & \\
a^2 + 2ab + b^2 & = & (a + b)^2 & \\
\int_0^\infty e^{-ax} dx & = & \frac{1}{a} & \\
\end{Beqnarray}}

```

Test:

$$\begin{array}{rcl}
 y & = & x^2 \quad (1) \\
 a^2 + 2ab + b^2 & = & (a + b)^2 \quad (2) \\
 \int_0^\infty e^{-ax} dx & = & \frac{1}{a} \quad (3)
 \end{array}$$

Bsp.
10-1-12

Das Paket implementiert auch einige Befehle zum Erzeugen wortwörtlicher Ausgaben. Für solche Anwendungen bietet allerdings das `fancyvrb`-Paket vom selben Autor bessere Benutzerschnittstellen (siehe Abschnitt 3.4.3) an.

10.1.4 epic – Eine erweiterte picture-Umgebung

Standard- $\text{\LaTeX}$ bietet eine `picture`-Umgebung, mit der aus Strichen und Punkten aufgebaute Graphiken von im Prinzip beliebiger Komplexität über Basisbefehle zum Zeichnen von Linien, Vektoren, Viertelkreisen und Bézier-Kurven generiert werden können. Das Erstellen komplexer Graphiken ist zwar möglich, erfordert aber einen hohen manuellen Einsatz. Die meisten Graphikbefehle für Bilder benötigen eine explizite Angabe der Koordinaten für jedes *Objekt*. Mit den High-Level-Befehlen kann die Anzahl der Koordinaten reduziert werden, die manuell berechnet werden müssen. Grundsätzlich kann man zwei verschiedene Ansätze heranziehen, um solche Befehle zu entwerfen:

- Man kann Befehle definieren, die Objekte gemäß ihrer Dimensionen relativ zueinander platzieren – `\shortstack` fällt in diese Kategorie.
- Es werden Befehle zur Verfügung gestellt, die den Großteil der Berechnung intern vornehmen und nur die Angabe von einigen wenigen Koordinatenpaaren benötigen – `\multipt` ist ein Beispiel für diesen Ansatz.

Der offensichtliche Vorteil, Befehle zu verwenden, die in die obengenannten Kategorien fallen, besteht nicht nur darin, dass sie anfangs leichter einzugeben sind, sondern auch darin, dass bei einer folgenden Veränderung der Positionierung der Aufwand für Neuberechnungen minimiert wird.

Der häufig verwendete Low-Level-Befehl `\line` hat starke Beschränkungen und Schwächen. Seine Argumente sind nicht intuitiv verständlich und benötigen umfangreiche Berechnungen – der Denkprozess beim Schreiben eines `\line`-Befehls beinhaltet häufig folgende Schritte:

1. Berechnung der Koordinaten für die beiden Endpunkte.
2. Berechnung des horizontalen und vertikalen Abstandes.
3. Übersetzung dieser Abstände in ein (x, y) -Koordinatenpaar zur Angabe der Steigung und in eine horizontale Länge zur Definition der Strichlänge.
4. Überprüfung, ob die gewünschte Steigung verfügbar ist und, wenn nicht, Wiederholung der Schritte 1 bis 3, bis eine zufriedenstellende Lösung gefunden ist.

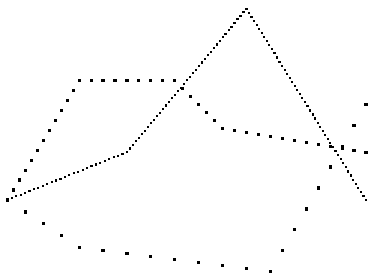
Das obige Verfahren ist sehr schwerfällig. Zudem ist die Länge des kürzesten verfügbaren Strichs in verschiedenen Steigungen aufgrund der Art und Weise, wie der `\line`-Befehl implementiert ist, nicht immer dieselbe. Das `epic`-Paket (von Sunil Podar) führt eine leistungsfähige High-Level-Benutzerschnittstelle zur `picture`-Umgebung [141] ein, die diese Mängel beseitigt. Ihr Hauptziel besteht darin, den Anteil an manueller Berechnung zu reduzieren, der für die Festlegung der *Objekt*positionen benötigt wird. Auf diese Weise ermöglicht das `epic`-Paket das Erstellen von anspruchsvollen Bildern mit einem geringeren Aufwand, als es bisher möglich war.

High-Level-Linienbefehle

Das Paket führt eine Reihe von leistungsfähigen Zeichenbefehlen ein und stellt gleichzeitig eine einfachere Syntax bereit. Diese Befehle benötigen nur die Koordinaten der Endpunkte und machen damit die anderen Schritte zur Definition einer Linie überflüssig.

```
\dottedline[objekt]{abstand}(x1,y1)(x2,y2)... (xn,yn)
```

Der Befehl `\dottedline` verbindet die angegebenen Koordinaten, indem er zwischen den einzelnen Koordinatenpaaren eine gepunktete Linie zieht. Es müssen mindestens zwei Koordinaten definiert sein. Die gepunktete Linie wird mit einem Punktabstand dargestellt, der im obligatorischen Argument *abstand* (in `\unitlength`) angegeben ist. Da die Anzahl der zu zeichnenden Punkte ganzzahlig sein muss, entspricht der tatsächliche Punktabstand möglicherweise nicht genau dem angeforderten.

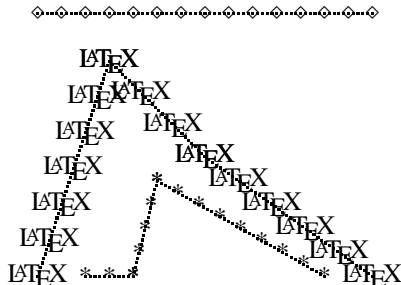


```
\usepackage{epic}
\setlength\unitlength{0.9pt}
\begin{picture}(150,80)(0,0)
\dottedline{2}(0,0)(50,20)(100,80)(150,0)
\thicklines
\dottedline{5}(0,0)(30,50)(70,50)(90,30)(150,20)
\dottedline{10}(0,0)(30,-20)(110,-30)(150,40)
\end{picture}
```

Bsp.
10-1-13

Standardmäßig (das heißt wenn das optionale Argument *objekt* nicht verwendet wird) erzeugt `\dottedline` winzige Quadrate, die intern mit dem Befehl `\picsquare` generiert werden. Die Größe der Quadrate hängt von der aktuellen Einstellung des Befehls `\thinlines`, `\thicklines` bzw. `\linethickness` ab. Tatsächlich benutzen die meisten epic-Befehle intern zur Darstellung der Linien `\picsquare`.

Durch Verwendung des optionalen Argumentes *objekt* kann jedes Objekt entlang des Linienzugs erzeugt werden, der durch die Koordinaten definiert wurde. Man beachte, dass einige Zeichen, wie „*“, in vielen Schriftschnitten nicht zentriert werden; die meisten anderen Zeichen und Objekte jedoch schon.



```
\usepackage{epic}
\setlength\unitlength{0.9pt}
\thicklines
\begin{picture}(140,110)(0,0)
\dottedline{2}(0,110)(140,110)
\dottedline[$\diamond$]{10}(0,110)(140,110)
\dottedline{2}(20,0)(40,0)(50,40)(120,0)
\dottedline[*]{10}(20,0)(40,0)(50,40)(120,0)
\dottedline{2}(0,0)(30,90)(70,50)(140,0)
\dottedline[\LaTeX]{20}(0,0)(30,90)(70,50)(140,0)
\end{picture}
```

Bsp.
10-1-14

`\dashline[dehn]{strichlänge}[abstand](x_1,y_1)(x_2,y_2)\dots(x_n,y_n)`

Der Befehl `\dashline` verbindet die angegebenen Punkte, indem er zwischen den einzelnen Koordinatenpaaren eine gestrichelte Linie zieht. Es müssen mindestens zwei Punkte angegeben werden. Intern werden die Striche mit dem Befehl `\dottedline` zusammengesetzt. Das obligatorische Argument *strichlänge* legt die Länge der einzelnen Striche fest, und das optionale Argument *abstand* gibt den Zwischenraum zwischen den Punkten an, die zur Bildung der Striche verwendet werden. Beide werden in `\unitlength` angegeben. Als Voreinstellung wird ein durchgehender Strich generiert.

<pre> - </pre>	<pre> \usepackage{epic} \setlength{\unitlength}{1mm} \begin{picture}(70,22)(0,-2) \dashline{3}[0.7](0,20)(63,20) \thicklines \dashline{3}(0,16)(63,16) \dashline[-30]{3}(0,12)(63,12) \dashline[+15]{3}(0,8)(63,8) \dashline[+30]{3}(0,4)(63,4) \dashline[+30]{3}[0.7](0,0)(63,0) \end{picture} </pre>

Bsp.
10-1-15

Gemäß der Definition des Befehls `\dashline` muss das optionale Argument *dehn* eine ganze Zahl zwischen -100 und ∞ sein. Sie gibt den Anteil an, um den die Anzahl von Strichen „gestreckt“ oder erhöht (*dehn* $>$ 0) bzw. „gestaucht“ oder verringert wird (*dehn* $<$ 0). Wenn *stretch* null ist, wird die Mindestanzahl an Strichen verwendet, die einen etwa gleichmäßigen Abstand von Leerraum und Strichen ergibt. Der Gedanke, der hinter dem prozentualen Parameter steckt, ist, dass bei Zeichnung von mehreren Strichen unterschiedlicher Strichlänge alle gestrichelten Linien mit gleichen *dehn*-Werten in etwa das gleiche Erscheinungsbild haben. Die Standardeinstellung für die *dehn*-Angabe kann durch Neudefinition des Befehls `\dashlinestretch` geändert werden:

```
\renewcommand\dashlinestretch{-50} % Nur ganze Zahlen erlaubt
```

Sein Wert gibt die Erhöhung oder Verringerung an, die für alle folgenden `\dashline`-Befehle gilt, außer für jene, bei denen *dehn* explizit als erstes optionales Argument angegeben ist.

`\drawline[dehn](x_1,y_1)(x_2,y_2)\dots(x_n,y_n)`

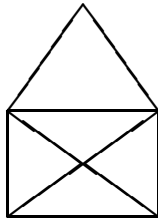
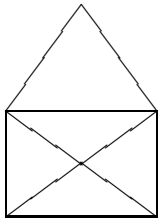
Der Befehl `\drawline` verbindet die angegebenen Punkte, indem er zwischen den einzelnen Koordinatenpaaren eine Linie zieht. Hierfür verwendet er Liniensegmente der nächstgelegenen Neigung, die in den Linienfonts von $\text{\LaTeX}$ verfügbar sind. Es müssen mindestens zwei Koordinaten angegeben werden. Da in den Linienfonts nur eine begrenzte Anzahl von Steigungen verfügbar ist, werden fehlende Steigungen generiert, indem sehr kurze Liniensegmente mit der jeweils nächsten Steigung aneinander gereiht werden. Hierdurch kann

Unerwünschte
gezackte Linien

es dazu kommen, dass einige Linien gezackt aussehen (im nächsten Beispiel trifft das auf alle geneigten Linien zu). Das ist der Preis, den man bezahlt, wenn man implizit Linien beliebiger Steigung verwenden darf. Das Problem lässt sich jedoch beheben, wenn man zusätzlich zu `epic` das Paket `eepic` verwendet.

Der Befehl `\drawline` kann dicke oder dünne Linien erzeugen, je nachdem, ob die Einstellung `\thinlines` oder `\thicklines` aktiv ist. Dies sind die einzigen Liniestärken, die für diesen Befehl zur Verfügung stehen.

Das optionale Argument *dehn* wirkt ähnlich wie jenes für den `\dashline`-Befehl. Wenn *dehn* null ist, wird die Mindestanzahl an Strichen verwendet, die erforderlich ist, um den Eindruck einer durchgehenden Linie zu erzeugen. Alle Striche werden dabei an den Enden „verbunden“. Wenn *dehn* größer null ist, werden mehr Striche zur Erzeugung der Linie verwendet, so dass der Linienverlauf glatter erscheint (vgl. auch die beiden Häuser im nächsten Beispiel).



```
\usepackage{epic} \setlength{\unitlength}{2mm}
\begin{picture}(25,14)
  \drawline(0,0)(0,7)(5,14)(10,7)
    (0,7)(10,0)(0,0)(10,7)(10,0)

  \thicklines
  \drawline[70](15,0)(15,7)(20,14)(25,7)
    (15,7)(25,0)(15,0)(25,7)(25,0)
\end{picture}
```

Bsp.
10-1-16

Wie mit dem Parameter `\dashlinestretch` für den Befehl `\dashline` kann man mit dem Parameter `\drawlinestretch` einen Standardwert für das prozentuale *dehn*-Argument des Befehls `\drawline` definieren.

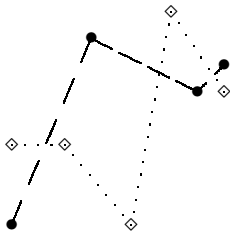
Wissenschaftliche Daten graphisch darstellen

Bei der Darstellung von wissenschaftlichen Daten ist es häufig wünschenswert, Graphen zu erstellen, welche die erhaltenen (zweidimensionalen) Datensätze in Bezug zueinander zeigen. Man kann beispielsweise eine Reihe von experimentell gewonnenen Datenpunkten auf eine bestimmte Art graphisch darstellen (z.B. durch gefüllte Kreise) und eine andere Reihe durch andere Symbole (z.B. durch Rauten). Zur Verdeutlichung kann man die einzelnen Datenpunkte mit Linien verbinden. Damit der Leser die resultierenden Kurven leichter voneinander unterscheiden kann, sind auch verschiedene Arten von „Linien“ denkbar.

Eine Möglichkeit, dies zu erreichen, ist, die experimentellen Ergebnisse durch eine Folge von einfachen `\put`-Befehlen graphisch darzustellen, gefolgt von dem Befehl `\dottedline`, `\dashline` oder `\drawline`, der die Datenpunkte verbindet. Mit anderen Worten: Die Koordinaten werden zweimal angegeben. Zur Vereinfachung dieses Vorgangs bietet `epic` die Umgebungen `dottedjoin`, `dashjoin` und `drawjoin`, die den obigen Befehlen entsprechen und dieselben obligatorischen und optionalen Argumente annehmen. Diese Umgebungen verwenden den neuen Befehl `\jput` (join und put). Er ist identisch mit dem regulären `\put`-Befehl von $\text{\LaTeX}$, mit der Ausnahme, dass er nur innerhalb dieser drei Umgebungen verwendet werden kann. Alle Objekte, die

im Gültigkeitsbereich von einer der drei Umgebungen per `\jput`-Befehl eingefügt werden, werden nicht nur gezeichnet, sondern auch durch Linien des entsprechenden Typs verbunden. Es obliegt dem Anwender, diese Objekte an ihren Positionen zu zentrieren.

Jede Instanz von einer der drei `..join`-Umgebungen definiert eine separate „Kurve“; daher sollte jeder Datensatz, der zu einer anderen „Kurve“ gehört, in eine separate `..join`-Umgebung eingebettet werden. Die Hauptmotivation, solche `..join`-Umgebungen zu entwerfen, war, Graphen zeichnen zu können, die verschiedene Kurven- und Linienarten verwenden.



Bsp.
10-1-17

```
\usepackage{epic} \setlength{\unitlength}{1pt}
\newcommand\cb{\makebox(0,0){$\bullet$}}
\newcommand\cd{\makebox(0,0){$\diamond$}}
\begin{picture}(80,80)
  \begin{dashjoin}[30]{10} \jput(0,0){\cb} \jput(30,70){\cb}
    \jput(70,50){\cb} \jput(80,60){\cb}
  \end{dashjoin}
  \begin{dottedjoin}[5] \jput(0,30){\cd} \jput(20,30){\cd}
    \jput(45,0){\cd} \jput(60,80){\cd} \jput(80,50){\cd}
  \end{dottedjoin}
\end{picture}
```

Das `epic`-Paket bietet eine weitere Möglichkeit, Graphen zu erzeugen, und zwar mithilfe des Befehls `\putfile{datei}{objekt}`. Er ähnelt $\text{\LaTeX}$ s `\put`-Befehl, außer dass die x - und y -Koordinaten, die vom `\put`-Befehl benötigt werden, aus einer externen Datei gelesen werden und an jedem dieser Punkte dasselbe *objekt* ausgegeben wird. Dieser Befehl wird bereitgestellt, weil $\text{\TeX}$ keine Funktion zur Gleitkommaberechnung hat. Diese ist erforderlich, wenn eine Parameterkurve dargestellt werden soll, die keine gerade Linie ist. Die Koordinaten der Punkte auf solchen Kurven lassen sich leicht mithilfe eines Programms in einer Computersprache generieren und können dann von $\text{\TeX}$ eingelesen werden. Die externe Datei muss die (x, y) -Koordinatenpaare enthalten, wobei pro Zeile nur ein Paar angegeben werden darf und zwischen den beiden Koordinaten ein Leerzeichen stehen muss. Als Kommentarzeichen kann `%` verwendet werden; allerdings sollte nach dem y -Eintrag mindestens ein Leerzeichen folgen, wenn der Kommentar auf derselben Zeile wie die Daten erscheint, da `%` das Zeilenumbruchzeichen maskiert.

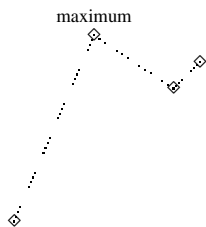
*Extern generierte
graphische Daten
laden*

Um beispielsweise eine glatte Kurve entlang einer Reihe von Koordinaten zu zeichnen, kann man folgendes Verfahren wählen:

1. Erstellung einer Datei mit den x, y -Koordinaten der Datenpunkte, die z.B. `plot.data` genannt wird.
2. Bei Bedarf Glättung der Daten.
3. Platzierung des Programmcodes `\putfile{plot.data}{\picsquare}` in einer `picture`-Umgebung im Dokument.

Wie der Name des Befehls andeutet, verwendet `\putfile` nicht den Befehl `\jput`, sondern den Befehl `\put`. Diese Wahl ist unglücklich, da die Verwendung von `\putfile` in einer der `..join`-Umgebungen zwar Objekte

an den Koordinaten zeichnet, aber sie nicht verbindet – obwohl technisch nichts gegen eine Verbindung spricht. Es gibt jedoch einen kleinen Trick, den man verwenden kann, um solch eine Verbindung zu erstellen. Dazu muss man sicherstellen, dass `\put` stets `\jput` in den Bildern ausführt. Da sich `\jput` genauso wie $\text{\LaTeX}$ s `\put`-Befehl verhält, wenn er außerhalb der `...join`-Umgebungen eingesetzt wird, gibt es keine Einwände, ihn global als Ersatz einzusetzen. Dieser Ansatz wird im nächsten Beispiel verfolgt.



```
\usepackage{epic} \renewcommand\put{\jput} % <- immer \jput nutzen
\begin{filecontents}{test.put}
0 0                                     % Daten im externer Datei
30 70                                  % Man beachte, dass die
60 50                                  % Koordinaten durch
70 60                                  % Leerzeichen separiert sind
\end{filecontents}
\newcommand\cd{\makebox(0,0){$\diamond$}}
\begin{picture}(80,80)
\begin{dashjoin}[6][2] \putfile{test.put}{\cd} \end{dashjoin}
\put(30,75){\makebox(0,0)[b]{\scriptsize maximum}}
\end{picture}
```

Bsp.
10-1-18

Objekte in regelmäßigen Abständen platzieren

Bei den bisher gezeigten Beispieldiagrammen fehlten noch die beschrifteten Achsen. `epic` hat keine Befehle, die diese Aufgabe vollständig übernehmen, bietet aber mit `\multiputlist` und `\grid` Werkzeuge, die bei den monotoneren Teilaufgaben helfen können.

$\text{\multiputlist}(x,y)(\Delta x,\Delta y)[pos]\{objekt_1,objekt_2,\dots,objekt_n\}$

Dieser Befehl ist eine Variante von $\text{\LaTeX}$ s `\multiput`-Befehl, mit dem *dasselbe* Objekt an Koordinaten mit regelmäßigen Abständen eingefügt werden kann. Der Befehl `\multiputlist` wirkt ähnlich, lässt aber *verschiedene* Objekte zu. Bei der Ausführung des Befehls `\multiputlist` werden die zu „platzierenden“ Objekte aus einer *Objektliste* ausgewählt, während die Koordinaten hochgezählt werden. Das erste Objekt wird an Position 1 platziert, das zweite an Position 2 usw. So kann man beispielsweise entlang der x -Achse Nummern ausgeben, wenn man Folgendes angibt:

```
\multiputlist(0,0)(10,0){1.00,1.25,1.50,1.75,2.00}
```

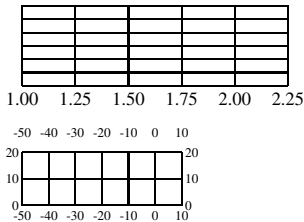
Die Objekte in der Liste können praktisch alles sein, einschließlich `\makebox`, `\framebox` oder mathematischer Symbole. Dieser Befehl zwingt der Anordnung verschiedener Objekte in einem Bild eine gewisse Regelmäßigkeit und Symmetrie auf.

$\text{\grid}(breite,höhe)(\Delta breite,\Delta höhe)[ursprung-x,ursprung-y]$

Der Befehl `\grid` erzeugt ein Gitter der Größe *breite* mal *höhe* in `\unitlength`-Einheiten. Vertikale Linien werden in Abständen von $\Delta breite$ und horizontale Linien in Abständen von $\Delta höhe$ gezeichnet. Bei Angabe des

dritten (optionalen) Argumentes wird der Rand des Gitters beschriftet, wobei die Anfangswerte die ganzzahligen Nummern *ursprung-x* und *ursprung-y* bilden. Sie werden entlang der Achsen um Δbreite und $\Delta\text{höhe}$ erhöht.

Der Befehl `\grid` erzeugt eine Box. Daher muss er mit `\put` an den entsprechenden Koordinaten platziert werden. Zum Beispiel:



```
\usepackage{epic}
\begin{picture}(100,60)
\put(0,45){\grid(100,30)(20,5)}
\scriptsize % Größe der Nummern einstellen
\multiputlist(0,40)(20,0){1.00,1.25,1.50,1.75,2.00,2.25}

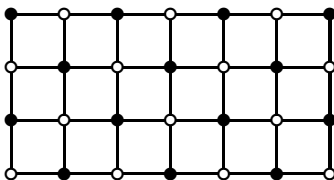
\put(0,0){\tiny\grid(60,20)(10,10)[-50,0]}
\end{picture}
```

Bsp.
10-1-19

Wenn mehr Flexibilität erforderlich ist, als der `\grid`-Befehl zur Erzeugung einer regelmäßigen zweidimensionalen Struktur bietet, kann `\matrixput` möglicherweise die Lösung sein.

$\text{\matrixput}(x,y)(\Delta x_1,\Delta y_1)\{n_1\}(\Delta x_2,\Delta y_2)\{n_2\}\{\text{objekt}\}$

Dieser Befehl ist das zweidimensionale Äquivalent des `\multiput`-Befehls von $\text{\LaTeX}$. Es ist jedoch effizienter, anstelle mehrerer `\multiput`-Befehle einen `\matrixput`-Befehl zu verwenden. Dieser Befehl ist besonders nützlich, um regelmäßige zweidimensionale Muster zu generieren.



```
\usepackage{epic} \setlength{\unitlength}{2pt}
\begin{picture}(62,32) \thicklines
\matrixput(0,0)(10,0){7}(0,10){4}{\circle{2}}
\matrixput(10,0)(20,0){3}(0,20){2}{\circle*{2}}
\matrixput(0,10)(20,0){4}(0,20){2}{\circle*{2}}
\matrixput(1,0)(10,0){6}(0,10){4}{\line(1,0){8}}
\matrixput(0,1)(10,0){7}(0,10){3}{\line(0,1){8}}
\end{picture}
```

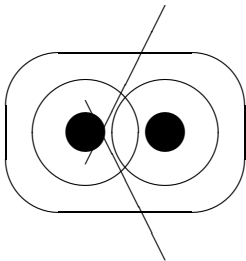
Bsp.
10-1-20

10.1.5 epic – Erweiterung des epic-Paketes

$\text{\LaTeX}$ verfügt über ein elementares, aber begrenztes Potential zum Zeichnen, das von `epic` um Befehle und Umgebungen zum Zeichnen von durchgehenden, gestrichelten und gepunkteten Linien erweitert wird und außerdem um neue Umgebungen, die zur graphischen Darstellung von Graphen geeignet sind (beschrieben im vorangegangenen Abschnitt).

Doch `epic` enthält noch viele der Begrenzungen von $\text{\LaTeX}$, was das Zeichnen von Bildern betrifft. Die Folge ist, dass viele der Funktionen noch immer sehr zeitaufwendig sind oder das Ergebnis keine hohe Qualität aufweist. In $\text{\LaTeX}$ werden spezielle Fonts verwendet, um Linien und Kreise zu zeichnen. Aus diesem Grund werden nur Linien mit bestimmten Steigungen unterstützt, und für das Zeichnen von Kreisen, Ovalen oder Kreisflächen steht nur eine begrenzte Anzahl von Durchmessern zur Verfügung.

Das folgende Beispiel zeigt einige dieser Begrenzungen. Hier sind die Fläche und der Kreis links zu klein (ohne dass eine Warnung ausgegeben wird), und die `\line`-Befehle führen zu Fehlern, weil die erforderliche Steigung nicht verfügbar ist. Hier schafft auch das Laden des `epic`-Paketes keine Abhilfe.



```
\usepackage{epic}
\begin{picture}(0,0)
  \put(0,0){\circle{80}} \put(0,0){\circle*{24}}
  \put(30,0){\circle{40}} \put(30,0){\circle*{16}}
  \put(15,0){\oval(90,60)}
  \put(0,12){\line(15,-2){30}}\put(0,-12){\line(15,2){30}}
\end{picture}
```

Bsp.
10-1-21

Man vergleiche dieses Ergebnis mit Beispiel 10-1-22 auf der gegenüberliegenden Seite, das die korrekte Ausgabe zeigt – es ist auffallend anders.

Ende der 80er Jahre wurde die Programmiersprache `pic` entwickelt, mit der eine „natürlichsprachliche“ Methode zur Beschreibung von einfachen Bildern und Diagrammen angeboten werden sollte (siehe [79]). Ein Präprozessor, wie GNUs `gpics`, kann diese Graphikbefehle in ein Format umwandeln, welches das UNIX-Formatierungsprogramm `troff` versteht. Weitaus interessanter für die $\text{\LaTeX}$ -Gemeinde ist jedoch, dass es auch $\text{\TeX}$ s `\special`-Befehle generieren kann, die viele `dvi`-Treiberprogramme unterstützen. Das `dvi`-zu-PostScript-Übersetzungsprogramm `dvips`, das in Abschnitt 10.4.2 beschrieben wird, ist beispielsweise in der Lage, diese Befehle zu interpretieren.

Das `eepic`-Paket, das von Conrad Kwok geschrieben wurde, stellt sowohl für $\text{\LaTeX}$ als auch für `epic` eine Erweiterung dar. Es hebt einige der Beschränkungen auf, die in $\text{\LaTeX}$, `epic` und `gpics` existieren, indem es mithilfe von $\text{\TeX}$ -Befehlen `gpics` `\special`-Befehle generiert. Da die Befehle des `eepic`-Paketes eine Obermenge von `epic` bilden, kann man es verwenden, um ein beliebiges Bild zu bearbeiten, das `epic`-Befehle enthält, und damit ein optisch besseres Ergebnis erzielen.

eepics Neuimplementierung von $\text{\LaTeX}$ -Befehlen

Die Erweiterungen in `eepic` ermöglichen es dem Anwender, Linien in beliebiger Steigung und Kreise in jeder Größe zu zeichnen. Die Begrenzungen für die Steigung von Vektoren bleiben jedoch erhalten. Das heißt, dass nur Steigungen in der Form x/y gehandhabt werden können, wobei x und y ganze Zahlen im Bereich $[-4, 4]$ darstellen.

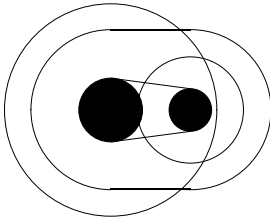
`\line(x,y){länge}`

Die Syntax des `\line`-Befehls entspricht der in $\text{\LaTeX}$ verwendeten, mit dem Unterschied, dass x und y jetzt Platzhalter für jede ganze Zahl sind, die von $\text{\TeX}$ akzeptiert wird. Außerdem entfällt der untere Grenzwert für das Argument *länge* (in Standard- $\text{\LaTeX}$ beträgt er ungefähr 3,5 mm).

```
\circle{durchmesser} \circle*{durchmesser} \oval(x,y)[teil]
```

Die Syntax für das Anlegen von leeren und gefüllten Kreisen mittels `\circle` und `\circle*` entspricht der in $\text{\LaTeX}$ verwendeten. Das Argument *durchmesser* kann jedoch jetzt für jede Zahl stehen, die von $\text{\TeX}$ akzeptiert wird, und es werden Kreise mit (exakt) dem angegebenen Durchmesser gezeichnet. Der Befehl `\oval` wurde so verändert, dass der maximale Durchmesser des Viertelkreises an den Ecken auf jeden Wert gesetzt werden kann, indem die Variable `\maxovaldiam` auf das gewünschte $\text{\TeX}$ -Maß gesetzt wird (voreingestellt ist 40pt).

Das folgende Beispiel nimmt Beispiel 10-1-21 auf der vorherigen Seite wieder auf, nur dass jetzt `eepic` geladen und `\maxovaldiam` verwendet wurde. Alle Elemente erscheinen im überarbeiteten Beispiel wie angegeben.



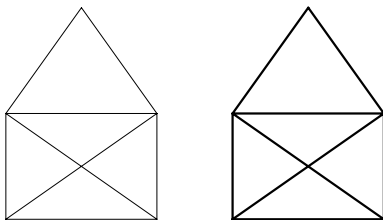
Bsp.
10-1-22

```
\usepackage{eepic} \setlength{\maxovaldiam}{60pt}
\begin{picture}(0,0)
  \put(0,0){\circle{80}} \put(0,0){\circle*{24}}
  \put(30,0){\circle{40}} \put(30,0){\circle*{16}}
  \put(15,0){\oval(90,60)}
  \put(0,12){\line(15,-2){30}} \put(0,-12){\line(15,2){30}}
\end{picture}
```

eepics Neuimplementierung der epic-Befehle

Das `epic`-Paket erzeugt standardgemäße dvi Dateien und benötigt lediglich die Standardfonts von $\text{\LaTeX}$. Die `epic`-Erweiterung, das `eepic`-Paket, erzielt beim Zeichnen von Linien ein schöneres Ergebnis, benötigt weniger Bearbeitungszeit und hat einen geringeren Speicherplatzbedarf – dafür ist die Ausgabe treiberabhängig. Es reimplementiert dafür die Befehle `\drawline`, `\dashline` und `\dottedline` (siehe Seite 620) sowie die entsprechenden `..join`-Umgebungen `dashjoin`, `dottedjoin` und `drawjoin` (siehe Seite 622).

Man vergleiche im folgenden Beispiel die diagonalen Linien mit jenen in Beispiel 10-1-16 auf Seite 622. Zu beachten ist, dass das Ergebnis von Strichzeichenbefehlen geglättet wird, wenn `eepic` zusammen mit `epic` geladen wird. Beide Pakete müssen in der richtigen Reihenfolge geladen werden.



Bsp.
10-1-23

```
\usepackage{epic,eepic} \setlength{\unitlength}{2mm}
\begin{picture}(25,14)
  \drawline(0,0)(0,7)(5,14)(10,7)
  (0,7)(10,0)(0,0)(10,7)(10,0)
  \thicklines
  \drawline[70](15,0)(15,7)(20,14)(25,7)
  (15,7)(25,0)(15,0)(25,7)(25,0)
\end{picture}
```

Das `eepic`-Paket führt auch eine Reihe von neuen Befehlen ein. Mit Ausnahme von `\path` haben diese Befehle kein Äquivalent in $\text{\LaTeX}$ und `epic`. Am Ende dieses Abschnitts wird auf Kompatibilitätsfragen bezüglich dieser Pakete eingegangen.

```
\allinethickness{maß} \Thicklines
```

Der Befehl `\allinethickness` legt die Strichstärke für alle Strichzeichenbefehle fest, auch für die geneigte Linien, Kreise, Ellipsen, Bögen, Ovale und Splines.

Nach Verwendung des Befehls `\Thicklines` werden alle folgenden Linien 1,5 mal dicker gezeichnet, als durch `\thicklines` festgelegt.

```
\path(x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
```

Der Befehl `\path` ist die schnelle Variante des Befehls `\drawline`. Da das optionale *dehn*-Argument des letztgenannten Befehls nicht zugelassen ist, erzeugt `\path` nur durchgezogene Linien. Der Befehl `\path` wird hauptsächlich zum Zeichnen von komplexen Pfaden verwendet.

```
\spline(x_1,y_1)(x_2,y_2)\dots(x_n,y_n)
```

Der Befehl `\spline` zeichnet eine Chaikin-Kurve, die nur durch den ersten und letzten Punkt geht. Alle anderen Punkte fungieren nur als Kontrollpunkte.

```
\ellipse{x-durchmesser}{y-durchmesser}
\ellipse*{x-durchmesser}{y-durchmesser}
```

Analog zu den Befehlen `\circle` und `\circle*` zeichnen die Befehle `\ellipse` und `\ellipse*` eine leere oder gefüllte Ellipse unter Verwendung der angegebenen Argumente *x-durchmesser* und *y-durchmesser*.

```
\arc{durchmesser}{anfangswinkel}{endwinkel}
```

Der Befehl `\arc` zeichnet einen Kreisbogen. Das erste Argument *durchmesser* wird in `\unitlength`-Einheiten angegeben. Sowohl der *anfangswinkel* als auch der *endwinkel* werden in Radianten angegeben, wobei der *anfangswinkel* innerhalb des Bereiches $[0, \frac{\pi}{2}]$ liegen muss und der *endwinkel* jeden Wert zwischen dem *anfangswinkel* und *anfangswinkel* + 2π annehmen kann. Bögen werden im Uhrzeigersinn erstellt, wobei der Winkel 0 auf dem Papier rechts liegt.

```
\filltype{bereichsfülltyp}
```

Der Befehl `\filltype` gibt den Typ der Bereichsfüllung für die Befehle `\circle*` und `\ellipse*` an. Die Anweisung selbst erstellt keine Zeichnung. Sie ändert lediglich die Interpretation von `*` in den beiden oben angeführten Befehlen. Der *bereichsfülltyp* kann folgende Werte annehmen: `black` (voreingestellt), `white` und `shade`. Durch Angabe des folgenden Befehls kann der Fülltyp z.B. auf weiß umgeschaltet werden: `\filltype{white}`.

Das *eepic*-Paket ist nicht notwendigerweise in allen $\text{\LaTeX}$ -Installationen verfügbar, und selbst wenn es vorhanden ist, wird es unter Umständen nicht vom gewählten Ausgabegerät unterstützt. Um Portabilitätsprobleme, die bei Verwendung dieses Paketes auftreten können, zu vermeiden und trotzdem die

Stärken von eepic zu nutzen, nämlich die Erzielung besserer Qualität, sollte der Anwender einige Vorsichtsmaßnahmen treffen:

- Vermeidung von `\line`-Befehlen. Stattdessen sollte man `\drawline` verwenden, denn $\text{\LaTeX}$ s `\line`-Befehl unterstützt nur eine begrenzte Anzahl von Steigungen.
- Vermeidung des `\arc`-Befehls. Wenn tatsächlich eine komplexe Kurve erstellt werden muss, sollte `\spline` verwendet werden.
- Vermeidung von langen gestrichelten Linien mit keinen oder kleinen Punktzwischenräumen, da diese in der originalen epic-Implementation einen großen Teil des $\text{\TeX}$ -Speichers belegen. Um eine gestrichelte Linie zu erstellen, sollte man den Befehl `\drawline` mit negativem Dehnwert verwenden.

Wenn die eigene Installation eepic nicht unterstützt und ein mit eepic erstelltes Dokument ausgedruckt werden muss, können die Makros zur Emulation von eepic verwendet werden. Diese werden durch das eepicemu-Paket definiert. Die erweiterten Befehle werden folgendermaßen emuliert:

*Emulieren der
eepic-Befehle*

- Kreise, die größer sind als 40pt, werden mit dem Befehl `\oval` erzeugt.
- Ellipsen werden ebenfalls mit `\oval` gezeichnet.
- Bögen generieren eine Warnung, werden aber ansonsten ignoriert.
- Splines werden mit `\drawline` angenähert.
- `\path` wird durch `\drawline` ersetzt.
- `\Thicklines` wird durch `\thicklines` ersetzt.
- `\allinethickness` wird durch `\thicklines` und `\linethickness` ersetzt.

Da das eepic-Paket verschiedene Befehle des epic-Paketes neu definiert, muss die Deklaration des eepic-Paketes nach der Deklaration des epic-Paketes erfolgen. Es ist zwar nicht unbedingt erforderlich, aber sehr empfehlenswert, bei Verwendung der eepic-Befehle stets das epic-Paket mitzuladen. Auf jeden Fall kann das eepic-Emulationspaket eepicemu nur benutzt werden, wenn beide angegeben sind.

10.1.6 Sprachen für besondere Zwecke

Auf Basis der `picture`-Umgebung von $\text{\LaTeX}$, möglicherweise erweitert um die Pakete epic und eepic, haben mehrere Paketautoren High-Level-Schnittstellen implementiert, um die Eingabe von graphischen Daten zu vereinfachen und weniger fehleranfällig zu machen. Dazu verwenden sie eine Syntax, die für Endbenutzer in einem bestimmten Anwendungsgebiet vertrauter ist. Einige der Systeme sind recht komplex; der *$\text{\LaTeX}$ Graphics Companion* [58] beschreibt einige von ihnen im Detail. Dieser Abschnitt gibt lediglich einen kurzen Einblick in die Möglichkeiten aus diesem Bereich und zeigt einige kurze Beispiele.

Für alle, die nicht auf ein Zeichenpaket zugreifen können, aber ein paar Kurven mit kontinuierlicher Steigung einfügen müssen, bietet das `curves`-Paket von Ian Maclaine-cross einige faszinierende Möglichkeiten. Es ermög-

licht, die Strichstärke der Kurven über einen großen Bereich zu variieren, die Endsteigungen zu kontrollieren und geschlossene Kurven mit kontinuierlicher Steigung zu definieren. Außerdem kann es mit `\arc` große Kreise und Kreisbögen erstellen und ermöglicht damit eine unabhängige Skalierung von Kurvenabzissen und -ordinaten, um sie an Graphen anzupassen. Darüber hinaus bietet es affine Skalierung, um Bögen oder Kreise elliptisch zu machen, und unterstützt Symbole und Strichmuster. Wie das folgende, einfache Beispiel zeigt, zeichnet `\curve` eine Kurve durch die angegebenen Koordinatenpaare, während `\closecurve` eine geschlossene Kurve mit kontinuierlichen Tangenten an allen Punkten zeichnet. Der Befehl `\tagcurve` fungiert generell wie `\curve`, außer dass das erste und letzte Segment nicht gezeichnet wird.

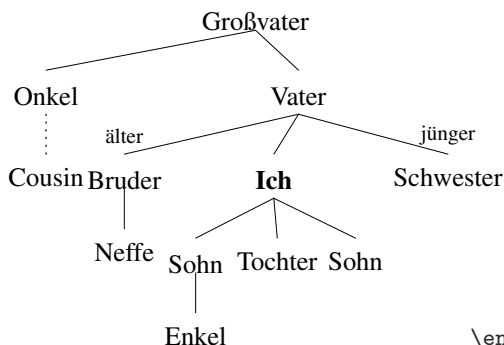


```
\usepackage{curves}
\setlength{\unitlength}{0.4pt}
\linethickness{0.7mm}
\begin{picture}(400,110)(-10,0)
  \curve(0,0, 40,100, 80,0)
  \closecurve(150,0, 190,100, 230,0)
  \tagcurve(380,0, 300,0, 340,100, 380,0, 300,0)
\end{picture}
```

Bsp.
10-1-24

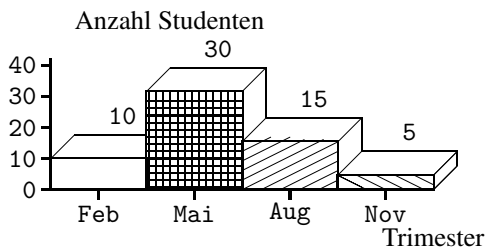
Hideki Isozakis `ectree`-Paket ermöglicht das Zeichnen von einfachen Baumstrukturen. Es bietet eine `bundle`-Umgebung, die den obersten Knoten beschriftet, dem ein oder mehrere Knoten untergeordnet werden können. Diese werden von `\chunk`-Befehlen definiert. Sie haben ein optionales Argument, mit welchem Kommentare auf den Linien hinzugefügt werden können. Der Linienstil kann mit dem Befehl `\drawwith` kontrolliert werden, indem als Argument einer der Strichzeichenbefehle des `epic`-Paketes eingegeben wird (beschrieben in Abschnitt 10.1.4). Die `bundle`-Umgebung und `\chunk`-Befehle lassen sich verschachteln, wie der folgende $\text{\LaTeX}$ -Programmcode zeigt.

```
\usepackage{epic, eepic, ectree}
\begin{bundle}{Großvater}
  \chunk{\begin{bundle}{Onkel}\strut}
    \chunk{Cousin}\drawwith{\dottedline{3}}
  \end{bundle}}
  \chunk{\begin{bundle}{Vater}\strut}
    \chunk[\footnotesize älter]{%
      \begin{bundle}{Bruder}
        \chunk{Neffe}
      \end{bundle}}
    \chunk{\begin{bundle}{\textbf{Ich}}\strut}
      \chunk{\begin{bundle}{Sohn}
        \chunk{Enkel}
      \end{bundle}}
      \chunk{Tochter}\chunk{Sohn}
    \end{bundle}}
    \chunk[\footnotesize jünger]{Schwester}
  \end{bundle}}
\end{bundle}
```



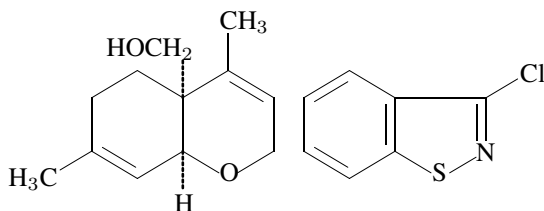
Bsp.
10-1-25

Das bar-Paket wurde von Joachim Bleser und Edmund Lang geschrieben, um Balkendiagramme zu erzeugen. Die Umgebung `barenv` enthält die Daten, die ein Balkendiagramm definieren. Jeder Datenpunkt wird über den Befehl `\bar` angegeben, dessen obligatorische Argumente die Ordinate des Eintrags und die Art der Schraffur festlegen. Des weiteren bietet das Paket eine Reihe von `\set . . .`-Befehlen, mit denen die Darstellung der Daten angepasst werden kann (siehe nächstes Beispiel).



```
\usepackage{epic, eepic, bar}
\begin{barenv}
\setlength\unitlength{0.9pt}
\setdepth{10} % 3-D Effekt
\setstretch{1.4} % Dehnung in y-Richtung
\setnumberpos{up} % Nummern oberhalb
\setxvaluetype{month} % Achsenbeschriftungstyp
\setxaxis{2}{12}{3}\setxname{Trimester}
\setyaxis{0}{40}{10}\setyname{Anzahl Studenten}
\bar{10}{1} \bar{30}{4}
\bar{15}{6} \bar{5}{7}
\end{barenv}
```

Wie bereits erwähnt, können über ein spezielles Paket sehr viel komplexere strukturelle Daten auf bequeme Art eingegeben werden. Ein Beispiel ist Shinsaku Fujitas $\hat{\text{X}}\text{M}\text{T}_{\text{E}}\text{X}$ -Paket zum Zeichnen von chemischen Diagrammen (siehe [50, 51] oder [58, Kapitel 6]). Dadurch, dass Befehlsnamen verwendet werden, die von Standardnomenklatur inspiriert wurden, die von Praktikern aus dem Bereich verwendet wird, können auch komplexe Formeln auf recht einfache Weise eingegeben werden. Im folgenden Beispiel wird das `hetarom`-Teilpaket verwendet, das für die Angabe der Struktur vertikaler heterozyklischer Verbindungen entworfen wurde.



```
\usepackage{epic, hetarom}
\decaheterov[af]{4==0}
{1==CH$_3$;6==H$_3$C;9A==H;%
{10}A==\lmoiety{HOCH$_2$}}
\hspace*{-15mm}
\nonaheterov[bjge]{1==S;2==N}{3==Cl}
```

10.2 Geräteabhängige Graphikunterstützung

Seit der Einführung von $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ im Jahre 1994 bietet $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ eine einheitliche Syntax, um jede Art von Graphikdatei einzubinden, die von den diversen Treibern gehandhabt werden kann. Außerdem sind alle Arten von Graphikbearbeitungen (wie Vergrößerung/Verkleinerung und Drehung) sowie Farbunterstützung verfügbar.

Diese Funktionen sind nicht Teil des $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ -Kernels, sondern werden vielmehr durch die Erweiterungspakete `color`, `graphics` und `graphicx` unterstützt. Da das $\text{T}_{\text{E}}\text{X}$ -Programm keine direkten Verfahren zur Graphikbearbeitung besitzt, müssen die Pakete auf die Funktionen zurückgreifen, die vom

„Treiber“ bereitgestellt werden, der zum Drucken der dvi-Datei verwendet wird. Leider unterstützen nicht alle Treiber dieselben Funktionen, und auch die internen Methoden, auf diese Erweiterungen zuzugreifen, variieren unter den Treibern. Folglich haben alle diese Pakete Optionen, wie dvips, die angeben, welcher externe Treiber verwendet wird. Durch dieses Verfahren werden unvermeidlichen geräteabhängige Daten an einer einzigen Stelle, der Dokumentenpräambel, eingefügt.

Die Pakete graphics und graphicx können beide verwendet werden, um L^AT_EX-Objekte zu skalieren, zu drehen und zu spiegeln oder Graphikdateien einzubinden, die mit anderen Programmen erstellt wurden. Der Unterschied zwischen den beiden besteht darin, dass graphics eine Kombination von Makros mit einer „Standard“- oder T_EX-ähnlichen Syntax verwendet, während das „erweiterte“ oder „verbesserte“ (eXtended) graphicx-Paket in den optionalen Argumenten des \includegraphics- und \rotatebox-Befehls Schlüssel-Wert-Paare benutzt.

10.2.1 Optionen für graphics und graphicx

Bei Verwendung der L^AT_EX-Graphikpakete wird zum Einbinden einer Graphik oder nach geometrischen Transformationen der notwendige Leerraum für das gesetzte Material auf der Ausgabeseite reserviert. Es ist jedoch die Aufgabe des *Gerätetreibers* (z.B. dvips, xdvi, dvipsone), die betreffende Einbindung oder Transformation durchzuführen und das korrekte Ergebnis zu zeigen. Weil verschiedene Treiber unterschiedliche Programmcodes benötigen, um Aktionen wie Drehungen auszuführen, muss der gewünschte Treiber als Option der Graphikpakete angegeben werden. Die Option dvips ist beispielsweise nötig, wenn eines der graphics-Pakete mit Tom Rokickis dvips-Programm verwendet wird, während die Option textures erforderlich ist, wenn graphics-Pakete auf einem Macintosh mit Blue Skys Textures-Programm eingesetzt werden.

Einige Treiber, wie Previewer, sind nicht in der Lage, alle gewünschten Funktionen durchzuführen. Dies kann dazu führen, dass die dargestellten Objekte mit dem umgebenden Text überlappen. Tabelle 10.1 auf der nächsten Seite zeigt die derzeit unterstützten Treiber und ihre potentiellen Beschränkungen. In unregelmäßigen Abständen wird auch Unterstützung für weitere Treiber angeboten, daher sollte man bei einem Treiber, der nicht in der Tabelle aufgeführt wird, die Online-Dokumentation des Paketes konsultieren, bzw. nach einer Treiber-Supportdatei im Internet suchen.

Der treiberspezifische Programmcode wird in Dateien mit der Erweiterung .def gespeichert – z.B. dvips.def für den PostScript-Treiber dvips. Da die meisten dieser Dateien von Dritten gepflegt werden, enthält die Standard-distribution von L^AT_EX nur eine Untermenge der verfügbaren Dateien und nicht unbedingt die neuesten Versionen. Während es normalerweise kein Problem ist, wenn L^AT_EX als Teil einer vollständigen T_EX-Installation eingerichtet wird, sollte man bei der manuellen Aktualisierung der L^AT_EX-Graphikpakete auf Inkompatibilitäten achten.

Es ist auch möglich, mithilfe der Deklaration \ExecuteOptions in der Konfigurationsdatei graphics.cfg einen Standardtreiber anzugeben. Zum Beispiel bewirkt die Deklaration \ExecuteOptions{emtex}, dass die emT_EX-

Einrichten eines
Standardtreibers

<i>Option</i>	<i>Autor des Treibers</i>	<i>Funktionsumfang</i>
<code>dvips</code>	T. Rokicki	Alle Funktionen (Option wird auch für <code>xdvi</code> genutzt)
<code>dviaw</code>	N. Beebe	Dateieinbindung nur mit Skalierung
<code>dvipdf</code>	S. Lesenko	Alle Funktionen
<code>dvilaser</code>	Arbortext	Dateieinbindung nur mit Skalierung
<code>dvipsone</code>	Y&Y	Alle Funktionen
<code>dvitops</code>	J. Clark	Alle Funktionen, aber ohne verschachtelte Rotationen
<code>dviwin</code>	H. Sendoukas	Dateieinbindung
<code>dviwindo</code>	Y&Y	Alle Funktionen
<code>emtex</code>	E. Mattes	Dateieinbindung, aber ohne Skalierung
<code>ln</code>	B. H Kelly	Dateieinbindung für DEC's LN03-Drucker
<code>oztex</code>	A. Trevorow	Dateieinbindung, Farbe, Rotation
<code>pdftex</code>	Hàn Thê Thành	Alle Funktionen
<code>pctexps</code>	PCTeX	Dateieinbindung, Farbe, Rotation
<code>pctexwin</code>	PCTeX	Dateieinbindung, Farbe, Rotation
<code>pctex32</code>	PCTeX	Alle Funktionen
<code>pctexhp</code>	PCTeX	Nur Dateieinbindung
<code>psprint</code>	A. Trevorow	Nur Dateieinbindung
<code>pubps</code>	Arbortext	Rotation, Dateieinbindung
<code>truetestex</code>	Kinch	Graphikeinbindung und einige Farben
<code>tcidvi</code>	Kinch	TrueTeX mit Extra-Unterstützung für Scientific Word
<code>textures</code>	Blue Sky	Alle Funktionen für Textures
<code>vtex</code>	Micropress	Alle Funktionen für VTeX

Tabelle 10.1: Überblick über Farb- und Graphikfunktionen der Gerätetreiber

Treiber als Standard verwendet werden. In diesem Fall nehmen die Graphikpakete den Treibercode für das emTeX T_EX-System auf einem PC auf, wenn das Paket ohne Treiberoption aufgerufen wird. Heutzutage werden die meisten T_EX-Installationen mit gebrauchsfertiger `graphics.cfg`-Datei ausgeliefert.

Zusätzlich zu den Treiberoptionen unterstützen die Pakete einige Optionen, die steuern, welche Funktionen aktiviert (oder deaktiviert) sind:

- `draft`** Unterdrückt alle „Sonder“-Funktionen, wie etwa die Einbindung externer Graphikdateien in der endgültigen Ausgabe. Das Layout der Seite ist nicht betroffen, weil L^AT_EX immer noch die Größeninformation über die Bounding Box des externen Objektes einliest. Diese Option ist von speziellem Interesse, während das Dokument noch entsteht und die (oft sehr großen) Graphikdateien nicht jedes Mal geladen werden sollen, wenn am Dokument gearbeitet wird. Wenn der `draft`-Modus aktiviert ist, wird das Bild durch eine Box der richtigen Größe ersetzt, die den Namen der externen Datei enthält.
- `final`** Das Gegenteil von `draft`. Diese Option kann nützlich sein, wenn z.B. als globale Option mit dem `\documentclass`-Befehl „`draft`“ definiert wurde (z.B. um überlaufende Zeilen anzuzeigen), aber nicht auch die Graphiken unterdrückt werden sollen.
- `hiresbb`** Sucht in PostScript-Dateien nach Bounding Box-Kommentaren, welche die Form `%HiResBoundingBox` haben (und typischerweise

reelle Werte haben), anstatt nach den %%BoundingBox-Standardkommentaren (die ganzzahlige Werte haben sollten). Mit dem `graphicx`-Paket sind diese und die vorherigen Optionen auch lokal für einzelne `\includegraphics`-Befehle verfügbar.

`hiderotate` Zeigt die gedrehten Objekte nicht an (z.B. wenn der Previewer Objekte nicht drehen kann und Fehlermeldungen erzeugen würde).

`hidescale` Zeigt skalierte Objekte nicht an (z.B. wenn der Previewer Skalierung nicht unterstützt).

10.2.2 Die `\includegraphics`-Syntax im `graphics`-Paket

Bei Verwendung des `graphics`-Paketes kann eine Bilddatei mit folgendem Befehl eingebunden werden:

```
\includegraphics*[ulx,uly] [orx,ory] {datei}
```

Wenn das Argument [*orx,ory*] vorhanden ist, gibt es die Koordinaten der oberen rechten Ecke des Bildes in Form von zwei TeX-Maßen an. Die voreingestellte Einheit sind PostScript-Punkte (TeX-Name: „big points“, `bp`); d.h. `[1in,1in]` ist dasselbe wie `[72,72]`. Wenn nur ein optionales Argument angegeben ist, geht das System davon aus, dass sich die untere linke Ecke des Bildes an der Position `[0,0]` befindet. Ansonsten wird dieser Punkt durch die Koordinaten [*ulx,uly*] definiert. Ohne optionale Argumente wird die Größe der Graphik ermittelt, indem die externe *datei* eingelesen wird (die entweder die Graphik oder eine Beschreibung derselben enthält, siehe unten).

Die Sternform des `\includegraphics`-Befehls „beschneidet“ die Größe des Bildes auf die angegebene Bounding Box. Im Normalfall (ohne `*`) überschreibt der Teil der Graphik, der über die angegebene Bounding Box hinausragt, den umgebenden Text.

Die Beispiele in diesem und dem nächsten Abschnitt verwenden ein kleines PostScript-Programm (in der Datei `w.eps`), die den Großbuchstaben „W“ und ein paar Linien zeichnet. Der Quellcode ist in Abbildung 10.1 auf der gegenüberliegenden Seite zu sehen. Ein besonderes Augenmerk sollte auf die `BoundingBox`-Deklaration gerichtet sein. Sie legt fest, dass das Bild am Punkt `100,100` beginnt (in PostScript-Punkten) und am Punkt `150,150` endet; d.h. es hat eine natürliche Größe von `50x50` PostScript-Punkten.

In den Beispielen wird der Befehl `\includegraphics` stets in eine `\fbox` (mit blauem Rahmen und null `\fboxsep`) eingebettet. Dies soll zeigen, wie viel Platz L^AT_EX für die eingebundene Graphik reserviert. Außerdem wird die Grundlinie durch eine horizontale Linie angedeutet, die mit dem Befehl `\HR` erzeugt wird, der als Abkürzung für `\rule{1em}{0.4pt}` definiert wurde.

Das erste Beispiel demonstriert die Einbindung der Graphik `w.eps` in seiner natürlichen Größe. Hier passen Bild und Bounding Box gut zusammen.



```
\usepackage{graphics,color}
\newcommand\HR{\rule{1em}{0.4pt}}
\newcommand\bluefbox[1]{\textcolor{blue}{%
  \setlength\fboxsep{0pt}\fbox{\textcolor{black}{#1}}}}
links\HR \bluefbox{\includegraphics{w.eps}}\HR rechts
```

Bsp.
10-2-1

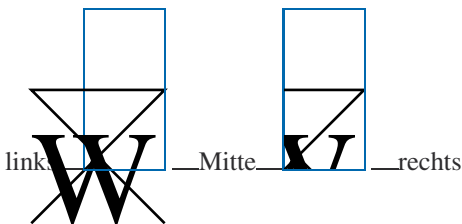
```

%!PS-Adobe-2.0
%%BoundingBox:100 100 150 150
100 100      translate % Ursprung bei 100 100 setzen
  0 0        moveto    % aktuelle Position definieren
  50 50      rlineto    % diagonale Linie markieren
  50 neg 0    rlineto    % horizontale Linie markieren
  50 50 neg   rlineto    % andere diagonale Linie markieren
stroke       % Linien zeichnen (Strich)
  0 0        moveto    % aktuelle Position neu definieren
/Times-Roman findfont % Times-Roman Font verwenden
  50          scalefont % auf 50 PostScript-Punkte skalieren
              setfont   % als aktuellen Font festlegen
(W) show     % Großbuchstabe W zeichnen

```

Abbildung 10.1: Inhalt der Datei w.eps

Als Nächstes wird eine Box definiert, die einem Teil des Bildes entspricht (und einem Bereich außerhalb), so dass einige Teile über die Boxgrenzen laufen und den Text überlagern, der um das Bild fließt. Bei Verwendung der Sternform des Befehls wird das Bild beschnitten, so dass es in die Box passt, wie der rechte Teil zeigt.



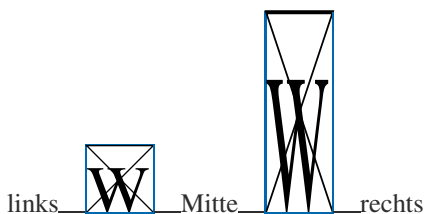
```

\usepackage{graphics,color}
% \bluefbox und \HR wie zuvor
links\HR
  \bluefbox{\includegraphics
    [120,120] [150,180]{w.eps}}
\HR Mitte\HR
  \bluefbox{\includegraphics*
    [120,120] [150,180]{w.eps}}
\HR rechts

```

Bsp.
10-2-2

In den übrigen Beispielen wird der Befehl `\includegraphics` mit anderen Befehlen des `graphics`-Paketes kombiniert. Dies soll zeigen, welche vielfältigen Möglichkeiten es zur Behandlung einer eingebundenen Graphik gibt. (Die genaue Syntax wird ausführlich in Abschnitt 10.3 besprochen.) Den Anfang macht eine Kombination aus `\scalebox`- und `\resizebox`-Befehlen. In beiden Fällen kann man entweder ein Maß angeben und das andere proportional skalieren oder beide Maße angeben, um das Bild zu verzerren.



```

\usepackage{graphics,color}
% \bluefbox und \HR wie zuvor
links\HR
  \bluefbox{\scalebox{.5}{%
    \includegraphics{w.eps}}}%
\HR Mitte\HR
  \bluefbox{\scalebox{.5}[1.5]{%
    \includegraphics{w.eps}}}%
\HR rechts

```

Bsp.
10-2-3



```
\usepackage{graphics,color}
% \bluefbox und \HR wie zuvor
links\HR
\bluefbox{\resizebox{10mm}{!}{}%
\includegraphics{w.eps}}}%
\HR Mitte\HR
\bluefbox{\resizebox{20mm}{10mm}{%
\includegraphics{w.eps}}}%
\HR rechts
```

Bsp.
10-2-4

Wenn man das Ganze noch dreht, wird die Sache noch viel interessanter. Im Vergleich zu Beispiel 10-2-1 auf Seite 634 sieht man, dass der von $\text{\LaTeX}$ reservierte Leerraum sehr viel größer ist. $\text{\LaTeX}$ „denkt“ in rechteckigen Boxen, also wählt es die kleinste Größe, die das gedrehte Objekt aufnehmen kann.



```
\usepackage{graphics,color}
% \bluefbox und \HR wie zuvor
links\HR
\bluefbox{\rotatebox{25}{%
\includegraphics{w.eps}}}%
\HR rechts
```

Bsp.
10-2-5

10.2.3 Die `\includegraphics`-Syntax im `graphicx`-Paket

Das erweiterte Graphikpaket `graphicx` verwendet ebenfalls den Befehlsnamen `\includegraphics`, bietet aber eine Syntax zur Einbindung externer Graphikdateien, die etwas transparenter und benutzerfreundlicher ist. Mit den heutigen $\text{\TeX}$ -Implementationen ist der entstehende Overhead vernachlässigbar, deshalb wird diese Schnittstelle sehr empfohlen.

`\includegraphics*[schlüssel-wert-liste]{datei}`

Die Sternform des Befehls gibt es nur aus Kompatibilitätsgründen zur Standardversion von `\includegraphics`, die in Abschnitt 10.2.2 beschrieben wurde. Sie entspricht dem `clip`-Schlüssel.

Die *schlüssel-wert-liste* ist eine durch Kommas getrennte Liste von *schlüssel=wert*-Paaren für Schlüssel, die einen Wert annehmen können. Bei booleschen Schlüsseln entspricht die reine Angabe des Schlüssels (ohne Wert) *schlüssel=true*; keine Angabe ist gleichbedeutend mit *schlüssel=false*. Mögliche Schlüssel sind:

bb Die Bounding Box für die Graphik. Das Wertefeld muss vier, durch Leerzeichen getrennte Werte enthalten.

bbllx, bbllly, bburx, bbury Die *x*- und *y*-Koordinaten für unten links und oben rechts (veraltet¹).

¹Nur aus Kompatibilitätsgründen vorhanden. Die Form `[bbllx=a, bbllly=b, bburx=c, bbury=d]` ist äquivalent zu `[bb = a b c d]`, daher sollte letztere benutzt werden.

- hiresbb** Lässt $\text{\LaTeX}$ nach `%%HiResBoundingBox`-Kommentaren suchen, anstatt nach dem üblichen `%%BoundingBox`. Einige Anwendungen verwenden diesen DSC-Schlüssel¹, um genauere Bounding Boxen zu definieren, da bei `%%BoundingBox` nur ganzzahlige Werte verwendet werden können. Der Schlüssel ist ein boolescher Wert, und d.h. entweder „true“ oder „false“.
- viewport** Hat vier Argumente (wie `bb`), aber in diesem Fall wird der Ursprung in Bezug auf die Bounding Box angegeben, die in der Datei definiert ist. Um ein 20bp großes Quadrat in der linken unteren Ecke des Bildes darzustellen, muss man z.B. `viewport=0 0 20 20` angeben.
- trim** Ähnlich wie der Schlüssel `viewport`, jedoch entsprechen die vier Maße dem Leerraum, der links, unterhalb, rechts und oberhalb der eingebundenen Graphik abgeschnitten werden soll.
- natheight, natwidth** Die natürliche Höhe und Breite des Bildes.²
- angle** Der Drehwinkel (in Grad, gegen den Uhrzeigersinn).
- origin** Der Ursprung für die Drehung, ähnlich wie der `origin`-Schlüssel des `\rotatebox`-Befehls, der auf Seite 650 und in Abbildung 10.2 auf Seite 650 beschrieben wird.
- width** Die benötigte Breite (die Breite des Bildes wird auf diesen Wert skaliert).
- height** Die benötigte Höhe (die Höhe des Bildes wird auf diesen Wert skaliert).
- totalheight** Die benötigte Gesamthöhe (Höhe + Tiefe des Bildes wird auf diesen Wert skaliert). Dieser Schlüssel sollte anstelle von `height` verwendet werden, wenn Bilder um mehr als 90 Grad gedreht werden, weil die Höhe verschwinden kann (und zu `depth` wird), so dass $\text{\LaTeX}$ möglicherweise die Anfrage des Benutzers nicht erfüllen kann.
- keepaspectratio** Eine boolesche Variable, die den Wert „true“ oder „false“ annehmen kann (Standardwerte siehe oben). Bei true wird das Bild nicht verzerrt, wenn sowohl `width` als auch `height` angegeben wurden, sondern skaliert, so dass weder `width` noch `height` die angegebenen Maße *überschreiten*.
- scale** Der Skalierungsfaktor.
- clip** Beschneidet die Graphik auf die Bounding Box. Er ist ein boolescher Wert, entweder „true“ oder „false“.

¹Von Adobe definierte „Document Structuring Conventions“.

²Diese Argumente können benutzt werden, um die untere linke Koordinate auf (0 0) und die obere rechte Koordinate auf (natwidth natheight) zu setzen. Damit sind sie gleichbedeutend mit `bb=0 0 w h`, wobei w und h die Werte sind, die für diese beiden Parameter angegeben sind.

<code>draft</code>	Schaltet lokal auf den Entwurfsmodus um. Ein boolescher Wertschlüssel, wie <code>clip</code> .
<code>type</code>	Der Graphiktyp; siehe Abschnitt 10.2.5.
<code>ext</code>	Die Dateierweiterung der Datei, welche die Bilddaten enthält.
<code>read</code>	Die Dateierweiterung der Datei, die von $\text{\LaTeX}$ gelesen wird, um nötigenfalls die Bildgröße zu ermitteln.
<code>command</code>	Befehl bzw. Programm, das auf die Datei angewendet wird.

Wenn für die ersten acht Schlüssel (`bb` bis `trim`) Größen ohne Einheiten angegeben werden, geht $\text{\LaTeX}$ von $\text{\TeX}$ s `bp`-Einheiten aus („big points“ sind äquivalent zu PostScript-Punkten).

Die ersten zehn Schlüssel (`bb` bis `natwidth`) definieren die Bildgröße. Diese Information muss für den Fall angegeben werden, dass $\text{\TeX}$ die Datei nicht lesen kann, die Datei falsche Größeninformationen enthält oder das Bild auf ein bestimmtes Rechteck zurechtgeschnitten werden soll.

Die nächsten sieben Schlüssel (`angle` bis `scale`) befassen sich mit der Skalierung oder Drehung der eingebundenen Objekte. Ähnliches erreicht man mit dem `graphics`-Paket und dem Befehl `\includegraphics`. Dafür muss man letzteren in das Argument eines `\resizebox`-, `\rotatebox`- oder `\scalebox`-Befehls platzieren (siehe die Beispiele in Abschnitt 10.2.2 und die ausführliche Besprechung dieser Befehle in Abschnitt 10.3).

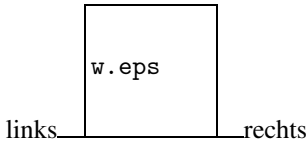
Zu beachten ist, dass Schlüssel von links nach rechts gelesen werden. Demnach bedeutet `[angle=90, totalheight=2cm]` eine Drehung um 90 Grad mit anschließender Skalierung auf eine Höhe von 2 cm, während `[totalheight=2cm, angle=90]` zu einer *Breite* von 2 cm führt.

Standardmäßig reserviert $\text{\LaTeX}$ für das Bild den Leerraum, der entweder in der Datei oder der *schlüssel-wert-liste* angegeben ist. Wenn ein Teil des Bildes aus diesem Bereich hinausragt, überschreibt es den umgebenden Text. Bei Verwendung der Sternform oder der `clip`-Option werden die Teile, die aus diesem Bereich herausfallen, nicht gedruckt.

Die letzten vier Schlüssel (`type`, `ext`, `read` und `command`) unterdrücken die Syntaxanalyse des Dateinamens. Wenn sie verwendet werden, sollte das Hauptargument *datei* keine Dateierweiterung haben (siehe Beschreibung des Befehls `\DeclareGraphicsRule` unten).

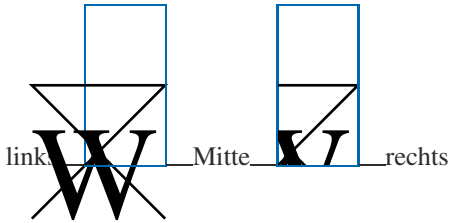
Unten werden einige der Beispiele aus Abschnitt 10.2.2 wiederholt. Dabei wird die Syntax des `graphicx`-Paketes verwendet, um die zusätzlichen Möglichkeiten zu zeigen, die das erweiterte Paket bietet. In den meisten Fällen ist die neue Form einfacher zu verstehen als die frühere Version. Im einfachsten Fall ohne optionale Argumente ist die Syntax für den Befehl `\includegraphics` in beiden Paketen gleich.

Bei Verwendung des `draft`-Schlüssels wird nur ein Rahmen dargestellt, der die Bounding Box zeigt. Das `graphics`-Paket bietet diese Funktion für einzelne Graphiken nicht an.

Bsp.
10-2-6

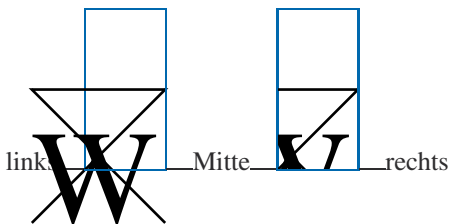
```
\usepackage{graphicx}
% \HR wie zuvor
links\HR
\includegraphics[draft]{w.eps}%
\HR rechts
```

Die Auswirkungen der Schlüssel `bb`, `clip`, `viewport` und `trim` werden in den folgenden Beispielen gezeigt. Man vergleiche hierzu Beispiel 10-2-2 auf Seite 635.

Bsp.
10-2-7

```
\usepackage{graphicx,color}
% \bluebox und \HR wie zuvor
links\HR\bluebox{\includegraphics
[bb=120 120 150 180]{w.eps}}%
\HR Mitte\HR
\bluebox{\includegraphics
[bb=120 120 150 180,clip]{w.eps}}%
\HR rechts
```

Anstelle der Angabe einer Bounding Box (Schlüssel `bb`) kann man dieses Ergebnis auch mit den Schlüsseln `viewport` oder `trim` erzeugen. Man beachte, dass im nächsten Beispiel zur Beschneidung ein negativer Wert angegeben wurde, wodurch der für das Bild reservierte Leerraum tatsächlich vergrößert wurde.

Bsp.
10-2-8

```
\usepackage{graphicx,color}
% \bluebox und \HR wie zuvor
links\HR\bluebox{\includegraphics
[viewport=20 20 50 80]{w.eps}}%
\HR Mitte\HR
\bluebox{\includegraphics
[trim= 20 20 0 -30,clip]{w.eps}}%
\HR rechts
```

Wenn das Bild skaliert werden soll, ist der Schlüssel `scale` zu verwenden. Mit diesem Schlüssel kann man das Bild jedoch nur gleichmäßig in beiden Richtungen skalieren.

Bsp.
10-2-9

```
\usepackage{graphicx,color}
% \bluebox und \HR wie zuvor
links\HR \bluebox{\includegraphics[scale=.4]{w.eps}}\HR rechts
```

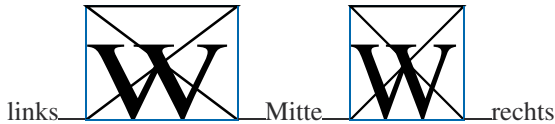
Um die Maße eines Bildes einem gegebenen Wert anzupassen, sollte man den Schlüssel `width` oder `height` verwenden (das jeweils andere Maß wird dann entsprechend skaliert). Wenn beide Schlüssel gleichzeitig verwendet werden, kann das Bild verzerrt werden, damit es in ein bestimmtes Rechteck passt (siehe nächstes Beispiel).



```
\usepackage{graphicx,color}
% \bluefbox und \HR wie zuvor
links\HR \bluefbox{\includegraphics
[width=15mm]{w.eps}}%
\HR Mitte\HR
\bluefbox{\includegraphics
[height=15mm,width=20mm]{w.eps}}%
\HR rechts
```

Bsp.
10-2-10

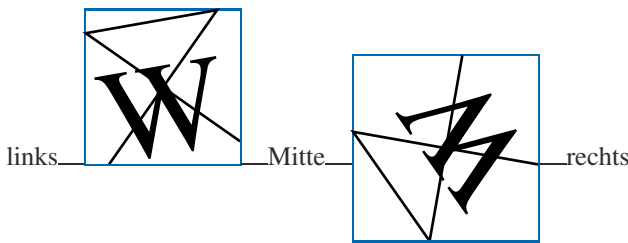
Mit dem Schlüssel `keepaspectratio` kann man sicherstellen, dass das Seitenverhältnis des Bildes erhalten bleibt. Wenn dieser angegeben ist, passt $\LaTeX$ das Bild, so gut es kann, an das Rechteck an.



```
\usepackage{graphicx,color}
% \bluefbox und \HR wie zuvor
links\HR \bluefbox{\includegraphics
[height=15mm,width=20mm]{w.eps}}%
\HR Mitte\HR
\bluefbox{\includegraphics[height=15mm,
width=20mm,keepaspectratio]{w.eps}}%
\HR rechts
```

Bsp.
10-2-11

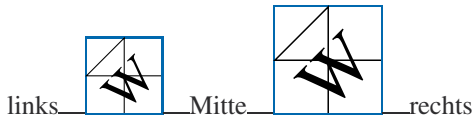
Eine weitere Komplexität entsteht durch Drehungen, die mit dem Schlüssel `angle` realisiert werden. Der Bezugspunkt für die Drehung ist der Bezugspunkt der Originalgraphik – in der Regel die untere linke Ecke, wenn die Graphik keine Tiefe hat. Durch Drehung um diesen Punkt ändern sich Höhe und Tiefe, so dass sich die Graphik in Bezug auf die Grundlinie rauf und runter bewegt. Dies zeigen die nächsten Beispiele.



```
\usepackage{graphicx,color}
% \bluefbox und \HR wie zuvor
links\HR
\bluefbox{\includegraphics
[angle=10]{w.eps}}%
\HR Mitte\HR
\bluefbox{\includegraphics
[angle=125]{w.eps}}%
\HR rechts
```

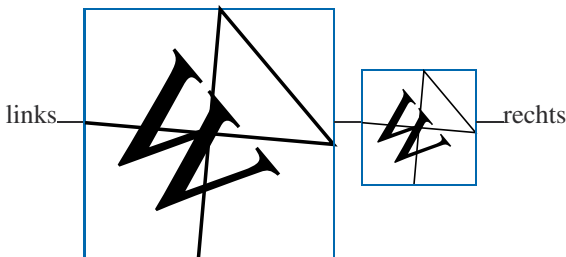
Bsp.
10-2-12

Der Spaß beginnt erst richtig, wenn sowohl ein Maß als auch ein Drehwinkel für ein Bild angegeben wird, denn hierbei ist die Reihenfolge wichtig. Das `graphicx`-Paket interpretiert die Schlüssel *von links nach rechts*. Besonders auffassen muss man, wenn ein Bild gedreht werden und eine bestimmte Höhe haben soll. Die nächsten Beispiele zeigen die Unterschiede zwischen der Angabe eines Drehwinkels vor und nach der Skalierung. Im ersten Fall wird das Bild zunächst gedreht, bevor das Ergebnis skaliert wird. Im zweiten Fall wird das Bild erst skaliert und dann gedreht.

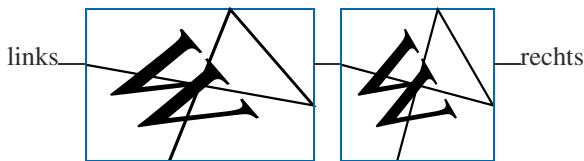
Bsp.
10-2-13

```
\usepackage{graphicx,color}
% \bluefbox und \HR wie zuvor
links\HR\bluefbox{\includegraphics
[angle=45,width=10mm]{w.eps}}%
\HR Mitte\HR
\bluefbox{\includegraphics
[width=10mm,angle=45]{w.eps}}%
\HR rechts
```

LaTeX betrachtet die Höhe und Tiefe der gedrehten Bounding Box separat. Der Schlüssel `height` bezieht sich nur auf die Höhe; d.h. die Tiefe bleibt unberücksichtigt. Im Allgemeinen sollte die Gesamthöhe eines (gedrehten) Bildes in einen gegebenen Freiraum passen, dafür sollte man den Schlüssel `totalheight` verwenden (siehe Abbildung 10.2 auf Seite 650 für eine Beschreibung der verschiedenen Maße, die eine LaTeX-Box definieren). Natürlich kann man die Schlüssel `height` und `width` und Drehungen beliebig kombinieren und verändern, um spezielle Effekte zu erzielen.

Bsp.
10-2-14

```
\usepackage{graphicx,color}
% \bluefbox und \HR wie zuvor
links\HR\bluefbox{%
\includegraphics
[angle=-50,height=15mm]%
{w.eps}}\HR
\bluefbox{%
\includegraphics
[angle=-50,totalheight=15mm]%
{w.eps}}\HR rechts
```

Bsp.
10-2-15

```
\usepackage{graphicx,color}
% \bluefbox und \HR wie zuvor
links\HR\bluefbox{\includegraphics
[angle=-60,totalheight=20mm,%
width=30mm]{w.eps}}\HR
\bluefbox{\includegraphics
[angle=-60,totalheight=20mm,%
width=30mm,keepaspectratio]%
{w.eps}}\HR rechts
```

10.2.4 Ändern der Voreinstellungen des graphicx-Paketes

Anstatt dieselben Schlüssel-Wert-Paare immer wieder im optionalen Argument der `\includegraphics`-Befehle einzugeben, kann man globale Werte für Schlüssel definieren, die solchen Befehlen zugeordnet sind. Für diesen Zweck gibt es die Deklaration `\setkeys`, die vom `keyval`-Paket angeboten wird. Dieses Paket wird automatisch eingebunden, wenn das `graphicx`-Paket verwendet wird.

```
\setkeys{bezeichner}{schlüssel-wert-liste}
```

Der *bezeichner* ist eine beliebige Zeichenfolge, die von demjenigen definiert wird, der das Makro geschrieben hat. Für `\includegraphics` wurde beispielsweise die Zeichenfolge `Gin` ausgewählt. Die *schlüssel-wert-liste* ist eine durch Kommas getrennte Liste von Schlüssel-Wert-Paaren.

Als Beispiel hierfür sei der Fall angeführt, in dem `graphicx` verwendet wird, um alle Abbildungen auf die Zeilenbreite zu skalieren. Dazu wird Folgendes angegeben:

```
\setkeys{Gin}{width=\linewidth}
```

Alle Bilder, die mit dem Befehl `\includegraphics` eingebunden werden, werden anschließend automatisch auf die aktuelle Zeilenbreite skaliert. (In solch einem Fall ist es empfehlenswert, anstelle von `\columnwidth` den Parameter `\linewidth` einzusetzen, da letzterer den Wert ändert, abhängig davon, wie die übergeordnete Umgebung, z.B. `quote`, aussieht.)

Auf ähnliche Weise kann man Standardwerte für jeden Schlüssel, der mit dem Befehl `\rotatebox` verwendet wird, so definieren (dem anderen Befehl, der eine Schlüssel-Wert-Syntax hat, wenn `graphicx` verwendet wird). Er hat den *bezeichner* `Grot`. Entsprechend gibt

```
\setkeys{Grot}{origin=ct}
```

an, dass in allen `\rotatebox`-Befehlen stets der Wert `ct` für den Schlüssel `origin` verwendet werden soll, sofern er nicht lokal überschrieben wird.

10.2.5 Deklarationen zum Einbinden von Bildern

Während Schlüssel-Wert-Paare nur gesetzt werden können, wenn das `graphicx`-Paket verwendet wird, können die in diesem Abschnitt beschriebenen Deklarationen sowohl mit dem `graphics`- als auch mit dem `graphicx`-Paket verwendet werden.

Verzeichnis für
Graphikdateien

Standardmäßig sucht $\text{\LaTeX}$ Graphikdateien in den Verzeichnissen, in denen es auch nach anderen Dateien sucht. Für größere Projekte kann es empfehlenswerter sein, alle Bilddateien zusammen in einem oder mehreren gemeinsamen Verzeichnis(sen) abzulegen. Eine Liste mit Verzeichnissen, die $\text{\LaTeX}$ mitteilt, wo Graphikdateien zu finden sind, kann mit dem Befehl `\graphicspath` angelegt werden. Der Befehl hat ein Argument: die Liste der Verzeichnisse, wobei jedes in geschweifte Klammern `{}` eingefasst wird (selbst wenn die Liste nur aus einem Verzeichnis besteht). Zum Beispiel bewirkt

```
\graphicspath{{./eps/}{./tiff/}}
```

dass $\text{\LaTeX}$ in den Unterverzeichnissen `eps` und `tiff` des aktuellen Verzeichnisses sucht.

Suchreihenfolge der
Dateierweiterung
definieren

Der Befehl `\DeclareGraphicsExtensions` ermöglicht es, das Verhalten des Systems zu definieren, wenn keine Dateierweiterung im Argument des Befehls `\includegraphics` angegeben ist. Sein Argument *{erweiterungs-liste}* ist eine durch Kommas getrennte Liste der Dateierweiterungen. Die vollständigen Dateinamen werden erzeugt, indem nacheinander jede Erweiterung der

erweiterungs-liste angehängt wird, bis eine Datei gefunden wird, die dem generierten Dateinamen entspricht.

Wenn der Befehl `\includegraphics` ohne Erweiterung angegeben wird und dieser Algorithmus das Vorhandensein einer Datei überprüft, um herauszufinden, welche Erweiterung zu verwenden ist, muss die Graphikdatei vorhanden sein, wenn das Dokument kompiliert wird. Wenn jedoch eine Dateierweiterung angegeben ist, wie etwa `\includegraphics{gr.eps}` statt `\includegraphics{gr}`, muss die Graphikdatei beim Start von $\text{\LaTeX}$ nicht vorhanden sein.¹ $\text{\LaTeX}$ muss jedoch die Größe des Bildes kennen. Diese muss in den Argumenten des `\includegraphics`-Befehls oder in einer Datei angegeben werden, die von $\text{\LaTeX}$ eingelesen wird. (Diese Datei kann entweder die Graphikdatei selbst oder eine andere Datei sein, die bei dem Schlüssel `read` angegeben wird oder aus der Liste der Dateierweiterungen konstruiert wird. Im letzteren Fall muss diese Datei während des $\text{\LaTeX}$ -Laufs vorhanden sein.)

Mit der unten gezeigten Deklaration sucht der Befehl `\includegraphics` zunächst nach der Datei `file.ps` und, wenn keine solche Datei existiert, nach der Datei `file.ps.gz`:

```
\DeclareGraphicsExtensions{.ps,.ps.gz}
\includegraphics{file}
```

Wenn man sicherstellen möchte, dass immer der vollständige Dateiname angegeben werden muss, sollte man nachfolgende Deklaration verwenden.

```
\DeclareGraphicsExtensions{}
```

Welche Aktion beim Antreffen einer Datei mit einer bestimmten Erweiterung durchgeführt werden soll, lässt sich mit folgendem Befehl steuern:

```
\DeclareGraphicsRule{erweiterung}{typ}{datei-lesen}{befehl}
```

Diese Deklaration kann mehrfach (für unterschiedliche Dateierweiterungen) verwendet werden. Die Bedeutung dieser Argumente wird unten beschrieben.

erweiterung Die Erweiterung der Graphikdatei, auf die diese Regel zutreffen soll. Um ein Standardverhalten für alle nicht explizit deklarierten Erweiterungen zu definieren, wird ein Stern (*) verwendet. Zum Beispiel bewirkt

```
\DeclareGraphicsRule{*}{eps}{*}{*}
```

dass alle nicht definierten Erweiterungen als EPS-Dateien behandelt werden und die entsprechenden Graphikdateien nach einem `%%BoundingBox`-Kommentar abgesucht werden.

typ Der „Typ“ der involvierten Datei. Alle Dateien desselben Typs werden mit demselben internen Befehl geladen (der in der entsprechenden Treiberdatei definiert werden muss). Beispielsweise sollten alle Dateien mit der Erweiterung `.ps`, `.eps` oder `.ps.gz` als Typ `eps` eingestuft werden.

¹Sie kann zum Beispiel mit einer geeigneten `\DeclareGraphicsRule`-Deklaration ad hoc generiert werden.

	<i>erweiterung</i>	<i>typ</i>	<i>datei-lesen</i>	<i>befehl</i>
<i>Basis-PostScript</i>	.ps	eps	.ps	
	.eps	eps	.eps	
<i>Dynamische Dekomprimierung</i>	.pz	eps	.bb	'gunzip -c #1
	.ps.gz	eps	.ps.bb	'gunzip -c #1
	.eps.gz	eps	.eps.bb	'gunzip -c #1
<i>MS-DOS-bezogene Formate</i>	.tif	tiff		
	.pcx	bmp		
	.bmp	bmp		
	.msp	bmp		
<i>Mac-bezogene Formate</i>	.pict	pict		
	.pntg	pntg		

Tabelle 10.2: Argumente von `\DeclareGraphicsRule`

datei-lesen Die Erweiterung der Datei, die gelesen werden soll, um die Größe der Graphik zu ermitteln. Sie kann mit der *erweiterung* übereinstimmen, aber bei komprimierten oder binären Bildern, die L^AT_EX nicht einfach interpretieren kann, wird die Größeninformation (die Bounding Box) normalerweise in einer separaten Datei vorgehalten. Beispielsweise könnten für die komprimierten gezippten PostScript-Dateien mit der Erweiterung `.ps.gz` entsprechende lesbare Dateien mit der Erweiterung `.ps.bb` existieren. Wenn das Argument *datei-lesen* leer ist (d.h. {}), sucht das System nicht nach einer externen Datei, um die Größe zu ermitteln, und die Größe muss in den Argumenten des Befehls `\includegraphics` angegeben werden. Wenn die Treiberdatei ein Verfahren zum Lesen der Größendatei des *typs* definiert, wird dieses Verfahren verwendet. Ansonsten wird das Verfahren zum Lesen von `.eps`-Dateien verwendet. Wenn kein anderes Format angegeben ist, wird daher die Größe einer Bitmap-Graphik bestimmt, indem die Syntax für PostScript-Graphiken verwendet wird (d.h. eine `%%BoundingBox`-Zeile gesucht wird).

befehl Der im Argument von `\special` einzufügende Befehl. Normalerweise ist *befehl* leer, aber bei komprimierten Dateien muss die Bilddatei möglicherweise vor der Einbindung in die zu druckende Datei entpackt werden – sofern der Treiber dies unterstützt. Beim dvips-Treiber kann man z.B. Folgendes definieren

```
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{'gunzip #1}
```

wobei das Argument `#1` den vollständigen Dateinamen angibt. In diesem Fall bewirkt das letzte Argument, dass dvips die Datei mit dem Befehl `gunzip` entpackt, bevor sie in der PostScript-Ausgabe eingefügt wird.

Tabelle 10.2 zeigt einige Beispielwerte, die in den Argumenten des Befehls `\DeclareGraphicsRule` verwendet werden können.

Das bislang beschriebene System kann Probleme aufwerfen, wenn die *erweiterung* nicht mit dem Argument *typ* übereinstimmt. Wenn man z.B. eine Reihe von PostScript-Dateien namens `datei.1`, `datei.2` ... hat, erkennt weder das `graphics`-Paket noch das `graphicx`-Paket automatisch, dass es sich um PostScript-Dateien handelt. Bei Verwendung des `graphicx`-Paketes kann dies erzwungen werden, indem man jedem `\includegraphics`-Befehl den Schlüssel `type=eps` mitgibt. Ein allgemeineres Verfahren ist, wie oben beschrieben, für den Typ `*` mit der `\DeclareGraphicsRule`-Deklaration einen Standardtyp zu definieren.

10.2.6 Eine Vorsichtsmaßnahme: Kapselung ist wichtig

Abschnitt 10.4 geht noch ausführlich auf PostScript ein, aber es ist wichtig, schon an dieser Stelle darauf hinzuweisen, dass PostScript eine Seitenbeschreibungssprache ist, die sich mit dem Erscheinungsbild von *vollständigen gedruckten Seiten* befasst. Dies erschwert es Autoren, kleinere PostScript-Graphiken, die mit externen Werkzeugen erstellt wurden, in ihre elektronischen (L^AT_EX-)Dokumente einzubinden. Zur Lösung dieses Problems hat Adobe das *Encapsulated PostScript*-Dateiformat (EPS oder EPSF) definiert, das der *PostScript Document Structuring Conventions Specification* [3] und der *Encapsulated PostScript File Format Specification* [2] entspricht.

Das EPS-Format definiert einen Standard, der es erlaubt, Dateien, die in der PostScript-Sprache geschrieben sind, in verschiedenen Umgebungen problemlos einzubinden. EPS-Dateien sollten in Hinblick auf die Verwendung verschiedener PostScript-Operatoren, die Manipulation des „graphic state“, des Interpreter-Stapels und globaler „dictionaries“ wohlgeformt sein, so dass sie keine destruktiven Änderungen an dem vom Textformatierungsprogramm festgelegten Seitenaufbau hervorrufen.

Die meisten modernen Graphikanwendungen generieren EPS-konforme Dateien, die problemlos in L^AT_EX verwendet werden können. Manchmal kann es jedoch vorkommen, dass man auf eine reine PostScript-Datei trifft, welche nicht die notwendigen Strukturierungsinformationen enthält. Um sie mit L^AT_EX zu verwenden, muss eine PostScript-Datei nicht exakt den oben erwähnten Strukturierungskonventionen entsprechen. Wenn die Datei „wohlgeformt“ ist (siehe oben), reicht es aus, dass die PostScript-Datei die Maße des Rechtecks enthält, welches das Bild begrenzt. Diese Maße werden L^AT_EX über die PostScript-Kommentarzeile `%%BoundingBox` zur Verfügung gestellt, z.B. folgendermaßen:

```
%!  
%%BoundingBox: ulx uly orx ory
```

Die erste Zeile gibt an, dass es sich um eine nicht konforme EPS-Datei handelt. Zu beachten ist, dass die Zeichen `%!` in den ersten zwei Spalten der Zeile platziert werden *müssen*. Die zweite Zeile, die für den vorliegenden Fall viel wichtiger ist, gibt die Größe des eingefügten Bildes in PostScript-Punkten an (72 PostScript-Punkte entsprechen einem Zoll, siehe auch Tabelle A.1 auf Seite 888). Die vier in dieser Zeile angegebenen Parameter entsprechen den *x*- und *y*-Koordinaten der unteren linken Ecke (`ulx` und `uly`) sowie jenen der

oberen rechten Ecke (orx und ory) des Bildes. Eine ganze A4-Seite (210mm mal 297mm) mit dem Nullpunkt in der unteren linken Ecke würde beispielsweise folgendermaßen deklariert:

```
%!
%%BoundingBox: 0 0 595 842
```

Wenn das Bild bei (100,200) beginnt und in einem Quadrat von 4 Zoll (288 Punkten) eingefasst werden soll, lautet der Befehl:

```
%!
%%BoundingBox: 100 200 388 488
```

In einem PostScript-Previewer, wie ghostview, kann man die Bounding Box eines Bildes sehr einfach ermitteln, indem man den Cursor über die Kanten des Bildes fährt und die entsprechenden Koordinaten ausliest. Im Allgemeinen empfiehlt es sich, ein oder zwei Punkte hinzuzufügen, um sicherzustellen, dass das gesamte Bild eingebunden wird, da bei der Berechnung durch den Interpreter Rundungsfehler entstehen können.

10.3 Manipulieren von graphischen Objekten

Neben dem `\includegraphics`-Befehl bieten die Pakete `graphics` und `graphicx` eine Reihe von Befehlen zur Verarbeitung von Graphiken.

Mit Ausnahme des Befehls `\rotatebox`, der auch eine Schlüssel-Wert-Syntax im `graphicx`-Paket unterstützt, haben diese Befehle in beiden Paketen dieselbe Syntax.

10.3.1 Skalieren einer L^AT_EX-Box

Mit dem Befehl `\scalebox` können Text oder andere L^AT_EX-Objekte um einen bestimmten Skalierungsfaktor vergrößert oder verkleinert werden.

```
\scalebox{h-skalierung}[v-skalierung]{objekt}
```

Das erste Argument gibt den Faktor an, um den beide Maße des *objekts* skaliert werden sollen. Das folgende Beispiel zeigt, wie das funktioniert.

Dies ist normaler Text.

Dies ist größer und noch größerer Text.

Dies ist kleiner Text.

```
\usepackage{graphics} % oder graphicx
\noindent Dies ist normaler Text.      \\
\scalebox{2}{Dies ist größer und noch} \\
\scalebox{3}{größerer Text.}          \\
\scalebox{0.5}{Dies ist kleiner Text.}
```

Bsp.
10-3-1

Ein zusätzliches optionales Argument gibt, wenn vorhanden, einen separaten vertikalen Skalierungsfaktor an. Es wird in den folgenden Beispielen

vorgestellt. Diese zeigen auch, wie mehrere Zeilen am Stück mithilfe des L^AT_EX-Standardbefehls `\parbox` skaliert werden können.

Bsp.
10-3-2



```
\usepackage{graphics} % oder graphicx
\fbbox{\scalebox{1.5}{%
\parbox{.5in}{Amerika \&\Europa}}}
\fbbox{\scalebox{1.5}[1]{%
\parbox{.5in}{Amerika \&\Europa}}}
```

`\reflectbox{objekt}`

Dieser Befehl ist eine bequeme Abkürzung für `\scalebox{-1}[1]{objekt}`, wie das folgende Beispiel zeigt:

Bsp.
10-3-3

Amerika?
Amerika?

```
\usepackage{graphics} % oder graphicx
\noindent Amerika?\reflectbox{Amerika?} \\\
Amerika?\scalebox{-1}[1]{Amerika?}
```

Auch interessante Spezialeffekte können erzielt werden. Beachtenswert ist insbesondere die Verwendung der `\makebox`-Befehle ohne horizontale Ausdehnung. Sie verbergen ihren Inhalt vor L^AT_EX und bieten damit die Möglichkeit, die Positionierung des gesetzten Materials fein abzustimmen.

Bsp.
10-3-4

Amerika?
Amerika?
Amerika?
Amerika?

```
\usepackage{graphics} % oder graphicx
\noindent Amerika?\scalebox{-1}{Amerika?} \\\
Amerika?\scalebox{1}[-1]{Amerika?} \\\
Amerika?\makebox[0mm][r]{%
\scalebox{-1}{Amerika?}} \\\
\makebox[0mm][l]{Amerika?}%
\scalebox{1}[-1]{Amerika?}
```

10.3.2 Skalieren auf eine bestimmte Größe

Auch die horizontale und vertikale Größe, in der L^AT_EX-Objekte gesetzt werden sollen, kann festgelegt werden:

`\resizebox*{h-maß}{v-maß}{objekt}`

Wenn das Seitenverhältnis des Objektes erhalten bleiben soll, reicht es aus, eines der Maße einzugeben und das andere Maß durch ein „!“-Zeichen zu ersetzen.

Bsp.
10-3-5



```
\usepackage{graphics} % oder graphicx
\fbbox{\resizebox{5mm}{!}{%
\parbox{14mm}{London,\ Berlin \&\ Paris}}}
\fbbox{\resizebox{!}{10mm}{%
\parbox{14mm}{London,\ Berlin \&\ Paris}}}
```

Wenn sowohl das *h-maß* als auch das *v-maß* explizit angegeben sind, kann es vorkommen, dass das Objekt verzerrt wird. Im folgenden Beispiel

wird die Grundlinie durch eine horizontale Linie dargestellt, die mit dem `\HR`-Befehl erzeugt wurde.

```
\usepackage{graphics} % oder graphicx
\HR\begin{tabular}{lll}
    K\"oln & Lyon & Oxford \\
    Rhein & Rh\^one & Thames
\end{tabular}\HR\par\bigskip
\HR\resizebox{2cm}{.5cm}{%
    \begin{tabular}{lll}
        K\"oln & Lyon & Oxford \\
        Rhein & Rh\^one & Thames
    \end{tabular}}\HR
```

Bsp.
10-3-6

Wie bei $\text{\LaTeX}$ -Befehlen üblich, die sich auf Boxmaße beziehen, können in den Argumenten die natürlichen Längen `\depth`, `\height`, `\totalheight` und `\width` verwendet werden:

```
\usepackage{graphics} % oder graphicx
\HR\fbbox{\resizebox{\width}{.7\height}{%
    \parbox{14mm}{London,\& Berlin \&\&Paris}}}\HR
\fbbox{\resizebox{\width}{.7\totalheight}{%
    \parbox{14mm}{London,\& Berlin \&\&Paris}}}\HR
```

Bsp.
10-3-7

Die `\resizebox`-Form ohne Stern führt ihre Berechnungen auf Grundlage der Höhe des $\text{\LaTeX}$ -Objektes durch, während die Sternform `\resizebox*` die Gesamthöhe (Höhe und Tiefe) der $\text{\LaTeX}$ -Box berücksichtigt. Die `tabular`-Umgebungen im nächsten Beispiel, die eine große Tiefe haben, machen den Unterschied deutlich.

```
\usepackage{graphics}
\HR\resizebox{20mm}{30mm}{%
    \begin{tabular}{lll}
        K\"oln & Lyon & Oxford \\
        Rhein & Rh\^one & Thames
    \end{tabular}}\HR
\HR\resizebox*{20mm}{30mm}{%
    \begin{tabular}{lll}
        K\"oln & Lyon & Oxford \\
        Rhein & Rh\^one & Thames
    \end{tabular}}\HR
```

Bsp.
10-3-8

10.3.3 Drehen einer $\text{\LaTeX}$ -Box

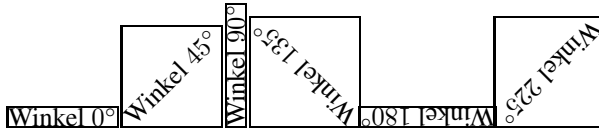
Mit dem Befehl `\rotatebox` können $\text{\LaTeX}$ -Objekte um einen Winkel gedreht werden. Eine alternative Methode, die eine Umgebung verwendet, wird in Abschnitt 10.3.4 beschrieben.

`\rotatebox{winkel}{objekt}`

Das Argument *objekt* wird in einer L^AT_EX-Box gesetzt und in einem Drehwinkel (in Grad) gegen den Uhrzeigersinn um den Bezugspunkt gedreht.

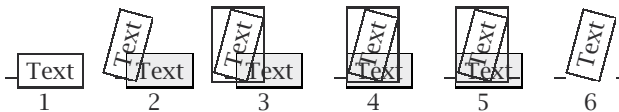
```
\usepackage{graphics} % oder graphicx
\newcommand\MyRot[1]{\frame
  {\rotatebox{#1}{Winkel
    $#1^\circ$}}}
\MyRot{0} \MyRot{45} \MyRot{90}
\MyRot{135} \MyRot{180} \MyRot{225}
```

Bsp.
10-3-9



Um zu verstehen, wo das gedrehte Objekt auf der Seite platziert wird, muss man sich näher mit dem Algorithmus befassen, der hierfür verwendet wird. Nachstehend werden die einzelnen Schritte gezeigt, die zur Drehung von `\fbox{text}` um 75 Grad notwendig sind. Schritt 1 zeigt den Text vor der Drehung. Die horizontale Linie auf der linken Seite kennzeichnet die Grundlinie. Zunächst wird das *objekt* (in diesem Fall `\fbox{text}`) in eine Box gesetzt. Diese Box hat einen Bezugspunkt, um den die Drehung standardmäßig ausgeführt wird. Dies ist in Schritt 2 dargestellt (die ursprüngliche Position des nicht gedrehten Objektes wird zur besseren Orientierung auch gezeigt). Anschließend berechnet der Algorithmus eine neue Bounding Box (d.h. den Platz, der für das gedrehte Objekt reserviert werden muss). Dies ist in Schritt 3 zu sehen. Als Nächstes wird das Objekt horizontal verschoben, so dass sich die linken Kanten der neuen und der alten Bounding Box an derselben Position befinden (Schritt 4). Danach wird T_EXs Satzposition verändert, so dass weiteres Material rechts der Bounding Box gesetzt wird. Dies zeigt die Linie, welche die Grundlinie in Schritt 5 darstellt. Schritt 6 zeigt das Endergebnis, wobei auch hier die Grundlinie auf beiden Seiten des rotierten Objektes angedeutet wird.

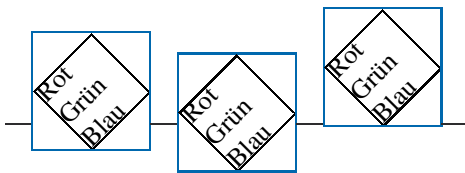
Drehalgorithmus



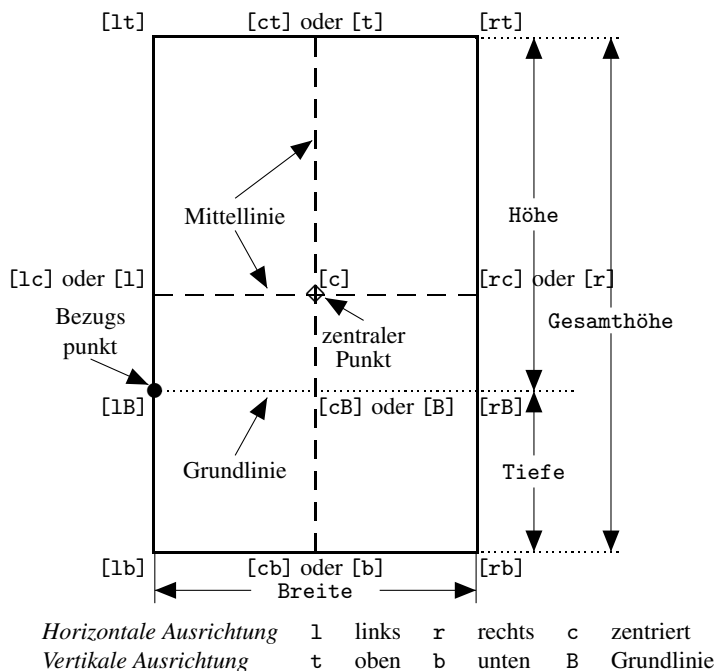
Bei komplexeren Gebilden ist es wichtig, dass man sich den Bezugspunkt der gedrehten Box merkt. Das folgende Beispiel zeigt, wie er verschoben wird, wenn man das Platzierungsargument des `\parbox`-Befehls verwendet.

```
\usepackage{color,graphics} % oder graphicx
\HR\bluefbox{\rotatebox{45}{%
  \fbox{\parbox{3em}{Rot\Grün\Blau}}}}%
\HR\bluefbox{\rotatebox{45}{%
  \fbox{\parbox[t]{3em}{Rot\Grün\Blau}}}}%
\HR\bluefbox{\rotatebox{45}{%
  \fbox{\parbox[b]{3em}{Rot\Grün\Blau}}}}\HR
```

Bsp.
10-3-10



Das erweiterte Graphikpaket `graphicx` bietet mehr Flexibilität, was die Angabe des Punktes betrifft, um den die Drehung durchgeführt wird. Hierzu verwendet es *schlüssel-wert*-Paare.



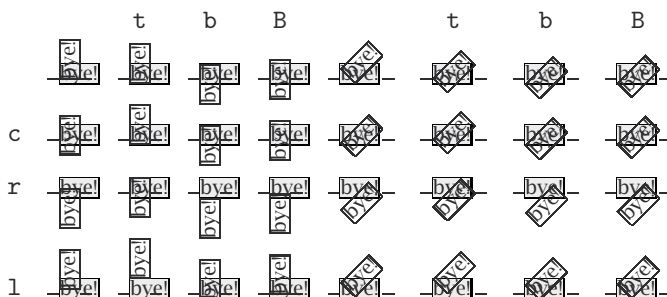
Bsp.
10-3-11

Abbildung 10.2: Eine $\text{\LaTeX}$ -Box und mögliche origin-Bezugspunkte

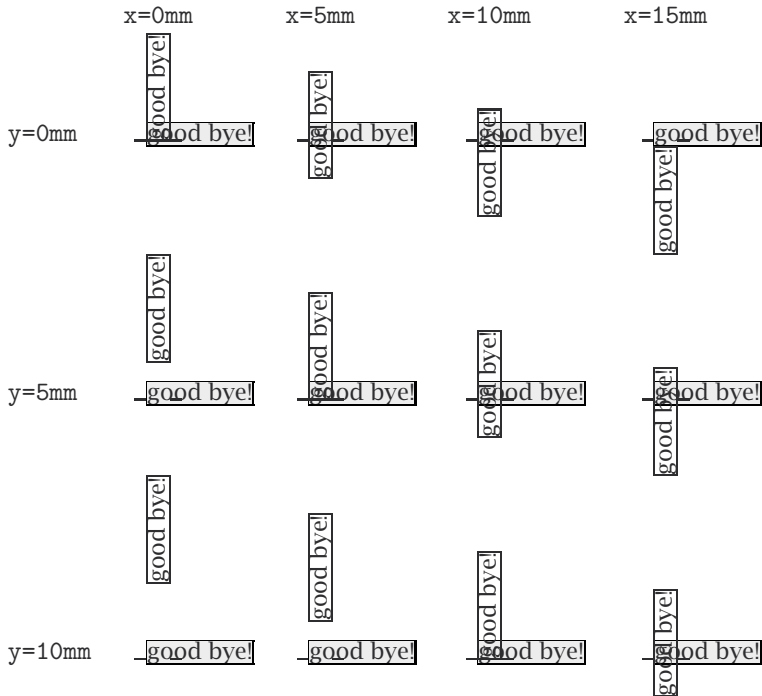
`\rotatebox[schlüssel-wert-liste]{winkel}{objekt}`

Für diesen Fall gibt es vier mögliche Schlüssel: `origin`, `x`, `y` und `units`. Die möglichen Werte für den Schlüssel `origin` sind in Abbildung 10.2 zu sehen (es kann jeweils ein Wert für die horizontale und vertikale Ausrichtung ausgewählt werden). Dort sieht man auch die tatsächlichen Positionen der einzelnen Kombinationen in Bezug auf die $\text{\LaTeX}$ -Box, die vom `objekt` erzeugt wurde.

Die Auswirkung der möglichen Kombinationen für den `origin`-Schlüssel lässt sich unten an einer Beispielbox nachverfolgen. Hier werden zwei Tabellen mit um 90 Grad und 45 Grad gedrehten Boxen gezeigt. Um diesen Effekt besser einordnen zu können, wird der nicht gedrehte Text dazu grau abgesetzt.



Wenn die Angabe von `origin` nicht ausreicht, kann man alternativ den Punkt, um den die Drehung ausgeführt werden soll, explizit durch seine x - und y -Koordinaten festlegen (relativ zum Bezugspunkt). Für diesen Zweck verwendet man die Schlüssel `x` und `y` und das Format `x=maß`, `y=maß`. Eine Matrix mit einigen Beispielwerten und ihren Auswirkungen auf eine Box, die um 90 Grad gedreht wird, folgt unten.



Die Interpretation des Argumentes *winkel* des Befehls `\rotatebox` kann über den Schlüssel `units` gesteuert werden. Er gibt die Anzahl der Einheiten (gegen den Uhrzeigersinn) in einem ganzen Kreis an. Voreingestellt ist 360. `units=-360` würde also heißen, dass die Drehwinkel im Uhrzeigersinn angegeben sind. Entsprechend ändert die Einstellung `units=6.283185` die Gradangaben in Radianen. Anstatt den Schlüssel `units` in jedem einzelnen `\rotatebox`-Befehl zu setzen, empfiehlt es sich, mit der Deklaration `\setkeys` eine Standardinterpretation zu definieren, wie es in Abschnitt 10.2.4 beschrieben ist.

10.3.4 Wiedersehen mit dem rotating-Paket

Die in diesem Abschnitt vorgestellte Funktion ähnelt der des `rotating`-Paketes von Sebastian Rahtz, das in Abschnitt 6.3.3 auf Seite 305 eingeführt wurde. Die Rotationsfunktion ist in diesem Paket über die Umgebungen `turn` und `rotate` realisiert; letztere erzeugt ein Objekt, das keinen Platz belegt. Das Ver-

wenden von Umgebungen hat den Vorteil, dass die gedrehten Objekte `\verb`-Befehle enthalten können. Die erweiterte Syntax des `\rotatebox`-Befehls wird nicht unterstützt, daher ist in den meisten Fällen der letztere Befehl vorzuziehen.

Drehe  ein wenig.

`\usepackage{rotating}`

Drehe `\begin{rotate}{-20}\Large\LaTeX\end{rotate}%`
`\begin{turn}{20}\verb=\LaTeX=\end{turn}` ein wenig.

Bsp.
10-3-12

10.4 Seitenbeschreibungssprachen – PostScript, PDF und SVG

Nachdem das elektronische Dokument gesetzt wurde, möchte man in der Regel die erzeugte Ausgabe-„Seite“ ansehen, entweder ausgedruckt auf Papier oder auf dem Computerbildschirm mit einem speziellen Programm oder im Browser oder (warum nicht?) auf einem Handy.

Im Laufe der Jahre wurden verschiedene Seitenbeschreibungssprachen entwickelt. Für Drucker hat PostScript, welches hauptsächlich eine Sprache zur Beschreibung der statischen Ausgabeseite ist, inzwischen eine Schlüsselrolle eingenommen. In den frühen 90er Jahren entwickelte Adobe eine abgespeckte PostScript-Version, das sogenannte Portable Document Format (PDF) [5]. PDF verwendet ein ähnliches Modell zur Beschreibung der Seiten wie PostScript, führt aber ein strukturiertes Format ein, um die Leistungsfähigkeit für die interaktive Ansicht zu verbessern. Außerdem fügt es Verknüpfungen und Kommentare für die Navigation hinzu.

Durch die drastisch gesunkenen Anschaffungspreise für PCs haben sich auch die Kosten für die Erstellung von elektronischen Dokumenten enorm reduziert. Das World Wide Web ermöglicht eine einfache, günstige und schnelle Verbreitung dieser Dokumente. Durch die Entwicklung der XML-Standards wurde ein einheitlicher Ansatz geschaffen, um sehr große, elektronisch gespeicherte Datenmengen zu handhaben und sie in verschiedene, individuell anpassbare Darstellungsformen umzuwandeln.

Inzwischen gibt es verschiedene Methoden, um $\text{\LaTeX}$ -Dokumente in PDF, HTML (XHTML) oder XML umzuwandeln, so dass die Information auf dem Internet bereitgestellt werden kann; mehrere Kapitel des *$\text{\LaTeX}$ Web Companion* [59] befassen sich mit diesen Methoden. Ein besonders interessanter Ansatz (der unten beschrieben ist) ist die Umwandlung von $\text{\LaTeX}$ -kodierte Daten in ein skalierbares Vektorgraphik (SVG)-Format.

Daher wird $\text{\LaTeX}$ auch weiterhin eine wichtige Rolle im integrierten weltweiten Cyberspace spielen. Besonders im Bereich der wissenschaftlichen Dokumente bleibt es ein wichtiges (Zwischen-)Format zur Erzeugung von qualitativ hochwertiger druckbarer PDF- oder Browser-fähiger SVG-Ausgabe.

Dieser Abschnitt gibt einen kurzen Überblick über die drei Seitenbeschreibungssprachen PostScript, PDF und SVG. Er geht kurz auf `dvips` ein, ein `dvi`-zu-PostScript-Übersetzungsprogramm, und erörtert dann `pspicture`, eine Erweiterung der $\text{\LaTeX}$ -Umgebung `picture`, die PostScript verwendet.

10.4.1 Die PostScript-Sprache

PostScript [4] ist eine Seitenbeschreibungssprache. Sie stellt Methoden zur Verfügung, um das Erscheinungsbild einer ausgedruckten Seite inklusive Text, Linien und Graphiken zu beschreiben.

PostScript ist eine geräte- und auflösungsunabhängige Programmiersprache, in der komplette Seiten als Einheit beschrieben werden. Die Sprache ist stapelorientiert und verwendet eine „umgekehrte polnische“ oder Postfix-Notation. Sie enthält auch Schleifen, Prozeduren und Vergleichsoperatoren und unterstützt viele Datentypen, zu denen auch reelle Zahlen, boolesche Datentypen, Felder, Zeichenketten und komplexe Objekte wie etwa „dictionaries“ gehören.

PostScript-Programme sind im Allgemeinen in Form eines ASCII-Quelltextes geschrieben, der einfach zu erzeugen, zu verstehen, zu übertragen und zu verarbeiten ist. PostScript ist auflösungs- und geräteunabhängig, d.h. dieselbe ASCII-Datei kann auf einem Computerbildschirm mit einem Previewer, wie etwa `ghostscript/ghostview`, angesehen werden und auf einem herkömmlichen Laserdrucker oder einer hochauflösenden Lichtsatzmaschine ausgedruckt werden.

Die PostScript-Sprache bietet folgende Möglichkeiten, die in jeder Kombination verwendet werden können.

- Aus Geraden, Bögen und kubischen Kurven können beliebige Formen erstellt werden. Die Formen können sich selbst schneiden und unzusammenhängende Abschnitte und Löcher enthalten.
- Die Zeichenfunktionen ermöglichen es, solche Formen mit Linien jeglicher Dicke zu umranden, sie mit einer beliebigen Farbe zu füllen oder sie zu verwenden, um andere Graphiken zu beschneiden.
- Text ist in dieses Graphikmodell nahtlos eingebunden. In PostScript werden Buchstaben und Symbole als graphische Formen behandelt, die mit jeglichem der Graphikoperatoren der Sprache manipuliert werden können. Dies gilt sowohl für Type 3-Fonts, in denen Zeichenformen als herkömmliche PostScript-Sprachprozeduren definiert sind, als auch für Type 1-Fonts, die auf einer speziellen, kleineren Sprache basieren, in der Zeichenformen über speziell kodierte Prozeduren definiert werden (siehe unten). Für komplexere Sprachen mit Tausenden von Zeichen (z.B. Chinesisch und Japanisch) können Kombinationsfonts (Type 0) verwendet werden.
- Bilder (wie etwa Photographien oder synthetisch hergestellte Bilder) können in jeder Auflösung und in vielen dynamischen Bereichen gerastert werden. PostScript enthält Funktionen, welche die Wiedergabe der Bilder auf dem Ausgabegerät steuern.
- Es werden mehrere Farbmodelle (RGB, HSB, CMYK, CIE) unterstützt. Konvertierung zwischen den Modellen ist möglich.
- Eine allgemeine Koordinatensystem-Funktion unterstützt alle Kombinationen linearer Transformationen inklusive Skalierung, Rotation, Spiegelung und Verzerrung. Diese Transformationen gelten einheitlich für alle

Elemente der Seite, inklusive Texte, Graphiken und gesampelter Bilder.

- Man kann „dictionaries“ für Vektorräume für Farbvalenzen, Zeichensätze, Formen, Bilder, Halbton- und andere Muster anlegen.
- Es stehen verschiedene Kompressionsfilter, wie z.B. JPEG und LZW, zur Verfügung, um den Datentransfer zwischen Computer und Drucker zu beschleunigen.

Type 1- und OpenType-Fontmerkmale

Als Komplement zur PostScript-Sprache definierte Adobe sein Type 1-Zeichensatzformat [1]. Ein Type 1-Zeichensatzprogramm besteht aus einem Klartext (ASCII)-Teil sowie einem kodierten und verschlüsselten Teil. Die Befehle der PostScript-Sprache, die in einem Type 1-Zeichensatzprogramm verwendet werden, folgen einer sehr viel strengeren Syntax als normale PostScript-Programme.

Wie PostScript ist auch Adobes Type 1-Modell völlig geräte- und auflösungsunabhängig. Es verwendet mathematische Ausdrücke, insbesondere Bézier-Kurven, um Zeichenkonturen zu definieren. Damit stellt es Flexibilität und Darstellungsgenauigkeit sicher. Die Zeichen sind in der Größe 1 Punkt in einem 1000 x 1000-Koordinatensystem definiert, das dann nach Belieben skaliert, gedreht und schräg gestellt werden kann. Es können Hinweise hinzugefügt werden, um die Darstellung auf einer Vielzahl von Geräten und in verschiedenen Pixeldichten so genau wie möglich zu gestalten.

Kürzlich entwickelten Adobe und Microsoft gemeinsam OpenType,¹ ein neues plattformübergreifendes Zeichensatzformat. Diese Erweiterung des TrueType-Formats unterstützt auch Daten im Type 1-Zeichensatzformat. Außerdem fügt OpenType weitere typographische Funktionen hinzu.

OpenType-Fontdateien können zwischen verschiedenen Plattformen (Macintosh und Windows) hin- und herkopiert werden. Damit verbessern sie die plattformübergreifende Übertragbarkeit von Dokumenten, die diese Fonts verwenden. Die Bitmap-, Kontur- und metrischen Daten werden in einer einzigen, plattformübergreifenden OpenType-Fontdatei kombiniert; dadurch wird die Schriftenverwaltung vereinfacht.

OpenType-Fonts basieren auf Unicode, einer internationalen, Multi-Byte-Zeichenkodierung, die praktisch alle Sprachen der Welt abdeckt. Damit erleichtert OpenType mehrsprachige Typographie, indem es mehrere Sprachzeichensätze in eine Fontdatei einbindet. Die Basisfonts von OpenType enthalten die Standardbandbreite an lateinischen Zeichen, die in der westlichen Welt verwendet werden, außerdem einige internationale Zeichen (z.B. das Euro-Symbol). Pro-Versionen fügen eine ganze Reihe an akzentuierten Zeichen hinzu, um zentral- und osteuropäische Sprachen, wie Türkisch und Polnisch, zu unterstützen, und viele enthalten zudem im selben Font kyrillische und griechische Zeichen.

Angesichts dessen, dass OpenType-Fonts mehr als 65.000 Zeichen enthalten können, bieten sie sehr viel mehr typographische Möglichkeiten, indem sie Basisfonts, Experten-Fonts und viele zusätzliche Zeichen in einer Datei kombinieren. Eine einzelne Fontdatei kann beispielsweise viele Nicht-

¹Siehe <http://partners.adobe.com/asn/developer/opentype/main.html>.

Standardzeichen enthalten, wie Mediävalglyphen, echte Kapitälchen, Brüche, Zierbuchstaben, hoch- und tiefgestellte Zeichen, Titelbuchstaben, kontextuelle und stilistische Alternativen und eine Vielzahl an Ligaturen.

OpenType regelt die „Übersetzung“ von Zeichen zu Glyphen. Insbesondere können seine Layoutfunktionen verwendet werden, um Glyphen zu positionieren oder zu ersetzen. Für jedes Zeichen gibt es einen Standardglyphen und ein bestimmtes Positionierverhalten. Die Anwendung von Layoutfunktionen auf ein oder mehrere Zeichen kann die Positionierung ändern oder einzelne Glyphen ersetzen.

Im Laufe der Jahre wurden Tausende von Schriften im Type1-Format verfügbar, einschließlich jener der weltweit wichtigsten Satzfirmen, wie Linotype, Agfa-Compugraphic, Monotype, Autologic und Varityper. Erst kürzlich hat Adobe die gesamte Adobe Type-Bibliothek (mit Tausenden von Schriften) nach OpenType konvertiert, und andere Organisationen werden Adobes Beispiel folgen.

In der T_EX-Welt kann derzeit Ω (Omega) OpenType-Fonts nutzen. Omega ist eine Erweiterung von T_EX, die von Yannis Haralambous und John Plaice entwickelt wurde, über Multi-Byte-Datenstrukturen verfügt und für die interne Zeichendarstellung auf Unicode zurückgreift.

10.4.2 Der PostScript-Treiber dvips

Tom Rokickis dvips-Programm¹ ist zweifellos der am weitesten verbreitete dvi-zu-PostScript-Treiber. Es ist ein sehr ausgereiftes Produkt, das zahlreiche wichtige und nützliche Funktionen hat. dvips bietet eine umfangreiche Unterstützung für den `\special`-Befehl; insbesondere unterstützt es die `pic`-Befehle des `eepic`-Paketes, die in Abschnitt 10.1.5 erörtert wurden.

Das dvips-Programm erzeugt automatisch fehlende Fonts, wenn `META-FONT` auf dem System existiert. Wenn ein Font nicht generiert werden kann, wird stattdessen eine skalierte Version desselben Fonts in einer anderen Größe verwendet (wobei dvips auf eine zu erwartende Verschlechterung des Erscheinungsbildes hinweisen wird). Die Funktion zur Erzeugung fehlender Fonts ist konfigurierbar und damit nicht nur auf die Ausführung von `META-FONT` begrenzt.

Die Ausgabe von dvips kann auf zwei Arten gesteuert werden: über Kommandozeilenoptionen für einen bestimmten Job und über Befehle in einer oder mehreren Konfigurationsdateien. Mithilfe solcher Konfigurationsdateien können Parameter einerseits global für das gesamte System eingestellt werden, andererseits auch spezifisch für einzelne Drucker bzw. individuell für einzelne Benutzer.

Beim Start von dvips wird nach einer globalen `config.ps`-Datei gesucht.²

Der dvips-Treiber hat eine Fülle von Kommandozeilenoptionen. Tabelle 10.3 auf der nächsten Seite gibt einen Überblick über diese Optionen.

¹Das Handbuch ist einsehbar unter http://www.ctan.org/tex-archive/dviware/dvips/dvips_man.pdf. Für eine ausführliche Beschreibung siehe auch [58, Kapitel 11].

²Diese Datei muss im Suchpfad von dvips vorhanden sein, der normalerweise `texmf/dvips/config` oder ähnlich lautet und sich unterhalb der `root`-Ebene des T_EX-Installationsverzeichnis befindet.

a*	Speicherplatz sparen, nicht Zeit	y #	Mit dvi-Vergrößerung multiplizieren
b #	Kopien pro Seite, z.B. für Poster	z*	Hyper-PostScript
c #	Kopien pro Seite (schnell)	A	nur ungerade Seiten ausdrucken
d #	Fehlersuche	B	nur gerade Seiten ausdrucken
e #	Maxdrift-Wert	C #	Anzahl sortierter Kopien
f*	Als Filter fungieren	D #	Auflösung
h f	Headerdatei hinzufügen	E*	versuchen, EPSF zu erstellen
i*	getrennte Dateien pro Abschnitt	F*	als letztes Zeichen Control-D schreiben
k*	Beschnittmarken drucken	G*	kleine Slots auf höhere Pos. verschieben
l #	letzte Seite	K*	Kommentare aus Headerdateien entfernen
m*	manuelle Papierzuführung	M*	keine Zeichensätze erstellen
n #	maximale Seitenanzahl	N*	keine strukturierten Kommentare
o f	Ausgabedatei	O c	Koordinatensprung setzen/ändern
p #	erste Seite (p=# absolut)	P s	config.\$s laden
pp#	einzelne Seite	R	ignorierte Systembefehle
ppn ₁ : n ₂	Seitenbereich	S #	maximale Abschnittsgröße in Seiten
q*	Bildschirm Ausgaben unterdrücken	T c	gewünschte Seitengröße angeben
r*	Reihenfolge der Seiten umkehren	U*	Bugfix für Xerox 4045-Drucker
s*	Ausgabe in Save/Restore einbetten	X #	horizontale Auflösung
t s	Papierformat	Y #	vertikale Auflösung
x #	dvi-Vergrößerung überschreiben	Z*	Bitmap-Fonts komprimieren
# = Nummer f = Dateiname s = Zeichenkette * = Suffix, '0' um auszuschalten c = durch Komma getrenntes Dimensionspaar (z.B. 3.2in, -32.1cm)			

Tabelle 10.3: Die wichtigsten Optionen des dvips-Programms

Mithilfe der Option `-d` des dvips-Programms kann man Fehler finden und nachvollziehen, was falsch gelaufen ist. Dazu muss eine ganze Zahl angegeben werden, welche die Klasse der anzuzeigenden Daten definiert. Um Informationen über mehrere Klassen zu erhalten, müssen lediglich die Nummern für die gewünschten Typen addiert werden. Es stehen folgende Typen zur Verfügung:

1	\specials	4	Fonts	16	Überschriften	64	Dateien
2	Pfade	8	Seiten	32	Fontkomprimierung	128	Speicher

Der Aufruf von dvips mit der Option `-d 4` liefert beispielsweise die Information, welche Fonts aufgerufen wurden und woher diese geladen wurden. Die Option `-d -1` (alle Marker sind gesetzt) zeigt ein sehr ausführliches Protokoll von allem, was dvips tut. Es produziert allerdings eine enorme Menge an Daten, so dass diese Funktion nur als letzte Maßnahme eingesetzt werden sollte, wenn präzisere Versuche fehlschlagen.

10.4.3 pspicture – Erweiterte picture-Umgebung für dvips

David Carlisles pspicture-Paket reimplementiert und erweitert L^AT_EXs picture-Umgebung mithilfe von PostScript-Befehlen, die in T_EXs \special-Befehle eingesetzt werden. Es hebt damit Beschränkungen von Standard-L^AT_EX auf, welches nur eine begrenzte Anzahl an Steigungen und Dicken für Linien und eine begrenzte Bandbreite von Durchmessern für Kreise anbietet.

Es gibt einige Überschneidungen von diesem Paket mit dem bereits beschriebenen eepic-Paket. Genaugenommen kann das pspicture-Paket als eine Art „Vertretung“ für das pict2e-Paket betrachtet werden, das von Leslie Lamport 1994 in der zweiten Ausgabe des *L^AT_EX Manuals* angekündigt, aber niemals geschrieben wurde.¹

pspicture hat jedoch den Nachteil, dass ein Bild nicht mehr mit einem dvi-Programm betrachtet werden kann, das keine Funktion zur Interpretation und Darstellung von PostScript-Befehlen hat.² Eine „arme Leute“-Lösung für diesen Fall ist, das Paket texpicture einzusetzen. Es verwendet, soweit wie möglich, die picture-Standardbefehle, aber lässt alle Bildobjekte weg, die nicht mit Standard-L^AT_EX gezeichnet werden können – ohne eine Warnung auszugeben. Natürlich wird das optische Ergebnis damit wahrscheinlich nicht der endgültigen Version entsprechen, aber zumindest wird das Dokument übersetzbar.

In der mit pspicture erzeugten dvi-Datei sind \special-Befehle eingebettet, die von Rokickis dvips-Treiber erkannt werden. Die Treiberdatei pspicture.ps enthält den PostScript-Code, auf den die \special-Befehle referenzieren. Dadurch kann ihn der PostScript-Interpreter lesen. Die Datei muss in der T_EX-Installation im entsprechenden dvips-Verzeichnis vorhanden sein, damit sie dvips bei Bedarf finden und einbinden kann.

Erweiterte oder geänderte Befehle

Das pspicture-Paket erweitert die Funktionalität verschiedener Befehle, die innerhalb der picture-Umgebung von L^AT_EX verfügbar sind.

Die Befehle \circle und \circle* ähneln ihren Gegenstücken in Standard-L^AT_EX, haben aber keine Beschränkung, was die Durchmesser betrifft. Die Dicke des Kreises wird mit dem Befehl \linethickness geändert. Die Größe des Kreises, der von \circle* erzeugt wird, ist unabhängig von \linethickness.

[\circle-
Erweiterungen](#)

`\oval [radius] (x,y) [teil]`

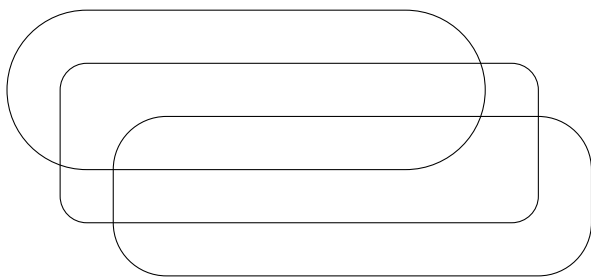
Der Befehl \oval funktioniert wie im L^AT_EX-Buch beschrieben; es gibt jedoch keinen maximalen Durchmesser für Kreisbögen, so dass das Oval (bei fehlendem optionalen Argument [teil]) stets aus zwei Halbkreisbögen besteht, die über zwei parallele Linien verbunden sind. Um ein „Rechteck mit abgerundeten Ecken“ zu erhalten, wurde zu Beginn des \oval-Befehls ein zweites optionales Argument radius hinzugefügt. Bei Verwendung dieser Option arbeitet

[\oval-
Erweiterungen](#)

¹Seit 2004 gibt es eine erste Implementation des pict2e-Paketes durch Hubert Gäßlein und Rolf Niepraschk.

²Wenn pdftex verwendet wird, um PDF direkt zu generieren, ergibt sich dasselbe Problem. In diesem Fall sollte pspicture nicht verwendet werden.

`\oval` mit Kreisbögen, die einen Radius von $\min(radius, x/2, y/2)$ haben. Das folgende Beispiel demonstriert den Unterschied.



```
\usepackage{pspicture}
\begin{picture}(200,120)
\put(90,40){\oval(180,60)}
\put(110,20){\oval[10](180,60)}
\put(130,0){\oval[20](180,60)}
\end{picture}
```

Bsp.
10-4-1

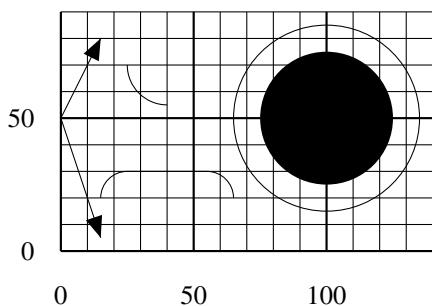
`\line-` und `\vector-`
Erweiterungen

Die Befehle `\vector` und `\line` werden im $\text{\LaTeX}$ -Buch beschrieben, haben aber keine Beschränkungen mehr in Bezug auf die Steigung. Die Dicke einer geneigten Linie wird mit dem Befehl `\linethickness` geändert. Die Pfeilspitzen, die vom Vektorbefehl erzeugt werden, haben eine dreieckige Form und sind standardmäßig größer als die von $\text{\LaTeX}$ vorgegebenen Werte. Die Größe kann mit dem unten beschriebenen Befehl `\arrowlength` gesteuert werden.

Die Befehle `\thinlines`, `\thicklines` und `\linethickness` ändern die Stärke *aller* Linien, einschließlich schräger Linien und Kreisbögen.

Alle anderen Befehle von $\text{\LaTeX}$ s `picture`-Umgebung, wie `\dashbox`, `\framebox`, `\makebox`, `\multiput`, `\put` und `\shortstack`, bleiben unverändert und verhalten sich wie im $\text{\LaTeX}$ -Buch beschrieben.

Das nächste Beispiel zeigt, wie das `pspicture`-Paket PostScript verwendet, um die `picture`-Umgebung von $\text{\LaTeX}$ zu erweitern. Um zu verdeutlichen, was passiert, wird im Beispiel zusätzlich `graphpaps` Befehl `\graphpaper` eingesetzt. Dieser zeichnet ein Koordinatengitter an der angegebenen Position mit den angegebenen Maßen (siehe erste Zeile in der `picture`-Umgebung). So sieht das Ergebnis von `pspicture` aus:

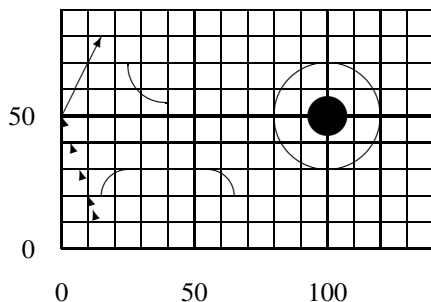


```
\usepackage{pspicture}\usepackage{graphpaps}
\begin{picture}(140,90)
\graphpaper(0,0)(140,90)
\put(0,50){\vector(1,2){15}}
\put(0,50){\vector(2,-6){15}}
\put(40,20){\oval(50,20)[t]}
\put(40,70){\oval(30,30)[b1]}
\put(100,50){\circle{70}}
\put(100,50){\circle*{50}}
\end{picture}
```

Bsp.
10-4-2

Die Auswirkungen dieser Erweiterungen sieht man deutlich, wenn man vergleicht, wie der obige Code mit der Standardumgebung `picture` von $\text{\LaTeX}$ dargestellt wird. Leider lassen sich diese Befehle nicht mit der `picture`-Umgebung kompilieren, da nicht unterstützte Argumente für die Befehle `\vector`, `\circle` und `\circle*` verwendet wurden. Daher muss anstelle von `pspicture` das Paket `texpicture` angegeben werden, wie das Beispiel unten

zeigt. Dank des darüber gelegten Koordinatengitters sind die Beschränkungen in Hinsicht auf den Einsatz von `pspicture` sehr gut sichtbar. Tatsächlich wird der zweite `\vector` nicht korrekt dargestellt, während die Durchmesser der beiden Kreise nicht mehr dem entsprechen, was gefordert wurde.



```
\usepackage{texpicture}\usepackage{graphpap}
\begin{picture}(140,90)
\graphpaper(0,0)(140,90)
\put(0,50){\vector(1,2){15}}
\put(0,50){\vector(2,-6){15}}
\put(40,20){\oval(50,20)[t]}
\put(40,70){\oval(30,30)[bl]}
\put(100,50){\circle{70}}
\put(100,50){\circle*{50}}
\end{picture}
```

Bsp.
10-4-3

Neue Befehle

Das `pspicture`-Paket führt ebenfalls eine Reihe neuer Befehle ein. Die Befehle `\Line` und `\Vector` vereinfachen das Zeichnen einer Linie, indem sie die Angabe „relativer Koordinaten“ ermöglichen.

<code>\put(x₁,y₁){\Line(x₂,y₂)}</code>	<code>\put(x₁,y₁){\Vector(x₂,y₂)}</code>

Die obige Syntax ergibt eine Linie (oder einen Vektor) zwischen den Punkten (x_1, y_1) und $(x_1 + x_2, y_1 + y_2)$.

<code>\put(x₁,y₁){\Curve(x₂,y₂){m}}</code>

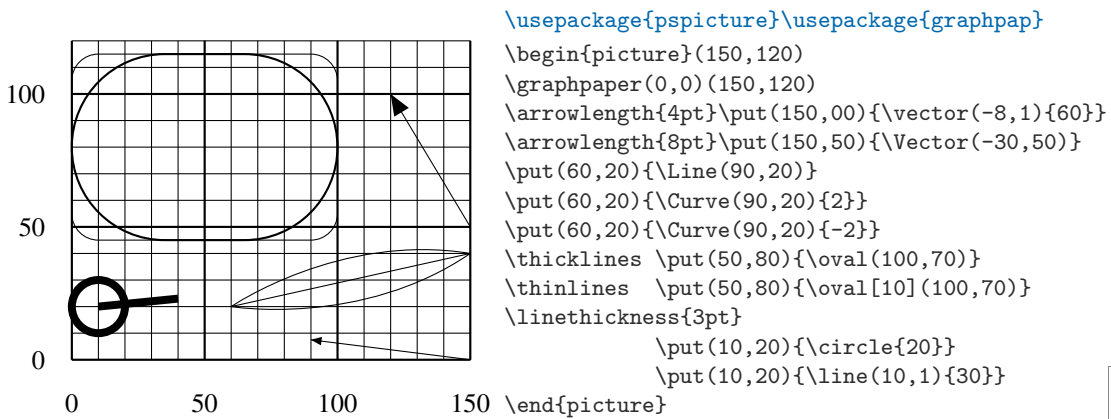
Der Befehl `\Curve` wirkt ähnlich wie `\Line`, erzeugt aber eine Linie, deren Kurvenverlauf mithilfe von m gesteuert wird (zunächst sollte man es mit 1 oder -1 versuchen). Der Wert von m muss nicht ganzzahlig sein. Negative Zahlen bewirken eine Krümmung in der entgegengesetzten Richtung wie positive Zahlen.

<code>\arrowlength{größe}</code>

Der Befehl `\arrowlength` gibt die Größe der dreieckigen Pfeilspitze an, die von den Befehlen `\vector` und `\Vector` erzeugt wird. Wie `\linethickness` ist er ein absoluter Wert (d.h. er wird nicht von `\unitlength` beeinflusst), der in einer $\text{\LaTeX}$ -Einheit angegeben wird.

Einige der zusätzlichen Funktionen, die nicht in der `picture`-Umgebung von Standard- $\text{\LaTeX}$ verfügbar sind, werden im nächsten Beispiel gezeigt. Auf die Möglichkeiten, mit `\line` und `\vector` beliebige Steigungen zu erzeugen, wurde bereits eingegangen. Es empfiehlt sich, die benutzerfreundlichere Schnittstelle der Befehle `\Vector`, `\Line` und `\Curve` zu verwenden (welche die Eingabe relativer Koordinaten ermöglicht). Der erste `\oval`-Befehl zeichnet eine normale Ellipse mit einer dicken Linie (mithilfe des Befehls `\thicklines`), während der zweite `\oval`-Befehl ein Rechteck mit abgerundeten Ecken und dünnen Rändern zeichnet (mithilfe des Befehls `\thinlines`).

Zum Schluss wird die Linienstärke mit dem Befehl `\linethickness` auf 3pt gesetzt. Man beachte die Auswirkung dieses Befehls auf die Kreise und Linien.



Bsp.
10-4-4

10.4.4 Das Portable Document Format

Adobes Portable Document Format (PDF) [5] ist ein direkter Abkömmling der PostScript-Sprache. Während PostScript eine reine Programmiersprache darstellt, ist PDF eine leichtere Graphiksprache der zweiten Generation, die für schnelleres Herunterladen und Anzeigen optimiert wurde. Die meisten Vorteile von PostScript bleiben erhalten: PDF sorgt für eine genaue Wiedergabe der Seite bis hin zum kleinsten Zeichen oder Leerraum und ist zusätzlich auf verschiedene Computerplattformen übertragbar. Aus diesen Gründen wird PDF immer häufiger im professionellen Druckbereich als Ersatz für PostScript eingesetzt. Außerdem sind dank Plug-and-Play-Technologie alle heutigen Browser in der Lage, neben HTML auch PDF-Dateien zu integrieren und diese anzuzeigen.

Die Hauptunterschiede zwischen PostScript und PDF sind:

- Es gibt keine eingebauten Programmiersprachen-Funktionen: PDF kann beispielsweise keine Werte berechnen.
- PDF garantiert vollständige Unabhängigkeit der Seiten, indem es die Ressourcen klar von den Seitenobjekten trennt.
- PDF-Dateien sind kompakt und können komplett durchsucht werden.
- Interaktive Hyperlinks erleichtern die Navigation durch die PDF-Datei.
- Mithilfe der PDF-Sicherheitsfunktionen können PDF-Dokumenten besondere Zugriffsrechte und digitale Signaturen zugewiesen werden.
- Fontkonturen müssen nicht in die Datei eingebettet werden, da PDF-Dateien genügend Fontinformationen besitzen, so dass PDF-fähige Anwendungen (z.B. Adobe Reader) das Erscheinungsbild eines Fonts imitieren können.
- PDF hat verbesserte Komprimierungsfunktionen, um die Größe der PDF-Dateien klein zu halten. Darüber hinaus können .png- und .jpeg-Graphiken direkt eingefügt werden.

- PDF 1.4 und höhere Versionen unterstützen ein transparentes Seitenbeschreibungsmo-
dell (PostScript verwendet ein nicht transparentes Modell) und bietet Multimedia-Unterstützung.
- PDF 1.4 und höhere Versionen verwenden Tagged PDF, eine Form von PDF, die Informationen über den Inhalt und die Struktur enthält. Tagged PDF ermöglicht, Daten einer Seite (Text, Graphiken, Bilder) zu extrahieren und wiederzuverwenden. Zum Beispiel kann Text mit Tagged PDF neu umbrochen werden, um auf tragbaren Geräten, wie Palm OS- oder Pocket PC-Systemen, angezeigt zu werden.
- PDF 1.5, das Ende 2003 herauskam, enthält neue Funktionen zur besseren Unterstützung von Multimedia-Anwendungen.

PDF kann auf vielen verschiedenen Computerplattformen angezeigt und ausgedruckt werden. Dazu muss lediglich der Adobe Reader¹ von Adobe heruntergeladen und installiert werden. Es gibt jedoch auch weitere PDF-Viewer. Die bekanntesten frei erhältlichen sind ghostscript², das PDF auch aus PostScript erzeugen kann, und Xpdf.³

PDF direkt mit T_EX erzeugen

Eine PostScript-Datei, die mit L^AT_EX erzeugt wurde, kann über ein „Distiller“-Programm nach PDF konvertiert werden. Das bekannteste und ausgereifteste unter ihnen ist der Acrobat Distiller von Adobe. Daneben leisten aber auch die Programme ghostscript und ImageMagicks convert (das auf letzterem aufbaut) gute Dienste.

Um PDF direkt, d.h. ohne den Zwischenschritt dvi, zu erzeugen, wurden pdfT_EX (siehe unten) und MicroPress VTeX⁴ entwickelt, die beide über einen eigenen, direkt PDF-generierenden T_EX-Mechanismus verfügen. Wenn es bereits eine dvi-Datei gibt, kann auch Mark Wicks' dvi-Treiber dvipdfm verwendet werden.⁵

Hàn Thê Thànhs pdfT_EX ist eine Erweiterung von T_EX, die PDF direkt aus T_EX-Quelldateien erzeugt [159]. Sie erweitert die Satzfunktionen von T_EX in einigen interessanten Bereichen [160, 161]. Seit 2002 ist pdfT_EX Teil der T_EX-Standarddistributionen.

Mit pdfT_EX können Kommentare, Hyperlinks und Lesezeichen in der erzeugten PDF-Ausgabedatei eingefügt werden. Das Programm kann TrueType-Fonts verarbeiten und unterstützt die Einbindung von Bildern im .png- und .jpeg-Format. Das üblichste Verfahren, d.h. die Einbindung von Encapsulated PostScript-Graphiken, wurde in diesem Programm durch die Einbindung von PDF-Dateien ersetzt. EPS-Dateien können mit verschiedenen Konvertern in PDF umgewandelt werden, z.B. mit ImageMagicks Hilfsprogramm convert oder eps2pdf (beide rufen intern ghostscript auf), dem Acrobat Distiller oder anderen PostScript-zu-PDF-Konvertern.

¹Kostenlos herunterzuladen unter <http://www.adobe.com/products/acrobat/readermain.html>.

²Siehe <http://www.cs.wisc.edu/~ghost/>.

³Siehe <http://www.foolabs.com/xpdf/home.html>.

⁴Siehe <http://www.micropress-inc.com/>.

⁵Siehe <http://gaspra.kettering.edu/dvipdfm/>.

Navigation spielt in PDF-Dokumenten eine große Rolle. Das `hyperref`-Paket (siehe Abschnitt 2.4.5 oder [59, Kapitel 2]), das von Sebastian Rahtz und Heiko Oberdiek entwickelt wurde, erweitert die Funktionalität der Querverweisbefehle von $\text{\LaTeX}$ (einschließlich Inhaltsverzeichnis, Bibliographie usw.), um `\special`-Befehle zu erzeugen, die ein `dvi`-Treiber oder `pdf\TeX` in Hypertext-Links umwandeln kann. Außerdem bietet das `hyperref`-Paket neue Befehle, mit denen der Anwender ad hoc Hypertext-Links erstellen kann, auch auf externe Dokumente und URLs.

Da PDF keine Programmiersprachenbefehle besitzt, kann es allgemeine reine PostScript-Befehle, wie etwa jene, die vom `pstricks`-Paket [58, Kapitel 4] verwendet werden, nicht verarbeiten. Daher werden diese Befehle nicht unterstützt.¹

$\text{\LaTeX}$ s Standardpakete `graphics` und `color` verfügen über die Option `pdftex`, mit deren Hilfe normale Befehle zur Farbgebung, Textgestaltung, Rotation und Einbindung von Graphiken verwendet werden können. Die Art der Graphikeinbindung stellt sicher, dass Graphiken stets nur einmal eingebunden werden, ungeachtet dessen, wie oft die Graphik verwendet wird (selbst wenn sie in verschiedenen Skalierungen verwendet oder auf unterschiedliche Art transformiert wird).

Richtiges PostScript oder PDF erzeugen

Die richtige PostScript- oder PDF-Ausgabe aus dem $\text{\LaTeX}$ -System zu erhalten, kann sich manchmal recht schwierig gestalten. Michael Shell hat im Zusammenhang mit den IEEEtran-Dokumentenklassendateien – wenn auch unabhängig von ihnen – ein „Testfluss“-Diagnoseprogramm entwickelt. Die Testdatei `testflow.tex` wird erst auf dem System des Anwenders kompiliert. Danach wird für die Ausgabe eine PostScript-Version (`testflow.ps`) und eine PDF-Version (`testflow.pdf`) erzeugt und auf dem Ausgabegerät ausgedruckt, damit sie mit Referenzdateien verglichen werden können. Die ursprüngliche Testdatei ist so konzipiert, dass die verschiedenen Komponenten des „Druck-Arbeitsflusses“ von $\text{\LaTeX}$ getestet werden. Der Test dient dazu, hilfreiche Informationen zu liefern, die Anwendern helfen, ihr $\text{\LaTeX}$ -System richtig zu konfigurieren, so dass eine gute PostScript- und PDF-Datei erzeugt wird.²

10.4.5 Skalierbare Vektorgraphiken

Seit Mitte der 90er Jahre haben das World Wide Web und die allgemeine Verfügbarkeit des PCs dazu beigetragen, dass die Erstellung, Pflege und Verbreitung elektronischer Dokumente weltweit günstiger, einfacher und schneller wurde. Viel wichtiger ist noch, dass die Entwicklung der XML-Standards und die allgemeine Verfügbarkeit plattformunabhängiger Skriptsprachen es ermöglicht, sehr große Mengen elektronischer Daten zu speichern, zu handhaben und sie in verschiedene, anpassbare Darstellungsformen umzuwandeln.

Für $\text{\LaTeX}$ -Dokumente gibt es eine Vielzahl von Verfahren, um sie in PDF, XHTML oder XML umzuwandeln, so dass die Daten im Internet bereitgestellt

¹Allgemeine PostScript-Befehle *können* mit `\TeX` von MicroPress verwendet werden, das einen eingebauten PostScript-Interpreter hat.

²Detaillierte Anweisungen und eine ausführliche Erläuterung sind erhältlich unter CTAN: `macros/latex/contrib/IEEEtran/testflow/testflow_doc.txt`.

werden können. Damit kann $\text{\LaTeX}$ auch weiterhin eine wichtige Rolle im integrierten, weltweiten Cyberspace einnehmen, insbesondere im wissenschaftlichen Bereich und auf Gebieten, auf denen hochwertiger Satz unerlässlich ist.

Nach einer kurzen Einführung in die skalierbaren Vektorgraphiken (SVG) geht der folgende Abschnitt darauf ein, wie $\text{\LaTeX}$ -kodierte Daten in das SVG-Format konvertiert werden können (siehe [60] für Einzelheiten).

SVG für portable Graphiken im Web

Seitdem das Web an Popularität und Komplexität zugenommen hat, suchen Anwender und Anbieter nach einer immer besseren, präziseren und skalierbaren graphischen Darstellung, die über die .gif- oder .png-Bilder hinausgehen, die heute üblicherweise auf den Webseiten verwendet werden. Um auf diese Bedürfnisse einzugehen, veröffentlichte das World Wide Web Consortium die SVG-Empfehlung, deren aktuelle Version 1.1 ist.¹

SVG ist eine Open-Standard-Vektorgraphiksprache, die zweidimensionale Graphiken mithilfe der XML-Syntax beschreibt. Mit dieser Sprache können Webseiten erstellt werden, die hochauflösende Computergraphiken enthalten.

Als XML-Instanz besteht SVG aus Unicode-Text. Dieses ist mit den üblichen Vektorgraphik-Funktionen ausgestattet. Sein fundamentaler Baustein ist das *Graphikobjekt*, dessen Modell folgende Teile enthält:

- Graphikpfade, die aus Polylinien, Bézier-Kurven und anderen Elementen bestehen:
 - Einfach oder komplex, geschlossen oder offen
 - (Gradient) gefüllt, (gradient) gestrichelt
 - Kann zum Abschneiden verwendet werden
 - Kann zum Erstellen geometrischer Formen verwendet werden
- Muster und Kennzeichnungen
- Vorlagen und Symbolbibliotheken
- Transformationen:
 - Standardkoordinatensystem: x ist rechts, y ist unten,² die Einheit ist ein Pixel
 - Darstellungsfeld gleicht einen Bereich in Weltkoordinaten mit einem Bereich auf dem Bildschirm ab
 - Transformationen ändern das Koordinatensystem (2×3 Transformationsmatrix für Computer; Translation, Rotationen, Skalierungen, und Scherung für Anwender)
 - Kann verschachtelt werden
- Einbindung von Bitmap- oder Rastergraphiken

¹Die *Scalable Vector Graphics (SVG) 1.1 Specification*, verfügbar unter <http://www.w3.org/TR/SVG11/>, wurde am 14. Januar 2003 veröffentlicht.

²Der Bezugspunkt des Anzeigebereiches befindet sich in der oberen linken Ecke. Bei PostScript, bei dem y von unten nach oben verläuft, ist der Bezugspunkt der Seite die untere linke Ecke.

- Abschneiden, Filterung und Rastereffekte; Alpha-Masken
- Animationen, Skripte und Erweiterungen
- Gruppierungen und Stile
- SVG-Fonts (unabhängig von Fonts, die auf dem System installiert sind)

Die SVG-Webseite des W3C (<http://www.w3.org/Graphics/SVG>) ist eine gute erste Informationsquelle und hat viele Links auf andere Seiten.

Ein $\text{\LaTeX}$ -Dokument in ein SVG-Dokument umwandeln

Wenn man nur ein reines $\text{\LaTeX}$ -Quelldokument hat (d.h. eines, das keine EPS-Dateien enthält und keine Erweiterungen verwendet, die $\text{\TeX}$ s `\special-`Befehle benötigen), kann man die dvi-Datei mit Adrian Frischau's `dvi2svg`-Programm in eine SVG-Datei umwandeln.¹

Über ein kleines UNIX-Skript namens `dvi2svg.sh` wird auf eine Java-Bibliothek zugegriffen. Das Skript wird folgendermaßen verwendet:

```
> dvi2svg.sh
Usage: dvi2svg.sh [options] [DVIFILE]
Options:
  -o [FILENAME] : Specify an output filename prefix. If not
                  set, dvi2svg will take the input filename.
  -d : set the debug mode to on(1)/off(0 default)
```

Ein Beispiel für die Verwendung des `dvi2svg`-Programms ist die Übersetzung von zwei Beispielen aus diesem Kapitel nach SVG. Die $\text{\LaTeX}$ -Datei `svgexa.tex` wird zunächst kompiliert. Danach wird die erzeugte dvi-Datei mit `dvi2svg.sh` verarbeitet, um die SVG-Datei `svgexa1` zu erzeugen. (Wenn die dvi-Datei mehr als eine Seite enthält, werden mehrere Ausgabeseiten generiert.)

```
> dvi2svg.sh svgexa.dvi -o svgexa
DEBUG from converter.DviToSvg => Converting file: svgexa.dvi
DEBUG from converter.DviToSvg => Writing result to: svgexa
DEBUG from converter.DviToSvg => Reader has been created
DEBUG from converter.DviToSvg => Writer has been created
Converting .....FINISHED
> ls -l svgexa*.svg
-rw-rw-r-- 1 goossens 23792 Jun 25 19:44 svgexa1.svg
```

Abbildung 10.3 zeigt die generierte SVG-Datei, so wie sie mit dem `squiggle`-Programm angezeigt wird.² Bei komplexeren $\text{\LaTeX}$ -Dateien (insbesondere jenen, in denen EPS- oder PDF-Daten eingebunden sind) kann man zunächst

¹Siehe <http://www.activemath.org/~adrianf/dvi2svg/>. Das `dvi2svg`-Programm enthält auch SVG-Fontkonturen für Zeichen, die in der dvi-Datei referenziert sind. Für alle Standard Computer Modern und $\text{\LaTeX}$ -Fonts wurden SVG-Fontinstanzen generiert, die mit der `dvi2svg`-Distribution verteilt werden.

²Der SVG-Browser `squiggle` ist Teil der Apache Batik-Distribution (<http://xml.apache.org/batik>). SVG kann auch mit Adobes Browser-Plug-In `svgview` betrachtet werden (<http://www.adobe.com/svg>).

mit dvips eine PostScript-Datei generieren und dann das pstoeedit-Programm von Wolfgang Glunz verwenden (eine Erläuterung zur Funktionsweise dieses Befehls findet man in [60]).

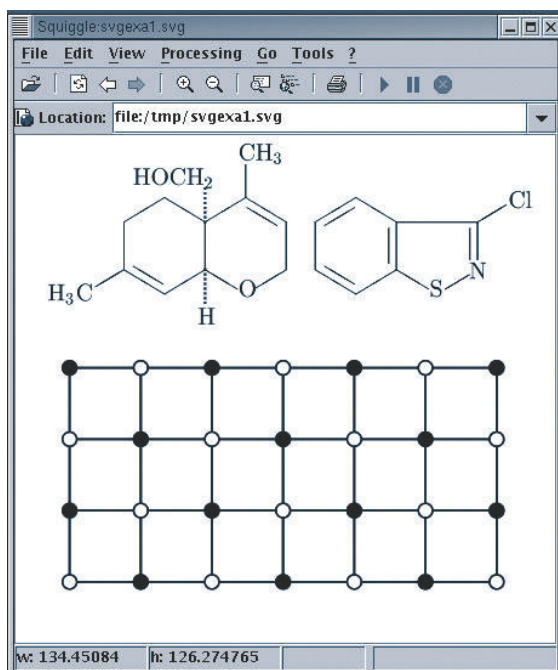


Abbildung 10.3: SVG aus einer dvi-Datei generiert

KAPITEL 11

Indexerstellung

11.1 Syntax der Indexeinträge	669
11.2 <code>makeindex</code>	675
11.3 <code>xindy</code> – Eine Alternative zu <i>MakeIndex</i>	688
11.4 Beeinflussung des Index mit $\LaTeX$ -Funktionen	702

Um Informationen zu einem bestimmten Thema in einem großen Dokument, Buch oder Nachschlagewerk zu erhalten, schlägt man normalerweise das Inhaltsverzeichnis oder, noch häufiger, den Index auf. Aus diesem Grund bildet das Stichwortverzeichnis einen sehr wichtigen Teil eines Dokumentes, nicht zuletzt deshalb, weil die meisten Anwender erst einmal den Index konsultieren, wenn sie Informationen zu einem bestimmten Thema suchen. Man sollte deshalb einen Index planen und ihn zusammen mit dem Haupttext anlegen [38]. Aus Gründen der Einheitlichkeit ist es sinnvoll, mit den nachfolgend beschriebenen Verfahren im Text spezielle Befehle zu verwenden, um ein bestimmtes Schlüsselwort im gesamten Dokument stets in der gleichen Weise im Text und im Index erscheinen zu lassen.

Dieses Kapitel gibt zunächst einen Überblick über die grundlegenden Befehle zur Indexerstellung, die Standard- $\LaTeX$ bietet. Außerdem erläutert es, welche Werkzeuge zur Verfügung stehen, um ein gut durchdachtes Indexregister zu erzeugen. Das *LaTeX-Manual* selbst geht nicht sehr detailliert auf die Syntax der `\index`-Einträge ein. Im *TUGboat* sind jedoch bereits verschiedene Artikel erschienen, die sich mit der Frage der Indexerstellung mit $\TeX$ oder $\LaTeX$ auseinander setzen. [49, 163, 164]. Die Syntax, die in Abschnitt 11.1 beschrieben ist, wird von *MakeIndex* [37, 105] und *xindy* [71, 78, 153] erkannt, den am häufigsten zur Indexerstellung verwendeten Programmen.

Abschnitt 11.2 beschreibt, wie der *MakeIndex*-Prozessor verwendet wird. Die Interpretation der Eingabedatei und das Format der Ausgabedatei werden durch Stilparameter gesteuert. Abschnitt 11.2.4 gibt einen Überblick über diese Parameter und führt mehrere einfache Beispiele an, die zeigen, wie sich die Veränderung der Parameter auf das ausgegebene Ergebnis auswirkt.

Abschnitt 11.3 stellt *xindy* vor, eine Alternative zu *MakeIndex*. Es empfiehlt sich, dieses Programm immer dann einzusetzen, wenn nicht englischsprachige Dokumente erstellt werden oder andere spezielle Anforderungen

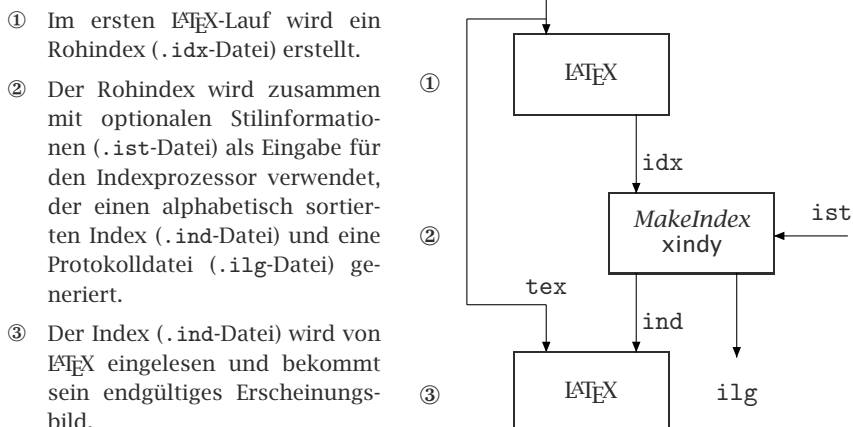


Abbildung 11.1: Flussdiagramm für die Indexerstellung mit den verschiedenen Hilfsdateien, die $\text{\LaTeX}$ und externe Indexprozessoren verwenden

vorliegen, wie etwa die Erstellung von technischen Stichwortverzeichnissen. Das `xindy`-Programm bietet vollkommene Flexibilität in Hinblick auf Vereinheitlichung und Sortierung von Indexeinträgen und ermöglicht eine beliebige Formatierung von Verweisen.

Der letzte Abschnitt geht auf verschiedene $\text{\LaTeX}$ -Pakete ein, die den Index erweitern und die Erstellung von mehreren Stichwortverzeichnissen ermöglichen. Zur Illustration wird dazu ein Beispiel angeführt.

Der Prozess der Indexerstellung wird schematisch in Abbildung 11.1 dargestellt. Es zeigt die einzelnen Schritte für die Erstellung eines Index mit $\text{\LaTeX}$ und `MakeIndex` oder `xindy`.

Abbildung 11.2 auf der gegenüberliegenden Seite zeigt anhand eines Beispiels die verschiedenen Stufen von der Eingabedatei bis hin zum endgültigen Index. Hier erkennt man sehr deutlich, welche Dateien in den Indexerstellungsprozess einbezogen sind. Abbildung 11.2(a) enthält einige Indexbefehle (`\index`), die in der Quelldatei eines Dokumentes vorkommen, wobei links der Befehle die zugehörigen Seitennummern stehen. In Abbildung 11.2(b) ist die von $\text{\LaTeX}$ daraus erstellte Rohindexdatei `.idx` zu sehen. Bei Erstellung mehrerer Stichwortverzeichnisse oder Glossare kann die Dateierweiterung differieren. Nach der Bearbeitung mit dem Indexprozessor wird aus der `.idx`-Datei eine alphabetisch sortierte Indexdatei `.ind` erzeugt, die $\text{\LaTeX}$ -Befehle zur Festlegung des Ausgabeformats enthält (Abbildung 11.2(c)). Das Endergebnis nach der Formatierung mit $\text{\LaTeX}$ ist in Abbildung 11.2(d) dargestellt.

Für die gemeinsame Nutzung von $\text{\LaTeX}$ und `MakeIndex` stehen eine Reihe von Markup-Konventionen zur Verfügung, die dem Benutzer die volle Kontrolle über das Ausgabeformat ermöglichen. Das `xindy`-Programm hat einen `MakeIndex`-Kompatibilitätsmodus, der dasselbe Format unterstützt. In Abschnitt 11.1, der die Syntax und Semantik des `\index`-Befehls beschreibt, werden stets die Standardeinstellungen verwendet.

Seite vi:	<code>\index{Tier}</code>	<code>\indexentry{Tier}{vi}</code>
Seite 5:	<code>\index{Tier}</code>	<code>\indexentry{Tier}{5}</code>
Seite 6:	<code>\index{Tier}</code>	<code>\indexentry{Tier}{6}</code>
Seite 7:	<code>\index{Tier}</code>	<code>\indexentry{Tier}{7}</code>
Seite 11:	<code>\index{Tierarten see{Tier}}</code>	<code>\indexentry{Tierarten see{Tier}}{11}</code>
Seite 17:	<code>\index{Tier@\emph{Tier}}</code> <code>\index{Haustier textbf}</code>	<code>\indexentry{Tier@\emph{Tier}}{17}</code> <code>\indexentry{Haustier textbf}{17}</code>
Seite 26:	<code>\index{Tier!Haustier!Katze}</code>	<code>\indexentry{Tier!Haustier!Katze}{26}</code>
Seite 32:	<code>\index{Tier!Insekt}</code>	<code>\indexentry{Tier!Insekt}{32}</code>
	(a) Die Eingabedatei	(b) Die .idx-Datei

<code>\begin{theindex}</code>	
<code>\item Haustier, \textbf{17}</code>	Haustier, 17
<code>\item Tier, vi, 5-7</code>	
<code>\subitem Haustier</code>	Tier, vi, 5-7
<code>\subsubitem Katze, 26</code>	Haustier
<code>\subitem Insekt, 32</code>	Katze, 26
<code>\item \emph{Tier}, 17</code>	Insekt, 32
<code>\item Tierarten, \see{Tier}{11}</code>	Tier, 17
<code>\indexspace</code>	Tierarten, <i>siehe</i> Tier
<code>\end{theindex}</code>	
(c) Die .ind-Datei	(d) Der formatierte Index

Abbildung 11.2: Schrittweise Entwicklung der Indexerstellung

11.1 Syntax der Indexeinträge

Dieser Abschnitt befasst sich mit der Syntax, die standardmäßig zur Erstellung der Indexeinträge mit $\text{\LaTeX}$ und *MakeIndex* oder *xindy* verwendet wird. Nach und nach werden Register mit immer komplexeren Elementen vorgestellt, wobei für jeden Fall sowohl die Eingabedatei als auch das aus ihr erstellte gesetzte Ergebnis zu sehen sind.

Die Abbildungen 11.3 bis 11.4 auf den Seiten 676–677 zeigen die Eingabe und die erzeugte Ausgabe eines kleinen $\text{\LaTeX}$ -Dokumentes, in denen verschiedene einfache Verwendungsmöglichkeiten des `\index`-Befehls zu sehen sind. Außerdem wird demonstriert, wie sich die Einbindung des `showidx`-Paketes auswirkt (siehe Abschnitt 11.4.2). Um einheitliche Indexeinträge zu erzeugen (siehe Abschnitt 11.1.7), wurden die Befehle `\Com` und `\Prog` definiert und eingesetzt. Die indexerstellende Umgebung `theindex` wurde neu definiert, so dass der Index auf eine Seite passt (Abschnitt 11.4.1 erläutert, wie dieses Ergebnis erzielt werden kann).

Nachdem die notwendigen `\index`-Befehle im Dokument eingeführt worden sind, soll der Index erzeugt werden, damit er im darauf folgenden Lauf wieder in das $\text{\LaTeX}$ -Dokument eingefügt werden kann. Wenn die Hauptdatei des Dokumentes z.B. `main.tex` heißt, sollten folgende Änderungen in dieser Datei vorgenommen werden:

Rohindex erstellen

- das `makeidx`-Paket mit einem `\usepackage`-Befehl einbetten,
- einen `\makeindex`-Befehl in der Dokumentenpräambel einfügen,

- einen `\printindex`-Befehl dort einfügen, wo der Index erscheinen soll – normalerweise am Ende, direkt vor dem Befehl `\end{document}`.

Danach kann das gesamte Dokument erneut mit $\text{\LaTeX}$ bearbeitet werden, wodurch die Datei `main.idx` erzeugt wird, die im Folgenden als `.idx`-Datei bezeichnet wird.

11.1.1 Einfache Indexeinträge

Jeder `\index`-Befehl bewirkt, dass $\text{\LaTeX}$ einen Eintrag in die `.idx`-Datei schreibt. Das folgende Beispiel zeigt einige einfache `\index`-Befehle, zusammen mit den Indexeinträgen, die sie generieren. Die Seitenzahl bezieht sich auf die Seite, die den Text enthält, in dem der `\index`-Befehl auftaucht. Wie in untenstehendem Beispiel gezeigt, führen mehrere gleiche Befehle auf derselben Seite (wie etwa `\index{stylistic}` auf Seite 23) nur zu einem Eintrag „23“ im Index.

style, 14	Seite iii: <code>\index{style}</code>
style , 16	Seite xi: <code>\index{Stylist}</code>
style, iii, 12	Seite 12: <code>\index{style}</code>
style , 15	<code>\index{styles}</code>
style file, 34	Seite 14: <code>\index{ style}</code>
styles, 12	Seite 15: <code>\index{style }</code>
Stylist, xi	Seite 16: <code>\index{ style }</code>
stylist, 34	Seite 23: <code>\index{stylistic}</code>
stylistic, 23	<code>\index{stylistic}</code>
	Seite 34: <code>\index{style file}</code>
	<code>\index{stylist}</code>

Leerzeichen
können schaden 

Ein besonderes Augenmerk richtet sich auf die Art und Weise, wie Leerzeichen in diesem Beispiel behandelt werden. Innerhalb der `\index`-Befehle werden Leerzeichen wortwörtlich in die `.idx`-Datei übernommen und von *MakeIndex* standardmäßig als normale Zeichen betrachtet, die den Buchstaben vorangestellt werden. Man achte im obigen Beispiel auf die `style`-Einträge auf den Seiten 14 und 16. Die Einträge mit führenden Leerzeichen werden an den Anfang des Index platziert, und zwar auf zwei verschiedene Zeilen, da das nachgestellte Leerzeichen auf Seite 16 die Zeichenkette um ein Zeichen verlängert. Durch diese Schreibweise entstehen vier verschiedene Einträge für denselben Begriff, ein Effekt, der wahrscheinlich nicht beabsichtigt war. Daher ist es wichtig, diese unerwünschten Leerzeichen aus den `\index`-Befehlen zu beseitigen, wenn *MakeIndex* verwendet wird. Als Alternative kann man beim Aufruf des Indexprozessors die Option `-c` angeben. Diese Option unterdrückt die Wirkung von führenden und angehängten Leerzeichen (siehe Abschnitte 11.2.2 und 11.3.1). Ein weiterer, zumindest in Sprachen wie Englisch häufig auftretender Fehler ist die uneinheitliche Schreibweise eines Wortes, etwa wechselweise mit kleinem und großem Anfangsbuchstaben (wie bei *Stylist* auf Seite xi), die zu zwei verschiedenen Indexeinträgen führt. In Sprachen wie Deutsch, in der „Arm“ (Körperteil) und „arm“ (mittellos) tatsächlich zwei verschiedene Einträge sind, ist dies durchaus beabsichtigt. Im Englischen sollten diese unerwünschten Doppeleinträge normalerweise beseitigt werden.

Bei Verwendung von `xindy` werden Leerzeichen automatisch komprimiert. Darüber hinaus unterstützt `xindy` die internationale Indexierung und behandelt damit Groß- und Kleinschreibung korrekt und automatisch auf länderspezifische Weise. Daher werden mit `xindy` die oben erwähnten Probleme nicht auftreten.

11.1.2 Erstellen von Untereinträgen

Für Indexeinträge sind bis zu drei Ebenen möglich (Haupt-, Unter- und Untereinträge zweiter Ordnung). Um solche Einträge zu erstellen, gibt man im Argument des `\index`-Befehls sowohl Haupt- als auch Untereinträge durch ein `!`-Zeichen getrennt an. Dieses Zeichen kann in der *MakeIndex*-Stildatei umdefiniert werden (siehe Tabelle 11.1 auf Seite 680).

Box, 21	Seite 3:	<code>\index{Dimension!Linie!Breite}</code>
Dimension der, 33	Seite 5:	<code>\index{Box!Parameter}</code>
Parameter, 5	Seite 9:	<code>\index{Dimension!Tabelle}</code>
Dimension	Seite 12:	<code>\index{Dimension!Linie!Tiefe}</code>
Abbildung, 17	Seite 17:	<code>\index{Dimension!Abbildung}</code>
Linie	Seite 21:	<code>\index{Box}</code>
Breite, 3	Seite 33:	<code>\index{Box!Dimension der}</code>
Tiefe, 12		
Tabelle, 9		

11.1.3 Seitenbereiche und Querverweise

Einen Seitenbereich für einen Indexeintrag kann man definieren, indem man den Befehl `\index{...|{}` an den Anfang des Bereiches und den Befehl `\index{...|})` an das Ende des Bereiches platziert. Seitenbereiche sollten innerhalb einer homogenen Seitennummerierung liegen (d.h. römische und arabische Seitenzahlen sollten nicht in denselben Bereich fallen). Wenn beide Enden eines Bereiches auf dieselbe Seite fallen, generieren *MakeIndex* und `xindy` übrigens nur eine Seitenzahl. Einzelne Einträge innerhalb eines Bereiches erzeugen keinen zusätzlichen Indexeintrag.

Auch für den Index können Querverweise erstellt werden. Hierfür wird die Kapselungsfunktion `see` verwendet. Da der „see“-Eintrag keine Seitennummern im Index erzeugt, kann der Befehl `\index{...|see{...}}` an einer beliebigen Stelle der Eingabedatei *nach* dem `\begin{document}`-Befehl platziert werden. Aus praktischen Erwägungen bietet es sich an, solche Querverweisbefehle an einer Stelle zusammenzufassen. Der erzeugte Text (voreingestellt *see*) ist in `\seename` gespeichert; verwendet man das `babel`-Paket wird dieser automatisch angepasst.

Fonts	Seite ii:	<code>\index{Tabelle {}</code>
Computer Modern, 13–25	Seite xi:	<code>\index{Tabelle })</code>
PostScript, 5	Seite 5:	<code>\index{Fonts!PostScript {}</code>
Text, <i>siehe</i> Textfonts		<code>\index{Fonts!PostScript })</code>
Tabelle, ii–xi, 14	Seite 13:	<code>\index{Fonts!Computer Modern {}</code>
	Seite 14:	<code>\index{Tabelle}</code>
	Seite 17:	<code>\index{Fonts!Text see{Textfonts}}</code>
	Seite 21:	<code>\index{Fonts!Computer Modern}</code>
	Seite 25:	<code>\index{Fonts!Computer Modern })</code>

11.1.4 Steuern der Darstellungsform

Es kann vorkommen, dass ein Eintrag nach einem bestimmten Schlüsselwort sortiert werden soll, im Ausdruck aber eine andere visuelle Repräsentation gewählt werden soll, wie z.B. bei griechischen Buchstaben, mathematischen Zeichen oder besonderen typographischen Zeichen. Dies wird durch die Syntax *schlüssel@darstellung* ermöglicht. Der *schlüssel* legt dabei die Position im Index fest, während die Zeichenkette *darstellung* den formatierten Text des Eintrags generiert.

Delta, 14	Seite 5:	<code>\index{neunundneunzig}</code>
δ , 23	Seite 14:	<code>\index{Delta}</code>
Delta-Flügler, 16	Seite 16:	<code>\index{Delta-Fluegler@Delta-Flügler}</code>
Flora , 19	Seite 19:	<code>\index{Flora@\textbf{Flora}}</code>
neunzig, 26	Seite 23:	<code>\index{Delta@\$\delta\$}</code>
xc, 28		<code>\index{tabular@\texttt{tabular}-Umgebung}</code>
neunundneunzig, 5	Seite 26:	<code>\index{neunzig}</code>
tabular-Umgebung, 23	Seite 28:	<code>\index{neunzig@xc}</code>

Verwendet man die Voreinstellungen von *MakeIndex*, werden Wörter mit Umlauten unter Umständen falsch einsortiert, weshalb im obigen Beispiel ein expliziter *schlüssel* angegeben wurde. Später werden noch einfachere Eingabemöglichkeiten diskutiert.

Bei einigen Stichwortverzeichnissen sollen bestimmte Seitenzahlen in besonderer Weise formatiert werden, so dass z.B. eine kursive Seitenzahl angibt, dass sich dieser Indexeintrag auf die wichtigste Textstelle zu diesem Begriff bezieht. Ein *n* kann nach einer Seitenzahl beispielsweise angeben, dass der Begriff auf den angegebenen Seiten in einer Fußnote steht. Mit *MakeIndex* kann eine einzelne Seitenzahl nach Belieben individuell formatiert werden, indem man die Syntax der Kapselungsfunktion ausnutzt, die durch das |-Zeichen gekennzeichnet ist. Alles, was nach dem |-Zeichen angegeben wird, wandelt *MakeIndex* in einen Befehl um, der als Argument die Seitenzahl bekommt, die zu dem Indexeintrag gehört. Der Befehl `\index{schlüssel|xxx}` erzeugt z.B. eine Seitenzahl in der Form `\xxx{n}`, wobei *n* die betreffende Seitenzahl darstellt. Auf ähnliche Weise erzeugen die Befehle `\index{schlüssel|(xxx)}` und `\index{schlüssel|)xxx}` einen Seitenbereich der Form `\xxx{n-m}`.

Bereits existierende Befehle (wie im nachfolgenden Beispiel `\textit`) oder selbstdefinierte Befehle können zur Kapselung von Seitenzahlen verwendet werden. Ein Dokument mit der Befehlsdefinition

```
\newcommand\nn[1]{\#1n}
```

würde beispielsweise den folgenden Ausdruck erzeugen:

tabular, ii, 21, 22n	Seite ii:	<code>\index{tabular textbf}</code>
Tabulator, 7, 34-37	Seite 7:	<code>\index{Tabulator}</code>
	Seite 21:	<code>\index{tabular textit}</code>
	Seite 22:	<code>\index{tabular nn}</code>
	Seite 34:	<code>\index{Tabulator (textit)}</code>
	Seite 37:	<code>\index{Tabulator)textit}</code>

Die Kapselungsfunktion *see* stellt einen Spezialfall dieser Funktion dar. Der `\see`-Befehl wird dabei durch das *makeidx*-Paket vordefiniert.

11.1.5 Ausgeben von Sonderzeichen

Um eines der Zeichen, die für *MakeIndex* oder *xindy* eine besondere Bedeutung haben (!, ", @ oder |)¹ im Index zu setzen, muss diesem ein "-Zeichen vorangestellt werden. Genauer gesagt, wird jedes Zeichen maskiert, das einem unmaskierten " folgt, das nicht Teil des \"-Befehls ist. Letzterer ermöglicht die Eingabe von Umlauten. Maskierte !, @, " und | werden wie normale Zeichen behandelt und verlieren ihre besondere Bedeutung. Bevor die Einträge alphabetisch sortiert werden, wird ein ", das maskierten Zeichen vorangestellt ist, gelöscht.

"-Zeichen, 1		<code>\index{" @\\texttt{" } see{vertikaler Balken}}</code>
@-Zeichen, 2	Seite 1:	<code>\index{maskieren (\verb+"")}</code>
, <i>siehe</i> vertikaler Balken		<code>\index{""-Zeichen@\\texttt{""}-Zeichen}</code>
Ausruf (!), 4	Seite 2:	<code>\index{Klammeraffe (\texttt{"@})}</code>
Ah!, 5		<code>\index{"@-Zeichen@\\texttt{"@}-Zeichen}</code>
Klammeraffe (@), 2	Seite 3:	<code>\index{Maedchen@M\{a}dchen}</code>
Mädchen, 3	Seite 4:	<code>\index{Ausruf ("!")}</code>
maskieren ("), 1	Seite 5:	<code>\index{Ausruf ("!")Ah"!}</code>

11.1.6 Erstellen eines Glossars

Zur Erstellung von Glossaren bietet L^AT_EX einen Befehl namens `\glossary`. Der `\makeglossary`-Befehl erzeugt eine Datei mit der Erweiterung `.glo`, die der `.idx`-Datei für `\index`-Befehle ähnelt. In der gleichen Weise, wie L^AT_EX `\index`-Befehle in `\indexentry`-Einträge umwandelt, übersetzt L^AT_EX die `\glossary`-Befehle in `\glossaryentry`-Einträge.

MakeIndex kann diese Glossarbefehle ebenfalls handhaben; jedoch müssen die Werte einiger Schlüsselwörter in einer Stildatei geändert werden (siehe hierzu die Stildatei `myglossary.ist`).

```
% MakeIndex style file myglossary.ist
keyword    "\\glossaryentry"           % Schlüsselwort für Glossareinträge
preamble   "\\n \\begin{theglossary}\\n" % Beginn des Glossars
postamble  "\\n\\n \\end{theglossary}\\n" % Ende des Glossars
```

Zusätzlich muss man eine passende `theglossary`-Umgebung definieren.

11.1.7 Definieren eigener Indexbefehle

Wie in der Einführung erwähnt, ist es sehr wichtig, bei identischen Namen oder Befehlen im gesamten Dokument, einschließlich Index, dieselbe visuelle Darstellung zu verwenden. Um dies sicherzustellen, kann man Anwenderbefehle definieren, die ähnliche Konstrukte stets auf dieselbe Weise in den Text und den Index einfügen.

¹Wie bereits erwähnt, können in *MakeIndex* die voreingestellten Zeichen, die normalerweise eine besondere Bedeutung haben, auch durch andere Zeichen ersetzt werden. Die Vorgehensweise hierfür ist auf Seite 683 erläutert.

Man kann z.B. den Befehl `\Index` definieren, dessen Argument gleichzeitig in den Text und in den Index eingefügt wird.

```
\newcommand\Index[1]{#1\index{#1}}
```

Wie unten ausführlicher erläutert wird, ist darauf zu achten, dass das Argument solch eines Befehls kein expandierbares Material (Makros) oder überflüssige Leerzeichen enthält. Im Allgemeinen gibt es bei einfachen Begriffen wie Wörtern keine Probleme, und dieses Verfahren kann verwendet werden. Man kann sogar einen Schritt weitergehen und einem Eintrag ein bestimmtes optisches Erscheinungsbild verleihen – z.B. ihn in einer Schreibmaschinenschrift setzen.

```
\newcommand\Indextt[1]{\texttt{#1}\index{#1@\texttt{#1}}}
```

Schließlich kann man bestimmte Begriffe gruppieren, indem man Befehle definiert, die eine generische Bedeutung haben. Beispielsweise könnten, wie in den folgenden Beispielen, $\LaTeX$ -Befehle und -Programmnamen durch spezielle Befehle verarbeitet werden:

```
\newcommand\bs{\symbol{'134}} % Backslash in Typewriter
                                % OT1/T1 ausgeben
\newcommand\Com[1]{\texttt{\bs#1}\index{#1@\texttt{\bs#1}}}
\newcommand\Prog[1]{\texttt{#1}\index{#1@\texttt{#1}-Programm}}
```

Der `\Com`-Befehl hängt sowohl im Text als auch im Index einen Backslash an den Befehlsnamen an und erleichtert damit die Arbeit des Eingebenden. Die Definition des `\bs`-Befehls ist notwendig, weil `\textbackslash`, wie in Abschnitt 7.3.5 auf Seite 357 erläutert, bei OT1-Fontkodierung ersetzt würde. Zur selben Zeit werden Befehle im Index namentlich sortiert, wobei das `\`-Zeichen während der Sortierung ignoriert wird. Auf ähnliche Weise lässt der `\Prog`-Befehl während der alphabetischen Sortierung den `\texttt`-Befehl außer Acht, weil Einträge wie `\index{\texttt{schlüssel}}` und `\index{schlüssel}` zu verschiedenen Einträgen im Index führen würden.

11.1.8 Besondere Erwägungen

Wenn ein `\index`-Befehl direkt im Text verwendet wird, wird sein Argument nicht bereits bei der Erstellung der `.idx`-Datei, sondern erst bei der Formatierung des Index expandiert. Wenn der `\index`-Befehl jedoch im Argument eines anderen Befehls steht, müssen Zeichen, die für $\TeX$ eine besondere Bedeutung haben, wie etwa das `\`-Zeichen, vor Expansion geschützt werden. Dieses Problem tritt am ehesten bei der Indexierung von Begriffen in Fußnoten auf oder wenn Befehle verwendet werden, deren Argument gleichzeitig in den Text und in den Index eingefügt wird (siehe Besprechung in Abschnitt 11.1.7). Selbst in diesem Fall können robuste Befehle in den „@“-Teil eines Eintrags eingefügt werden, wie bei `\index{Rose@\textit{Rose}}`. Zerbrechliche Befehle müssen jedoch mit dem Befehl `\protect` geschützt werden.

Wie bei jedem Argument eines Befehls muss die gleiche Anzahl an öffnenden und schließenden Klammern angegeben werden. Da der `\index`-Befehl in seinem Argument Sonderzeichen wie `%` oder `\` zulässt, wenn der Befehl im

Haupttext verwendet wird, gibt es allerdings eine Anomalie, was die Klammerung betrifft: geschweifte Klammern in den Befehlen `\{` und `\}` zählen mit. Eine Eingabe wie `\index{\{}` oder ähnlich wäre demnach falsch.

11.2 makeindex – Ein Programm zur Formatierung und Sortierung des Index

Im vorangegangenen Abschnitt wurden Beispiele vorgestellt, in denen das *MakeIndex*-Programm mit Standardeinstellungen verwendet wurde. Dieser Abschnitt geht zunächst näher auf das *MakeIndex*-Programm ein, bevor Verfahren besprochen werden, die das Verhalten des Programms ändern.

11.2.1 Erstellen eines formatierten Index

Um einen formatierten Index zu erstellen, sollte man das *MakeIndex*-Programm durch Eingabe des folgenden Befehls aufrufen (main ist hierbei der Name der Eingabedatei):

```
makeindex main.idx
```

Dies erzeugt die Datei `main.ind`, die im Folgenden `.ind`-Datei genannt wird. Sofern *MakeIndex* keine Fehlermeldungen ausgegeben hat, kann $\LaTeX$ für das Dokument erneut aufgerufen werden. Danach erscheint im Dokument das fertig erstellte Indexregister. (Wenn der Index nicht erneut generiert werden soll, kann der `\makeindex`-Befehl jetzt wieder aus der Präambel entfernt werden.) Beim Auftreten von Fehlermeldungen siehe Seite 679.

Wenn man beim Durchlesen des Index Fehler findet, sollte man diese beheben, indem man die betreffenden `\index`-Befehle im Dokument ändert und die `.ind`-Datei ein zweites Mal erstellt. Dafür muss $\LaTeX$ *vor* und *nach* dem letzten Schritt aufgerufen werden.

Im Folgenden ist ein Beispiel für einen *MakeIndex*-Lauf dargestellt. Zunächst wird die `.idx`-Datei `main.idx` erzeugt, indem $\LaTeX$ die Eingabedatei bearbeitet, die in Abbildung 11.3 auf der nächsten Seite zu sehen ist. Wie im Beispiel deutlich zu erkennen ist, werden zwei Dateien erstellt, und zwar die sortierte `.ind`-Indexdatei `main.ind`, die als Eingabedatei für $\LaTeX$ dient, und die Indexprotokolldatei `.ilg` namens `main.ilg`, die (in diesem Fall) mit der Bildschirmausgabe übereinstimmt. Beim Auftreten von Fehlern sind in der Protokolldatei alle Fehlermeldungen sowie die Nummern der Zeilen aufgeführt, in denen die Fehler in der Eingabedatei aufgetreten sind. Abbildung 11.4 auf Seite 677 zeigt das Ergebnis des nachfolgenden $\LaTeX$ -Laufs. Die Beispiele verwenden das `showidx`-Paket, um die Indexeinträge im Rand anzuzeigen (siehe Abschnitt 11.4.2).

```
> makeindex main
This is makeindex, version 2.13 [07-Mar-1997] (using kpathsea).
Scanning input file 11-2-1.idx....done (11 entries accepted, 0 rejected).
Sorting entries....done (37 comparisons).
Generating output file 11-2-1.ind....done (24 lines written, 0 warnings).
Output written in 11-2-1.ind.
Transcript written in 11-2-1.ilg.
```

```

\usepackage{makeidx,showidx,multicol}
\usepackage[ngerman]{babel} \usepackage[ansinew]{inputenc}

\newcommand\bs{\symbol{'134}}% Backslash drucken
\newcommand\Com[1]{\texttt{\bs#1}\index{#1@\texttt{\bs#1}}}
\newcommand\Prog[1]{\texttt{#1}\index{#1@\texttt{#1}-Programm}}

\makeatletter
\renewenvironment{theindex}
  {\begin{multicols}{2}[\section*{\indexname}]}%
  {\setlength\parindent{0pt}\let\item\@idxitem}
  {\end{multicols}}
\makeatother
\makeindex

\section{Erzeugen eines Index}

Mithilfe des \textsf{showidx}-Paketes können Anwender die Erstellung der
Indexeinträge\index{Index!Einträge!anzeigen} direkt dort überprüfen, wo sie
erstellt werden.

Die Einträge\index{Index!Einträge!erstellen} werden mit dem
\Com{index}-Befehl in das Stichwortverzeichnis übernommen. Genauer gesagt
wird das Argument des \Com{index}-Befehls unverändert in die Hilfsdatei
\texttt{idx} geschrieben. Dies geschieht jedoch nur, wenn der
\Com{makeindex}-Befehl in der Dokumentenpräambel angegeben wurde.

\section{Aufbereiten des Index}

Um einen Index\index{Index!aufbereiten} auszugeben, muss zuvor die
\texttt{idx}-Datei mithilfe eines externen Programms wie \Prog{makeindex}
in eine \texttt{ind}-Datei umgewandelt werden.
\begin{verbatim}
  makeindex filename
\end{verbatim}

\section{Ausgabe des Index}\index{Enddurchlauf}

Während des Enddurchlaufs kann der Index in das Dokument
\index{Index!einfügen} eingefügt werden, indem an der Textstelle, an
welcher der Index erscheinen soll (normalerweise am Ende des Dokumentes),
ein \Com{printindex}-Befehl gesetzt wird. Dieser Befehl fügt die von
\Prog{makeindex} bearbeitete \texttt{ind}-Datei ein, woraufhin \LaTeX{} die
in ihr enthaltenen Daten setzt.

\printindex

```

Abbildung 11.3: Beispiel für den Einsatz von \index-Befehlen und des showidx-Paketes. Diese Datei wird zunächst mit $\text{\LaTeX}$ bearbeitet, danach wird der Indexprozessor ausgeführt, bevor $\text{\LaTeX}$ ein zweites Mal aufgerufen wird.

Bsp.
11-2-1

1 Erzeugen eines Index

Mithilfe des showidx-Paketes können Anwender die Erstellung der Indexeinträge direkt dort überprüfen, wo sie erstellt werden.

Die Einträge werden mit dem \index-Befehl in das Stichwortverzeichnis übernommen. Genauer gesagt wird das Argument des \index-Befehls unverändert in die Hilfsdatei idx geschrieben. Dies geschieht jedoch nur, wenn der \makeindex-Befehl in der Dokumentenpräambel angegeben wurde.

2 Aufbereiten des Index

Um einen Index auszugeben, muss zuvor die idx-Datei mithilfe eines externen Programms wie makeindex in eine ind-Datei umgewandelt werden.

makeindex filename

3 Ausgabe des Index

Während des Enddurchlaufs kann der Index in das Dokument eingefügt werden, indem an der Textstelle, an welcher der Index erscheinen soll (normalerweise am Ende des Dokumentes), ein \printindex-Befehl gesetzt wird. Dieser Befehl fügt die von makeindex bearbeitete ind-Datei ein, woraufhin L^AT_EX die in ihr enthaltenen Daten setzt.

Index

Enddurchlauf, 1	erstellen, 1
Index	\index, 1
aufbereiten, 1	\makeindex, 1
einfügen, 1	makeindex-Programm, 1
Einträge	
anzeigen, 1	\printindex, 1

Index!Einträge!anzeigen
Index!Einträge!erstellen
index@\index
index@\index
makeindex@\makeindex
Index!aufbereiten
makeindex@makeindex-
 Programm
Enddurchlauf
Index!einfügen
printindex@\printindex
makeindex@makeindex-
 Programm

Abbildung 11.4: Einsatz von \index-Befehlen und des showidx-Paketes (Ausgabe). Diese Abbildung zeigt den Index, der von der Eingabedatei generiert wurde, die in Abbildung 11.3 auf der gegenüberliegenden Seite dargestellt ist. Alle Indexeinträge werden im Rand abgebildet, so dass Fehler und doppelte Einträge leicht zu finden sind.

11.2.2 Einzelne Optionen des *MakeIndex*-Programms

Im Folgenden werden die Optionen des *MakeIndex*-Aufrufs beschrieben:

```
makeindex [-ciglqr] [-o ind] [-p nr] [-s stil] [-t log] [idx0 ...]
```

- c Ermöglicht die Komprimierung von Leerzeichen. Standardmäßig wird im Indexschlüssel jedes Leerzeichen berücksichtigt. Die Option *-c* ignoriert führende und angehängte Leerzeichen sowie Tabulatorzeichen und behandelt mehrere aufeinanderfolgende Leerzeichen zwischen den Wörtern als eines.
- i Verwendung der Standardeingabe (*stdin*) als Eingabedatei. Bei Aktivierung dieser Option ohne gleichzeitige Verwendung von *-o* wird die Ausgabe in die Standardausgabe (*stdout*, den voreingestellten Ausgabestrom) geschrieben.
- g Verwendung der deutschen Sortierreihenfolge gemäß DIN-Norm 5007. In diesem Fall wird die normale Sortierreihenfolge, nach der *MakeIndex* Wörter einordnet (Symbole, Zahlen, Großbuchstaben, Kleinbuchstaben), durch die deutsche Sortierreihenfolge (Symbole, Kleinbuchstaben, Großbuchstaben, Zahlen) ersetzt. Zusätzlich versetzt diese Option *MakeIndex* in die Lage, die deutschen $\TeX$ -Befehle "a, "o, "u, und "s zur besseren Sortierung als ae, oe, ue bzw. ss einzuordnen. Das Maskierungszeichen muss in einer Stildatei umdefiniert werden (siehe Seite 683); ansonsten wird eine Fehlermeldung ausgegeben, und *MakeIndex* bricht ab. Allerdings ist diese Option erst ab *MakeIndex* Version 2.13 korrekt implementiert.
- l Verwendung der Sortierreihenfolge nach Buchstaben. Bei der voreingestellten Sortierreihenfolge werden Leerzeichen vor den Buchstaben des Alphabets einsortiert. Bei der Sortierreihenfolge nach Buchstaben werden Leerzeichen ignoriert. Dementsprechend werden Indexeinträge wie „Punkt ab Stand“ und „Punktabstand“ in beiden Sortierarten unterschiedlich einsortiert.
- q Betrieb im stillen Modus. Es werden keine Meldungen auf dem Bildschirm ausgegeben (*stderr*). Normalerweise werden Arbeits- und Fehlermeldungen sowohl auf den Bildschirm *stderr* als auch zur Protokolldatei gesandt. Die Option *-q* deaktiviert die *stderr*-Meldungen.
- r Diese Option deaktiviert die automatische Seitenbereichserzeugung. Standardmäßig werden drei oder mehr aufeinanderfolgende Seiten automatisch zu einem Bereich zusammengefasst (z.B. 1-5). Durch Angabe von *-r* wird die Funktion deaktiviert, wodurch Seitenbereiche nur noch durch explizite Bereichsoperatoren erzeugt werden.
- o *ind* Verwendung von *ind* als Indexausgabedatei. Standardmäßig wird der Dateigrundname der ersten Eingabedatei *idx0*, verknüpft mit der Erweiterung *.ind*, als Name für die Ausgabedatei verwendet.
- p *nr* Mit dieser Option wird die Nummer der ersten Seite für die Indexausgabedatei auf *nr* gesetzt. Dies ist dann nützlich, wenn die Indexdatei separat formatiert werden soll. Außer reinen Zahlen sind für *nr* drei

Spezialfälle zugelassen, nämlich *any*, *odd* und *even* (alle, gerade und ungerade). In diesen Fällen wird die Nummer der ersten Seite festgelegt, indem die letzte Seitenzahl aus der *.log*-Datei des letzten $\LaTeX$ -Laufs ausgelesen wird. Der Name der *.log*-Datei wird bestimmt, indem der Dateigrundname der ersten Rohindexdatei (*idx0*) mit der Erweiterung *.log* verknüpft wird. Die letzte Seite der Quelldatei wird ermittelt, indem in der Protokolldatei von hinten nach vorne nach der ersten in der Datei vorkommenden Zahl gesucht wird, die in eckigen Klammern eingeschlossen ist. Sollte keine Seitenzahl vorhanden sein oder die *.log*-Datei nicht gefunden werden, dann wird die erste Seitenzahl nicht gesetzt. Die drei Spezialfälle haben folgende Bedeutung:

any Die Nummer der ersten Seite ist um eins größer als die letzte Seite der Quelldatei.

odd Die Nummer der ersten Seite ist die erste ungerade Seite, die der letzten Seite der Quelldatei folgt.

even Die Nummer der ersten Seite ist die erste gerade Seite, die der letzten Seite der Quelldatei folgt.

-s stil Verwendung von *stil* als Stildatei. Für diese Datei gibt es keine Standardeinstellung. Die Umgebungsvariable *INDEXSTYLE* legt den Suchpfad für die Stildatei fest.

-t log Verwendung von *log* als Protokolldatei. Standardmäßig wird für den Namen der Protokolldatei der Dateigrundname der ersten Eingabedatei *idx0* verknüpft mit der Erweiterung *.ilg* verwendet.

11.2.3 Fehlermeldungen

MakeIndex gibt auf dem Bildschirm die Anzahl der eingelesenen und geschriebenen Zeilen aus und teilt dem Anwender mit, wie viele Fehler gefunden wurden. Gleichzeitig zeichnet die Protokolldatei, die standardmäßig die Erweiterung *.ilg* hat, Meldungen auf, die über die Art der Fehler informieren. *MakeIndex* kann Fehlermeldungen erzeugen, wenn es etwas von der *.idx*-Datei liest oder etwas in die *.ind*-Datei überträgt. Jede Fehlermeldung gibt Auskunft über die Art des Fehlers und führt die Nummer der Zeile an, in welcher der Fehler in der Datei auftrat.

In der Lesephase bezieht sich die Zeilennummer auf die *.idx*-Datei.

*Fehler in der
Lesephase*

Extra ‘!’ at position ...

Das Argument des *\index*-Befehls enthält mehr als zwei unmaskierte *!*-Zeichen. Eventuell müssen eines oder mehrere von ihnen maskiert werden.

Extra ‘@’ at position ...

Das Argument des *\index*-Befehls enthält zwei oder mehr unmaskierte *@*-Zeichen ohne trennendes *!*. Eventuell muss eines der *@*-Zeichen maskiert werden.

Extra ‘|’ at position ...

Das Argument des *\index*-Befehls enthält mehr als ein unmaskiertes *|*-Zeichen. Eventuell sollten die zusätzlichen Zeichen maskiert werden.

Schlüsselwort	Voreinstellung	Beschreibung
keyword (s)	"\\indexentry"	Befehl, der <i>MakeIndex</i> mitteilt, dass sein Argument ein Indexeintrag ist.
arg_open (c)	'{'	Öffnendes Begrenzungszeichen des Argumentes
arg_close (c)	'}'	Schließendes Begrenzungszeichen des Argumentes
range_open (c)	'('	Öffnendes Begrenzungszeichen, das den Anfang des expliziten Seitenbereiches angibt.
range_close (c)	')'	Schließendes Begrenzungszeichen, welches das Ende des expliziten Seitenbereiches angibt.
level (c)	'!'	Begrenzungszeichen für Untereinträge.
actual (c)	'@'	Symbol, das angibt, dass der nachfolgende Teil des Eintrags in der Indexdatei erscheinen soll.
encap (c)	' '	Symbol, das angibt, dass der Rest der Argumentliste als Kapselungsfunktion für die Seitenzahl verwendet werden soll.
quote (c)	'\"'	Symbol, das das nachfolgende Zeichen maskiert.
escape (c)	'\\'	Symbol, das nur eine Sonderbedeutung besitzt, wenn es direkt vor dem Maskierungszeichen <code>quote</code> verwendet wird. In diesem Fall verliert das Maskierungszeichen seine Sonderbedeutung und beide Zeichen werden in die Ausgabedatei übertragen. Dieses Zeichen wurde einbezogen, weil <code>\</code> in $\text{\TeX}$ das Zeichen für Umlaute ist. Die beiden Symbole <code>quote</code> und <code>escape</code> müssen unterschiedlich sein.
page_compositor (s)	"-"	Trennzeichen für Seitenbereiche

(s) Wert vom Typ `string` (eingeschlossen in Anführungszeichen)

(c) Wert vom Typ `char` (eingeschlossen in Anführungsstriche)

Tabelle 11.1: Stilparameter für *MakeIndex* (Eingabe)

Illegal null field

Das Argument des `\index`-Befehls ergibt keinen Sinn, weil eine Zeichenkette leer ist, die nicht leer sein sollte. Die Eingabe `\index{!komisch}` erzeugt diesen Fehler, da sie einen Untereintrag „komisch“ definiert, ohne einen Haupteintrag zu besitzen. Aus dem gleichen Grund ist auch die Eingabe `\index{@komisch}` inkorrekt, da sie für den Sortiervorgang einen Leerstring definiert.

Argument ... too long (max 1024)

Das Dokument enthielt einen `\index`-Befehl mit einem sehr langen Argument. Wahrscheinlich wurde die rechte geschweifte Klammer vergessen, die das Argument begrenzen sollte.

Fehler in der Schreibphase

In der Schreibphase bezieht sich die Zeilennummer in der Fehlermeldung auf die `.ind`-Datei.

Unmatched range opening operator

Zu dem `\index{...|()}`-Befehl fehlt der zugehörige schließende `\index{...|})`-Befehl. Der Text „...“ muss in beiden Befehlen absolut identisch sein.

Unmatched range closing operator

Zu dem `\index{...|})`-Befehl fehlt der zugehörige öffnende `\index{...|()`-Befehl.

Extra range opening operator

Zwei `\index{...|()`-Befehle erscheinen im Dokument ohne trennenden `\index{...|})`-Befehl.

Inconsistent page encapsulator ... within range

MakeIndex wurde angewiesen, einen Seitenbereich für einen Eintrag sowie eine einzelne Seitenzahl innerhalb dieses Bereiches einzufügen, die anders formatiert ist – zum Beispiel in einem Befehl wie `\index{Katze|see{Tier}}` zwischen einem `\index{Katze|()`-Befehl und einem `\index{Katze|})`-Befehl.

Conflicting entries

MakeIndex ist der Meinung, dass es dieselbe Seitenzahl auf zwei verschiedene Arten formatieren soll – z.B. durch die Befehle `\index{Katze|see{...}}` und `\index{Katze}`, die sich auf derselben Seite befinden.

MakeIndex kann eine Anzahl anderer Meldungen erzeugen, die auf schwerwiegende Fehler in der `.idx`-Datei hinweisen. Wenn solch ein Fehler ausgegeben wird, bedeutet dies normalerweise, dass die `.idx`-Datei in irgendeiner Weise defekt ist. Wenn $\text{\LaTeX}$ bei der Erstellung der `.idx`-Datei keine Fehlermeldung generiert hat, ist es sehr unwahrscheinlich, dass eine fehlerhafte `.idx`-Datei erzeugt wurde. Andernfalls sollte die `.idx`-Datei auf die exakte Fehlerursache hin untersucht werden.

11.2.4 Anpassen des Index mit *MakeIndex*

Bei *MakeIndex* sind die Formate der Ein- und Ausgabedateien nicht festgelegt, sondern können an die Anforderungen einer bestimmten Anwendung angepasst werden. Um diese Formatunabhängigkeit zu erreichen, wird das *MakeIndex*-Programm von einer Stildatei gesteuert, die normalerweise mit dem Dateityp `.ist` gekennzeichnet ist (siehe auch Abbildung 11.1 auf Seite 668). Diese Datei besteht aus einer Reihe von Schlüssel-Wert-Paaren. Die Schlüsselwörter können in Eingabe- bzw. Ausgabe-Stilparameter aufgeteilt werden. Tabelle 11.1 auf der gegenüberliegenden Seite beschreibt die verschiedenen Schlüsselwörter und ihre Voreinstellungen für die Interpretation der Eingabedatei. Sie zeigt beispielsweise, wie man das Trennzeichen zur Kennzeichnung der Indexebene (`level`, mit `!` als Voreinstellung) verändern kann.

Tabelle 11.2 auf der nächsten Seite beschreibt die verschiedenen Schlüsselwörter und ihre Voreinstellungen, mit denen die Übersetzung der Eingabedaten in $\text{\LaTeX}$ -Befehle gelenkt wird. Hier findet man Hinweise, wie die verschiedenen Ebenen (mithilfe diverser `item`-Schlüsselwörter) formatiert werden. In Beispielen wird näher beschrieben, wie diese verschiedenen Schlüsselwörter der Eingabe- und Ausgabedatei in der Praxis verwendet werden können. *MakeIndex*-Stildateien verwenden die Syntax von UNIX-Zeichenketten. Um in der Ausgabe ein einzelnes `\` zu erhalten, muss man daher `\\` eingeben.

Die folgenden Abschnitte zeigen, wie man den Index ganz individuell gestalten kann, indem man ein paar Änderungen an den Werten der Parameter-voreinstellungen vornimmt, die den Index steuern.

Schlüsselwort	Voreinstellung	Beschreibung
Kontext		
preamble (s)	"\\begin{theindex}\n"	Befehl, der den Index einleitet
postamble (s)	"\n\n\\end{theindex}\n"	Befehl, der den Index abschließt
Startseite		
setpage_prefix (s)	"\n\\setcounter{page}{}"	Start des Befehls, mit dem die Nummer der Seite festgelegt wird
setpage_suffix (s)	"}\n"	Zugehöriger Suffix
Neue Buchstaben-Symbolgruppe		
group_skip (s)	"\n\n\\indexspace\n"	Vertikaler Platz zwischen zwei Gruppen
heading_prefix (s)	" "	Präfix für die Überschrift einer neuen Gruppe
heading_suffix (s)	" "	Suffix für die Überschrift einer neuen Gruppe
headings_flag (n)	0	flag≠0: Einfügung des Buchstabens (>0 Großbuchstaben) oder Symbols, das die neue Gruppe charakterisiert, umgeben von heading_prefix und heading_suffix; flag=0: keine Einfügung.
Haupt- und Untereinträge		
item_0 (s)	"\n\\item "	Befehl, der einen Haupteintrag einleitet
item_1 (s)	"\n \\subitem "	dito für Einträge der Ebene 1 mit Vorgänger auf Ebene ≥ 1
item_2 (s)	"\n \\subsubitem "	dito für Einträge der Ebene 2 mit Vorgänger auf Ebene ≥ 2
item_01 (s)	"\n \\subitem "	Befehl vor Einträgen der Ebene 1 mit Vorgänger auf Ebene 0
item_12 (s)	"\n \\subsubitem "	dito für Einträge der Ebene 2 mit Vorgänger auf Ebene 1
item_x1 (s)	"\n \\subitem "	Befehl vor Einträgen der Ebene 1 mit übergeordnetem Eintrag ohne Seitenzahl
item_x2 (s)	"\n \\subsubitem "	dito für Einträge der Ebene 2
Trennzeichen		
delim_0 (s)	", "	Trennzeichen zwischen Eintrag und erster Seitenzahl auf Ebene 0
delim_1 (s)	", "	dito für Ebene 1
delim_2 (s)	", "	dito für Ebene 2
delim_n (s)	", "	Trennzeichen zwischen Seitenzahlen
delim_r (s)	" - "	Trennzeichen für einen Seitenbereich
Kapselungsfunktion für Seitenzahlen		
encap_prefix (s)	"\\"	Präfix vor der Kapselungsfunktion der Seiten
encap_infix (s)	"{"	Infix für die Kapselungsfunktion der Seiten
encap_suffix (s)	"}"	Suffix für die Kapselungsfunktion der Seiten
Reihenfolge der Seiten		
page_precedence (s)	"rRnaA"	Reihenfolge: A, a bedeutet alphabetische Groß-/Kleinschreibung, n Zahlen; r und R klein- und großgeschriebene römische Ziffern
Zeilenumbruch		
line_max (n)	72	maximale Länge einer Zeile der Ausgabedatei
indent_space (s)	"\t\t"	Einrückung umbrochener Zeilen
indent_length (n)	16	Länge der Einrückung umbrochener Zeilen

„\n“ und „\t“ erzeugen neue Zeile / ein Tab; (s) Wert vom Typ string; (n) Wert vom Typ number

Tabelle 11.2: Stilparameter für *MakeIndex* (Ausgabe)

Unabhängiger Index

Die im folgenden dargestellte Stildatei `mybook.ist` definiert einen unabhängigen Index für ein Buch. Unabhängig bedeutet hier, dass die Datei separat formatiert werden kann. Dies kann sinnvoll sein, wenn der Eingabetext des Buches „eingefroren“ ist (d.h. die Seitenzahlen sich nicht mehr ändern) und man nur den Index neu formatieren möchte.

```
% MakeIndex style file mybook.ist
preamble
"\documentclass[12pt]{book} \n\n \\\begin{document} \n
  \\\begin{theindex}\n"
postamble
"\n\n\\end{theindex} \n \\\end{document}\n"
```

Angenommen, die Rohindexbefehle befänden sich in der Datei `mybook.idx`, dann kann man *MakeIndex* unter Angabe des Namens der Stildatei folgendermaßen aufrufen:

```
makeindex -s mybook.ist -o mybookind.tex mybook
```

Hier wurde bewusst kein Standardname für die Ausgabe der Quelldatei gewählt, um zu vermeiden, dass die Dokumentausgabedatei (wahrscheinlich `mybook.dvi`) zerstört wird. Wenn sich der Index in der Datei `mybook.ind` befindet, wird das formatierte Ergebnis ebenfalls in `mybook.dvi` ausgegeben, womit die originale `.dvi`-Datei überschrieben wird.

Um mit dem Index auf der richtigen Seite zu beginnen, kann man zusätzlich noch die Seitenzahl angeben, auf welcher der Index beginnen soll (z.B. 181 in folgendem Beispiel).

```
makeindex -s mybook.ist -o mybookind.tex -p 181 mybook
```

MakeIndex kann auch die $\text{\LaTeX}$ -Protokolldatei `mybook.log` lesen, um die Seitennummer zu ermitteln, an welcher der Index beginnen soll (siehe dazu die Option `-p`, die auf Seite 678 beschrieben ist).

Ändern der „Sonderzeichen“

Das nächste Beispiel zeigt, wie man die Interpretation von Sonderzeichen in der Eingabedatei verändern kann. Zu diesem Zweck müssen die neuen Sonderzeichen in einer Stildatei festgelegt werden. Mithilfe von Tabelle 11.1 auf Seite 680 wurde im folgenden Beispiel das `@`-Zeichen (siehe Seite 672) durch das Gleichheitszeichen ersetzt, das Ebenentrennzeichen `!` (siehe Seite 671) durch `>` und das Anführungszeichen `"` (siehe Seite 673) durch `!` (das Standard-Ebenentrennzeichen).

```
% MakeIndex style file myinchar.ist
actual '='      % = anstatt voreingestelltem @
quote '!'      % !                                "
level '>'      % >                                !
```

Diese Einstellungen lassen sich gut zusammen mit den Optionen `german` oder `ngerman` des `babel`-Paketes verwenden, da man nun die Anführungszeichen (") als Abkürzung für die Umlautkonstruktion `\`" verwenden kann.

"-Zeichen, 1	Seite 1:	<code>\index{\texttt{"}-Zeichen}</code>
=-Zeichen, 2	Seite 2:	<code>\index{\texttt{@}-Zeichen}</code>
@-Zeichen, 2	Seite 2:	<code>\index{\texttt{!=}-Zeichen}</code>
Ausruf (!), 4	Seite 3:	<code>\index{Maedchen=M\{a}dchen}</code>
Ah!, 5	Seite c:	<code>\index{Maedchen=M"adchen}</code>
Brücke, 5	Seite v:	<code>\index{Bruecke=Br"ucke}</code>
Brücke, V	Seite 5:	<code>\index{Br"ucke}</code>
Brücke, v	Seite V:	<code>\index{Br\"ucke}</code>
Dimension	Seite 3:	<code>\index{Dimension>Linie>Breite}</code>
Linie	Seite 4:	<code>\index{Ausruf (!)}</code>
Breite, 3	Seite 5:	<code>\index{Ausruf (!)>Ah!!}</code>
Mädchen, c		
Mädchen, 3		

Im Beispiel wird eine weitere Eigenschaft für die Sortierung von *MakeIndex* deutlich: Die Konstruktionen `"` und `\` werden als unterschiedliche Einträge betrachtet (`Br"ucke` und `Br\"ucke`, `M"adchen` und `M\"adchen`, auch wenn im letzteren Fall der eigentliche Eintrag, nämlich `Maedchen`, identisch ist). Daher ist es wichtig, dieselbe Eingabekonstruktion im gesamten Dokument zu verwenden.

Ändern des Ausgabeformats

Auch das Ausgabeformat des Index kann man selbst gestalten. Als erstes könnte man z.B. einen Index erstellen, in dem jede Buchstabengruppe von der nächsten durch einen großen Buchstaben getrennt ist. Dieses Ergebnis kann man mit der unten aufgeführten Stildatei `myhead.ist` erreichen (siehe Tabelle 11.2 auf Seite 682 für nähere Angaben). Das Resultat dieser Maßnahme ist in Abbildung 11.5 dargestellt.

```
% MakeIndex style file myhead.ist
heading_prefix "{\\bfseries\\hfil "      % vor Zeichen einfügen
heading_suffix "\\hfil}\\nopagebreak\\n" % nach Zeichen einfügen
headings_flag      1                    % Überschriften an
                                           % (Großbuchstaben)
```

Ein weiterer Schritt könnte die Ausrichtung der Seitenzahlen am rechten Rand sein, wobei zwischen dem Text und der Seitenangabe Punkte eingefügt werden, die das Auge leiten. Ein Beispiel hierfür ist in Abbildung 11.6 auf der gegenüberliegenden Seite zu sehen. Dieser Effekt lässt sich erreichen, indem man folgende Befehle hinzufügt:

```
% MakeIndex style file myright.ist
delim_0    "\\dotfill "
delim_1    "\\dotfill "
delim_2    "\\dotfill "
```

Der $\text{\LaTeX}$ -Befehl `\dotfill` kann durch ausgefalleneren Befehle ersetzt werden, aber das Grundprinzip bleibt dasselbe.

Symbole	Seite 2:	<code>\index{\texttt{"@}-Zeichen}</code>
@-Zeichen, 2	Seite 3:	<code>\index{Dimension!Linie!Breite}</code>
B	Seite 5:	<code>\index{Box!Parameter}</code>
Box, 21		<code>\index{Fonts!PostScript}</code>
Dimension der, 33	Seite 9:	<code>\index{Dimension!Tabelle}</code>
Parameter, 5	Seite 12:	<code>\index{Dimension!Linie!Tiefe}</code>
D	Seite 17:	<code>\index{Dimension!Abbildung}</code>
Dimension	Seite 21:	<code>\index{Box}</code>
Abbildung, 17		<code>\index{Fonts!Computer Modern}</code>
Linie	Seite 33:	<code>\index{Box!Dimension der}</code>
Breite, 3		<code>\index{Linie!Tiefe}</code>
Tiefe, 12	Seite 41:	<code>\index{Linie!Breite}</code>
Tabelle, 9	Seite 48:	<code>\index{Linie!Tiefe}</code>
F		
Fonts		
Computer Modern, 21		
PostScript, 5		
L		
Linie		
Breite, 41		
Tiefe, 33, 48		

Abbildung 11.5: Beispiel für die Anpassung des Ausgabeformats

@-Zeichen	2	Seite 2:	<code>\index{\texttt{"@}-Zeichen}</code>
Box	21	Seite 3:	<code>\index{Dimension!Linie!Breite}</code>
Dimension der ...	33	Seite 5:	<code>\index{Box!Parameter}</code>
Parameter	5		<code>\index{Fonts!PostScript}</code>
Dimension		Seite 9:	<code>\index{Dimension!Tabelle}</code>
Abbildung	17	Seite 12:	<code>\index{Dimension!Linie!Tiefe}</code>
Linie		Seite 17:	<code>\index{Dimension!Abbildung}</code>
Breite	3	Seite 21:	<code>\index{Box}</code>
Tiefe	12		<code>\index{Fonts!Computer Modern}</code>
Tabelle	9	Seite 33:	<code>\index{Box!Dimension der}</code>
Fonts			<code>\index{Linie!Tiefe}</code>
Computer Modern	21	Seite 41:	<code>\index{Linie!Breite}</code>
PostScript	5	Seite 48:	<code>\index{Linie!Tiefe}</code>
Linie			
Breite	41		
Tiefe	33, 48		

Abbildung 11.6: Leitpunkte im Index hinzufügen

Arbeiten mit zusammengesetzten Seitenzahlen

Wie bereits beschrieben, akzeptiert *MakeIndex* fünf elementare Arten von Seitenzahlen: Ziffern, Groß- und Kleinbuchstaben sowie groß- und kleingeschriebene römische Zahlen. Daneben können aber auch zusammengesetzte Seitenzahlen verarbeitet werden. Das Trennzeichen für solche Seitenzahlen wird

durch das *MakeIndex*-Schlüsselwort `page_compositor` festgelegt. Die Voreinstellung ist der Trennstrich (-); siehe Tabelle 11.1 auf Seite 680. Die Sortierreihenfolge der verschiedenen Arten von Seitenzahlen wird dagegen durch das Schlüsselwort `page_precedence` bestimmt (die voreingestellte Reihenfolge ist `rRnA`; siehe Tabelle 11.2 auf Seite 682).

Das erste Beispiel befasst sich zunächst mit einfachen Seitenzahlen. Angenommen die Seiten mit den Nummern ii, iv, 1, 2, 5, a, c, A, C, II und IV enthalten einen `\index`-Befehl mit dem Inhalt „Stil“. Mit der Standard-`page_precedence` von `rRnA` würde dieser Begriff im Index dann wie unten gezeigt gesetzt werden. Die Einträge c und C werden dabei als römische Ziffern betrachtet, und nicht etwa als Buchstaben:

Stil, ii, iv, c, II, IV, C, 1, 2, 5, a, A

Diese Sortierreihenfolge kann man mithilfe des `page_precedence`-Schlüsselwortes ändern, wie in der Stildatei `mypages.ist` zu sehen ist.

```
% MakeIndex style file mypages.ist
page_precedence "rNaaR"
```

Eine erneute Bearbeitung der obigen Indexeinträge mit *MakeIndex* ergibt nun:

Stil, ii, iv, c, 1, 2, 5, A, a, II, IV, C

Wie man sieht, werden auch hier Buchstaben wie „c“ als römische Ziffern interpretiert. Solange *MakeIndex* keine Möglichkeit bietet, dieses Verhalten zu beeinflussen, lassen sich einfache, mit Buchstaben nummerierte Seiten damit leider nur mit bedingt verwenden – entweder akzeptiert man eine potenziell fehlerhafte Reihenfolge innerhalb der Seitenverweise oder man ist gezwungen, diese manuell nachzuarbeiten.

Die Situation sieht bei zusammengesetzten Seitenzahlen leicht anders aus, d.h. bei Seiten die Nummerierungen wie „A-3“ tragen (wobei „A“ etwa die Anhangnummer und „3“ die Seite innerhalb dieses Anhangs darstellen). In dieser doch recht häufigen Nummerierungsvariante wird „C“ zu den Buchstaben gezählt – „I“ aber nach wie vor zu den römischen Ziffern. Somit kann man hier acht alphabetisch nummerierte Anhänge verwenden, bevor man in Probleme mit der Reihenfolge läuft. Wie bereits erwähnt, ist der Bindestrich das voreingestellte Separierungszeichen. Angenommen, es gibt einen Verweis auf das Wort *Stil* auf folgenden Seiten (in unsortierter Reihenfolge): C-3, 1-1, D-1-1, B-7, F-3-5, 2-2, D-2-3, A-1, B-5 und A-2. Nach der Bearbeitung mit *MakeIndex* wird daraus der folgende sortierte Index erzeugt:

Stil, 1-1, 2-2, A-1, A-2, B-5, B-7, C-3, D-1-1, D-2-3, F-3-5

Haben die Verweise stattdessen die Gestalt C--3, 1--1, ..., dann erhält man lauter Fehlermeldungen und einen leeren Index. In einem solchen Fall muss man das Separierungszeichen mithilfe des Schlüsselwortes

Probleme
bei Buchstaben als
Seitennummern

page_compositor in einer Stildatei passend abändern:

```
% MakeIndex style file mypagsep.ist
page_compositor "---"
```

Verarbeitet man die Indexeinträge mit dieser Stildatei, werden die Seitennummern korrekt erkannt und sortiert:

Stil, 1-1, 2-2, A-1, A-2, B-5, B-7, C-3, D-1-1, D-2-3, F-3-5

Da *MakeIndex* nur ein Separierungszeichen kennt, lassen sich komplexer aufgebaute Seitennummerierungen (wie etwa „A-4.1“) mit dem Programm nicht verarbeiten.

11.2.5 *MakeIndex*-Fallstricke

Der `\index`-Befehl versucht, wann immer möglich sein Argument unverändert in die `.idx`-Datei zu schreiben.¹ Dieses Verhalten hat eine Reihe verschiedener Konsequenzen. Wenn der Indextext Befehle enthält (wie in `\index{\Prog}`), ist der Eintrag wahrscheinlich falsch einsortiert, weil dieser Eintrag im Haupttext unter dem Sortierschlüssel `\Prog` einsortiert wurde (mit dem Sonderzeichen `\` als Anfangszeichen für die Sortierung), egal wie die Definition des `\Prog`-Befehls lautet. Auf der anderen Seite wird `\Prog` expandiert, wenn er im Argument eines anderen Befehls verwendet wird, bevor er in die `.idx`-Datei geschrieben wird; die Platzierung im Index hängt dann von der Expansion von `\Prog` ab. Dasselbe geschieht, wenn man `\index` innerhalb eigener Definitionen verwendet. Alle Befehle innerhalb des Indexargumentes werden expandiert (es sei denn, es handelt sich um robuste Befehle oder ihnen ist `\protect` vorangestellt).

Bei der Sortierung geht *MakeIndex* davon aus, dass Seiten, die mit römischen Zahlen nummeriert sind, jenen mit arabischen Zahlen vorangestellt werden. Arabische Zahlen kommen wiederum vor Buchstaben. Diese Reihenfolge lässt sich nur mit gewissen Einschränkungen ändern, und es gibt generelle Probleme bei der Verwendung von Buchstaben als Seitennummern (siehe die Diskussion auf der vorherigen Seite).

MakeIndex ordnet Symbole (das heißt Zeichenfolgen, die mit einem nicht alphanumerischen Zeichen beginnen) im Index vor Zahlen und diese vor alphanumerischen Einträgen ein. Die Symbole werden anhand ihres ASCII-Wertes einsortiert (was nicht notwendigerweise der korrekten Sortierreihenfolge der verwendeten Sprache entspricht). Bei der Sortierung von Wörtern werden groß- und kleingeschriebene Wörter gleich behandelt, aber bei identischen Einträgen kommt die großgeschriebene Variante vor der kleingeschriebenen. Zahlen werden in aufsteigender Reihenfolge sortiert.

Leerzeichen werden als normale Zeichen betrachtet. Sie zählen sowohl bei der alphabetischen Sortierung als auch bei der Entscheidung mit, ob zwei Einträge identisch sind (siehe auch das Beispiel auf Seite 670). Wenn z.B. „□“ ein Leerzeichen markiert, dann erzeugen die Befehle `\index{Katze}`,

¹Die Art, wie $\text{\LaTeX}$ versucht, Expansionen zu verhindern, ist nicht immer erfolgreich. Das `index`-Paket (siehe Abschnitt 11.4.3) verwendet einen anderen Ansatz, der in *allen* Fällen eine Expansion verhindert.

`\index{␣Katze}` und `\index{Katze␣}` drei separate Einträge. Im Ausdruck sehen alle drei (fast) gleich aus. Auf ähnliche Weise erzeugen auch `\index{ein␣Leerzeichen}` und `\index{ein␣␣Leerzeichen}` zwei verschiedene Einträge, die im Ausdruck identisch aussehen. Aus diesem Grund ist es wichtig, dass man überflüssige Leerzeichen vermeidet. Vorsicht ist z.B. geboten, wenn das Argument eines `\index`-Befehls in der Eingabedatei am Zeilenende umbrochen wird. Die *MakeIndex*-Option `-c` schaltet dieses Verhalten aus, entfernt führende und angehängte Leerzeichen und komprimiert sonstige enthaltene Leerzeichen zu einem. Es ist empfehlenswert, diese Option stets eingeschaltet zu haben.

11.3 xindy – Eine Alternative zu *MakeIndex*

Das Programm *xindy* von Roger Kehr und Joachim Schrod ist ein flexibles System zur Indexerstellung, das eine Alternative zu *MakeIndex* darstellt. Es überwindet einige der Beschränkungen von *MakeIndex*, insbesondere der Erstellung von Indexregistern in nicht englischen Sprachen. Die Verwendung von *xindy* empfiehlt sich in folgenden Fällen:

- Es gibt einen **Index mit nicht englischen Wörtern**, ansonsten soll alles so wie mit *MakeIndex* funktionieren.
Die Migration aus *MakeIndex* ist einfach, weil *xindy* verwendet werden kann, ohne dass die Indexeinträge im Dokument geändert werden müssen. Eine Kompatibilitäts-Stildatei erzeugt Ergebnisse, die den Standardeinstellungen von *MakeIndex* entsprechen. Der Hauptunterschied besteht darin, dass die Sortierung der Indexeinträge problemlos funktioniert.
- Es soll sichergestellt werden, dass der Index **konsistenter** ist als jener, der mit *MakeIndex* erzeugt wird.
Da *MakeIndex* alle Begriffe wortwörtlich nimmt, muss die visuelle Darstellung, wie in Abschnitt 11.1.4 auf Seite 672 erläutert, explizit angegeben werden. Dies gilt insbesondere dann, wenn zur Darstellung $\LaTeX$ -Befehle benötigt werden. Wenn an einer Stelle die Angabe einer speziellen Darstellung vergessen wurde, wird der Index inkonsistent. Das *xindy*-Programm nimmt übliche $\LaTeX$ -Darstellungen und berechnet aus ihnen den Indexschlüssel – damit müssen Unterschiede zwischen Indexschlüssel und visueller Darstellung nicht mehr jedes Mal explizit angegeben werden. (Zum Beispiel sind die unterschiedlichen Definitionen von `\Index` und `\Indextt` aus Abschnitt 11.1.7 auf Seite 673 nicht mehr notwendig.) Natürlich kann man trotzdem noch spezielle Darstellungen in den Indexeinträgen angeben.
- Der Index soll stärker **auf Korrektheit überprüft** werden.
Wenn im Index Querverweise vorkommen, die mit *see* eingeleitet werden (vgl. Besprechung in Abschnitt 11.1.3 auf Seite 671), überprüft *xindy*, ob der Eintrag, auf den verwiesen wird, tatsächlich existiert. Auf diese Weise lassen sich tote Verknüpfungen vermeiden.
- Es soll ein **technischer Index auf effiziente Weise** erstellt werden.
Technische Stichwortverzeichnisse enthalten häufig $\LaTeX$ -Befehle in den

Indexschlüsseln, z.B. um Symbole oder Logos aufzunehmen. xindy ermöglicht es, den eigentlich benötigten Indexschlüssel anhand vorgegebener Regeln zu konstruieren. Dies gibt dem Anwender die Möglichkeit, die eigentlichen Indexeinträge automatisch aus den $\text{\LaTeX}$ -Befehlen heraus zu generieren, so dass jede Verwendung eines technischen Begriffs im Index aufgenommen wird. Voraussetzung dafür ist das Erstellen der benötigten Konstruktionsregeln.

- Es soll ein **Index mit „ungewöhnlichen“ Begriffen** erstellt werden. Bei einigen Begriffen gelten aus historischen Gründen besondere Sortierregeln. Zum Beispiel werden manchmal die Namen von Orten und Personen anders sortiert, als sie geschrieben werden – „St. Martin“ wird je nach Kontext als „Martin“ oder als „Sankt Martin“ einsortiert, „van Beethoven“ wird als „Beethoven“ eingeordnet usw. Symbole sind ein weiteres Beispiel für eine mehr oder weniger willkürliche Sortierreihenfolge im Index. Wie auch immer diese definiert ist, sie sollte im gesamten Dokument einheitlich sein.

Das xindy-Programm bietet diese Vorteile, weil es viele der festen Annahmen von *MakeIndex* verworfen hat, die in internationalen Dokumenten mit beliebigen Arten von Referenzen keine Gültigkeit haben. Stattdessen bietet xindy einen flexiblen Rahmen, um Regeln zur Indexerstellung aufzustellen, zusammen mit einem einfachen *MakeIndex*-ähnlichen Skript für Standardaufgaben. Die Leistungsfähigkeit von xindy ergibt sich aus fünf zentralen Eigenschaften:

Internationalisierung xindy lässt sich einfach für Sprachen mit verschiedenen Alphabeten und/oder unterschiedlichen Sortierregeln konfigurieren. Es ist möglich, zusätzliche Buchstaben oder komplette Alphabete zu definieren und eine Reihe von Regeln zur Sortierung und Gruppierung dieser Zeichen bereitzustellen. Momentan werden durch die xindy-Distribution etwa 50 Sprachen unterstützt.

Modulare Konfiguration xindy wird durch kombinierbare und wiederverwendbare Deklarationen konfiguriert. Für Standard-Indexaufgaben müssen $\text{\LaTeX}$ -Anwender lediglich die verfügbaren Module benutzen.

Normalisierung des Markups Ein ermüdendes Problem, das im Zusammenhang mit technischen oder fremdsprachigen Stichwortverzeichnissen entsteht, betrifft das Markup und nicht textuelles Material. xindy bietet die Möglichkeit, Eingabevarianten und unterschiedliche Kodierungen für dasselbe Objekt zu ignorieren oder Markup-Elemente, wie den Formelmodus, aus dem Eintrag zu entfernen.

Benutzerdefinierbare Textreferenzen Ein Indexeintrag verweist auf eine bestimmte Stelle im Text. Ausgefallene Stichwortverzeichnisse bestehen möglicherweise nicht nur aus Seitenzahlen, sondern auch aus Buchnamen, Gesetzesparagraphen und strukturierten Artikelnummern (z.B. „I-20“, „Genesis 1, 31“). Das xindy-Programm ermöglicht eine beliebige Sortierung und Gruppierung dieser Referenzen.

Konfigurierbare Auszeichnung xindy bietet eine vollständige Kontrolle des Markups. Diese Funktion ist in der Regel für $\text{\LaTeX}$ -Anwender nicht von Bedeutung, aber sie ist sehr praktisch für die Indexierung von Nicht- $\text{\TeX}$ -Material.

Verfügbarkeit Wenn das xindy-Programm nicht Teil der $\text{\TeX}$ -Distribution ist, bietet die Webseite des Programms (www.xindy.org) Distributionen für viele verschiedene Betriebssysteme und weitere Referenzdokumentationen. CTAN enthält ebenfalls xindy-Distributionsdateien.

11.3.1 Erzeugen des formatierten Index mit xindy

Das xindy-Programm enthält den Befehl `texindy`, mit dem das Programm für Standardaufgaben auf einfache, *MakeIndex*-ähnliche Weise eingesetzt werden kann. Die Optionen, die äquivalent zu jenen von *MakeIndex* sind, werden an dieser Stelle nicht mehr im Detail erörtert; hierzu sei auf Abschnitt 11.2.2 verwiesen. Auf die Optionen `-M` und `-L` wird in den folgenden Abschnitten näher eingegangen.

```
texindy [-gilqr] [-o ind] [-t log] [-L sprache] [-M modul] [idx0 ...]
```

- `-i` Verwendung der Standardeingabe (`stdin`) als Eingabedatei.
- `-o ind` Verwendung von *ind* als Indexausgabedatei.
- `-t log` Verwendung von *log* als Protokolldatei.
- `-q` Betrieb im stillen Modus.
- `-g` Verwendung des deutschen Modus (äquivalent zu `-L german-din` `-M german-sty`).
- `-l` Sortierreihenfolge nach Buchstaben; voreingestellt ist die Sortierung nach Wörtern (äquivalent zu `-M letter-order`).
- `-r` Deaktivierung der automatischen Seitenbereichserzeugung (äquivalent zu `-M no-ranges`).
- `-M modul` Verwendung der xindy-Stildatei *modul.xdy* zur Konfigurierung der Bearbeitung.
- `-L sprache` Verwendung von *sprache* als Sprachkonfiguration für die Sortierung der Wörter.

Die Dateien *idx0*, *idx1* usw. enthalten Rohindexeinträge. Bei Angabe von mehreren Ausgabedateien ist es sinnvoll, die Option `-o` zu verwenden, um die Ausgabedateien mit Namen zu versehen. Standardmäßig wird der Name der Ausgabedatei aus *idx0* generiert.

Bei Verwendung der Option `-c`, `-p` oder `-s` wird eine Warnung ausgegeben, die darüber informiert, dass diese *MakeIndex*-Optionen nicht unterstützt werden. Tatsächlich sind xindy-Stildateien selbst geschriebene Module, die mit der Option `-M` angegeben werden; für Einzelheiten zur Erstellung der Module siehe Abschnitt 11.3.4.

Der Befehl `texindy` komprimiert standardmäßig Leerzeichen, da die Autoren der Meinung sind, dass man dies von einem Indexprozessor erwartet. Tatsächlich arbeitet das gesamte $\TeX$ -Programmpaket standardmäßig unter der Annahme, dass mehrere Leerzeichen effektiv nur ein Leerzeichen sind. Wenn sich `xindy` auch hierin *MakeIndex*-kompatibel verhalten soll, kann man das Modul `keep-blanks` verwenden, wie Abschnitt 11.3.3 erläutert.

Mit der *MakeIndex*-Option `-p` kann ein $\LaTeX$ -Befehl in die `.ind`-Datei geschrieben werden, der den Seitenzähler setzt. Eventuell wird dafür sogar die Protokolldatei von $\LaTeX$ analysiert. Das `xindy`-Programm hat keine solche Option, und dies ist beabsichtigt. Die Autoren von `xindy` glauben, dass ein separates $\LaTeX$ -Dokument für den Index zu fehleranfällig ist und der bessere Ansatz darin besteht, die $\LaTeX$ -Datei mit dem Befehl `\printindex` in das Hauptdokument einzubinden.

Der `texindy`-Befehl ignoriert standardmäßig unbekannte $\TeX$ -Befehle, unter der Annahme, dass sie keinen Text erzeugen. Außerdem kennt er typische texterzeugende Befehle, wie `\LaTeX` und `\BibTeX`, und handhabt sie korrekt. Wenn eigene Befehlsdefinitionen erstellt werden, die Text produzieren, oder von einem Paket bereitgestellte Befehle verwendet werden, wird der Eintrag falsch sortiert. Entweder muss im Indexeintrag ein expliziter Sortierschlüssel angegeben werden, wie in `\index{prog@\Prog}`, oder eine `xindy`-Stildei mit einer Vereinheitlichungsregel geschrieben werden, wie in Abschnitt 11.3.4 erläutert.

*$\LaTeX$ -Befehle
indexieren*

Zu beachten ist, dass die Verwendung von Indexeinträgen in Argumenten von Befehlen seine eigenen Tücken hat, zum Beispiel in einer Überschrift wie `\section{Properties of \Prog\index{\Prog}}`. In diesem Fall werden Befehle möglicherweise expandiert, bevor sie in die `.idx`-Datei geschrieben werden, und die Platzierung im Index hängt von der Expansion von `\Prog` ab.

11.3.2 Internationale Indexierung mit `xindy`

Die meisten nicht englischen Sprachen stellen eine zusätzliche Herausforderung für die Indexbearbeitung dar. Sie haben akzentuierte Zeichen oder sprachspezifische Zeichen, die in Bezug auf die Sortierung speziellen Regeln gehorchen. Normalerweise reicht es nicht aus, die Akzente zu ignorieren, und natürlich kann man die binäre Kodierung der Zeichen in einer bestimmten Sprache nicht für die Sortierung verwenden. Selbst wenn man es wollte, wäre es sogar recht schwierig, die binäre Kodierung zur Sortierung zu verwenden – die meisten $\LaTeX$ -Implementierungen geben Nicht-ASCII-Zeichen als $\LaTeX$ -Kommandos aus, z.B. wird „ß“ als `\IeC {\ss}` ausgegeben.

Die Realität sieht anders aus: Entweder werden fremde Zeichen mit Makros eingegeben oder das `inputenc`-Paket wird verwendet. Zum Beispiel wird ein $\LaTeX$ -Anwender in Westeuropa auf einem Linux-System für alle seine Dokumente wahrscheinlich `\usepackage[latin1]{inputenc}` hinzufügen (oder auf einer neueren Linux-Distribution die Option `utf8`), während Windows-Anwender die `inputenc`-Option `ansinew` oder `utf8` nutzen. Damit enthält die `Rohindex`-Datei plötzlich viele $\LaTeX$ -Befehle, da alle sprachspezifischen und akzentuierten Zeichen als Befehle ausgegeben werden. Bei *MakeIndex* muss der Autor Sortierschlüssel und Schlüssel für den Druck für solche Indexeinträge separat angeben. Bei einigen wenigen Indexeinträgen ist das zu bewältigen,

Argument der *texindy* -L-Option

albanian	finnish	kurdish-bedirxan	slovak-small
croatian	french	kurdish-turkish-i	slovak-large
czech	general	latvian	slovenian
danish	german-din	lithuanian	spanish-modern
dutch	german-duden	lower-sorbian	spanish-traditional
dutch-ij-as-ij	greek-translit	norwegian	swedish
english	hungarian	polish	turkish
esperanto	icelandic	portuguese	upper-sorbian
estonian	kurdish	romanian	

general ist die voreingestellte Sprachoption, sie bietet Definitionen, die für alle westeuropäischen Sprachen gleichermaßen geeignet sind. Sie unterstützt keine Sonderbehandlung von sprachspezifischen Zeichen.

Weitere Sprachoptionen sind für *xindy* verfügbar, können aber nicht einfach mit *texindy* verwendet werden.

Tabelle 11.3: Von *texindy* unterstützte Sprachen

ist jedoch sehr fehleranfällig, wenn es für alle Einträge vorgenommen werden muss, die sprachspezifische Zeichen enthalten. Außerdem funktioniert die automatische Erstellung von Indexeinträgen mithilfe von $\text{\LaTeX}$ -Befehlen (wie in Abschnitt 11.1.7 empfohlen) nicht mehr.

Das *xindy*-Programm bietet Lösungen für dieses Problem. Es kennt die $\text{\LaTeX}$ -Makros für sprachspezifische Zeichen und verarbeitet sie wie erforderlich. Damit können neue Alphabete und ihre Sortierreihenfolge sowie komplexere, mehrphasige Sortierregeln definiert werden, die das richtige Sortierschema beschreiben. So können typische reale Anforderungen, wie etwa folgende, bewältigt werden:

Deutsch Im Deutschen gibt es zwei verschiedene Sortierschemata, um Umlaute zu handhaben: normalerweise wird *ä* wie *ae* sortiert, aber in Telefonbüchern oder Wörterbüchern wird es wie *a* sortiert. Das erste Schema ist als *DIN-Reihenfolge* bekannt, das zweite als *Duden-Reihenfolge* [45].

Spanisch Im Spanischen ist die Ligatur *ll* eine separate Buchstabengruppe, die nach *l* und vor *m* vorkommt.

Französisch Im Französischen werden in der ersten Sortierphase diakritische Zeichen ignoriert, so dass *cote*, *côte*, *coté* und *côté* gleich einsortiert werden. In der nächsten Phase werden akzentuierte Zeichen, die sich nur in den Akzenten unterscheiden, von rechts nach links durchsucht. Buchstaben mit diakritischen Zeichen kommen demzufolge nach Buchstaben ohne diese. *cote* und *côte* stehen damit am Anfang (ohne Akzent auf dem *e*), danach folgen Wörter mit *o* vor Wörtern mit *ô*.

Das *xindy*-Programm bietet *Sprachmodule* für eine wachsende Anzahl von Sprachen. Solch ein Sprachmodul definiert das Alphabet mit allen sprachspezifischen Zeichen, ihren Sortierregeln und auf diese Sprache angepassten Definitionen für Buchstabengruppen. Außerdem werden akzentuierte Zeichen, die üblicherweise in dieser Sprache verwendet werden, korrekt verarbeitet.

Die vordefinierten Sprachmodule umfassen west- und osteuropäische Sprachen. Derzeit gibt es keine Unterstützung für asiatische Sprachen.

Es gibt etwa 50 vordefinierte Sprachen, wovon 35 sofort mit *texindy* einsetzbar sind. Sie sind in Tabelle 11.3 auf der vorherigen Seite aufgeführt; ausgewählt werden sie mit der *texindy*-Option `-L`. Die anderen vordefinierten Sprachen haben nicht lateinische Schriften, ihre Verwendung wird in der *xindy*-Dokumentation beschrieben.

Es besteht auch die Möglichkeit, sich eigene *xindy*-Sprachmodule zu schaffen. Das *xindy*-Hilfsprogramm *make-rules* vereinfacht dieses Verfahren, wenn die gewünschte Sprache folgende Kriterien erfüllt:

- Das Schriftsystem verwendet ein Alphabet mit Buchstaben.
- Die Sortierreihenfolge basiert auf diesen Buchstaben (und auf den Akzenten).
- Zur Sortierung ist kein spezieller Rückgriff auf den Kontext notwendig; Akzente beeinflussen nur die Sortierreihenfolge der akzentuierten Buchstaben.

Weitere Informationen zur Erstellung von Sprachmodulen mit und ohne *make-rules* sind auf der Webseite von *xindy* (www.xindy.org) zu finden. Neue Module werden vom *xindy*-Projekt gern aufgenommen.

11.3.3 Module für alltägliche Aufgaben

Wie *MakeIndex* kann auch *xindy* konfiguriert werden, indem man eine persönliche Stildatei erstellt, wie in Abschnitt 11.3.4 erläutert. Die meisten Anwender benötigen jedoch nicht den vollen Umfang der *xindy*-Konfiguration. Ihnen reicht es, die üblichen Probleme mit den existierenden Modulen zu lösen.

Zur Vereinfachung der üblichen Aufgaben stellt *xindy* eine Reihe von Modulen bereit, die in Tabelle 11.4 auf der nächsten Seite aufgeführt sind. Sie bieten Standardlösungen zur Sortierung, Erstellung von Seitenbereichen und Layoutanforderungen. Wenn es keine weiteren Anforderungen gibt, kann man einen internationalen Index ohne persönliche Stildatei erstellen. Dazu muss man nur die gewünschte Sprachoption und die ausgewählten Module auf der *texindy*-Befehlszeile angeben. Bei Verwendung des *texindy*-Befehls gibt es drei Kategorien von Modulen:

Automatische Module Diese Module stellen Funktionalitäten bereit, die zu *MakeIndex* konform sind. Sie lassen sich bei Benutzung des Befehls *texindy* nicht abschalten. Wenn eine solche Funktionalität nicht erwünscht ist, muss der Befehl *xindy* direkt verwendet werden (siehe Abschnitt 11.3.4).

Standardmodule Einige Module werden standardmäßig aktiviert, sie können mit *texindy*-Optionen abgeschaltet werden.

Zusatzmodule Man kann ein oder mehrere Module mit der *xindy*-Option `-M` auswählen.

<i>xindy-Modul</i>	<i>Kategorie</i>	<i>Beschreibung</i>
<i>Sortierung</i>		
word-order	Standard	Leerzeichen kommen vor den Buchstaben des Alphabets: „Computer Modern“ wird vor „Computereingabe“ einsortiert; Wörter mit Präfixen stehen demnach am Anfang. Dieses Verhalten lässt sich mit der <code>texindy</code> -Option <code>-1</code> abschalten.
letter-order	Zusatz	Leerzeichen werden ignoriert: „Computer Modern“ wird nach „Computereingabe“ einsortiert. Dieses Verhalten lässt sich mit der <code>texindy</code> -Option <code>-1</code> einschalten.
keep-blanks	Zusatz	Führender und angehängter Leerraum (Leerzeichen und Tabs) wird nicht ignoriert und Leerraum innerhalb des Begriffs bleibt unverändert.
ignore-hyphen	Zusatz	Bindestriche werden ignoriert: „QED-Zeichen“ wird wie „QEDZeichen“ einsortiert.
ignore-punctuation	Zusatz	Alle Arten von Satzzeichen werden ignoriert: Bindestriche, Punkte, Kommas, Schrägstriche, Klammern usw.
numeric-sort	Auto	Nummern werden nicht wie Zeichen, sondern numerisch einsortiert: „V64“ erscheint vor „V128“.
<i>Seitennummern</i>		
page-ranges	Standard	Begriffe, die auf mehr als zwei aufeinander folgenden Seiten vorkommen, werden als Bereich angegeben: „1-4“. Dieses Verhalten lässt sich mit <code>-r</code> ausschalten.
ff-ranges	Zusatz	Verwendet die implizite „ff“-Notation für Bereiche bis zu drei Seiten und explizite Bereiche danach: 2f, 2ff, 2-6.
ff-ranges-only	Zusatz	Verwendet nur implizite Bereiche: 2f, 2ff.
book-order	Zusatz	Sortiert Seitennummern mit üblichem Buchnummerierungs-Schema korrekt – zuerst römische Zahlen, dann arabische Zahlen, dann andere: i, 1, A-1.
<i>Markup und Layout</i>		
tex	Auto	Stellt grundlegende T _E X-Konventionen bereit.
latex-loc-fmts	Auto	Bietet L ^A T _E X-Formatierungsbefehle zur Kapselung von Seitennummern.
latex	Auto	Verwendet L ^A T _E X-Konventionen sowohl in Rohindexeinträgen als auch im ausgegebenen Begriff, beinhaltet auch <code>tex</code> .
makeindex	Auto	Emuliert die Standard-Eingabesyntax und das Maskierungsverhalten von <i>MakeIndex</i> .
latin-lettergroups	Auto	Layout enthält einen einzelnen lateinischen Buchstaben als Überschrift über jedem Block von Wörtern mit gleichem Anfangsbuchstaben.
german-sty	Zusatz	Verwendet die Umlautauszeichnung der <code>babel</code> -Optionen <code>german</code> und <code>ngerman</code> .

Wenn zwei Einträge bis auf die ignorierten Zeichen identisch sind, werden diese Zeichen nicht ignoriert.

Tabelle 11.4: xindy-Standardmodule

Das automatische Modul `latex-loc-fmts` behandelt einen Unterschied zwischen *xindy* und *MakeIndex*. In *MakeIndex* kann man eine allgemeine Schreibweise zur Kapselung verwenden, um Seitenzahlen mit einem beliebigen Befehl einzuklammern (siehe Abschnitt 11.1.4). In *xindy* muss man für jeden Befehl eine eigene Referenzkategorie mit einer entsprechenden Markup-Definition bereitstellen (siehe Seite 700). Das Modul `latex-loc-fmts` stellt Definitionen für die üblichsten Kapselungen (`textbf` und `textit`) bereit.

11.3.4 Stildateien für individuelle Lösungen

Das *xindy*-Programm ist ein Werkzeug, das sich sehr gut konfigurieren lässt. Die gewählte Funktionalität wird in einer Stildatei angegeben. Das Programm `texindy` bietet für die meisten Zwecke einen bequemen Zugriff, indem es aus bestehenden Modulen eine virtuelle Stildatei erstellt. Wenn die angebotenen Funktionen erweitert, die Funktionalität geändert oder ein eigenes Indexierschema erstellt werden soll, muss *xindy* direkt verwendet und eine neue Stildatei geschrieben werden. Diese wird dann wie ein weiteres Modul behandelt. Die bestehenden *xindy*-Module können wiederverwendet werden.

Dieser Abschnitt demonstriert, wie man *xindy* mit eigenen Stildateien verwenden kann. Er beschreibt die grundlegenden Konzepte des *xindy*-Programms und zeigt Beispiele für typische Erweiterungen.

xindy-Stildateien können auch genutzt werden, um Stichwortverzeichnisse für Nicht- $\text{\LaTeX}$ -Dokumente zu erstellen (z.B. XML-Dokumente, andere Unicode-basierte Markup-Systeme). Die Funktionen, die für diesen Zweck erforderlich sind, werden in diesem Abschnitt nicht beschrieben, weil sie den Rahmen dieses Buches sprengen würden. Interessierte Leser finden weitere Informationen hierzu auf der Webseite von *xindy*.

Um das Prinzip der *xindy*-Stildateien zu verstehen, wird im Folgenden näher auf das Basismodell eingegangen, das *xindy* verwendet. Abbildung 11.7 auf der nächsten Seite zeigt die einzelnen Verarbeitungsschritte. Eine *xindy*-Stildatei enthält Regeln zur Vereinheitlichung, zur Sortierung, zur Erstellung von Referenzbereichen und zur Spezifikation von Markup. Mithilfe dieser Deklarationen definiert sie, wie der Rohindex aus der `.idx`-Datei in einen getaggten Index in der `.ind`-Datei umgewandelt wird.

*Das xindy-
Verarbeitungsmodell*

- Vereinheitlichungsregeln geben an, wie aus einem Rohschlüssel ein Sortierschlüssel berechnet wird. Im Rohindex können Rohschlüssel, die den gleichen Eintrag repräsentieren, unterschiedlich geschrieben sein. Dies kann dadurch entstehen, dass $\text{\LaTeX}$ je nach Kontext Befehle expandiert oder auch nicht. Oder der Autor verwendet unterschiedliche Schreibweisen, z.B. `ä`, `\a` oder `"a`. Durch die Regeln ist keine manuelle Eingabe zur Vereinheitlichung notwendig.

Dieser Aspekt ist auch bei der Verwendung von $\text{\LaTeX}$ -Befehlen für Logos von Bedeutung, da *xindy* alle Befehle ignoriert. Wenn man z.B. `\MF` für `METAFONT` nutzt und in den Index aufnimmt, wird der Eintrag nicht unter "M" einsortiert. Mit einer dokumentspezifischen Vereinheitlichungsregel kann man für die korrekte Einordnung sorgen, ohne dass man jedes Mal `\index{METAFONT@MF}` schreiben muss.

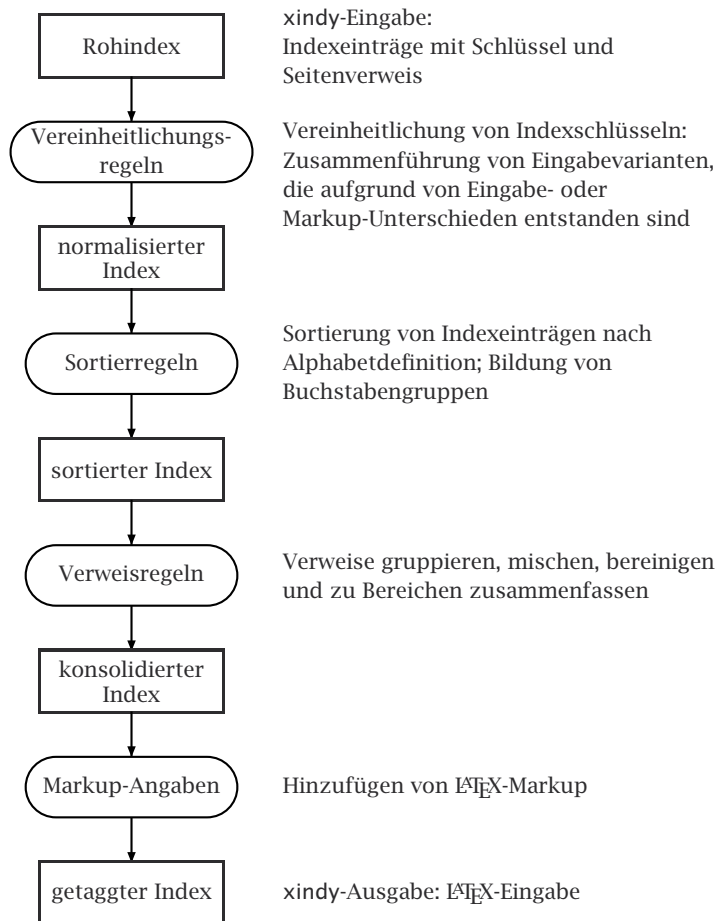


Abbildung 11.7: xindy-Verarbeitungsmodell

- Sortierregeln deklarieren Alphabete sowie die Reihenfolge innerhalb der Alphabete. Das Alphabet kann neben einzelnen Zeichen auch aus mehreren Zeichen bestehen, die zu Sortierungszwecken eine Einheit bilden (z.B. 11 in spanisch). Diese neuen Zeichen müssen in Bezug zu anderen Zeichen eingeordnet werden. Ein xindy-Sprachmodul besteht aus Alphabet-Deklarationen, Sortierregeln und Definitionen von Buchstabengruppen.
- Nach der Sortierung werden Einträge mit demselben Sortierschlüssel in einem konsolidierten Indexeintrag zusammengefasst. Er enthält mehrere Seitenangaben und einen Druckschlüssel. Von den Rohschlüsseln wird der erste, der im Dokument auftaucht, als Druckschlüssel ausgewählt. Sortieren, Gruppieren und Mischen der Seitenangaben, aus denen die zu druckenden Seitenangaben zusammengestellt werden, ist eine komplexe Aufgabe, die auf unterschiedlichste Weise beeinflusst werden kann.
- Markup-Angaben beschreiben, welche $\text{\LaTeX}$ -Befehle zu den konsolidierten Indexeinträgen hinzugefügt werden, wodurch ein getaggter Index entsteht, der als Eingabe für $\text{\LaTeX}$ verwendet werden kann.

Direkter Aufruf von xindy

Die xindy-Optionen sind jenen, die mit texindy verfügbar sind, sehr ähnlich. Stildateien werden wie jedes andere Modul angegeben.

```
xindy [-qvV] [-o ind] [-d magic] [-t log] [-L sprache]
      [-C codepage] [-M modul] [idx0 idx1 ...]
```

- o *ind* Verwendung von *ind* als Indexausgabedatei.
- M *modul* Verwendung der xindy-Stildatei *modul.xdy* zur Konfiguration der Verarbeitung.
- L *sprache* Verwendung von *sprache* als Sprachkonfiguration für die Festlegung der Wortreihenfolge.
- C *codepage* Verwendung von *codepage* als interne Basiskodierung zur Sortierung. Diese Option wird zur Feinsteuerung der Sprachmodule verwendet. Sie ist nur für nicht lateinische Skripte erforderlich.
- q Betrieb im stillen Modus.
- v Betrieb im ausführlichen Modus.
- V Ausgabe der Versionsnummer und Beendigung.
- d *magic* Erzeugung von Debug-Meldungen; *magic* legt fest, welche xindy-Komponente sie ausgeben soll.

Erstellen einer xindy-Stildatei

Eine xindy-Stildatei startet in der Regel, indem sie vordefinierte Module lädt, die einen Großteil der gewünschten Funktionen bereitstellen. Es ist zu beachten, dass auch jene Module, die in Tabelle 11.4 auf Seite 694 als *automatisch* (auto) aufgeführt sind, explizit benannt werden müssen. Danach kann man eigene Definitionen bereitstellen, welche die bereits geladenen Module erweitern oder überschreiben.

```
;;; xindy Beispiel-Stildatei

;; Klausel für alle vordefinierten Sprachen von texindy
(require "tex/inputenc/latin.xdy")

;; Vereinheitlichungsregeln zur Normalisierung des Markups
;; doppelter Backslash erforderlich, weil es eine regexp ist
(merge-rule "\\PS *" "PostScript")

;; automatische texindy-Module verwenden
(require "texindy.xdy")
;; Standard- und Zusatzmodule müssen angegeben werden
(require "page-ranges.xdy") (require "book-order.xdy")

;; Markup-Änderung:
;; Seiten-Listeneinträge durch LaTeX-Befehl abgrenzen
(markup-location-list :sep "\\page ")
```

Stildatei-Syntax

Das vorherige Beispiel einer xindy-Stildatei zeigte einige der Syntaxelemente, die verfügbar sind. Hier ein paar genauere Definitionen:

- Prinzipiell besteht eine Stildatei aus einer Liste von deklarativen Klauseln in Klammern, die mit einem Deklarationsnamen beginnt, gefolgt von mehreren Parametern.
- Ein Parameter kann entweder mit einer Zeichenkette oder einer Option beginnen. Eine Option hat ein Schlüsselwort, geschrieben als `:opt`, und kann einen Parameter haben, üblicherweise eine Zeichenkette, aber auch eine Zahl oder einen festen Wert wie `none`. Wie der Name andeutet, sind Optionen nicht zwingend erforderlich. Welche Optionen gültig sind, hängt von der Funktion ab. Ein Parameter kann auch aus einer Liste von Parametern in Klammern bestehen, wie einige der nachfolgenden Beispiele zeigen.
- Kommentare beginnen mit einem Semikolon und enden am Zeilenende. Die Beispiele zeigen typische Verwendungen für den Einsatz unterschiedlich vieler Semikolons: eines für Kommentare im Absatz (nach xindy-Klauseln), zwei für Absatzkommentare vor Programmcode und drei für Kommentare mit „Abschnittsüberschriften“ für die Stildateien. Dies ist jedoch nur eine Konvention – in allen Fällen beginnt der Kommentar nach dem ersten Semikolon.
- Zeichenketten werden in Anführungszeichen angegeben. Zeilenumbrüche sind erlaubt. Innerhalb einer Zeichenkette gilt die Tilde als Maskierungszeichen, die dem folgenden Buchstaben eine besondere Bedeutung zuweist. `~n` gibt beispielsweise einen Zeilenumbruch an.

Vereinheitlichungs- und Sortierregeln

Vereinheitlichungsregeln helfen, Rohindexeinträge anzugleichen, bevor diese sortiert und gruppiert werden. Sie können verwendet werden, um unterschiedliche Notationen zu vereinheitlichen und den Eintrag von den Markup-Angaben zu trennen, die für die Sortierung irrelevant sind. Wenn verschiedene Indexeinträge zusammengeführt werden, erscheinen sie als ein Eintrag und haben folglich dasselbe Aussehen in der Ausgabe. Genauer gesagt sehen alle so aus wie der erste Eintrag, der im Dokument vorkommt. Man kann übrigens nur einzelne Wörter vereinheitlichen, nicht ganze Ausdrücke.

Eine Vereinheitlichungsregel hat zwei Parameter. Sie deklariert, dass alle Vorkommen des ersten Parameters innerhalb eines Wortes durch den zweiten Parameter ersetzt werden. Innerhalb des zweiten Parameters können die virtuellen Zeichen `~b` und `~e` verwendet werden: `~b` wird vor allen anderen Zeichen einsortiert, während `~e` nach allen Zeichen kommt. Diese beiden virtuellen Zeichen werden nicht ausgegeben, weil Vereinheitlichungsregeln dazu verwendet werden, den Sortierschlüssel aus dem Rohschlüssel zu erzeugen – und Sortierschlüssel sind interne Bezeichner der Einträge.

Indexeinträge vereinheitlichen

Zum Beispiel können im Stichwortverzeichnis eines Atlases Orte, die *St* im Namen haben, auch als *Sankt* geschrieben werden. Diese unterschiedlichen Schreibweisen sollten trotzdem in einem Indexeintrag vereinheitlicht werden. Mit anderen Worten: Die Indexierung von *St Augustin* und *Sankt*

Augustin sollte nur einen Indexeintrag ergeben.

```
( merge-rule "St" "Sankt" )
```

In einer Vereinheitlichungsregel kann man auch ein Muster (regulärer Ausdruck) und eine Ersatzzeichenkette angeben. Sogenannte *extended regexps* stellen die Standardeinstellung dar, sie sind im Standard POSIX 1003.2 definiert. Auf UNIX-Systemen findet man ihre Beschreibung in der Manualseite von *egrep*. Man kann auch *reguläre Basisausdrücke* mit der Option `:bregexp` in der Vereinheitlichungsregel verwenden. Die Ersatzzeichenkette kann sich auf Unterausdrücke beziehen. Dies führt zwar zu leistungsfähigen Definitionen, die aber häufig die Erstellung und die Fehlersuche erschweren. Zu beachten ist außerdem, dass die Verwendung von regulären Ausdrücken die Verarbeitung verlangsamt. Um XML-Tags ohne spitze Klammern zu indizieren, kann man Folgendes schreiben:

Vereinheitlichung mit regulären Ausdrücken

```
( merge-rule "<(.*)>" "\1" )
```

Dies bewirkt, dass `\index{<HTML>}` und `\index{HTML}` in einem Eintrag zusammengefasst werden, was möglicherweise nicht der gewünschte Effekt ist. Um beide separat, aber nacheinander aufzuführen, kann man `<HTML>` in `HTML~e` umwandeln, und zwar folgendermaßen:

```
( merge-rule "<(.*)>" "\1~e" )
```

Sortierregeln geben an, wie Zeichen oder Zeichenfolgen sortiert werden (d.h. an welcher Stelle sie im Alphabet platziert werden sollen). Eine Sortierregel besteht aus zwei Zeichenketten. Die erste Zeichenkette wird wie die zweite sortiert. In der zweiten kann mithilfe von `~b` und `~e` die Sortierreihenfolge angegeben werden (Erläuterung siehe oben).

Buchstabengruppen

Das xindy-Programm überprüft jede Buchstabengruppe, um festzustellen, ob sie mit dem Präfix eines Sortierschlüssels in einem Eintrag übereinstimmt. Die längste Übereinstimmung weist den Indexeintrag dieser Buchstabengruppe zu. Wird keine Übereinstimmung gefunden, dann wird der Indexeintrag in die Gruppe `default` verschoben.

Die folgenden Definitionen fügen alle Einträge mit den angegebenen Präfixen in dieselbe Buchstabengruppe *ABC* ein:

Buchstabengruppen kombinieren

```
( define-letter-group "ABC" :prefixes ("a") )
( define-letter-group "ABC" :prefixes ("b") )
( define-letter-group "ABC" :prefixes ("c") )
```

Bei Stichwortverzeichnissen, die etwas unausgewogen sind, z.B. im Buchstaben *X*, empfiehlt es sich manchmal, eine zusätzliche Buchstabengruppe einzufügen. Die folgende Definition fügt eine zusätzliche Gruppe namens *xsl*

Zusätzliche Buchstabengruppen

ein, die alle Einträge enthält, die mit `xsl:` beginnen. Diese Einträge werden vor allen anderen Einträgen einsortiert, die mit `x` beginnen.

```
( define-letter-group "xsl" :before "x" :prefixes ("xsl:") )
```

Referenzbereiche

Die Liste der Verweise hinter einem Indexeintrag kann aus mehreren Arten von Referenzen bestehen, die in einer unter Umständen dokumentabhängigen Reihenfolge angeordnet werden müssen. Möglicherweise kommen römische Zahlen vor arabischen und diese wiederum vor Kombinationen aus Buchstaben und arabischen Zahlen. Dieses Schema tritt üblicherweise in einem Buch auf, das ein Vorwort, Haupttext und Anhänge hat. Auch innerhalb einer Referenzart werden die Verweise geordnet. Die Verweise können in Bereichen wie *10-15* oder *5ff* zusammengefasst werden. Wie man sieht, bietet xindy vielfältige Möglichkeiten zur Sortierung und Bereichsbildung.

*Minimale
Seitenbereichslänge*

Um beispielsweise die minimale Länge für automatisch generierte Seitenbereiche zu ändern, muss man lediglich die Definition für Seitenbereiche ändern:

```
( define-location-class "pages"
  ("arabic-numbers")
  :min-range-length 4 )
```

*Seitenbereiche
unterdrücken*

Um für römische Zahlen die Generierung von Seitenbereichen zu deaktivieren, muss die Option `:min-range-length` folgendermaßen geändert werden:

```
( define-location-class "pages"
  ("roman-numerals-lowercase")
  :min-range-length none )
```

Komplexe Referenzen

Wenn der Rohindex Verweise mit nicht numerischen Elementen und eine ungewöhnliche Syntax enthält (z.B. `Pasta: :II.4`), muss man ein spezielles Alphabet definieren, um xindy mitzuteilen, wie es die Referenzen sortieren soll. Damit kann man dann eine Referenzart definieren, welche die Syntax für Verweise, einschließlich Trennzeichen beschreibt:

```
( define-alphabet "my-chapters"
  ("Vorspeisen" "Pasta" "Fleisch" "Dessert") )
( define-location-class "my-index"
  ("my-chapters" :sep "::  
"roman-numerals-uppercase" :sep "."  
"arabic-numbers") )
```

Referenzen mit spezieller Formatierung

Das xindy-Programm nutzt ein sehr flexibles Verfahren zur Formatierung, Sortierung und Gruppierung von Ortsangaben mit besonderer Bedeutung. Dazu markiert man im Dokument Indexeinträge, die eine spezielle Formatierung erhalten sollen, wie `\index{schlüsselwort|definition}`. In xindy definiert man dann ein Attribut mit entsprechenden Markup-Angaben.

Man kann auch festlegen, wie verschiedene Referenzarten interagieren sollen: ob sie vermischt oder separat aufgeführt werden, zusammengefasste Bereiche zwischen ihnen zugelassen sind oder Einträge, die Teil eines Bereiches sind, unterdrückt werden oder nicht.

Die folgenden Beispiele zeigen verschiedene Variationen, wie Verweise mit spezieller Formatierung verarbeitet werden können.

Input:	1 4 5 6 7 7 9 10
Beispiel 1: mischen, zusammenfassen, unterdrücken	1 4-7 9 10
Beispiel 2: mischen, zusammenfassen	1 4-7 7 9 10
Beispiel 3: nicht mischen, zuerst Definitionen	7 9 1 4-7 10

Beispiel 1: Seitenangaben mischen, zusammenfassen und einzelne Seiten unterdrücken.

```
;; Definition und Standard mischen
(define-attributes (("definition" "default"))))

;; zusammenfassende Bereiche zulassen, innerhalb eines Bereiches
;; Verweise auf Definitionen unterdrücken
(merge-to "definition" "default" :drop)

;; Markup definieren
(markup-location :attr "definition"
                 :open "\textbf{" :close "}" )
```

Beispiel 2: Seitenangaben mischen und zusammenfassen.

```
;; Definition und Standard vermischen
(define-attributes (("definition" "default"))))

;; zusammenfassende Bereiche zulassen, Verweise auf Definition
;; innerhalb eines Bereiches erhalten
(merge-to "definition" "default")

;; Markup definieren
(markup-location :attr "definition"
                 :open "\textbf{" :close "}" )
```

Beispiel 3: Seitenangaben nicht mischen, Definitionen zuerst auflisten.

```
;; Definition und Standard separat, Definitionen zuerst
(define-attributes (("definition") ("default"))))

;; Markup definieren
(markup-location :attr "definition"
                 :open "\textbf{" :close "}" )
```

Zu beachten ist, dass `define-attributes` einen Parameter in Klammern besitzt. Er besteht entweder aus einer Liste von Attributnamen, die in Klammern eingeschlossen sind, oder einer Liste von Zeichenketten, die jeweils

einzelnen eingeklammert werden. Alle Attribute, die sich innerhalb eines Klammerspaares befinden, werden gemischt. Bei mehreren Attributen würde ein Ausdruck wie

```
(( "definition" "important" ) ( "default" ))
```

angeben, dass Definitionen mit der Gruppe der wichtigen Verweise gemischt werden können, aber nicht mit Standardverweisen.

11.4 Beeinflussung des Index mit L^AT_EX-Funktionen

Dieser Abschnitt geht auf L^AT_EX-Funktionen ein, welche die Indexerstellung unterstützen. Er zeigt Möglichkeiten auf, um das Layout des Stichwortverzeichnis zu verändern und mehrere Stichwortverzeichnisse zu erzeugen.

11.4.1 Ändern des Layouts

Man kann die Umgebung `theindex` neu definieren, die standardmäßig zum Drucken des Index verwendet wird. Das Layout der `theindex`-Umgebung und die Definition der Befehle `\item`, `\subitem` und `\subsubitem` werden in den Klassendateien `article`, `book` und `report` festgelegt. In der `book`-Klasse findet man folgende Definitionen:

```
\newenvironment{theindex}
{
  \@restonecoltrue\if@twocolumn\@restonecolfalse\fi
  \columnseprule \z@ \columnsep 35\p@
  \twocolumn[\@makeschapterhead{\indexname}]%
  \@mkboth{\MakeUppercase\indexname}{\MakeUppercase\indexname}%
  \thispagestyle{plain}\parindent\z@
  \parskip \z@\@plus .3\p@\relax \let\item\@idxitem
  {\if@restonecol\onecolumn\else\clearpage\fi}
\newcommand\@idxitem {\par\hangindent 40\p@}
\newcommand\subitem {\par\hangindent 40\p@ \hspace*{20\p@}}
\newcommand\subsubitem{\par\hangindent 40\p@ \hspace*{30\p@}}
```

Auch wenn dieser Programmcode in einer recht niedrigen internen Sprache programmiert wurde, ist es wahrscheinlich nicht schwer zu entziffern, was er bewirkt. Zunächst prüft er auf zweispaltigen Modus und speichert das Ergebnis. Anschließend setzt er einige Abstandsparameter, setzt den Seitenstil wieder auf `plain` und ruft `\twocolumn` auf. Schließlich ändert er `\item`, um `\@idxitem` auszuführen. Dieser erzeugt einen Absatz mit hängendem Einzug von 40 Punkten. Eine flexiblere Neuimplementierung (die den Befehl `ifthen` verwendet) könnte folgendermaßen aussehen:

```
\renewenvironment{theindex}
{
  {\ifthenelse{\boolean{@twocolumn}}%
    {\setboolean{@restonecol}{false}}%
    {\setboolean{@restonecol}{true}}%
  \setlength\columnseprule{0pt}%
  \setlength\columnsep{35pt}%
}
```

```

\twocolumn[\chapter*{\indexname}]%
\markboth{\MakeUppercase\indexname}%
          {\MakeUppercase\indexname}%
\setlength\parindent{0pt}%
\setlength\parskip{0pt plus 0.3pt}%
\thispagestyle{plain}%
\let\item\@idxitem }
{\ifthenelse{\boolean{@restonecol}}{\onecolumn}{\clearpage}}

```

Durch Anpassung dieser Definition sind kleinere Änderungen möglich, wie etwa die Änderung des Seitenstils oder der Spaltentrennung.

Ebenso kann der Index anstatt aus zwei aus drei Spalten bestehen. Dafür lässt sich das multicol-Paket und die multicols-Umgebung verwenden:

```

\renewenvironment{theindex}{%
  \begin{multicols}{3}[\chapter*{\indexname}] [10\baselineskip]%
  \addcontentsline{toc}{chapter}{\indexname}%
  \setlength\parindent{0pt}\pagestyle{plain}\let\item\@idxitem
\end{multicols}}

```

Auf der aktuellen Seite werden mindestens 10 Zeilen Freiraum benötigt; ansonsten beginnt der Index auf einer neuen Seite. Neben der Erstellung eines Titels an erster Position wird im Inhaltsverzeichnis (.toc) die Überschrift als „Kapitel“ eingegeben und der Seitenstil auf plain geändert. Danach wird der \item-Befehl neu definiert, um die Indexeinträge handhaben zu können (siehe oben). Die Einträge selbst werden mithilfe der multicols-Umgebung in drei Spalten gesetzt.

11.4.2 showidx, repeatindex, tocibind, indxcite – Kleine Helfer

Es gibt verschiedene kleine nützliche L^AT_EX-Pakete, welche die Indexerstellung unterstützen. Eine Auswahl wird in diesem Abschnitt vorgestellt, weitere sind im Online-Katalog [171] aufgeführt.

Das Paket showidx (von Leslie Lamport) kann helfen, die Einträge im Index zu verbessern und mögliche Probleme zu lokalisieren. Es gibt alle \index-Befehle im Rand der ausgedruckten Seite aus. Abbildung 11.3 auf Seite 676 zeigt die Benutzung.

Indexeinträge im Rand darstellen

Das Paket repeatindex (von Harald Harders) wiederholt die oberste Ebene eines Indexeintrags, wenn die Seite oder die Spalte innerhalb einer Liste von Untereinträgen umbrochen wird. Dies hilft dem Leser, den Untereintrag dem zugehörigen Haupteintrag zuzuordnen.

Seitenumbrüche intelligent steuern

Das Paket tocibind (von Peter Wilson) kann verwendet werden, um das Inhaltsverzeichnis, die Bibliographie und den Index im *Inhaltsverzeichnis* hinzuzufügen. Für weitere Informationen zu diesem Paket siehe Seite 51.

Auflistung im Inhaltsverzeichnis

Das Paket indxcite (von James Ashton) erzeugt automatisch ein Autorenverzeichnis aus den Referenzen, die mit B^BT_EX eingetragen wurden. Diese Funktionalität ist auch mit den Bibliographiepaketen natbib und jurabib verfügbar, die beide ausführlich in Kapitel 12 beschrieben werden.

Automatisches Autorenverzeichnis

11.4.3 index – Erzeugen mehrerer Indexe

Das `index`-Paket (geschrieben von David Jones und verteilt als Teil des `camel-Paketes`) verbessert die Indexierung durch $\text{\LaTeX}$ in mehreren Bereichen:

- Mehrfachindexe werden unterstützt.
- Zur Erstellung der Rohindex-Dateien (wie die Standard-`.idx`-Datei) wird ein zweistufiger Prozess verwendet, ähnlich dem, der zur Erstellung der `.toc`-Datei verwendet wird. Zunächst werden die Indexeinträge in die `.aux`-Datei geschrieben und dann am Ende des Laufs in die `.idx`-Datei kopiert. Bei diesem Ansatz geht auch bei großen Dokumenten, die aus mehreren Dateien bestehen (die mit dem `\include`-Befehl eingebunden wurden), der Index nicht mehr verloren, wenn nur ein Teil des Dokumentes mit `\includeonly` formatiert wird. Dieses Verfahren erschwert jedoch die Erstellung eines Kapitelindex.
- Für den `\index`-Befehl wird eine Sternform eingeführt. Das Argument des Befehls wird nicht nur in den Index eingefügt, sondern auch im laufenden Text gesetzt.
- Um die Eingabe zu erleichtern, aktiviert der Befehl `\shortindexingon` eine Kurzschreibweise. Damit kann man anstelle von `\index{foo}` die Schreibweise `^{\text{foo}}` und anstelle von `\index*{\text{foo}}` die Schreibweise `_{\text{foo}}` verwenden. Diese Kurzschreibweisen werden mit dem Befehl `\shortindexingoff` ausgeschaltet. Da der Unterstrich und der Zirkumflex im Formelmodus eine besondere Bedeutung haben, können diese Kurzschreibweisen dort nicht verwendet werden.
- Das Paket umfasst die Funktionalität des `showidx`-Paketes. Mit dem Befehl `\proofmodetrue` können Indexeinträge in den Rand geschrieben werden. Stil und Größe der Schrift im Rand können mit dem Befehl `\indexproofstyle{stil}` angepasst werden. Stilinformationen für diese Randnotizen werden in seinem Argument eingegeben (zum Beispiel `\indexproofstyle{\footnotesize\itshape}`).
- Bei Verwendung des `index`-Paketes wird das Argument nie expandiert. In Standard- $\text{\LaTeX}$ wird bei Verwendung von `\index{\command}` die Expansion von `\command` manchmal in die `.idx`-Datei geschrieben (siehe Abschnitt 11.2.5 auf Seite 687). Mit dem `index`-Paket wird `\command` selbst stets in die `.idx`-Datei geschrieben. Das ist in den meisten Fällen hilfreich, aber für Makroentwickler kann es zur Stolperfalle werden. In Abschnitt 11.1.7 wird empfohlen, dass man Befehle definiert, die automatisch Indexeinträge hinzufügen. Diese Befehle erwarten häufig, dass `\index` ihr Argument expandiert und funktionieren deshalb möglicherweise nicht, wenn das `index`-Paket verwendet wird. Das Ergebnis der automatischen Indexierung muss auf jeden Fall überprüft werden – dies ist allerdings in jedem Fall empfehlenswert.

Ein neuer Index kann mit dem Befehl `\newindex` deklariert werden. Der Befehl `\renewindex`, der eine identische Syntax hat, wird zur Neudefinition von bestehenden Indexen verwendet.

```
\newindex{tag}{roh-erw}{verarb-erw}{indextitel}
```

Das erste Argument *tag* ist ein Kurzbezeichner, das als Verweis auf den Index verwendet wird. Insbesondere die Befehle `\index` und `\printindex` werden neu definiert, so dass sie ein optionales Argument annehmen können – nämlich das Tag des Index, auf den sie verweisen. Wenn dieses optionale Argument fehlt, wird der Index mit dem Tag „default“ verwendet. Dies entspricht dem normalen Index. Das zweite Argument *roh-erw* gibt die Erweiterung der Rohindex-Datei an, in welche L^AT_EX die unverarbeiteten Einträge für diesen Index schreiben soll (beim Standardindex ist dies `.idx`). Das dritte Argument *verarb-erw* gibt die Erweiterung der Indexdatei an, in welcher L^AT_EX den verarbeiteten Index erwartet (beim Standardindex ist dies `.ind`). Das vierte Argument *indextitel* gibt den Titel an, den L^AT_EX am Anfang des Index ausgibt.

Das nächste Beispiel zeigt die Einstellung, die zur Erstellung dieses Buches verwendet wurde. Die Präambel enthielt folgende Einstellung:

```
\RequirePackage{index}
\proofmodetrue           % zum Korrekturlesen der Indexeinträge
\newindex{xauthor}{adx}{and}{Personen}
\newindex{xcmds}{cdx}{cnd}{Index der Befehle und Konzepte}
```

Im hinteren Teil des Buches wurde der Index dann mit den Befehlen `\printindex[xcmds]` und `\printindex[xauthor]` ausgegeben. Für jede generierte Rohindexdatei (z.B. `lb2.adx` für die Liste der Autoren) wurden die Einträge mit *MakeIndex* bearbeitet, um die entsprechende formatierte Indexdatei für L^AT_EX zu erzeugen:

```
makeindex -o lb2.and -t lb2.alg lb2.adx
```

All diese Hilfsmittel tragen dazu bei, die richtigen Seitennummern zu ermitteln, aber die eigentliche Schwierigkeit bleibt bestehen: nützliche Indexeinträge für den Leser auszuwählen. Dieses Problem muss der Anwender bisher noch selbst lösen (wenn er Glück hat, mit Hilfe).

Auch der Index dieses Buches wurde von einem professionellen Indexgestalter zusammengestellt, nämlich Richard Evans von Infodex Indexing Services in Raleigh, North Carolina. Richard arbeitete eng mit Frank zusammen, um einen umfassenden Index zu erzeugen, der den Lesern nicht nur hilft, Namen von Objekten (Paketen, Programmen, Befehlen usw.) zu finden, sondern auch Aufgaben, Konzepte und Ideen, die in dem Buch beschrieben wurden. Doch hierzu seine eigenen Worte (aus der Infodex FAQ unter <http://www.mindspring.com/~infodex>):

Frage: Warum braucht man einen Indexexperten? Kann der Computer nicht den Index erzeugen?

Antwort: Genau so gut, wie ein Textverarbeitungsprogramm das Buch schreiben kann. Indexe sind kreative Schöpfungen, die den Intellekt und die Analyse des Menschen erfordern.

L^AT_EX kann das Index-Markup verarbeiten, aber nur ein menschlicher Indexexperte kann entscheiden, was ausgezeichnet werden soll. Ein großes Dankeschön an Richard für seine hervorragende Arbeit.

KAPITEL 12

Quellenverweise

12.1 Einleitung	707
12.2 Das numerische Schema.	715
12.3 Das Autor-Jahr-Schema	722
12.4 Das Autor-Nummer-Schema	737
12.5 Das Kurztitelschema	741
12.6 Mehrere Bibliographien in einem Dokument	771

12.1 Einleitung

Quellenverweise sind Querverweise auf bibliographische Daten, wie z.B. auf Veröffentlichungen mit weiteren Informationen zu einem Thema, sowie Quellenangaben zu angeführten Zitaten. Man muss sicherlich nicht alles durch Quellen belegen, aber Hintergrundinformationen zu strittigen Fragen und Literaturhinweise zu verwendetem Material sollten durchaus angegeben werden.

Es gibt viele verschiedene Möglichkeiten, Bibliographien und Quellenverzeichnisse zusammenzustellen. Nötigenfalls lassen sie sich manuell erstellen; normalerweise werden sie jedoch automatisch aus einer Datenbank mit bibliographischen Informationen erzeugt. Dieses Kapitel stellt einige der zahlreichen Darstellungsformen bibliographischer Quellen vor und beleuchtet verschiedene Traditionen, in einem Dokument auf solche Quellen zu verweisen.

Zu Beginn erfolgt eine kurze Einführung in die wichtigsten, heute üblichen Zitierschemata. Darauf folgt eine Beschreibung des Standard-Markups für Bibliographien in $\LaTeX$ und der Schnittstelle zu dem Programm $\BibTeX$, mit dessen Hilfe sich solche Listen automatisch aus einer (passend vorbereiteten) Datenbank erzeugen lassen. In Kapitel 13 folgen anschließend noch nähere Einzelheiten zu $\BibTeX$. Im vorliegenden Kapitel liegt das Augenmerk zunächst darauf, wie man Literaturverzeichnisse mit $\BibTeX$ erzeugen kann.

Auf Grundlage dieses Wissens beschäftigt sich das Kapitel dann intensiv mit der Frage, auf welche Weise $\LaTeX$ die verschiedenen Zitierschemata unterstützt. Als die erste Auflage dieses Buches geschrieben wurde, unterstützte $\LaTeX$ im Wesentlichen das „numerische“ System für Quellenverweise. Ein

Jahrzehnt später sieht die Lage völlig anders aus: Heute werden die meisten wichtigen Zitierschemata durch Erweiterungspakete gut unterstützt.

Das Kapitel schließt mit einigen Paketen, mit deren Hilfe man in einem Dokument mehrere Literaturverzeichnisse führen kann. Bei manuell erstellten Verzeichnissen ist das nicht weiter schwierig. Es wird jedoch zu einer Herausforderung, wenn man gleichzeitig noch Daten mit $\text{\LaTeX}$ austauschen möchte.

12.1.1 Verweisschemata für Bibliographien

Es gibt vier gängige Methoden um auf Quellen zu verweisen: per „Kurtztitel“, per „Autor und Jahr“, per „Autor und Nummer“ sowie „rein numerisch“. Die erste Form findet sich vornehmlich in geisteswissenschaftlicher Literatur, die zweite wird hauptsächlich in natur- und sozialwissenschaftlichen Werken verwendet. Die anderen beiden Schemata sind weniger verbreitet, obgleich die letzte in der $\text{\LaTeX}$ -Welt recht üblich ist, da sie von Leslie Lamport aktiv gefördert wurde und ursprünglich die einzige in $\text{\LaTeX}$ unterstützte Verweisform war. Abseits von $\text{\LaTeX}$ ist ihre Variante „numerisch nach erstem Vorkommen“ ebenfalls recht beliebt.

Das Kurtztitelschema

Im Kurtztitelverfahren wird der Verweis auf eine Quelle direkt im Text aufgeführt, entweder eingebettet oder als Fußnote, oftmals in der Form „Hart, *Hart's Rules*, S. 52“. Im Kontext der Veröffentlichung kann auch die Form „Goossens et al., *LGC*“ auftauchen, wenn sich eine Abkürzung des Titels eingebürgert hat. Es gibt viele Abwandlungen. So kann z.B. ein Werk, wenn es das erste Mal vorkommt, mit sehr detaillierten Daten zur Quelle angeführt werden und spätere Verweise benutzen dann eine kürzere Form, etwa den Namen des Autors und einen Kurtztitel oder das Erscheinungsjahr. Bei mehreren, direkt aufeinander folgenden Verweisen auf dieselbe Quelle findet sich auch *a.a.O.* oder *Ebd.* (bzw. *Ibid.* in englischen Texten), anstelle eines wiederholten Verweises. Das jurabib-Paket bietet ein Kurtztitelssystem mit verschiedensten Anpassungsmöglichkeiten (siehe Abschnitt 12.5.1).

Da in diesem System normalerweise beim ersten Vorkommen einer Quelle ein ausführlicher Literaturhinweis erfolgt, kann man auf ein Quellenverzeichnis oder eine Bibliographie, die alle zitierten Werke an einer Stelle auflisten, verzichten.

Das Autor-Jahr-Schema

Im Autor-Jahr-System (häufig auch als Harvard-System bezeichnet, nach einer seiner bekannteren setzerischen Varianten) erfolgen die Quellenangaben ebenfalls direkt im Text. In diesem Fall werden jedoch der Name des Autors (oder der Autoren) und das Erscheinungsjahr aufgeführt. Der ausführliche Literaturhinweis steht in einem Literaturverzeichnis oder einer Bibliographie. Wenn ein Autor in einem Jahr mehr als ein Werk veröffentlicht hat, wird das Jahr durch einen Kleinbuchstaben ergänzt (z.B. 2001a, 2001b).

Im Laufe der Jahre gab es viele Versuche, Autor-Jahr-Quellenverweise in $\text{\LaTeX}$ zu unterstützen. Mit dem natbib-Paket (das in Abschnitt 12.3.2 näher beschrieben wird) liegt nun eine sehr flexible und umfassende Lösung vor.

In allen Zitierschemata, die mit Autorennamen arbeiten, wird auf Werke von drei oder mehr Autoren normalerweise mit dem Namen des ersten gefolgt von *et al* verwiesen. Gerade im Autor-Jahr-Schema kann dieses Verfahren zu mehrdeutigen Verweisen führen, wenn verschiedene Autorengruppen um den

gleichen Hauptautor im gleichen Jahr ein Werk veröffentlicht haben. Dieses Problem¹ wird im nächsten Beispiel deutlich.

Einträge mit mehreren Autoren können problematisch sein, z.B. (Goossens et al. 1997) und (Goossens et al. 1997) oder noch schlimmer (Goossens et al. 1997; Goossens et al. 1997).

Literatur

Goossens, M., S. Rahtz, and F. Mittelbach (1997). *The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript*. Tools and Techniques for Computer Typesetting. Reading, MA, USA: Addison-Wesley Longman.

Goossens, M., B. User, J. Doe, et al. (1997). Mehrdeutige Quellenverweise. Eingereicht bei IBM Journal of Research and Development.

Bsp.
12-1-1

```
\usepackage{chicago}
\bibliographystyle{chicago}
```

Einträge mit mehreren Autoren können problematisch sein, z.B. `\shortcite{LGC97}` und `\shortcite{test97}` oder noch schlimmer `\shortcite{LGC97,test97}`.

```
\bibliography{tex}
```

Die Bibliographie des vorigen Beispiels wurde aus der B²T_EX-Beispieldatenbank `tex.bib` erzeugt, die in Abbildung 12.2 auf Seite 714 zu sehen ist. Sie dient als Grundlage für die meisten Beispiele in diesem Kapitel. Hier wurde der B²T_EX-Stil `chicago` verwendet, welcher Bibliographien und Verweise entsprechend der Vorschläge des Werkes *The Chicago Manual of Style* [38] gestalten soll.

Mehrdeutige Quellenangaben lassen sich in diesem Fall beheben, indem man alle Autoren anführt. Dieser Lösungsansatz erzeugt jedoch sehr lange Verweise und ist ab einer bestimmten Anzahl von Autoren schlichtweg nicht mehr praktikabel. Stattdessen kann man auch Kleinbuchstaben an die Jahreszahl anhängen, auch wenn die Verweise sich eigentlich auf verschiedene Autorengruppen beziehen. Diese Strategie wird z.B. in [29] befürwortet. Ein manuell erstelltes Literaturverzeichnis (vgl. Abschnitt 12.1.2) lässt sich leicht in dieser Weise abwandeln. Für B²T_EX benötigt man jedoch eine B²T_EX-Stildatei, die solche Fälle erkennt und automatisch die passenden Daten erzeugt. Der Stil `chicago` eignet sich hier z.B. nicht, aber alle B²T_EX-Stile, die mit `makebst` (siehe Abschnitt 13.5.2) erstellt wurden, verfügen über diese Funktion:

Einträge mit mehreren Autoren könnten problematisch sein, wie z.B. Goossens et al. [1997a] und Goossens et al. [1997b] oder sogar Goossens et al. [1997a,b]. Oder auch nicht.

Literatur

M. Goossens, S. Rahtz, and F. Mittelbach. *The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997a. ISBN 0-201-85469-4.

M. Goossens, B. User, J. Doe, et al. Mehrdeutige Quellenverweise. Eingereicht bei IBM J. Res. Dev., 1997b.

Bsp.
12-1-2

```
\usepackage{natbib}
\bibliographystyle{abbrvnat}
```

Einträge mit mehreren Autoren könnten problematisch sein, wie z.B. `\cite{LGC97}` und `\cite{test97}` oder sogar `\cite{LGC97,test97}`. Oder auch nicht.

```
\bibliography{tex}
```

¹Weitere Probleme mit diesem und vielen anderen B²T_EX-Stilen sind die Verwendung fest verdrahteter sprachspezifischer Texte, wie etwa „and“, und die Konvertierung von Titeln in Kleinbuchstaben, was in diesem und einigen weiteren Beispielen zu sichtbar wird. Anpassungsmöglichkeiten werden in diesem und dem nächsten Kapitel diskutiert.

Das Autor-Nummer-Schema

Im Autor-Nummer-System werden die Verweise aus dem Namen des Autors (oder der Autoren) und einer Nummer, die normalerweise in runden oder eckigen Klammern steht, zusammengesetzt. Die Nummer gibt an, welches der Werke des Autors gerade zitiert wird. Im zugehörigen Literaturverzeichnis werden alle Veröffentlichungen je Autor (oder Autorengruppe) durchnummeriert. In der $\text{\LaTeX}$ -Welt ist dieses System unüblich, da es sich nur schwer manuell erstellen lässt. Soweit den Autoren dieses Buches bekannt ist, wird dieses Schema zur Zeit nicht von $\text{\BibTeX}$ unterstützt, was sich jedoch zukünftig noch ändern kann. Eine Variante dieses Schemas besteht darin, alle Publikationen fortlaufend zu nummerieren. Dazu gibt es auch passende $\text{\BibTeX}$ -Stile.

Das numerische Schema

Im numerischen System, schließlich, werden alle Werke in der Bibliographie fortlaufend nummeriert. Quellenverweise im Text verweisen auf diese Nummern, die normalerweise in eckigen oder runden Klammern stehen. Manchmal werden stattdessen auch hochgestellte Nummern verwendet. In einer leichten Abwandlung, die als „Alpha“-Stil bekannt ist, bestehen die Verweise aus dem Namen des Autors und dem Erscheinungsjahr des Werkes. Das bibliographische Label und der Quellenverweis würden dann z.B. „[Knu86]“ lauten.

Ein Argument gegen dieses System – das z.B. in *The Chicago Manual of Style* [38] angeführt wird – sind die hohen Publikationskosten, da das nachträgliche Hinzufügen oder Löschen eines Quellenverweises eine neue Nummerierung nach sich ziehen kann, so dass viele Seiten des Manuskriptes auf kostenintensive (und fehleranfällige) Weise geändert werden müssen. Durch automatisch erzeugte Querverweise, wie $\text{\LaTeX}$ sie erzeugt, verliert dieses Argument teilweise seine Gültigkeit. Tatsächlich ist das nummerierte System in $\text{\LaTeX}$ die voreingestellte Variante.

Numerisch nach erstem Vorkommen

Eine recht beliebte Form des numerischen Systems nummeriert die Werke fortlaufend nach ihrem ersten Vorkommen im Text (und führt sie auch in dieser Reihenfolge in der Bibliographie auf). Sie lässt sich mit $\text{\LaTeX}$ ziemlich leicht erzielen. Die nächsten beiden Abschnitte und Abschnitt 12.2.3 erklären, wie man Quellenverweise im Inhaltsverzeichnis vermeidet, damit sie die erwartete Reihenfolge nicht durcheinander bringen.

12.1.2 Markup für Quellenverweise und Bibliographie

Die $\text{\LaTeX}$ -Standardumgebung zum Erzeugen eines Literaturverzeichnisses oder einer Bibliographie heißt `thebibliography`. Gemäß ihrer Voreinstellung generiert sie automatisch eine passende Überschrift und richtet eine vertikale Liste ein, in der jede Publikation als einzelner Eintrag erscheint.

```
\begin{thebibliography}{\breitestes label}
\bibitem[label1]{zitierschlüssel1} bibliographische information
\bibitem[label2]{zitierschlüssel2} bibliographische information
...
\end{thebibliography}
```

Das Argument *breitestes label* dient zur Bestimmung des richtigen Einzugs für die einzelnen Einträge. Wenn die Werke fortlaufend nummeriert sind, sollte es z.B. die Anzahl der Einträge enthalten.

Einzelne Publikationen werden mit einem `\bibitem`-Befehl eingeleitet. Sein obligatorisches Argument ist ein eindeutiger *zitierschlüssel*, mit dem im Text auf dieses Werk verwiesen wird. Das optionale Argument definiert eine textliche Entsprechung, die im Quellenverweis und als *label* im Verzeichnis verwendet wird. Sofern dieses Argument nicht angegeben ist, werden die Publikationen entsprechend der Voreinstellung mit arabischen Ziffern nummeriert. Innerhalb einer Publikation kann man mit dem Befehl `\newblock` umfangreichere Informationen voneinander absetzen. Je nach von der Klasse erzeugtem Layout kann dies durch einen normalen Freiraum, zusätzliche Abstände oder eine neue Zeile geschehen.

Literatur

- [1] Goossens, M., S. Rahtz und F. Mittelbach (1997). *The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript*. Reading, MA, USA: Addison-Wesley Longman.
- [2] Goossens, M., B. User, J. Doe (1997). Mehrdeutige Quellenverweise.

```
\begin{thebibliography}{2}
\bibitem[LGC97]{Goossens, M., S.~Rahtz
und F.~Mittelbach (1997).
\newblock \emph{The \LaTeX{} Graphics
Companion: Illustrating Documents with
\TeX{} and PostScript}. \newblock Reading, MA,
USA: Ad\~di\~son\~Wes\~ley Longman.
\bibitem{GUD97} Goossens, M., B.~User, J.~Doe
(1997). \newblock Mehrdeutige Quellenverweise.
\end{thebibliography}
```

Bsp.
12-1-3

Das manuelle Erstellen einer derartigen Bibliographie ist auf jeden Fall mühsam und aufwendig, und das Ergebnis lässt sich normalerweise nicht wieder verwenden, da nahezu alle Journale und Verlagshäuser über eigene Layoutrichtlinien mit anderen Formatierungen verfügen. Daher nutzt man im Allgemeinen besser BibT_EX, ein Programm, das aus einer Datenbank mit bibliographischen Informationen fertigen L^AT_EX-Code erzeugt. Dies wird im nächsten Abschnitt besprochen.

Man beachte, dass die Verweise in der Reihenfolge nummeriert werden, wie sie in der Bibliographie erscheinen, sofern `\bibitem` ohne optionales Argument verwendet wird. Bei einer manuell erzeugten Bibliographie muss man die Einträge dann also selbst nach ihrem ersten Auftreten nummerieren und sortieren. Mit BibT_EX hingegen wird das entsprechende Ergebnis automatisch erzeugt.

Innerhalb eines Dokumentes erfolgen die Verweise auf ein Werk mithilfe des `\bibitem`-Argumentes *zitierschlüssel*. L^AT_EX verfügt dazu über den Befehl `\cite`, der einen solchen Schlüssel als Argument übernimmt. Er kann sogar eine durch Kommas unterteilte Liste solcher Schlüssel enthalten und verfügt zusätzlich über ein optionales Argument für weitere Informationen, wie etwa Seitenzahlen oder Kapitelnummern. Seine Syntax ist in Abschnitt 12.2.1 genau beschrieben. Für Kurztitel- und Autor-Jahr-Schemata gibt es noch weitere Quellenverweisbefehle, sobald die unterstützenden Pakete geladen sind.

*Manuelle Sortierung
nach erstem
Vorkommen*

12.1.3 Erzeugen bibliographischer Daten mit BibT_EX

Das Programm BibT_EX sammelt alle in einem Dokument vorkommenden Zitierschlüssel, schlägt diese in einer bibliographischen Datenbank nach und erzeugt daraus eine vollständige thebibliography-Umgebung, die dann im folgenden Durchlauf von L^AT_EX geladen werden kann. Je nach verwendetem

- ① Aufruf von $\text{\LaTeX}$, das eine Reihe von `\citation`-Verweisen in der Hilfsdatei `.aux` erzeugt.
- ② Aufruf von $\text{\BibTeX}$, das die Hilfsdatei einliest, die Literaturverweise in einer Datenbank nachschlägt (eine oder mehrere `.bib`-Dateien) und dann eine Datei (die `.bbl`-Datei) erstellt, welche die formatierten Literaturangaben enthält. Ihr Format wird dabei durch die Stildatei (die `.bst`-Datei) festgelegt. Warnungen und Fehlermeldungen werden in der Protokolldatei (der `.blg`-Datei) aufgezeichnet. Man beachte, dass $\text{\BibTeX}$ nie die Original- $\text{\LaTeX}$ -Quelldatei liest.
- ③ Erneuter Aufruf von $\text{\LaTeX}$, das nun die `.bbl`-Datei mit den bibliographischen Daten liest.
- ④ Dritter Aufruf von $\text{\LaTeX}$ zur Auflösung aller Verweise.

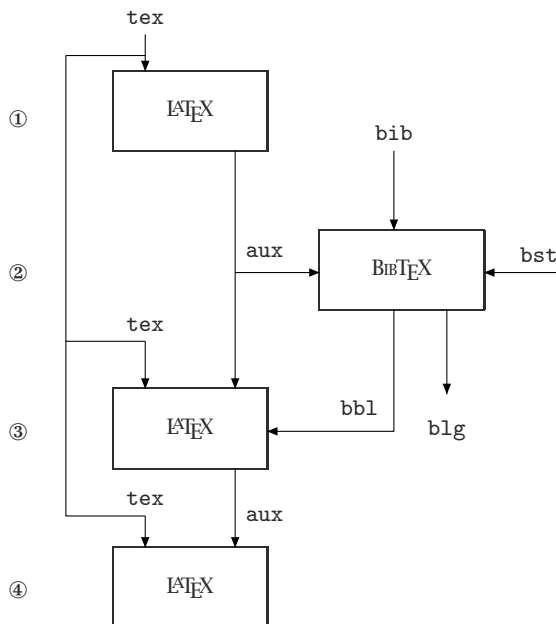


Abbildung 12.1: Flussdiagramm des Zusammenspiels von $\text{\BibTeX}$ und $\text{\LaTeX}$

Mit $\text{\BibTeX}$ nach erstem
Vorkommen sortieren

$\text{\BibTeX}$ -Stil kann $\text{\BibTeX}$ die Einträge nach einem bestimmten Schema sortieren (wie z.B. Autornamen, Erscheinungsjahr) oder ein Literaturverzeichnis erzeugen, in dem die Einträge in der Reihenfolge ihres Vorkommens im Text erscheinen. Man beachte, dass ein solcher „nicht sortierender“ Stil automatisch eine Bibliographie in der Reihenfolge des ersten Vorkommens generiert, wie es bei vielen Verlagen üblich ist. Ein Beispiel für einen derartigen $\text{\BibTeX}$ -Stil ist `unsrt`.

Das Zusammenspiel von $\text{\LaTeX}$ und $\text{\BibTeX}$ ist in Abbildung 12.1 schematisch dargestellt. Mindestens drei $\text{\LaTeX}$ -Läufe sind erforderlich: der erste um Daten für $\text{\BibTeX}$ zu erzeugen, der nächste um das Ergebnis aus dem $\text{\BibTeX}$ -Lauf zu laden und der letzte um die im vorherigen Lauf hinzugefügten Querverweise zum Literaturverzeichnis aufzulösen.

```
\bibliography{datei-liste} \bibliographystyle{format}
```

Damit $\text{\BibTeX}$ weiß, welche Datenbanken durchsucht werden sollen, um Verweise aufzulösen, werden ihre Namen, durch Kommas getrennt (und ohne die Erweiterung `.bib`), als Argument an den Befehl `\bibliography` übergeben. Dieser Befehl wird an der Stelle positioniert, an welcher letztlich die Bibliographie erscheinen soll. Außerdem muss man $\text{\BibTeX}$ mitteilen, wie die Einträge formatiert werden sollen. Dies geschieht in der Präambel mithilfe des Befehls `\bibliographystyle`, der als Argument einen geeigneten $\text{\BibTeX}$ -Stil erhält. Es ist natürlich wichtig, dass die im Dokument verwendeten *zitierschlüssel* genau einen Eintrag in der oder den Datenbankdateien identifizieren, damit der Quellenverweis beim Verarbeiten des Dokumentes aufgelöst werden kann.

Damit BibTeX auch ohne Analyse der L^AT_EX-Quelldateien auf die Informationen zugreifen kann, schreiben diese Befehle zwei Zeilen in die .aux-Datei. Aus dem gleichen Grund schreiben der Befehl \cite und alle seine Varianten ihre *schlüssel* in diese Datei. In Beispiel 12-1-2 würde die .aux-Datei beispielsweise (unter anderem) die folgenden Zeilen enthalten:

```
\bibstyle{abbrvnat}
\citation{LGC97}
\citation{test97}
\bibdata{tex}
```

Diese Befehle dürfen nicht mit den im Dokument zu verwendenden verwechselt werden. Sie dienen einzig dazu, die interne Kommunikation zwischen L^AT_EX und BibTeX zu erleichtern. Wenn man fälschlicherweise anstelle von \bibliography den Befehl \bibdata verwendet, wird L^AT_EX das Dokument zwar problemlos verarbeiten, BibTeX wird sich jedoch beklagen, dass sich in der .aux-Datei keine Datenbankinformationen befinden.

Der genaue Aufbau eines BibTeX-Eintrags wird in Kapitel 13 beschrieben. Um die Beispiele in den nächsten Abschnitten besser nachvollziehen zu können, sollte man jedoch wissen, dass ein BibTeX-Eintrag grundlegend aus den folgenden drei Teilen besteht:

1. Einem *Publikationstyp* (z.B. “book”, “article”, “inproceedings”, “phdthesis”).
2. Einem *vom Benutzer gewählten Zitierschlüssel*, anhand dessen die Veröffentlichung eindeutig zu identifizieren ist. Will man im Dokument auf den Eintrag verweisen, so muss das Argument *zitierschlüssel* des \cite-Befehls mit diesem Schlüsselwort identisch sein (auch bezüglich seiner Groß-/Kleinschreibung).
3. Einer *Reihe von Feldern* (bibliographischen Kategorien), die aus einem Feldnamen bestehen, dem in doppelten Anführungsstrichen oder geschweiften Klammern die entsprechenden Angaben folgen (z.B. author, journal und title).

In Abbildung 12.2 auf der nächsten Seite ist eine Beispieldatenbank zu sehen. Die meisten Beispiele werden anhand dieser Datenbank erstellt, um zu zeigen, wie sich die Darstellungsformen der einzelnen BibTeX-Stildateien voneinander unterscheiden.

Es gibt verschiedene Konventionen, um aussagekräftige Zitierschlüssel für Datenbankeinträge zu erstellen. Eines der bekanntesten ist das so genannte Harvard-System, in dem der Nachname des Autors (in Kleinbuchstaben) durch einen Doppelpunkt mit dem Erscheinungsjahr verbunden wird (z.B. schmidt:1987).

Die BibTeX-Einträge werden von BibTeX aus der Literaturdatenbank (der .bib-Datei) eingelesen. Die Formatierung der Einträge wird durch die zugehörige Stilvorlage (die .bst-Datei) festgelegt, die in einer stapelorientierten Sprache geschrieben ist. Diese Sprache wird vom BibTeX-Programm interpretiert (siehe Abschnitt 13.6).

```

@String{ttct = "Tools and Techniques for Computer
              Typesetting" }

@Book{LGC97,
  author = "Michel Goossens and Sebastian Rahtz
            and Frank Mittelbach",
  title = "The {\LaTeX} Graphics Companion:
            Illustrating Documents with {\TeX}
            and {\PostScript}",
  publisher = "Ad{\-d}i{\-s}on-Wes{\-l}ey Longman",
  address = "Reading, MA, USA",
  pages = "xxi + 554",
  year = "1997",
  ISBN = "0-201-85469-4",
  series = ttct
}

@UNPUBLISHED{test97,
  author = "Michel Goossens and Ben User
            and Joe Doe and others",
  title = "Mehrdeutige Quellenverweise",
  year = "1997",
  note = "Eingereicht bei " # ibmjrd
}

@Book{LWC99,
  author = "Michel Goossens and Sebastian Rahtz",
  title = "The {\LaTeX} {\Web} companion:
            integrating {\TeX}, {\HTML},
            and {\XML}",
  publisher = "Ad{\-d}i{\-s}on-Wes{\-l}ey Longman",
  address = "Reading, MA, USA",
  pages = "xxii + 522",
  year = "1999",
  ISBN = "0-201-43311-7",
  note = "With Eitan M. Gurari and Ross Moore
            and Robert S. Sutor",
  series = ttct
}

@Book{Knuth-CT-a,
  Author = "Donald E. Knuth",
  Title = "The {\TeX}book",
  Publisher = "Ad{\-d}i{\-s}on-Wes{\-l}ey",
  Address = "Reading, MA, USA",
  Volume = "A",
  Series = "Computers and Typesetting",
  pages = "ix + 483",
  year = 1986,
  isbn = "0-201-13447-0",
}

@Article{Knuth:TB10-1,
  Author = "Donald E. Knuth",
  Title = "{\Typesetting Concrete
            Mathematics}",
  Journal = "TUGboat",
  Volume = "10",
  Number = "1",
  Pages = "31--36",
  year = 1989,
  month = apr,
  issn = "0896-3207"
}

@Book{vLeunen:92,
  author = "Mary-Claire van Leunen",
  gender = "sf",
  title = "A handbook for scholars",
  publisher = "Oxford University Press",
  address = "Walton Street, Oxford OX2 6DP, UK",
  pages = "xi + 348",
  year = "92"
}

@manual{GNUMake, key = {make},
  title = {{GNU Make}, A Program for Directing
  Recompilation}, organization= "Free
  Software Foundation", address = "Boston,
  Massachusetts", ISBN={1-882114-80-9}, year = 2000}

@book{G-G,
  TITLE = {{Gutenberg Jahrbuch}},
  EDITOR = {Hans-Joachim Koppitz},
  PUBLISHER = {Gutenberg-Gesellschaft, Internationale
  Vereinigung f\ur Geschichte und
  Gegenwart der Druckkunst e.V.},
  ADDRESS = {Mainz, Deutschland},
  NOTE = {Enthält Fakten zur Geschichte und
  Gegenwart der Druckkunst. Gegründet von
  Aloys Ruppel. Erscheint seit 1926.}
}

@misc{oddy,
  title = "{{TUGboat} The Communications of the
            {\TeX} User Group}",
  howpublished = "Quarterly published.",
  year = {1980ff},
}

@InProceedings{MR-PQ,
  author = "Frank Mittelbach and Chris Rowley",
  title = "The Pursuit of Quality: How can
            Automated Typesetting achieve the
            Highest Standards of Craft
            Typography?",
  pages = "261--273",
  crossref = "EP92"}

@InProceedings{Southall,
  Author = "Richard Southall",
  Title = "Presentation Rules and Rules of
            Composition in the Formatting of
            Complex Text",
  Pages = "275--290",
  crossref = "EP92"}

@Proceedings{EP92,
  title = "{EP92}---Proceedings of Electronic
            Publishing, '92",
  shorttitle = "{EP92}",
  editor = "Christine Vanoirbeek and Giovanni
            Coray",
  publisher = "Cambridge University Press",
  address = "Cambridge",
  year = 1992,
  booktitle = "{EP92}---Proceedings of Electronic
            Publishing, '92"
}

```

Abbildung 12.2: B_BT_EX-Beispieldatenbank tex.bib

Die Einträge dieser Datenbank folgen bewusst keinen einheitlichen Konventionen, sondern enthalten z.B. klein-, groß- oder gemischt geschriebene Feldnamen und verschiedene Einrückungen, anhand derer sich in späteren Beispielen einige Funktionen und Probleme aufzeigen lassen. Sie lässt sich durch eines der in Abschnitt 13.4 beschriebenen Werkzeuge normalisieren.

BibTeX weiß, welche Felder für welchen Publikationstyp zwingend oder optional sind und welche ignoriert werden (vgl. Tabelle 13.1 auf Seite 791). Bei fehlenden Daten werden Warnungen, wie etwa „author name required“ (Auturname erforderlich), ausgegeben. Die Stildatei kann sowohl das Aussehen des Verweises im Haupttext als auch des tatsächlichen Bibliographieeintrags in der thebibliography-Umgebung steuern.

Man sollte immer im Hinterkopf behalten, dass BibTeX für die Verwaltung von Quellenverweisen nicht erforderlich ist (außer im Falle des jurabib-Paketes und der Pakete für mehrfache Literaturverzeichnisse). Man kann auch ohne BibTeX eine Bibliographie erzeugen, indem man die Einträge entsprechend der in Abschnitt 12.1.2 beschriebenen Syntax selbst erstellt. Außerdem ist es relativ einfach, in Sonderfällen die von BibTeX erzeugten Daten manuell nachzubearbeiten. Um ein völlig autarkes L^AT_EX-Dokument zu erhalten, kann man sogar den Inhalt der .bb1-Datei in das Dokument aufnehmen.

12.2 Das numerische Schema

12.2.1 Standard-L^AT_EX – Numerische Verweise

Wie bereits zuvor erwähnt, ist das numerische System die voreingestellte Zitiermethode und sie wird von Standard-L^AT_EX direkt unterstützt. Damit funktioniert sie als Einzige ohne Laden zusätzlicher Pakete rein mithilfe der verfügbaren Markup-Befehle. Literaturverweise innerhalb des Textes eines L^AT_EX-Dokumentes werden dann mit dem Befehl `\cite` gekennzeichnet.

<code>\cite[<i>text</i>]{<i>schlüssel</i>}</code> <code>\cite[<i>text</i>]{<i>schlüssel1</i>,<i>schlüssel2</i>,...}</code>

Der Befehl `\cite` verknüpft jeden Zitierschlüssel aus der Liste seines obligatorischen Argumentes mit dem Schlüssel im Argument eines `\bibitem`-Befehls der thebibliography-Umgebung zu einem Quellenverweis. Wie viele andere Feldnamen und sonstige Bezeichnungen in L^AT_EX werden auch die Zitierschlüssel nach Groß- und Kleinschreibung unterschieden.

Die erzeugten Quellennummern entsprechen der Reihenfolge, in der die Schlüssel der `\bibitem`-Befehle innerhalb der thebibliography-Umgebung auftreten bzw., sofern die `\bibitem`-Befehle über ein optionales Argument verfügen, den Daten dieses Argumentes.

Das optionale Argument *text* ist ein zusätzlicher Hinweis, der zusammen mit dem vom `\cite`-Befehl erzeugten Text ausgegeben wird. Dies wird auch im nächsten Beispiel deutlich. Zum Vergleich wurde im ersten Quellenverweis ein geschützter Leerraum (~) eingefügt und im zweiten ein kleiner Leerraum (\,). Solche typographischen Feinheiten sollten natürlich im gesamten Dokument einheitlich gehandhabt werden.

`\bibliographystyle{plain}`

Bsp.
12-2-1

Die Farbunterstützung für L^AT_EX ist in [2, Kap. 9] und das hyperref-Paket in [1, S. 35–67] beschrieben.

Die Farbunterstützung für L^AT_EX ist in `\cite[Kap.~9]{LGC97}` und das `\texttt{hyperref}`-Paket in `\cite[S.\,35--67]{LWC99}` beschrieben.

Um Platz zu sparen, entfällt das Literaturverzeichnis in vielen der Beispiele in diesem Kapitel. In diesen Fällen wird der Befehl `\bibliography{tex}`

Ein Hinweis zu den
Beispielen in diesem
Kapitel

beim automatischen Erzeugen der Beispiele für dieses Buch am Ende des jeweiligen Beispieldokumentes platziert. Daher sollte man Beispiele wie 12-2-1 folgendermaßen lesen: Das Ergebnis wird erzeugt, indem $\text{\LaTeX}$ unter Verwendung des Stils `plain` (angezeigt) und der Datenbank `tex.bib` (nicht angezeigt; siehe Abbildung 12.2) die Bibliographie generiert. Dazu enthielt das eigentliche Dokument, mit dessen Hilfe das Beispiel erzeugt wurde, gegen Ende die Anweisungen `\newpage\bibliography{tex}`.

```
\nocite{schlüssel-liste}
```

Unter $\text{\LaTeX}$ kann man auch die `\nocite`-Variante des `\cite`-Befehls verwenden. Er dient einzig dem Zweck, die Schlüssel aus dem Argument *schlüssel-liste* in die `.aux`-Datei zu schreiben, so dass die entsprechenden Quellenangaben im Literaturverzeichnis erscheinen, auch wenn das Werk gar nicht zitiert wird. Aus technischen Gründen muss er *nach* dem Befehl `\begin{document}` stehen, obwohl er keine Ausgabe erzeugt und daher eigentlich in der Präambel besser aufgehoben wäre. Er lässt sich beliebig oft verwenden. Als Sonderform überträgt der Befehl `\nocite{*}` alle Einträge der gewählten $\text{\LaTeX}$ -Daten in das Literaturverzeichnis.

Wie zuvor bereits erwähnt, wird der `\cite`-Befehl durch das Argument *schlüssel-liste* mit einem oder mehreren Bibliographieeinträgen verknüpft. Die Literaturangabe, die dann tatsächlich im formatierten Text erscheint, hängt von der gewählten Stilvorlage für die Bibliographie ab.

Anpassen der Literaturhinweise und der Bibliographie

Leider ist Standard- $\text{\LaTeX}$ nicht mit einer einfach einstellbaren Schnittstelle zum Anpassen der Formatierung von Literaturhinweisen ausgerüstet. Um beispielsweise die voreingestellten eckigen in runde Klammern zu ändern, muss man daher den internen $\text{\LaTeX}$ -Befehl `\@cite` umdefinieren.

Darüber hinaus verwendet der `\cite`-Befehl auch noch einen internen temporären Schalter `@tempwa`, um `\@cite` anzuzeigen, ob ein optionales Argument vorhanden ist. Um mit diesem optionalen Argument zu arbeiten, muss also zunächst der Wert dieses Schalters ermittelt werden. Der Befehl `\@cite` besitzt zwei obligatorische Argumente: die Liste der erhaltenen Verweise und den Hinweis (sofern vorhanden, sonst ein Leerstring). Im nächsten Beispiel wird zunächst `#1` gesetzt und, wenn `@tempwa` wahr ist, folgt darauf ein Komma und dann `#2`. Danach wird die Klammer geschlossen. Der Befehl `\nolinebreak[3]` sorgt dafür, dass ein Zeilenumbruch nach dem Komma nur im Notfall in Erwägung gezogen wird.

```
\bibliographystyle{plain}    \usepackage{ifthen}
\makeatletter
\renewcommand\@cite[2]{({#1\ifthenelse{\boolean{@tempwa}}{
    \nolinebreak[3] #2}{}})}
\makeatother
```

Die Farbkombination für $\text{\LaTeX}$ ist in `\cite{LGC97}` und das `\texttt{hyperref}`-Paket in `\cite[S.~35--67]{LWC99}` beschrieben.

Bsp.
12-2-2

Die Farbkombination für $\text{\LaTeX}$ ist in (2) und das `hyperref`-Paket in (1, S. 35–67) beschrieben.

Lädt man das cite-Paket (siehe Abschnitt 12.2.2), dann braucht man den Befehl `\@cite` nicht mehr für derartige Zwecke umzudefinieren.

Bei der `thebibliography`-Umgebung ist die Lage leider auch nicht viel besser: Die vorgegebene Lösung bietet nur wenige Anpassungsmöglichkeiten. Um die Kurzbezeichnung (Label) vor jeder Publikation umzugestalten (etwa durch Verzicht auf die Klammern) muss man den internen $\LaTeX$ -Befehl `\@biblabel` ändern.

Literatur

1. D. E. Knuth. *The $\TeX$ book*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
\bibliographystyle{abbrv}
\makeatletter
\renewcommand\@biblabel[1]{#1.}
\makeatother
\nocite{Knuth-CT-a,Knuth:TB10-1}
\bibliography{tex}
```

Bsp.
12-2-3

2. D. E. Knuth. *Typesetting Concrete Mathematics*. *TUGboat*, 10(1):31–36, Apr. 1989.

Pakete, die eine Variante des Autor-Jahr-Schemas umsetzen (wie z.B. `apalike`, `chicago` und `natbib`), definieren den Befehl `\@biblabel` normalerweise schlichtweg so um, dass er sein Argument unterschlägt und nichts ausgibt. In dem entsprechenden Literaturverzeichnis schlägt man ja den Namen des Verfassers nach, so dass keine Kurzbezeichnung erforderlich ist. Dabei ist das `natbib`-Paket jedoch noch relativ vorsichtig: Wenn es entdeckt, dass `\@biblabel` zuvor undefiniert wurde, behält es die geänderte Definition bei.

Wie bereits erwähnt werden die einzelnen Bestandteile eines `\bibitem`-Befehls, wie etwa die Angabe der Autoren oder des Titels, im Literaturverzeichnis durch `\newblock`-Befehle voneinander getrennt. Diese Befehle werden auch von den meisten `BibTeX`-Stilen automatisch eingefügt. Normalerweise werden die einzelnen Elemente zusammen im gleichen Absatz gesetzt. Bevorzugt man jedoch eine „offene“ Bibliographie, in der jede Information in einer neuen Zeile beginnt und die nachfolgenden Zeilen innerhalb eines Blocks um die Länge `\bibindent` (voreingestellt auf 1,5em) eingezogen werden, so sollte man die Klassenoption `openbib` verwenden. Diese Option wird von allen Standardklassen unterstützt. Das Ergebnis ist im nächsten Beispiel zu sehen. Dort werden außerdem mit einer geänderten `\@biblabel`-Definition hochgestellte Label erzeugt.

Literatur

- ¹ M. Goossens and S. Rahtz.
The $\LaTeX$ Web companion: integrating $\TeX$, HTML, and XML.

Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1999.

With Eitan M. Gurari and Ross Moore and Robert S. Sutor.

- ² D. E. Knuth.

The $\TeX$ book, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
\documentclass[openbib]{article}
\bibliographystyle{abbrv}
\setlength\bibindent{24pt}

\makeatletter
\renewcommand\@biblabel[1]
{\textsuperscript{#1}}
\makeatother
\nocite{LWC99,Knuth-CT-a}
\bibliography{tex}
```

Bsp.
12-2-4

12.2.2 cite – Bessere numerische Verweise

Eine Unzulänglichkeit, die rasch ins Auge fällt, wenn man die in $\text{\LaTeX}$ vorgegebene Methode für Quellenverweise verwendet, besteht darin, dass es die Reihenfolge der Quellen aus dem Argument *schlüssel-liste* des `\cite`-Befehls beibehält. Entsprechend zeigt das nächste Beispiel eine sehr seltsam sortierte Zahlenliste (der nicht aufgelöste Verweis wurde absichtlich hinzugefügt):

In [2, 1, 3, ?, 4] findet man gute Informationen zu $\text{\TeX}$ und $\text{\LaTeX}$.
`\bibliographystyle{plain}`
 In `\cite{LGC97,LWC99,Knuth-CT-a,Knuth:CT-b,Knuth:TB10-1}` findet man gute Informationen zu `\TeX` und `\LaTeX`.

Bsp.
12-2-5

Die Situation lässt sich leicht verbessern, indem man, wie im folgenden Beispiel, das `cite`-Paket von Donald Arseneau lädt:

In [?, 1–4] findet man gute Informationen zu $\text{\TeX}$ und $\text{\LaTeX}$.
`\usepackage{cite}` `\bibliographystyle{plain}`
 In `\cite{LGC97,LWC99,Knuth-CT-a,Knuth:CT-b,Knuth:TB10-1}` findet man gute Informationen zu `\TeX` und `\LaTeX`.

Bsp.
12-2-6

Das `cite`-Paket sortiert die Quellennummern entsprechend seiner Voreinstellung in aufsteigender Reihenfolge und stellt drei oder mehr aufeinander folgende Nummern als Nummernbereich dar. Nicht numerische Label werden an den Anfang gestellt, so wie im vorigen Beispiel das „?“ aus dem nicht aufgelösten Verweis. Wer keine sortierten Einträge wünscht, kann diese mit der Option `nosort` global abschalten. Die Zusammenfassung zu Bereichen lässt sich durch die Option `nocompress` unterdrücken.

Das `cite`-Paket bietet eine Reihe von Befehlen zum Anpassen der formatierten Verweise. So kann man z.B. mithilfe der Befehle `\citeleft` und `\citeright` festlegen, welche Zeichen links bzw. rechts des Verweistextes erscheinen sollen. Mit ihrer Hilfe kann man, wie im nächsten Beispiel, die eckigen Klammern gegen runde austauschen (vgl. hierzu auch Beispiel 12-2-2 auf Seite 716). Außerdem lässt sich der Befehl `\citimid` so umdefinieren, dass er den Quellenverweis und die optionalen ergänzenden Angaben etwa durch ein Semikolon und ein Leerzeichen voneinander trennt.

Anpassen des Layouts
der Quellenverweise

Die Farbkunterstützung für $\text{\LaTeX}$ ist in (2) und das `hyperref`-Paket in (1; S. 35–67) beschrieben.

`\usepackage{cite}` `\bibliographystyle{plain}`
`\renewcommand\citeleft{() \renewcommand\citeright{}}`
`\renewcommand\citimid{;\nolinebreak[3]}`
 Die Farbkunterstützung für `\LaTeX` ist in `\cite{LGC97}` und das `\texttt{hyperref}`-Paket in `\cite[S.~35--67]{LWC99}` beschrieben.

Bsp.
12-2-7

Ein weiterer wichtiger Gesichtspunkt beim Umgang mit Quellenverweisen ist die Steuerung ihres Verhaltens am Zeilenende. Nimmt man z.B. den Text „siehe [2–3,7,13]“, so kann man hier einerseits alle Zeilenumbrüche verbieten, oder andererseits einen Umbruch nach dem „siehe“, den Kommas oder dem Halbgeviertstrich im Nummernbereich erlauben.

Das `cite`-Paket ist so voreingestellt, dass es Zeilenumbrüche vor einem Quellenverweis mit `\nolinebreak[3]`, und solchen nach einem Komma zwischen Verweis und ergänzenden Angaben mit `\nolinebreak[2]` entgegenwirkt. Zeilenumbrüche nach Halbgeviertstrichen in Bereichsangaben und nach

Anpassen von
Umbrüchen in
Quellenverweisen

Kommas zwischen einzelnen Quellennummern lässt es so gut wie gar nicht zu. Die drei letztgenannten Fälle lassen sich durch Umdefinieren der Befehle `\citemid`, `\citedash` und `\citepunct` steuern. Mit folgenden Zeilen kann man z.B. Umbrüche nach Bindestrichen verhindern, dabei aber solche nach Kommas erleichtern:

```
\renewcommand\citedash{\mbox{--}\nolinebreak}
\renewcommand\citemid{\nolinebreak[1] }
\renewcommand\citepunct{\nolinebreak[1]\hspace{.13em plus .1em minus .1em}}
```

Hier gilt es einige interessante Punkte zu beachten. Alle drei Definitionen steuern nicht nur den Zeilenumbruch, sondern fügen auch die erforderlichen Satzzeichen hinzu: einen Bindestrich für einen Bereich, ein Komma und ein ganzes Leerzeichen vor den ergänzenden Angaben oder ein Komma und einen kleinen Leerraum zwischen einzelnen Quellenverweisen. Sollen zwischen den Quellenverweisen gar keine Leerzeichen stehen, dann kann man den Befehl `\citepunct` so umdefinieren, dass er nur ein Komma enthält. Der andere wichtige und vielleicht überraschende Aspekt ist der `\mbox`-Befehl um den Bindestrich. Diese Box ist erforderlich, um den Zeilenumbruch durch $\text{\LaTeX}$ an dieser Stelle zu steuern. $\text{\TeX}$ fügt automatisch nach jedem expliziten Trenn- oder Bindestrich eine Trennstelle ein. Solange man den Strich also nicht in einer Box versteckt, hat der `\nolinebreak`-Befehl keinerlei Auswirkungen, da die intern eingefügte Trennstelle weiterhin einen Umbruch an dieser Position zulässt. Und der `\hspace`-Befehl, schließlich, ermöglicht eine gewisse Dehnung oder Stauchung. Bevorzugt man stattdessen einen festen Leerraum, so muss man die Komponenten `plus` und `minus` entfernen.

Der hohe Penalty-Wert vor einem Quellenverweis ist fest im Code verankert. Er wird jedoch nur eingefügt, wenn man nicht selbst schon einen entsprechenden Wert im Dokument festgelegt hat. Der Ausdruck „siehe~\cite{..}“ wird zum Beispiel immer beachtet, so dass kein Umbruch zwischen „siehe“ und dem Quellenverweis erfolgt.

Mithilfe eines weiteren Formatierungsbefehls, `\citeform`, kann man die einzelnen Quellennummern umgestalten. Er ist zunächst so eingestellt, dass er nichts bewirkt, so dass die Label nicht verändert werden. Im nächsten Beispiel sind sie eingefärbt. Es gibt natürlich auch andere Gestaltungsmöglichkeiten, wie etwa das Hinzufügen von Klammern in Beispiel 12-2-9.

Anpassen der Quellenverweisnummern

```
\usepackage{cite,color} \bibliographystyle{plain}
\renewcommand\citeform[1]{\textcolor{blue}{#1}}
```

Die Farbunterstützung für $\text{\LaTeX}$ ist in [2] und das `hyperref`-Paket in [1, S. 35–67] beschrieben.

Die Farbunterstützung für $\text{\LaTeX}$ ist in `\cite{LGC97}` und das `\texttt{hyperref}`-Paket in `\cite[S.~35--67]{LWC99}` beschrieben.

Bsp.
12-2-8

```
\citen{schlüssel-liste}
```

Das Paket verfügt über den zusätzlichen Befehl `\citen` (Alternativnamen `\citenum` oder `\citeonline`), mit dessen Hilfe man eine Nummernliste ohne die umgebenden Befehle `\citeleft` und `\citeright` erhält (also z.B. ohne die voreingestellten Klammern). Alle anderen Formatierungen bleiben erhalten. Im nächsten Beispiel werden die einzelnen Quellenverweise in Klammern

gesetzt. Das sieht in Kombination mit den voreingestellten Klammern um den gesamten Verweis zugegebenermaßen recht seltsam aus.

```
\usepackage[nospace]{cite}
\bibliographystyle{plain} \renewcommand\citeform[1]{(#1)}
\citen{LGC97,LWC99,test97,vLeunen:92} aber
(1)–(3),(5) aber [(4), §5] \cite[\S5]{Knuth-CT-a}
```

Bsp.
12-2-9

Das cite-Paket bietet eine Reihe von Optionen für typische Konfigurationswünsche oder um das Verhalten des Paketes anderweitig zu beeinflussen. Einige davon wurden bereits besprochen. Hier folgt nun eine vollständige Aufstellung:

adjust/noadjust Aktiviert (Voreinstellung) bzw. deaktiviert den „intelligenten“ Umgang mit Leerzeichen vor einem `\cite`- oder einem `\citen`-Befehl. Leerzeichen vor solchen Befehlen werden per Voreinstellung als normaler Wortzwischenraum gestaltet. Wenn man `see\cite{..}` schreibt, wird automatisch ein Leerzeichen eingefügt.

compress/nocompress Aktiviert (Voreinstellung) bzw. deaktiviert das Zusammenfassen aufeinander folgender Nummern zu Bereichen.

sort/nosort Aktiviert bzw. deaktiviert das Sortieren der Nummern.

space Nach Kommas wird ein ganzer Wortzwischenraum eingesetzt und ein Zeilenumbruch an dieser Stelle wird nicht behindert. Per Voreinstellung (Option ist nicht ausgewählt) wird ein kleiner Leerraum eingefügt und der Umbruch erschwert, aber nicht verboten.

nospace Löscht in der Nummernliste die Leerzeichen nach den Kommas, wobei jedoch das Leerzeichen nach dem Komma vor den ergänzenden Angaben erhalten bleibt. Die Auswirkung dieser Option ist in Beispiel 12-2-9 zu sehen. Sie ist nicht das Gegenteil der Option `space`!

verbose Normalerweise warnt cite bei Quellenverweisen nur einmal je nicht definierter Quelle. Mit dieser Option wird die Warnung bei jedem Verweis auf die entsprechende Quelle wiederholt.

Die neueste Version des cite-Paketes kann Quellenverweise auch als hochgestellte Nummern ausgeben, wenn das Paket mit der Option `superscript` (oder `super`) geladen wird. Diese Funktion wurde früher durch das eigenständige Paket `overcite` bereitgestellt (das ebenfalls von Donald Arseneau stammt). Dieses Paket ist aus Gründen der Kompatibilität auch weiterhin verfügbar.

Quellenverweise mit
hochgestellten
Nummern

Wird der `\cite`-Befehl mit optionalem Argument verwendet, dann erscheint die gesamte Liste der Quellenverweise so, als wäre das cite-Paket ohne die `superscript`-Option geladen worden.

Unter dem Einfluss der Option `superscript` oder `super` wirken sich die Konfigurationsbefehle `\citeleft`, `\citeright` und `\citimid` nur noch auf Quellenverweise aus, die über ein optionales Argument verfügen, während `\citedash`, `\citepunct` und `\citeform` weiterhin alle Verweise betreffen. Nähere Einzelheiten dazu, wie sich diese Befehle verwenden lassen, wurden auf den Seiten 718–719 besprochen.

```

\usepackage[superscript]{cite} \bibliographystyle{plain}
\usepackage{color}
\renewcommand\citeform[1]{\textcolor{blue}{\#1}}
\renewcommand\citeleft{() \renewcommand\citeright{()}}
In \cite{LGC97,LWC99,Knuth-CT-a,Knuth:CT-b,Knuth:TB10-1}
findet man gute Informationen zu \TeX{} und \LaTeX{}.
Zu \texttt{hyperref} siehe \cite[S.~35--67]{LWC99}.

```

In^{2,1-4} findet man gute Informationen zu $\TeX$ und $\LaTeX$. Zu `hyperref` siehe (1, S. 35–67).

Bsp.
12-2-10

Man muss sein Quelldokument wahrscheinlich nicht ändern, egal ob die `superscript`-Option verwendet wird oder nicht. Insbesondere wird der Leer- raum vor dem Quellenverweis ignoriert, wenn die Verweise hochgestellt sind. Im Prinzip kann man die Option einfach hinzufügen, ohne die Quelldokumen- te verändern zu müssen, vorausgesetzt die numerischen Verweise wurden nicht in die Formulierung der Sätze integriert, wie es etwa im vorigen Beispiel der Fall war.

Wenn als Label für die Quellenverweise hochgestellte Nummern verwen- det werden, muss man besonders auf die Interpunktion um die Verweise her- um achten. Das `cite`-Paket verschiebt entsprechend seiner Voreinstellung Satz- zeichen, die auf einen Verweis folgen, vor den Verweis. Alle Satzzeichen, die auf diese Weise wandern, sind in dem Befehl `\CiteMoveChars` gespeichert. Die Voreinstellung enthält „.,;:“, die Zeichen ! und ? sind nicht enthalten, können jedoch hinzugefügt werden. Durch dieses Verfahren kann es zu ei- ner Verdoppelung von Punkten kommen. Dieser Fall wird vom Paket wahrge- nommen und eines der Satzzeichen wird dann unterdrückt (vgl. den zweiten Verweis im nächsten Beispiel).

```

\usepackage[superscript]{cite} \bibliographystyle{plain}
\ldots\ Buch-~\cite{Knuth-CT-a}; siehe auch Goossens
et al.~\cite{LGC97}.

```

... Buch;² siehe auch
Goossens et al.¹

Bsp.
12-2-11

Leider versagt das Unterdrücken doppelter Punkte bei Abkürzungen mit Großbuchstaben oder wenn *nach* dem Punkt ein `@` steht. Das nächste Bei- spiel zeigt dazu mögliche Lösungen. Man beachte jedoch, dass die Lösung `U.S.A@.` nur für das `cite`-Paket gilt, wobei man außerdem ohne Verweis ei- nen falschen Abstand erhält (da man eigentlich behauptet, dass der Satz nach der Abkürzung endet)!

```

\usepackage[super]{cite} \bibliographystyle{plain}
et al.\@ \cite{LGC97}. \hfil et al.\ \cite{LGC97}.\par
U.S.A..? U.S.A.? U.S.A. \cite{unknown}. \hfil U.S.A@. \cite{unknown}.

```

et al.¹ et al.¹
U.S.A..² U.S.A..²

Bsp.
12-2-12

Amerikanische Konventionen erfordern unter Umständen noch eine wei- tere Besonderheit, die nicht unterstützt wird: In einem direkt vorangehenden Zitat muss das abschließende Satzzeichen integriert werden (wie z.B. in *The Chicago Manual of Style* [38] befürwortet). Der englische Eingabetext muss dann entsprechend manuell angepasst werden.

For details see “The $\TeX$ book”.¹ But wanted is “The $\TeX$ book.”¹

```

\usepackage[super]{cite} \bibliographystyle{plain}
For details see ‘‘The \TeX book’’ \cite{Knuth-CT-a}.
But wanted is ‘‘The \TeX book.’’ \cite{Knuth-CT-a}

```

Bsp.
12-2-13

Die zentralen Optionen des cite-Paketes wurden auf auf Seite 720 besprochen. Zum Hochstellen der Verweisnummern existieren noch drei weitere Möglichkeiten. Durch die Option `nomove` werden Satzzeichen nicht vor den hochgestellten Verweis verschoben. Die Option `ref` bewirkt, dass vor Verweisen mit einem optionalen Argument das Wort „Ref.“ steht. Dies wird intern über eine Änderung des `\citeleft`-Befehls erreicht. Um also einen anderen Text einzustellen oder von eckigen etwa auf runde Klammern zu wechseln, muss man die Konfigurationsbefehle ändern, anstatt diese Option zu wählen.

Die Farbunterstützung für $\LaTeX$ ist in „LGC“² und das `hyperref`-Paket in „LWC“ [Ref. 1, S. 35–67] beschrieben.

```
\usepackage[super,ref]{cite} \bibliographystyle{plain}
Die Farbunterstützung für \LaTeX{} ist in "LGC"
\cite{LGC97} und das \texttt{hyperref}-Paket in
"LWC" \cite[S.~35--67]{LWC99} beschrieben.
```

Bsp.
12-2-14

Die Option `biblabel` schließlich stellt auch die Label in der Bibliographie hoch. (Sie behalten normalerweise ihre Position, auch wenn man die Option `superscript` bzw. ihren Alias `super` verwendet.)

12.2.3 notoccite – Unsortierte Quellenverweise, aber richtig

Damit die Publikationen in der Bibliographie in genau der gleichen Reihenfolge erscheinen, wie sie im Dokument zitiert werden, sollte man Stile für unsortierte Quellenverweise verwenden (wie z.B. den $\BibTeX$ -Stil `unsrt`). Dieser Ansatz funktioniert jedoch nicht, wenn auch Verweise in Überschriften oder Legenden von Gleitobjekten erscheinen. In diesem Fall tauchen diese Verweise ebenfalls im Inhalts-, Abbildungs- und entsprechenden Verzeichnissen auf. Dadurch werden sie an den Anfang der Bibliographie verschoben, obwohl sie eigentlich erst viel später im Text vorkommen.

Dieses Problem lässt sich umgehen, indem man für `\caption`, `\section` und ähnliche Befehle ein optionales Argument ohne den Verweis verwendet, so dass die Verweise nicht in den Verzeichnissen auftauchen. Sofern auch an diesen Stellen Verweise verwendet werden müssen, kann man als „manuelle“ Lösung zunächst alle Hilfsdateien vom letzten $\LaTeX$ -Lauf löschen, dann einen $\LaTeX$ -Lauf durchführen und danach $\BibTeX$ starten. So erhält $\BibTeX$ nur die Quellenverweise aus dem Hauptdokument. Dieses Verfahren ist natürlich fehleranfällig und es kann sein, dass die Reihenfolge der Verweise trotzdem durcheinander gerät, wenn man seinen Artikel schließlich gedruckt sieht.

Donald Arseneau hat zur Lösung dieses Problems das kleine Paket `notoccite` entwickelt, das den internen Befehl `\@starttoc` so umdefiniert, dass Quellenverweise in Verzeichnissen keine `\citation`-Befehle für $\BibTeX$ erzeugen. Mit dem einfachen Laden dieses Paketes ist das Problem in jedem Fall behoben, es sei denn man verwendet noch andere Pakete, die den Befehl `\@starttoc` umdefinieren. So lässt sich `notoccite` beispielsweise nicht mit dem `hyperref`-Paket oder den AMS-Dokumentenklassen kombinieren.

12.3 Das Autor-Jahr-Schema

Je nach Satzstruktur verwendet das Autor-Jahr-Schema normalerweise eine von zwei verschiedenen Verweisarten: Sofern der Name des Autors sowieso

im Satz vorkommt, wird er nicht innerhalb der Klammern wiederholt. Andernfalls werden sowohl der Autor als auch das Erscheinungsjahr aufgeführt. Dieser Stil erzeugt für die L^AT_EX-Standardsyntax ein unlösbares Problem, da diese nur über einen einzigen `\cite`-Befehl verfügt.

Daher musste jeder, der eine Unterstützung für das Autor-Jahr-Schema entwickeln wollte, die L^AT_EX-Syntax zum Zitieren von Quellen erweitern. Das folgende Beispiel zeigt die beiden Formen und ihre Umsetzung (durch zwei neue Befehle) im natbib-System.

Knuth (1989) zeigt ... Eine	<code>\usepackage{natbib}</code>
Erklärung dazu findet sich im	<code>\citet{Knuth:TB10-1}</code> zeigt \ldots Eine Erklärung dazu
maßgebenden Handbuch über	findet sich im maßgebenden Handbuch über <code>\TeX{}</code>
$\TeX$ (Knuth, 1986).	<code>\citep{Knuth-CT-a}</code> .

Bsp.
12-3-1

Eine Erweiterung der L^AT_EX-Syntax für das Zitieren von Veröffentlichungen löst das Problem jedoch nicht vollständig. Um die verschiedenen Arten von Quellenverweisen zu erzeugen, die das Autor-Jahr-Schema benötigt, müssen die Daten, welche das Literaturverzeichnis über das optionale Argument des `\bibitem`-Befehls zurückgibt, zunächst strukturiert werden. Ohne spezielle Strukturierung können die Daten für die textuellen Querverweise nicht extrahiert werden (z.B. nur das Jahr in Klammern). Aus dem Eintrag

```
\bibitem[Donald~E. Knuth 1986]{Knuth-CT-a} Donald~E. Knuth.
\newblock \emph{The {\TeX}book}, Band~A von \emph{Computers
and Typesetting}. \newblock Addison-Wesley, Reading, MA,
USA, 1986.
```

kann der `\cite`-Befehl zwar den Verweis „(Donald E. Knuth 1986)“ erzeugen, nicht aber „Donald E. Knuth (1986)“ oder nur „Knuth“ bzw. „1986“. Man muss auch sichergehen, dass `\bibitem` nicht das Label anzeigt, aber das ist relativ leicht zu erreichen.

Alle Lösungen für Autor-Jahr-Zitierschemata legen zu diesem Zweck für das optionale Argument von `\bibitem` eine besondere Syntax fest. Bei manchen Lösungen ist diese recht einfach aufgebaut. Das Paket `chicago` benötigt zum Beispiel nur

```
\bibitem[\protect\citeauthoryear
                {Goossens, Rahtz, und Mittelbach}
                {Goossens et al.}{1997}]{LGC97}
```

Derartige Daten können im Bedarfsfalle immer noch manuell erzeugt werden. Andere Pakete gehen hier viel weiter und bilden sehr umfangreiche Datenstrukturen. Das Paket `jurabib` z.B. erfordert den folgenden Aufbau des Argumentes (für das gleiche Werk):

```
\bibitem[Goossens\jbbfsasep Rahtz\jbbstasep Mittelbach\jbdy
{1997}]{book}{1997}{xxi + 554}{Reading, MA,
USA\bpubaddr{ Ad{\-d}i{\-s}on-Wes{\-l}ey Longman\bbbdsepe}
1997}{The {\LaTeX} Graphics Companion: Illustrating Documents
with {\TeX} and {\PostScript}}{LGC97}
```

Wie noch zu sehen sein wird (Abschnitt 12.5.1), gewährt dieser Ansatz eine große Flexibilität bei der Gestaltung von Literaturhinweisen. Es ist jedoch auch klar, dass niemand eine solche Bibliographieumgebung manuell erstellen möchte. Daher besteht die einzig brauchbare Lösung hier darin, die Einträge mithilfe eines externen Werkzeuges wie $\text{\LaTeX}$ automatisch zu erzeugen.

12.3.1 Frühe Versuche

Im Laufe der Jahre wurden viele eigenständige Zusatzpakete für das Autor-Jahr-Zitierschema entwickelt. Leider definierte jedes von ihnen einen anderen Satz von Benutzerbefehlen. Typischerweise bestehen diese Erweiterungen aus einem $\text{\LaTeX}$ -Paket mit den Benutzerbefehlen und einem oder mehreren $\text{\LaTeX}$ -Stilen zum Erzeugen der thebibliography-Umgebung mit einer passenden Syntax im optionalen Argument des `\bibitem`-Befehls.

Das *chicago*-Paket etwa, das die Empfehlungen des *The Chicago Manual of Style* [38] umzusetzen sucht, bietet die folgende Befehlsliste (sowie Variationen ohne Klammern, die alle auf NP enden, wie z.B. `\citeNP`):

	<code>\usepackage{chicago}</code>	
	<code>\bibliographystyle{chicago}</code>	
(Goossens, Rahtz, and Mittelbach 1997)	<code>\cite{LGC97}</code>	<code>\\</code>
(Goossens, Rahtz, and Mittelbach)	<code>\citeA{LGC97}</code>	<code>\\</code>
Goossens, Rahtz, and Mittelbach (1997)	<code>\citeN{LGC97}</code>	<code>\\</code>
(Goossens and Rahtz 1999)	<code>\shortcite{LWC99}</code>	<code>\\</code>
(Goossens and Rahtz)	<code>\shortciteA{LWC99}</code>	<code>\\</code>
Goossens and Rahtz (1999)	<code>\shortciteN{LWC99}</code>	<code>\\</code>
(1999)	<code>\citeyear{LWC99}</code>	<code>\\</code>
1999	<code>\citeyearNP{LWC99}</code>	

Bsp.
12-3-2

Zum *chicago*-Paket gibt es verschiedene kompatible $\text{\LaTeX}$ -Stile (*chicago*, *chicagoa*, *jas99*, *named* und *newapa*), die sich allerdings nur für englischsprachige Bibliographien eignen, da sie festverdrahtete Texte wie „and“ generieren. Sie sind alle weiterhin im Gebrauch, auch wenn das *chicago*-Paket heutzutage nur noch in wenigen $\text{\LaTeX}$ -Distributionen zu finden ist. An seiner Stelle kann das Paket *natbib* die nötigen Benutzerbefehle zur Verfügung stellen.

Das Paket *authordate1-4* von David Rhead kennt hingegen nur zwei Benutzerbefehle. Es ist das ursprüngliche Supportpaket zu den $\text{\LaTeX}$ -Stilen *authordate1* bis *authordate4* und setzt die Empfehlungen der Cambridge und der Oxford University Press sowie verschiedener britischer Standards um.

	<code>\usepackage{authordate1-4}</code>	
	<code>\bibliographystyle{authordate2}</code>	
(Goossens <i>et al.</i> , 1997) oder (1997)	<code>\cite{LGC97}</code> oder <code>\shortcite{LGC97}</code>	

Bsp.
12-3-3

Als letztes Beispiel sei das *harvard*-Paket von Peter Williams und Thorsten Schnier aufgeführt. Im Gegensatz zu den beiden erstgenannten wurde das *harvard*-Paket weiter entwickelt und für $\text{\LaTeX}2_{\epsilon}$ überarbeitet. Es verfügt über eine Reihe interessanter Funktionen. So nennt beispielsweise der erste Hinweis auf eine Quelle alle Autoren, während weitere Verweise eine abgekürzte

Aufzählung verwenden, sofern es nicht explizit anders gewünscht wird. Die Benutzerbefehle sind im nächsten Beispiel zu sehen.

		<code>\usepackage{harvard}</code>	
		<code>\bibliographystyle{agsm}</code>	
(Goossens, Rahtz & Mittelbach 1997)		<code>\cite{LGC97}</code>	<code>\\</code>
(Goossens et al. 1997)	zweiter	<code>\cite{LGC97} \hfill zweiter</code>	<code>\\</code>
(Goossens, Rahtz & Mittelbach 1997)	erzwungen	<code>\cite{*{LGC97}}\hfill erzwungen</code>	<code>\\</code>
Goossens et al. (1997)		<code>\citeasnoun{LGC97}</code>	<code>\\</code>
(siehe Goossens et al. 1997)		<code>\citeaffixed{LGC97}{siehe}</code>	<code>\\</code>
Goossens et al.		<code>\citenam{LGC97}</code>	<code>\\</code>
Knuth's (1986)		<code>\possessivecite{Knuth-CT-a}</code>	

Bsp.
12-3-4

Das `harvard`-Paket benötigt eine besonders aufbereitete Bibliographieumgebung, in welcher der Befehl `\bibitem` durch `\harvarditem` ersetzt wird, einem Befehl, der durch seine spezielle Syntax alle erforderlichen Daten für Autor-Jahr-Verweise bereitstellt. Einige $\text{\LaTeX}$ -Stile (unter anderem `agsm`, `dcu`, `kluwer` und `nederlands`) machen sich diese besondere Syntax zunutze. Auch hier gilt, dass diese Stile in einigen Aspekten (etwa `\possessivecite` im obigen Beispiel) von einer englischsprachigen Bibliographie ausgehen.

Viele dieser Pakete setzen das Autor-Jahr-Schema recht ordentlich um. Trotzdem blieb die Situation lange unbefriedigend, angesichts verschiedener Pakete mit eigener Syntax, von denen jedes nur ein halbes Dutzend $\text{\LaTeX}$ -Stile unterstützte. Die Lage besserte sich erst mit der `natbib`-Lösung von Patrick Daly, die Thema des nächsten Abschnitts ist.

12.3.2 natbib – Anpassungsfähige Autor-Jahr-Verweise

Auch wenn die meisten Verlage angeben, welches Format sie für ein Literaturverzeichnis bevorzugen, ist es nicht immer offensichtlich, wie man von einem System zu einem anderen wechseln kann, wenn man Quelltexte für verschiedene Stile aufbereiten muss.

Als Lösung für das im letzten Abschnitt beschriebene Problem einer nicht kompatiblen Syntax entwickelte Patrick Daly das Paket `natbib` (für „NATurwissenschaftliche BIBliographie“). Das Paket akzeptiert unterschiedliche `\bibitem`-Varianten (auch `\harvarditem`), wie sie von den verschiedenen $\text{\LaTeX}$ -Stilen erzeugt werden. So war es Benutzern zum ersten Mal möglich, (fast) alle Autor-Jahr-Stile von $\text{\LaTeX}$ mit einer einheitlichen Befehlssyntax für die Quellenverweise anzusprechen.

Das `natbib`-Paket ist mit Paketen wie `babel`, `chapterbib`, `hyperref`, `index` und `showkeys` kompatibel, sowie mit verschiedenen Dokumentenklassen, einschließlich der $\text{\LaTeX}$ -Standardklassen, `amsbook` und `amsart` sowie den Klassen der KOMA-Script-Sammlung und `memoir`. Es lässt sich nicht mit dem `cite`-Paket kombinieren, verfügt jedoch mittels Optionen über ähnliche Funktionen zum Sortieren und Zusammenfassen.

Soweit das Autor-Jahr-Schema gefragt ist, bildet das `natbib`-Paket somit eine einheitliche, flexible Schnittstelle für die meisten bibliographischen Stile. Es kann auch für numerische Verweise verwendet werden, wie noch in Abschnitt 12.4.1 zu sehen sein wird.

Die Basissyntax

Die beiden zentralen Befehle des `natbib`-Paketes sind `\citet` für im Text eingebettete Quellenverweise und `\citep` für Verweise in „parenthesen“, also in Klammern.

```
\citet[nach]{schlüssel-liste}   \citet[vor][nach]{schlüssel-liste}
\citep[nach]{schlüssel-liste}   \citep[vor][nach]{schlüssel-liste}
```

Beide Befehle verfügen über ein obligatorisches Argument (die *schlüssel-liste*, die sich auf ein oder mehrere Publikationen bezieht) und ein oder zwei optionale Argumente, durch welche zusätzliche Angaben vor oder nach dem Quellenverweis erfolgen können. Der Standardbefehl von L^AT_EX, `\cite`, verfügt über nur ein optionales Argument, das dem *nach*-Text entspricht. Daher verwenden die Befehle die folgende Syntax: Ist nur ein optionales Argument vorhanden, so entspricht dieses dem *nach*-Text (also den Angaben, die dem eigentlichen Quellenverweis folgen). Bei zwei optionalen Argumenten ist das erste der *vor*- und das zweite der *nach*-Text. Um nur Angaben vor dem Quellenverweis zu erhalten (mit dem *vor*-Text) muss man ein leeres zweites Argument hinzufügen, wie in Zeile 4 und 8 im nächsten Beispiel zu sehen. Man beachte auch, dass `natbib` den Befehl `\cite` so umdefiniert, dass er sich wie `\citet` verhält.¹

Goossens et al. (1997)	<code>\usepackage{natbib}</code>	
Goossens et al. (1997, Kap. 2)	<code>\citet{LGC97}</code>	<code>\\</code>
Goossens et al. (siehe 1997, Kap. 2)	<code>\citet[Kap.~2]{LGC97}</code>	<code>\\</code>
nur Vortext: Goossens et al. (siehe 1997)	<code>\citet[siehe][Kap.~2]{LGC97}</code>	<code>\\</code>
(Goossens et al., 1997)	<code>nur Vortext: \citet[siehe] []{LGC97}</code>	<code>\\[5pt]</code>
(Goossens et al., 1997, Kap. 2)	<code>\citep{LGC97}</code>	<code>\\</code>
(Goossens et al., 1997, Kap. 2)	<code>\citep[Kap.~2]{LGC97}</code>	<code>\\</code>
(siehe Goossens et al., 1997, Kap. 2)	<code>\citep[siehe][Kap.~2]{LGC97}</code>	<code>\\</code>
nur Vortext: (siehe Goossens et al., 1997)	<code>nur Vortext: \citep[siehe] []{LGC97}</code>	

 Bsp.
12-3-5

Beide Befehle verfügen über Sternformen, `\citet*` und `\citep*` (mit ansonsten gleicher Syntax), die alle Autoren auflisten, sofern die entsprechenden Daten verfügbar sind.² Diese Formen funktionieren jedoch nur, wenn die Funktion von dem verwendeten B^BT_EX-Stil unterstützt wird. Anders ausgedrückt müssen die Daten im optionalen Argument von `\bibitem` zur Verfügung stehen, andernfalls wird immer die gekürzte Liste ausgegeben.

	<code>\usepackage{natbib}</code>	
	<code>\bibliographystyle{lb2exa}</code>	
Goossens, Rahtz, und Mittelbach (1997)	<code>\citet*[LGC97]</code>	<code>\\</code>
(siehe Goossens, Rahtz, und Mittelbach, 1997)	<code>\citep*[siehe] []{LGC97}</code>	

 Bsp.
12-3-6

Ein Vorgriff auf
Kapitel 13

Im Unterschied zu den ersten Beispielen in diesem Kapitel, sind die generierten Texte („und“ im aktuellen Fall) diesmal korrekt ins Deutsche übersetzt.

¹Genauer gesagt wird `\cite` dann zu `\citet` umdefiniert, wenn man das `natbib`-Paket im Autor-Jahr-Modus einsetzt, um den es ja in diesem Abschnitt geht. Im Autor-Nummer-Modus (siehe Abschnitt 12.4.1) verhält er sich wie `\citep`.

²Wenn man gleichzeitig das `jurabib`-Paket verwenden möchte (siehe Abschnitt 12.5.1), sollte man die Sternformen meiden, da sie von diesem Paket nicht unterstützt werden.

Dazu wurde `lb2exa`, ein mit `custom-bib` (siehe Abschnitt 13.5.2 auf Seite 828) erzeugter und auf die deutsche Sprache angepasster `BBTEX`-Stil, verwendet. Dieser Stil kommt auch in den weiteren Beispielen zum Tragen, sofern nichts anderes angegeben ist.

Es gibt noch zwei weitere Varianten der Befehle: `\citealt` funktioniert genau wie `\citet`, erzeugt jedoch keine Klammern, und `\citealp` entspricht `\citep` ohne Klammern. Augenscheinlich sind einige der gesetzten Ergebnisse nahezu identisch.

Bsp.
12-3-7

	<code>\usepackage{natbib}</code>	
Goossens et al. 1997	<code>\citealt{LGC97}</code>	<code>\\</code>
Goossens et al., 1997	<code>\citealp{LGC97}</code>	<code>\\</code>
Goossens, Rahtz, und Mittelbach 1997	<code>\citealt*[LGC97]</code>	<code>\\</code>
Goossens, Rahtz, und Mittelbach, 1997	<code>\citealp*[LGC97]</code>	<code>\\</code>
Goossens und Rahtz, 1999, S. 236 etc.	<code>\citealp[S.~236]{LWC99}</code>	<code>etc.</code>

Beim Autor-Jahr-Schema möchte man gelegentlich nur den Autor (bzw. die Autoren) oder das Jahr angeben. Dazu verfügt das `natbib`-Paket über die folgenden zusätzlichen Befehle (`\citeauthor*` ist mit `\citeauthor` identisch, wenn keine vollständigen Autorenangaben vorliegen):

Bsp.
12-3-8

	<code>\usepackage{natbib}</code>	
Goossens et al.	<code>\citeauthor{LGC97}</code>	<code>\\</code>
Goossens, Rahtz, und Mittelbach	<code>\citeauthor*[LGC97]</code>	<code>\\</code>
1997 und (1997)	<code>\citeyear{LGC97}</code>	<code>und \citeyearpar{LGC97}</code>

Mithilfe des Befehls `\citetext` lassen sich noch komplexere Kombinationen aus Quellenverweisen und Texten zusammenstellen. Er verfügt über ein obligatorisches Argument, das mit den gleichen Klammern umgeben wird, wie sie die anderen Befehle für Quellenverweise verwenden. Durch Kombination dieses Befehls mit `\citealp` oder anderen Befehlen, die keine Klammern setzen, lassen sich verschiedenste Verweistexte zusammenstellen.

Bsp.
12-3-9

(siehe Goossens et al., 1997	<code>\usepackage{natbib}</code>
oder Knuth, 1986)	<code>\citetext{siehe \citealp{LGC97} oder \citealp{Knuth-CT-a}}</code>

Es kann vorkommen, dass ein Quellenverweis am Satzanfang steht und der Name des (ersten) Autors des zitierten Werkes mit einem Kleinbuchstaben beginnt. Für diesen Fall sind die bisher besprochenen Befehle unbrauchbar. Das `natbib`-Paket löst dieses Problem, indem es für alle Befehle Varianten bereitstellt, die den ersten Buchstaben immer groß schreiben. Sie sind leicht zu merken: Der Befehlsname entspricht jeweils dem des Originalbefehls, jedoch mit großem Anfangsbuchstaben. So erhält man beispielsweise `\Citet*` anstelle von `\citet*`. Es folgen einige weitere Beispiele:

 *Groß geschriebene
Namen erzwingen*

Bsp.
12-3-10

	<code>\usepackage{natbib}</code>
Normaler Quellenverweis: van Leunen (92)	Normaler Quellenverweis: <code>\citet{vLeunen:92}</code> <code>\\</code>
Van Leunen (92) oder Van Leunen 92	<code>\Citet{vLeunen:92}</code> oder <code>\Citealt{vLeunen:92}</code> <code>\\</code>
(Van Leunen, 92) oder Van Leunen, 92	<code>\Citep{vLeunen:92}</code> oder <code>\Citealp{vLeunen:92}</code> <code>\\</code>
Van Leunen	<code>\Citeauthor{vLeunen:92}</code>

Als letzte Zugabe kann man in `natbib` Alternativtexte für Quellenverweise formulieren, um diese anstelle der normalen Autor-Jahr-Kombination zu verwenden. Sie werden mit `\defcitealias` definiert (normalerweise in der Präambel) und über `\citetalias` oder `\citepalias` abgerufen.

```

\usepackage{natbib}
\defcitealias{LGC97}{Dogbook-II}

\citet{LGC97} = \citetalias{LGC97}  \
\citep{LGC97} = \citepalias{LGC97}  \
\defcitealias{LGC97}{Dogbook-II~2. Auf\1.}
Alias geändert: \citepalias[siehe] [] {LGC97}

```

Goossens et al. (1997) = Dogbook II
 (Goossens et al., 1997) = (Dogbook II)
 Alias geändert: (siehe Dogbook II 2.Aufl.)

Bsp.
12-3-11

Mit den in diesem Abschnitt vorgestellten Befehlen bietet das `natbib`-Paket (mit kleinen Abweichungen) den gleichen Funktionsumfang wie andere Lösungen für das Autor-Jahr-Schema (z.B. die in Abschnitt 12.3.1 beschriebenen Pakete). Zusätzlich bietet es Funktionen an, die man sonst nirgendwo findet. Andererseits verfügt `natbib` in einigen Fällen nicht direkt über genau entsprechende Befehle. So gibt es z.B. für den `\possessivecite`-Befehl des `harvard`-Paketes (der in Beispiel 12-3-4 gezeigt wurde) keine unmittelbare Entsprechung im `natbib`-Paket, er lässt sich jedoch leicht nachbauen. Man kann ihn entweder direkt mit `\citeauthor` und `\citeyearpar` nachahmen, wie in der ersten Zeile des nächsten Beispiels geschehen, oder einen eigenen Befehl dafür definieren, wenn diese Konstruktion häufiger benötigt wird (in der deutschen Sprache würde der Apostroph wegfallen).

```

\usepackage{natbib} \bibliographystyle{agsm}
\newcommand\possessivecite[1]{\citeauthor{#1}'s \citeyearpar{#1}}

\citeauthor{Knuth-CT-a}'s \citeyearpar{Knuth-CT-a}  \
\possessivecite{Knuth-CT-a}

```

Knuth's (1986)
 Knuth's (1986)

Bsp.
12-3-12

Mehrfache Quellenverweise

In Standard- $\text{\LaTeX}$ kann man mehrfache Quellenverweise vornehmen, indem man dem `\cite`-Befehl mehrere Zitierschlüssel im *schlüssel-liste*-Argument übergibt. Das gleiche Verfahren lässt sich auch für die Befehle `\citet` und `\citep` (sowie ihre Varianten) anwenden. Das `natbib`-Paket prüft dann automatisch, ob benachbarte Verweise in der *schlüssel-liste* die gleichen Autoren beinhalten. In diesem Fall werden die Namen der Autoren nur einmal ausgegeben. Dazu müssen die Autorennamen natürlich identisch geschrieben sein. Für das `natbib`-Paket sind z.B. „D. Knuth“ und „Donald Knuth“ zwei verschiedene Autoren.

```

\usepackage{natbib}

\citet{LGC97,LWC99}  \
\citep{LGC97,LWC99}  \
\citep{Knuth:TB10-1,Knuth-CT-a}

```

Goossens et al. (1997); Goossens und Rahtz (1999)
 (Goossens et al., 1997; Goossens und Rahtz, 1999)
 (Knuth, 1989, 1986)

Bsp.
12-3-13

Die letzte Zeile des vorigen Beispiels verdeutlicht ein mögliches Problem, wenn man in einem Verweisbefehl mehrere Schlüssel verwendet: Die Verweise werden in der Reihenfolge gesetzt, wie sie in der *schlüssel-liste* erscheinen.

Mithilfe der Option `sort` werden die Quellenverweise nach ihrer Reihenfolge in der Bibliographie sortiert, normalerweise zuerst nach Autor und dann nach dem Erscheinungsjahr.

Bsp.
12-3-14

(Knuth, 1986, 1989)

```
\usepackage[sort]{natbib}
\citep{Knuth:TB10-1,Knuth-CT-a}
```

Auch wenn alle Quellenverweisbefehle mehr als einen Zitierschlüssel im *schlüssel-liste*-Argument unterstützen, beschränkt man sich hierzu doch besser auf den Befehl `\citep`. Schon `\citet` erzeugt eher fragwürdige Ergebnisse. Die Lage verschlimmert sich noch durch die Verwendung der optionalen Argumente: Der Befehl `\citet` setzt jeden zusätzlichen *vor*-Text vor jedes Jahr (was man durchaus als Fehler des Paketes bezeichnen könnte). Allgemeiner gesagt wird nicht so richtig deutlich, auf was sich diese ergänzenden Texte eigentlich beziehen sollen. Daher ist es besser die Quellenverweise zu trennen, wenn man noch ergänzenden Text hinzufügen möchte.

Bsp.
12-3-15

(siehe Southall, 1992; Knuth, 1986, S. 55)
(siehe Knuth, 1986, 1989, S. 55)
Southall (siehe 1992); Knuth (siehe 1986,
S. 55)
Knuth (siehe 1986, 1989, S. 55)

```
\usepackage{natbib}
\citep[siehe][S.~55]{Southall,Knuth-CT-a} \\
\citep[siehe][S.~55]{Knuth-CT-a,Knuth:TB10-1} \\
\citet[siehe][S.~55]{Southall,Knuth-CT-a} \\
\citet[siehe][S.~55]{Knuth-CT-a,Knuth:TB10-1}
```

Nur beim ersten Verweis alle Autoren auflisten

Das *harvard*-Paket führt im ersten Quellenverweis zu einem Werk automatisch alle Autoren an, bei nachfolgenden Verweisen dann nur noch eine abgekürzte Liste. Dieser Stil ist in manchen Disziplinen sehr beliebt und wird vom *natbib*-Paket durch die Option `longnamesfirst` unterstützt. Man vergleiche das nächste Beispiel mit dem Beispiel 12-3-4 auf Seite 725.

Bsp.
12-3-16

(Goossens, Rahtz & Mittelbach 1997) erster Verweis
(Goossens et al. 1997) zweiter
(Goossens, Rahtz & Mittelbach 1997) erzwungen
Goossens et al. (1997)
(e.g., Goossens et al. 1997)
Goossens et al.

```
\usepackage[longnamesfirst]{natbib}
\bibliographystyle{agsm}
\citep{LGC97} \hfill erster Verweis \\
\citep{LGC97} \hfill zweiter \\
\citep*{LGC97}\hfill erzwungen \\
\citet{LGC97} \\
\citep[e.g.,][{}]{LGC97} \\
\citeauthor{LGC97}
```

Einige *BiBTeX*-Stildateien sind recht ausgeklügelt programmiert. Der *BiBTeX*-Stil *agsm* aus dem letzten Beispiel verweigert z.B. die Abkürzung der Liste, wenn er bemerkt, dass eine verkürzte Autorenliste zu einem mehrdeutigen Verweis führen würde. Nachdem das Beispiel um den Quellenverweis `test97` ergänzt wurde, erschienen alle Verweise plötzlich nur noch in der Langform.¹ *BiBTeX*-Stile, die mit `makebst` erzeugt werden, vermeiden mehrdeutige Verweise, indem sie das Jahr um ein Suffix ergänzen. Andere *BiBTeX*-Stile, wie

¹Dieses Verhalten verblüffte den Autor ziemlich, als er es beim Vorbereiten der Beispiele zum ersten Mal bemerkte.

zum Beispiel `chicago`), erzeugen diese jedoch ohne Bedenken (vergleiche Beispiel 12-3-18 unten).

		<code>\usepackage[longnamesfirst]{natbib}</code>
		<code>\bibliographystyle{agsm}</code>
(Goossens, Rahtz & Mittelbach 1997)	erster Verweis	<code>\citep{LGC97} \hfill erster Verweis \\\</code>
(Goossens, Rahtz & Mittelbach 1997)	zweiter	<code>\citep{LGC97} \hfill zweiter \\\</code>
(Goossens, User, Doe et al. 1997)	erster Verweis	<code>\citep{test97}\hfill erster Verweis \\\</code>
(Goossens, User, Doe et al. 1997)	zweiter	<code>\citep{test97}\hfill zweiter</code>

Bsp.
12-3-17

Einige Publikationen haben so viele Autoren, dass man sie immer nur mit einer verkürzten Liste zitieren möchte, selbst beim ersten Vorkommen. Dazu muss man ihre Zitierschlüssel durch Kommas getrennt im Argument der `\shortcites`-Deklaration aufführen. Das nächste Beispiel zeigt auch, dass die Verwendung des `chicago`-Stils zu mehrdeutigen Quellenverweisen führen kann (vgl. Zeile 1 und 2 mit Zeile 5).

		<code>\usepackage[longnamesfirst]{natbib}</code>
		<code>\bibliographystyle{chicago}</code>
		<code>\shortcites{LGC97}</code>
(Goossens et al., 1997)	erster Verweis	<code>\citep{LGC97} \hfill erster Verweis \\\</code>
(Goossens et al., 1997)	zweiter	<code>\citep{LGC97} \hfill zweiter \\\</code>
(Goossens, Rahtz, and Mittelbach, 1997)	erzwungen	<code>\citep*{LGC97}\hfill erzwungen \\\</code>
(Goossens, User, Doe, et al., 1997)	erster Verweis	<code>\citep{test97}\hfill erster Verweis \\\</code>
(Goossens et al., 1997)	zweiter	<code>\citep{test97}\hfill zweiter</code>

Bsp.
12-3-18

Anpassen des Layouts der Quellenverweise

Bisher zeigten alle Verweise in den Beispielen runde Klammern. Das ist jedoch bei Weitem nicht die einzige Möglichkeit, die das `natbib`-Paket bietet. Intern kennt das Paket mehr als 20 verschiedene $\LaTeX$ -Stile. Wird einer davon mit einem `\bibliographystyle`-Befehl ausgewählt, dann sucht das Paket auch ein dazu passendes Layout für die Quellenverweise aus. Mit dem Stil `agu` (American Geophysics Union) erhält man beispielsweise folgendes Ergebnis:

<i>Goossens et al.</i> [1997]	<code>\usepackage{natbib} \bibliographystyle{agu}</code>
[<i>Knuth</i> , 1986; <i>Goossens and Rahtz</i> , 1999]	<code>\citett{LGC97} \\\ \citep{Knuth-CT-a,LWC99} \\\</code>
[siehe <i>Knuth</i> , 1986, Kap. 2]	<code>\citep[siehe] [Kap.~2]{Knuth-CT-a}</code>

Bsp.
12-3-19

Gemäß der Voreinstellung wird das Layout für die Verweise durch den ausgewählten $\LaTeX$ -Stil bestimmt, oder durch die Vorgaben des `natbib`-Paketes, sofern es einen bestimmten Stil nicht kennt. Durch Hinzufügen einer `\citestyle`-Deklaration kann man für die Quellenverweise einen anderen $\LaTeX$ -Stil verwenden, als den der Bibliographie. Im nächsten Beispiel wird für die Quellenverweise der Stil `agsm` verwendet, während `agu` der Hauptstil bleibt. Bei einem Vergleich dieses Beispiels mit Beispiel 12-3-19 sieht man, dass die Formatierung des Textes erhalten bleibt (zum Beispiel Autornamen

in Kursivschrift), die Klammern und die Trennung zwischen Autoren und Jahr haben sich jedoch geändert.

Bsp.
12-3-20

Goossens et al. (1997)

(*Knuth* 1986, *Goossens and Rahtz* 1999)

(siehe *Knuth* 1986, Kap. 2)

```
\usepackage{natbib} \bibliographystyle{agu}
\citestyle{agsm}
\citet{LGC97} \ \ \citep{Knuth-CT-a,LWC99} \ \
\citep[siehe] [Kap.~2]{Knuth-CT-a}
```

Man kann das Layout auch über Optionen beeinflussen: Die Optionen `round` (Voreinstellung der meisten Stile), `square`, `curly` oder `angle` ändern die Art der verwendeten Klammern, während `colon`¹ (Voreinstellung der meisten Stile) und `comma` das Trennzeichen zwischen mehrfachen Quellenverweisen beeinflussen. Im nächsten Beispiel werden die Voreinstellungen des `agu`-Stils durch Laden des `natbib`-Paketes mit zwei Optionen überschrieben; vergleiche das Ergebnis mit Beispiel 12-3-19.

Bsp.
12-3-21

Goossens et al. {1997}

{*Knuth*, 1986, *Goossens and Rahtz*, 1999}

{siehe *Knuth*, 1986, Kap. 2}

```
\usepackage[curly,comma]{natbib}
\bibliographystyle{agu}
\citet{LGC97} \ \ \citep{Knuth-CT-a,LWC99} \ \
\citep[siehe] [Kap.~2]{Knuth-CT-a}
```

Eine weitere Methode zum Anpassen des Layouts, die sich hauptsächlich an Entwickler von Paketen und/oder Klassendateien richtet, ist die Deklaration `\bibpunct`. Sie verfügt über sieben Argumente (von denen das erste optional ist) zum Definieren verschiedener Aspekte eines Formates für Quellenverweise. Sie wird typischerweise verwendet, um das für einen bestimmten `LaTeX`-Stil voreingestellte Layout eines Quellenverweises zu definieren. So enthält das `natbib`-Paket z.B. viele Definitionen wie die folgende:

```
\newcommand\bibstyle@chicago{\bibpunct{(){}}{;}{a}{,}{,}{,}}
```

Diese Definition wird ausgewählt, wenn man `chicago` als `LaTeX`-Stil festlegt oder dem Befehl `\citestyle` als Argument übergibt. Ähnliche Deklarationen lassen sich auch für `LaTeX`-Stile hinzufügen, die `natbib` nicht direkt unterstützt. Dies geht am einfachsten, indem man solche Deklarationen in der lokalen Konfigurationsdatei `natbib.cfg` sammelt. Nähere Einzelheiten zur Bedeutung der Argumente kann man der Dokumentation entnehmen, die dem `natbib`-Paket beiliegt.

Bei sich überschneidenden Angaben greifen die folgenden Regeln: Interne `\bibstyle@<name>`-Deklarationen erhalten die geringste Priorität, gefolgt von den Optionen der `\usepackage`-Deklarationen. Beide werden durch eine explizite `\bibpunct`- oder `\citestyle`-Deklaration in der Präambel überschrieben.

Normalerweise lässt das `natbib`-Paket einen Zeilenumbruch in der Autorenlis- te eines Quellenverweises zu. Mithilfe der Option `nonamebreak` kann man festlegen, dass alle Autorennamen eines einzelnen Verweises in einer Zeile erscheinen. Dies ist zwar normalerweise keine gute Idee, da auf diese Weise leicht überlaufende Boxen entstehen können, andererseits lassen sich so jedoch einige `hyperref`-Probleme lösen.

*Alle Autorennamen in
eine Zeile zwingen*

¹Trotz ihres Namens erzeugt diese Option ein Semikolon.

Anpassen des Layouts der Bibliographie

Die Umgebung `thebibliography` des `natbib`-Paketes fügt vor dem Literaturverzeichnis automatisch eine Überschrift ein. Entsprechend seiner Voreinstellung wählt `natbib` hierzu eine nicht nummerierte Überschrift der höchsten Gliederungsebene, wie etwa `\chapter*` für `book`-ähnliche Klassen oder `\section*` für die Klasse `article` und ihre Varianten. Die eigentliche Überschrift wird in dem Befehl `\bibsection` gespeichert. Um die Voreinstellungen zu ändern muss man also diesen Befehl umdefinieren. Dadurch lässt sich z.B. die Überschrift unterdrücken oder man wählt eine nummerierte Form.

Das `natbib`-Paket unterstützt auch einen Sonderfall: Mit der Option `sectionbib` weist man das Paket an, die Überschrift immer mit `\section*` zu bilden, auch wenn die höchste Gliederungsebene eigentlich `\chapter` wäre. Diese Option erweist sich als nützlich, wenn man die Pakete `natbib` und `chapterbib` miteinander kombiniert (siehe Abschnitt 12.6.1).

Zwischen `\bibsection` und dem Anfang der Auflistung führt `natbib` den Befehl `\bibpreamble` aus, sofern dieser definiert ist. Mit seiner Hilfe kann man zwischen der Überschrift und dem eigentlichen Literaturverzeichnis noch zusätzlichen Text einfügen. Durch Definieren des Befehls `\bibfont` kann man außerdem die Schrift der Bibliographie festlegen. Mit seiner Hilfe lässt sich das Verzeichnis auch noch anders abwandeln, etwa indem man es mit `\raggedright` im Flattersatz setzt. Man beachte, dass sowohl `\bibpreamble` als auch `\bibfont` zunächst nicht definiert sind, so dass man sie erst mit `\newcommand` einrichten muss. Der Befehl `\bibsection` hingegen wird mit `\renewcommand` umdefiniert.

Schließlich stehen noch zwei Längenparameter zur Anpassung zur Verfügung. Die erste Zeile in jeder Quellenangabe wird linksbündig gesetzt, alle weiteren Zeilen werden um den in `\bibhang` gespeicherten Wert (voreingestellt auf `1em`) eingerückt. Der vertikale Abstand zwischen den einzelnen Einträgen ist in der elastischen Länge `\bibsep` gespeichert. Ihr voreingestellter Wert orientiert sich normalerweise an der Länge `\itemsep`, die für andere Listen definiert wird.

Um die verschiedenen Möglichkeiten zu veranschaulichen, wird hier Beispiel 12-1-2 auf Seite 709 wiederholt, jedoch mit verschiedensten Anpassungen (die nicht unbedingt immer eine Verbesserung darstellen!). Man beachte, dass am Ende von `\bibpreamble` der Befehl `\par` gesetzt wird. Ohne ihn würden sich die Einstellungen in `\bibfont` auch auf den eingefügten Text auswirken! Das Problem der Konvertierung von Titeln in Kleinbuchstaben besteht weiterhin.

```
\usepackage{natbib}
\bibliographystyle{abbrvnat}
\renewcommand\bibsection{\section{\refname}}
\newcommand\bibpreamble{Zusätzlicher Text zwischen Überschrift
                        und Verzeichnis.\par}
\newcommand\bibfont{\footnotesize\raggedright}
\setlength\bibhang{30pt} \setlength\bibsep{1pt plus 1pt}
```

Einträge mit mehreren Autoren könnten problematisch sein, wie z.B. `\cite{LGC97}` und `\cite{test97}` oder sogar `\cite{LGC97,test97}`. Oder auch nicht.

```
\bibliography{tex}
```


Einträge mit mehreren Autoren könnten problematisch sein, wie z.B. Goossens et al. [1997a] und Goossens et al. [1997b] oder sogar Goossens et al. [1997a,b]. Oder auch nicht.

1 Literatur

Zusätzlicher Text zwischen Überschrift und Verzeichnis.

M. Goossens, S. Rahtz, and F. Mittelbach. *The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997a. ISBN 0-201-85469-4.

M. Goossens, B. User, J. Doe, et al. Mehrdeutige quellenverweise. Eingereicht bei IBM J. Res. Dev., 1997b.

Bsp.
12-3-22

Publikationen ohne Autoren- oder Jahresdaten

Um das Autor-Jahr-Zitierschema verwenden zu können, müssen die Einträge im Literaturverzeichnis die benötigten Daten enthalten. Bei fehlenden Daten können Verweise mit dem Befehl `\citet` oder seinen Varianten seltsame Ergebnisse liefern.

Verfügt die Veröffentlichung nicht über einen Autor, sondern nur über einen Herausgeber, dann greifen die meisten L^AT_EX-Stile auf Letzteren zurück. Fehlen jedoch beide, dann werden sehr unterschiedliche Lösungsansätze verfolgt. B^AT_EX-Dateien im „Harvard“-Stil (wie z.B. `agsm`) verwenden die ersten drei Buchstaben des „key“-Feldes (Schlüssel), sofern eines existiert. Ansonsten benutzen sie die ersten drei Buchstaben des Feldes „organization“ (wobei Zeichenfolgen wie „The_“ ignoriert werden), oder sie führen den ganzen Titel an. Bei fehlendem Erscheinungsjahr wird der Text „n.d.“ (no date) verwendet. Auf diese Weise erhält man verwertbare Einträge, außer wenn nur ein Teil des „key“-Feldes ausgewählt wird:

Bsp.
12-3-23

Koppitz (n.d.) / *TUGboat The Communications of the T_EX User Group* (1980ff) / mak (2000)

```
\usepackage{natbib} \bibliographystyle{agsm}
\citet{G-G} / \citet{oditty} / \citet{GNUMake}
```

Mit `makebst` erzeugte B^AT_EX-Stile (wie etwa `unsrtnat`) verfolgen bei den gleichen Einträgen die folgende Strategie: Sofern ein „key“-Feld (Schlüssel) vorhanden ist, wird dieses komplett als „Autor“ angeführt. Ansonsten werden die ersten drei Buchstaben eines vorhandenen „organization“-Feldes verwendet (wobei Text wie „The_“ ignoriert wird). Ansonsten erscheinen die ersten drei Buchstaben des Verweislables. Ein fehlendes Erscheinungsjahr wird einfach ausgelassen. Bei im Text eingebetteten Verweisen wird so nur der Name des Autors aufgeführt. In solchen Fällen, oder wenn das Schlüsselfeld benutzt wird, sollte man den Befehl `\citet` wohl am besten zugunsten von `\citep` vermeiden, damit für den Leser deutlicher zu erkennen ist, dass man wirklich auf eine Quelle verweist und nicht nur beiläufig einen Autor erwähnt.

Bsp.
12-3-24

Koppitz / odd [1980ff] / make
[Koppitz] / [odd, 1980ff] / [make]

```
\usepackage{natbib} \bibliographystyle{unsrtnat}
\citet{G-G} / \citet{oditty} / \citet{GNUMake} \\\
\citep{G-G} / \citep{oditty} / \citep{GNUMake}
```


Als letztes Beispiel folgen die Ergebnisse, die der BibTeX-Stil `chicago` erzeugt. Der Verweis auf das GNU-Handbuch ist hier sehr schön (der volle Name der Organisation erscheint), aber der Eintrag mit fehlendem Datum sieht seltsam aus.

Koppitz (Koppitz) / odd (80ff) / Free Software Foundation (2000)

```
\usepackage{natbib} \bibliographystyle{chicago}
```

(Koppitz, Koppitz) / (odd, 80ff) / (Free Software Foundation, 2000)

```
\citest{G-G} / \citest{oddiy} / \citest{GNUMake} \\\citep{G-G} / \citep{oddiy} / \citep{GNUMake}
```

Bsp.
12-3-25

Erzwingen des Autor-Jahr-Stils

Das `natbib`-Paket ist so voreingestellt, dass es in Kombination mit den meisten BibTeX-Stilen Autor-Jahr-Verweise erzeugt. Man kann dieses Schema zudem explizit erzwingen, indem man das Paket mit der Option `authoryear` lädt.

Damit dieses Verfahren funktioniert, muss der BibTeX-Stil jedoch auch Autor-Jahr-Daten an das Dokument zurückmelden. Ausschließlich für numerische Verweise entwickelte `.bst`-Dateien, wie etwa das L^AT_EX-eigene `plain`, können diese Daten jedoch nicht liefern. In solchen Fällen ignoriert das `natbib`-Paket die `authoryear`-Option und gibt, wenn man den Befehl `\citest` oder eine seiner Varianten verwendet, Warnungen über fehlende Autordaten und ähnliche Verweise wie die folgenden aus:

(author?) [3] / (author?) [1] / (author?) [2]

```
\usepackage{natbib} \bibliographystyle{plain} \\\citest{G-G} / \citest{oddiy} / \citest{GNUMake}
```

Bsp.
12-3-26

Hier wechselt man am besten zu einem BibTeX-Stil, welcher das Autor-Jahr-Schema unterstützt, wie z.B. `plainnat` anstelle von `plain`.

Automatisches Indexieren von Quellenverweisen

Ab jeder Stelle in einem Dokument kann man Quellenverweise mithilfe des Befehls `\citeindextrue` in den Index aufnehmen. Von dieser Stelle bis zum nächsten `\citeindexfalse`-Befehl (oder dem Ende der aktuellen Gruppe) erzeugen alle Varianten der Befehle `\citest` und `\citep` Einträge in der Indexdatei (sofern eine solche erstellt wird). Solange `\citeindextrue` aktiv ist, erzeugen auch `\bibitem`-Befehle in der `thebibliography`-Umgebung Indexeinträge. Wenn dies nicht erwünscht ist, muss man vor Beginn der Umgebung (zum Beispiel bevor man `\bibliography` aufruft) den Befehl `\citeindexfalse` ausgeben.

Das Indexformat wird durch den internen Befehl `\NAT@idxtxt` gesteuert. Seine Definition ist folgendermaßen vorgegeben:

```
\newcommand\NAT@idxtxt{\NAT@name\ \NAT@open\NAT@date\NAT@close}
```

Entsprechend erzeugt er Einträge wie „Knuth (1986)“. Verweise ohne Autoren- oder Jahresdaten erscheinen höchstwahrscheinlich immer etwas seltsam. Die Quellenverweise aus Beispiel 12-3-24 erzeugen die folgenden Einträge:

```
\indexentry{{Koppitz}\ []}{6} \indexentry{{odd}\ [1980ff]}{6} \indexentry{{make}\ []}{6}
```

Damit der Befehl beispielsweise nur den Namen des Autors ausgibt, kann man ihn in der Datei `natbib.cfg` oder in der Dokumentenpräambel undefinieren. In letzterem Falle darf man die Befehle `\makeatletter` und `\makeatother` nicht vergessen!

Mithilfe des Paketes `index` von David Jones (siehe Abschnitt 11.4.3) kann man auch einen separaten Index für Quellenverweise erzeugen. Durch den Befehl `\newindex` lassen sich zudem mehrere Indexe erstellen. Dazu muss man in der Präambel zunächst den Index deklarieren und dann die automatische Indexierung von Quellenverweisen mit diesem Index verknüpfen:

```
\usepackage{index}
\newindex{default}{idx}{ind}{Index}           % der Hauptindex
\newindex{cite}{cdx}{cnd}{Index der Quellenverweise}
\renewcommand\citeindextype{cite}
```

Später bestimmt man mit dem Befehl `\printindex[cite]`, wo im Dokument der Index für Quellenverweise ausgegeben werden soll.

BibTeX-Stile für natbib

Wie bereits in der Einleitung erwähnt, wurde das `natbib`-Paket für verschiedene BibTeX-Stile entwickelt, die in irgendeiner Weise mit dem Autor-Jahr-Zitierschema arbeiten. Zusätzlich zu diesen Stilen Dritter unterstützt `natbib` alle Stile, die sich mit der `custom-bib`-Sammlung erzeugen lassen (siehe Abschnitt 13.5.2 auf Seite 828). Selbst enthält es die drei Stile `abbrvnat`, `plainnat` und `unsrtnat`, die Erweiterungen der entsprechenden Standardformate darstellen. Sie wurden so an `natbib` angepasst, dass man einige der Funktionen des Paketes verwenden kann, die anderweitig nicht verfügbar wären. Diese Stile unterstützen außerdem einige zusätzliche Felder, die sich im Zeitalter der elektronischen Veröffentlichungen als nützlich erwiesen:

- `doi` Für elektronische Zeitschriften und ähnliche Materialien. Der Digital Object Identifier (DOI) ist ein System zum Identifizieren und Austauschen von geistigem Eigentum in digitalen Umgebungen und ist wahrscheinlich robuster als eine URL (nähere Einzelheiten unter <http://www.doi.org>). Das Feld ist optional.
- `eid` Da elektronische Zeitschriften normalerweise keine Seitenzahlen enthalten, verwenden sie eine besondere Kennzeichnung für die Reihenfolge (die EID) um Artikel in einer Zeitschrift zu lokalisieren. Das Feld ist optional und wird, sofern es existiert, anstelle der Seitenzahl verwendet.
- `isbn` Die internationale Standardbuchnummer (ISBN) ist eine zehnstellige eindeutige Identifikationsnummer (siehe www.isbn.org). Die ISBN ist in ISO-Standard 2108 definiert und ist schon seit mehr als 30 Jahren im Gebrauch. Das Feld ist optional.
- `issn` Die internationale Standardseriennummer (ISSN) ist eine achtstellige Identifikationsnummer für regelmäßig erscheinende Veröffentlichungen (siehe www.issn.org). Das Feld ist optional.
- `url` Der Uniform Resource Locator (URL) zur Identifikation von Quellen im Worldwide Web. Das Feld ist optional. Da URL-Adressen normalerweise

sehr lang sein können und in Schreibmaschinenschrift gesetzt werden, kann es zu Problemen mit dem Zeilenumbruch kommen. Daher werden sie automatisch mit einem `\url`-Befehl umgeben, der eine einfache vorgegebene Definition erhält, sofern er noch nicht definiert ist. Mithilfe des `url`-Paketes (siehe Abschnitt 3.1.8) kann man so die Handhabung von Zeilenumbrüchen erheblich verbessern, da sich URLs dann auch an Interpunktionszeichen umbrechen lassen.

12.3.3 bibentry – Bibliographieeinträge im Fließtext

Anstatt alle zitierten Werke in einem Literaturverzeichnis zusammenzufassen ist es manchmal auch erforderlich, direkt alle Daten anzugeben, wenn eine Quelle das erste Mal zitiert wird. Zu diesem Zweck entwickelte Patrick Daly passend zum `natbib`-Paket das `bibentry`-Paket.

<code>\nobibliography{BIBTEX-datenbank-liste}</code>	<code>\bibentry{schlüssel}</code>

Dieser Befehl funktioniert folgendermaßen: Anstelle des üblichen `\bibliography`-Befehls, der die von `BIBTEX` geschriebene `.bbl`-Datei lädt und die Bibliographie setzt, verwendet man `\nobibliography` mit der gleichen Liste von `BIBTEX`-Datenbankdateien. Dieser Befehl liest die `.bbl`-Datei und verarbeitet die Daten, so dass an anderen Stellen im Dokument Verweise auf Einträge erfolgen können. Ein Quellenverweis mit vollständigen bibliographischen Daten wird mit dem Befehl `\bibentry` erzeugt. Die herkömmlichen Autor-Jahr-Verweise lassen sich mit jedem der `natbib`-Befehle ausgeben. Dazu folgendes Beispiel:

Nähere Einzelheiten in Knuth, D. E.,
Typesetting Concrete Mathematics, *TUGboat*,
10, 31–36, 1989. Allgemeine Informationen
finden sich in Knuth, D. E., *The TeXbook*,
vol. A of *Computers and Typesetting*,
Addison-Wesley, Reading, MA, USA, 1986.
Wie in *Knuth* [1989] gezeigt ...

```
\usepackage{bibentry,natbib}
\bibliographystyle{agu}
\raggedright \setlength\parindent{12pt}
\nobibliography{tex}
Nähere Einzelheiten in \bibentry{Knuth:TB10-1}.
Allgemeine Informationen finden sich in
\bibentry{Knuth-CT-a}.

Wie in \citet{Knuth:TB10-1} gezeigt \ldots
```

Bsp. 12-3-27

Mögliche Fallstricke 

Dabei sind eine Reihe von Punkten zu beachten: Der `\nobibliography`-Befehl muss im Hauptteil des Dokumentes stehen, und zwar vor dem ersten `\bibentry`-Befehl. In der Präambel wird `\nobibliography` völlig ignoriert und ein `\bibentry`-Befehl, der vor `\nobibliography` erfolgt, gibt keine Daten aus. Daher steht dieser Befehl am besten direkt nach `\begin{document}`.

Ein weiteres mögliches Problem ergibt sich aus der Wahl des `BIBTEX`-Stils. Das Paket `bibentry` erfordert eine besondere Struktur der Einträge in der `.bbl`-Datei. Sie müssen durch Leerzeilen voneinander getrennt sein und der `\bibitem`-Befehl muss entweder mit einem Leerzeichen oder einem Zeilenvorschub vom eigentlichen Eintrag abgesetzt werden. Bei `BIBTEX`-Stilen, die mit dem Programm `makebst` erzeugt wurden, wird diese Struktur automatisch erzwungen. Andere `BIBTEX`-Stile können jedoch unbrauchbar sein, auch solche, die ansonsten mit `natbib` funktionieren.

Der `\bibentry`-Befehl entfernt automatisch einen Punkt am Eintragsende, so dass die Einträge auch mitten im Satz aufgeführt werden können. Wenn der Eintrag jedoch weitere Satzzeichen enthält, wie etwa einen Punkt als Teil eines „note“-Feldes, dann kann der endgültige Text immer noch seltsam anmuten. In diesem Fall hilft wohl nur das Anpassen des BibTeX-Datenbankeintrags.

Man kann auch gleichzeitig ein Literaturverzeichnis verwenden und mit `\bibentry` vollständige Quellenangaben im Text erzeugen. Dazu platziert man den `\bibliography`-Befehl an der Stelle, an welcher das Literaturverzeichnis erscheinen soll. Zudem wird direkt nach `\begin{document}` der `\nobibliography*`-Befehl eingefügt. Diese Variante verfügt über kein eigenes Argument, da die BibTeX-Datenbankdateien bereits im `\bibliography`-Befehl festgelegt sind. Dadurch erscheinen alle mit `\bibentry` zitierten Publikationen automatisch auch in der Bibliographie, da auf die gleiche `.bbl`-Datei zugegriffen wird.

12.4 Das Autor-Nummer-Schema

Wie bereits in der Einleitung erwähnt, gibt es derzeit keine BibTeX-Stildatei mit einem Autor-Nummer-Schema, in der die Publikationen jedes Autors einzeln durchnummeriert sind. Wenn die Werke jedoch in der gesamten Bibliographie fortlaufend nummeriert werden sollen, dann bieten BibTeX und das `natbib`-Paket, das bereits im Zusammenhang mit dem Autor-Jahr-Schema besprochen wurde, weit reichende Unterstützung.

12.4.1 Wiedersehen mit dem `natbib`-Paket

Das `natbib`-Paket wurde ursprünglich nur für das Autor-Jahr-Schema entwickelt. Trotzdem kann es ebenso auch Autor-Nummer-Verweise sowie rein numerische Verweise erzeugen. Diese beiden Verweisarten werden mithilfe von BibTeX-Stilen bereitgestellt, die speziell für nummerierte Literaturverzeichnisse entworfen wurden, ähnlich denjenigen BibTeX-Stilen, die normalerweise für Autor-Jahr-Verweise benutzt werden.

Entsprechend seiner Voreinstellung erzeugt das `natbib`-Paket Autor-Jahr-Verweise. Will man jedoch hauptsächlich Quellen nach dem numerischen oder dem Autor-Nummer-Schema zitieren, dann sollte man das `natbib`-Paket mit der Option `numbers` laden.

Zum Vergleich wird hier Beispiel 12-3-5 auf Seite 726 mit geladener Option `numbers` wiederholt. Diese Option bringt automatisch die Optionen `square` und `comma` mit sich. Wer runde Klammern bevorzugt, muss daher die vorgegebene Auswahl mit der Option `round` überschreiben.

Goossens et al. [1]	<code>\usepackage[numbers]{natbib}</code>	
Goossens et al. [1, Kap. 2]	<code>\citet{LGC97}</code>	<code>\\</code>
Goossens et al. [siehe 1, Kap. 2]	<code>\citet[Kap.~2]{LGC97}</code>	<code>\\</code>
nur Vortext: Goossens et al. [siehe 1]	<code>\citet[siehe][Kap.~2]{LGC97}</code>	<code>\\</code>
[1]	nur Vortext: <code>\citet[siehe][]{LGC97}</code>	<code>\\[5pt]</code>
[1, Kap. 2]	<code>\citep{LGC97}</code>	<code>\\</code>
[siehe 1, Kap. 2]	<code>\citep[Kap.~2]{LGC97}</code>	<code>\\</code>
nur Vortext: [siehe 1]	<code>\citep[siehe][Kap.~2]{LGC97}</code>	<code>\\</code>
	nur Vortext: <code>\citep[siehe][]{LGC97}</code>	

Bsp.
12-4-1

Wie man sieht, erzeugt der `\citet`-Befehl nun Quellenverweise im Autor-Nummer-Stil, während `\citep` rein numerische Verweise erstellt. Stellt man `natbib` so ein, dass es numerische Quellenverweise vornimmt, dann verhält sich der $\text{\LaTeX}$ -Befehl `\cite` wie der Befehl `\citep`. Im Autor-Jahr-Modus definiert das `natbib`-Paket ihn zu einer Kurzform für den Befehl `\citet` um.

Alle Varianten von `\citet` und `\citep`, die bereits in Abschnitt 12.3.2 besprochen wurden, sind auch im numerischen Modus verfügbar, obwohl hier nur einige von ihnen sinnvoll sind. So gibt zum Beispiel `\citep*` den gleichen Verweis aus, wie `\citep`, da in der Klammer ohnehin keine Autoren aufgeführt werden.

Goossens, Rahtz, und Mittelbach [1]	<code>\usepackage[numbers]{natbib}</code>	
Goossens et al.	<code>\citet*{LGC97}</code>	<code>\\</code>
Goossens, Rahtz, und Mittelbach	<code>\citeauthor{LGC97}</code>	<code>\\</code>
1997 oder [1997]	<code>\citeauthor*{LGC97}</code>	<code>\\</code>
	<code>\citeyear{LGC97}</code> oder <code>\citeyearpar{LGC97}</code>	

Bsp.
12-4-2

Die Befehle `\citealt` und `\citealt*` sollte man wohl eher meiden, da eine Verweisnummer ohne Klammern sehr wahrscheinlich falsch gedeutet wird. An manchen Stellen kann jedoch `\citealp` hilfreich sein, um die einzelne Nummer zu erhalten, die sich dann etwa mit `\citetext` kombinieren lässt.

Goossens et al. 1	<code>\usepackage[numbers]{natbib}</code>	
Goossens, Rahtz, und Mittelbach 1	<code>\citealt{LGC97}</code>	<code>\\</code>
1	<code>\citealt*{LGC97}</code>	<code>\\</code>
1, S. 236 usw.	<code>\citealp{LGC97}</code>	<code>\\</code>
	<code>\citealp[S.~236]{LGC97}</code> usw.	

Bsp.
12-4-3

In einigen Zeitschriften werden die Nummern der numerischen Quellenverweise hochgestellt. Das `natbib`-Paket unterstützt dieses Verfahren durch Laden der Option `super`. In diesem Fall erzeugt das Standardbeispiel (vgl. Beispiel 12-4-1) folgendes Ergebnis:

Goossens et al. ¹	<code>\usepackage[super]{natbib}</code>	
Goossens et al. ¹ , Kap. 2	<code>\citet{LGC97}</code>	<code>\\</code>
Goossens et al. siehe ¹ , Kap. 2	<code>\citet[Kap.~2]{LGC97}</code>	<code>\\</code>
nur Vortext: Goossens et al. siehe ¹	<code>\citet[siehe][Kap.~2]{LGC97}</code>	<code>\\</code>
1	<code>nur Vortext: \citet[siehe][]{LGC97}</code>	<code>\\[5pt]</code>
1 (Kap. 2)	<code>\citep{LGC97}</code>	<code>\\</code>
1 (Kap. 2)	<code>\citep[Kap.~2]{LGC97}</code>	<code>\\</code>
nur Vortext: 1	<code>\citep[siehe][Kap.~2]{LGC97}</code>	<code>\\</code>
	<code>nur Vortext: \citep[siehe][]{LGC97}</code>	

Bsp.
12-4-4

Wie man sieht, erzeugt das optionale Argument recht fragwürdige Ergebnisse; im Falle von `\citep` erscheint das Argument *vor* gar nicht erst. Bei diesem Zitierformat ist es daher normalerweise am besten, einfach die Grundformen der Befehle zu verwenden.

Für hochgestellte Quellenverweise entfernt `natbib` mögliche Leerzeichen vor den Verweisbefehlen, damit die Nummer direkt am vorhergehenden Wort

erscheint. Im Gegensatz zu den Ergebnissen des cite-Paketes wandern Satzzeichen jedoch nicht vor den Quellenverweis und es gibt auch keine Überprüfung auf doppelte Punkte. Zur Veranschaulichung wird hier noch einmal Beispiel 12-2-11 von Seite 721 wiederholt:

Bsp.
12-4-5

```
... von Knuth2; siehe auch \usepackage[super]{natbib}
Goossens et al.1. \ldots von Knuth~\citep{Knuth-CT-a}; siehe auch \citet{LGC97}.
... von Knuth;2 siehe auch \par %% An zwei Stellen manuell korrigiert:
Goossens et al.1 \ldots von Knuth;\citep{Knuth-CT-a} siehe auch \citet{LGC97}
```

Die Pakete natbib und cite sind leider nicht kompatibel (beide verändern die L^AT_EX-internen Mechanismen für Quellenverweise), so dass man leider wie oben die Quelle ändern muss, wenn natbib verwendet werden soll.

Sortieren und Zusammenfassen numerischer Verweise

Wie in Abschnitt 12.2.2 zu sehen, sortiert das cite-Paket mehrfache Verweise und fasst sie optional zu Bereichen zusammen. Auch das natbib-Paket verfügt über solche Funktionen, die sich mit den Optionen sort bzw. sort&compress aktivieren lassen.

Die Option sort wurde bereits im Zusammenhang mit Autor-Jahr-Verweisen angesprochen. In numerischen Querverweisen (also mit den Optionen numbers und super) werden entsprechend die Nummern sortiert. Zur Veranschaulichung wird hier Beispiel 12-2-5 von Seite 718 noch einmal wiederholt, allerdings dieses Mal ohne den nicht definierten Quellenverweis.

Bsp.
12-4-6

In [1, 2, 3, 4] findet man gute Informationen zu T_EX und L^AT_EX.

```
\usepackage[sort]{natbib} \bibliographystyle{plain}
In \citep{LGC97,LWC99,Knuth-CT-a,Knuth:TB10-1}
findet man gute Informationen zu \TeX{} und \LaTeX{}.
```

Durch die Option sort&compress werden die Nummern nicht nur sortiert, sondern soweit möglich auch zu Bereichen zusammengefasst. Im Autor-Jahr-Modus hat diese Option den gleichen Effekt wie sort.

Bsp.
12-4-7

In [1–4] findet man gute Informationen zu T_EX und L^AT_EX.

```
\usepackage[sort&compress]{natbib}\bibliographystyle{plain}
In \citep{LGC97,LWC99,Knuth-CT-a,Knuth:TB10-1}
findet man gute Informationen zu \TeX{} und \LaTeX{}.
```

Regeln zur Auswahl des numerischen Modus

Wie bereits zuvor erwähnt ist das natbib-Paket auf den Autor-Jahr-Modus voreingestellt. In den letzten beiden Beispielen wählte natbib jedoch den numerischen Modus, ohne dass es durch die Optionen numbers oder super dazu veranlasst wurde. Der Grund liegt darin, dass der B^BT_EX-Stil plain seinen \bibitem-Befehlen keine Autor-Jahr-Daten mitgibt. Immer wenn ein \bibitem-Befehl ohne die erforderlichen Daten vorkommt, wechselt natbib automatisch in den numerischen Modus. Selbst die Option authoryear wird in diesem Falle nicht beachtet.

Unterstützt ein BibTeX-Stil jedoch das Autor-Jahr-Schema, dann kann man mit einem der folgenden Verfahren auf den numerischen Modus umschalten (die Liste ist nach zunehmender Priorität sortiert):

1. Indem man mit dem Befehl `\bibliographystyle` einen Bibliographiestil wählt, der ein numerisches Format für Quellenverweise vorgibt (welches z.B. in einer lokalen Konfigurationsdatei, einer Klassen- oder einer Paketdatei definiert ist).
2. Indem man die Option `numbers` oder `super` angibt, wie bei den meisten Beispielen in diesem Abschnitt.
3. Indem man explizit `\bibpunct` mit `n` oder `s` als viertem obligatorischem Argument verwendet (Einzelheiten in der Paketdokumentation).
4. Indem man explizit `\citestyle` mit dem Namen eines vordefinierten numerischen Stils für Literaturverzeichnisse verwendet.

Anpassen von natbib im numerischen Modus

Der Großteil der Optionen und Parameter zum Anpassen des natbib-Paketes wurde bereits auf den Seiten 730–732 besprochen. Im numerischen Modus stehen jedoch noch zwei weitere Befehle zum Verändern des Layouts zur Verfügung. Die Verweisnummern sind so voreingestellt, dass sie in der Grundschrift formatiert werden. Definiert man jedoch den Befehl `\citenumfont` (als Befehl mit einem Argument), dann formatiert er die Verweisnummer entsprechend seiner Einstellung.

Ebenso kann man das Format der Nummer im Literaturverzeichnis ändern, indem man den Befehl `\bibnumfmt` mithilfe von `\renewcommand` umdefiniert.¹ Die vorgegebene Definition dieses Befehls erzeugt normalerweise eckige Klammern um die Nummer.

Bilder werden an anderer Stelle besprochen, siehe (1, 2).

Literatur

1. M. Goossens, S. Rahtz, and F. Mittelbach. *The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997. ISBN 0-201-85469-4.
2. D. E. Knuth. *The T_EXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0.

```
\usepackage[numbers,round]{natbib}
\bibliographystyle{abbrvnat}
\newcommand\bibfont{\small\raggedright}
\setlength\bibhang{30pt}           % ignoriert!
\setlength\bibsep{1pt plus 1pt}
\newcommand\citenumfont[1]{\textbf{#1}}
\renewcommand\bibnumfmt[1]{\textbf{#1.}}
```

Bilder werden an anderer Stelle
besprochen, siehe `\citep[LGC97,Knuth-CT-a]`.

```
\bibliography{tex}
```

Bsp.
12-4-8

Während `\bibsection`, `\bibpreamble`, `\bibfont` und `\bibsep` ihre Funktion beibehalten, hat der Parameter `\bibhang` hier keinerlei Auswirkungen, da sich der Einzug bei einer nummerierten Bibliographie nach der Breite der größten Zahl richtet.

¹Das Paket verhält sich leider nicht ganz konsistent hinsichtlich der Vorgabe bzw. fehlenden Vorgabe von Einstellungen für die Anpassungsbefehle. Daher muss man je nach Kontext entweder `\newcommand` oder `\renewcommand` verwenden.

12.5 Das Kurztitelschema

12.5.1 jurabib – Anpassungsfähige Kurztitelverweise

Das jurabib-Paket von Jens Berger als Paket für das Kurztitelschema zu bezeichnen, lässt ihm nicht wirklich Gerechtigkeit widerfahren (kein Wortspiel beabsichtigt), da es auch die anderen Zitierschemata unterstützt.

Neben dem Kurztitel-Zitierschema unterstützt es auch Autor-Jahr-Verweise (durch die Befehlsschnittstelle des natbib-Paketes), verschiedene Optionen für geisteswissenschaftliche Werke und besondere Funktionen zum Zitieren juristischer Werke wie etwa Kommentare (daher der Name jurabib).

Das Paket verwendet ein erweitertes Optionskonzept, bei dem Optionen mithilfe einer Schlüssel-Wert-Syntax festgelegt werden. Es unterstützt mehr als 30 Optionen, von denen jede mehrere Werte annehmen kann. So lassen sich Literaturhinweise im Text und die entsprechenden Quellenangaben im Literaturverzeichnis nach den verschiedensten Gesichtspunkten darstellen. In diesem Buch kann nur eine kleine Auswahl der Möglichkeiten vorgestellt werden. Weitere Informationen findet man in der Paketdokumentation, die in deutscher und englischer Sprache verfügbar ist.

Da sich die `\usepackage`-Deklaration nicht sehr gut zum Verwalten so vieler Optionen eignet, bietet das jurabib-Paket als Alternative den Befehl `\jurabibsetup`. Er lässt sich in der Präambel oder der Konfigurationsdatei `jurabib.cfg` des Paketes verwenden (um die Voreinstellungen für alle Dokumente festzulegen). Werden beim Laden des Paketes oder mit `\jurabibsetup` in der Präambel andere Einstellungen vorgenommen, so überschreiben diese die globalen Vorgaben. Für die Beispiele in diesem Abschnitt gelten zunächst die folgenden Voreinstellungen:

*Voreinstellungen für
alle Beispiele in diesem
Abschnitt!*

```
\jurabibsetup{titleformat=colonsep,commabeforerest=true}
```

Diese werden dann je nach Bedarf überschrieben. Ihre Bedeutung wird im Folgenden erläutert.

Anders als natbib benötigt das jurabib-Paket speziell gestaltete BibTeX-Stildateien. Es erwartet einen `\bibitem`-Befehl mit besonders aufgebaute optionalen Argument, das verschiedenste Daten an die Quellenverweissbefehle auf Benutzerebene zurückgeben kann (siehe Seite 723). Diese BibTeX-Stile verwenden außerdem eine Reihe zusätzlicher Felder, die in Kombination mit jurabib hilfreich sind.

Die verschiedenen Eigenschaften von jurabib werden anhand der kleinen BibTeX-Datenbank aus Abbildung 12.3 auf der nächsten Seite, sowie der bisher verwendeten Datenbank (Abbildung 12.2 auf Seite 714) dargestellt. Soweit nicht anders angemerkt, wird allen Beispielen in diesem Abschnitt am Ende implizit die Zeile

```
\newpage\bibliography{tex,jura}
```

angehängt, wenn sie verarbeitet werden.

Die Basissyntax

Genau wie natbib erweitert auch das jurabib-Paket den L^AT_EX-Standardbefehl für Quellenverweise, `\cite`, um ein zweites optionales Argument.

oder das Trennzeichen zwischen Autor und Bearbeiter ändern. Beide Möglichkeiten werden im ersten Teil des nächsten Beispiels eingesetzt. Man kann den Bearbeiter auch vor den Autor setzen. Diese Lösung ist in zwei Varianten im zweiten Teil des Beispiels zu sehen.

Bsp.
12-5-2

Brox/Walker
Otto Palandt–*Heinrichs*, § 26
Heinrichs, Otto Palandt, § 26
Heinrichs in: Otto Palandt, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{annotatorformat=italic,
               annotatorlastsep=divis}

\cite{aschur} \ \ \cite[Heinrichs][\S\,26]{bgb} \ \
\jurabibsetup{annotatorfirstsep=comma}
\cite[Heinrichs][\S\,26]{bgb} \ \
\jurabibsetup{annotatorfirstsep=in,
               annotatorformat=normal}
\cite[Heinrichs][\S\,26]{bgb}
```

Man kann Autoren und Bearbeiter auch noch anders klar voneinander abheben: Mithilfe der Option `authorformat` und dem Schlüsselwort `and` werden die Schrägstriche durch Kommas und ein „und“ ersetzt; das Schlüsselwort `dynamic` verwendet eine andere Schrift, wenn ein *bearbeiter* existiert; und das Schlüsselwort `year` verschiebt die Publikation direkt hinter den Autor.

Mithilfe der Option `authorformat` kann man die Formatierung der Autorennamen noch weiter gestalten. Im Folgenden sind einige Beispiele dazu aufgeführt. Die Paketdokumentation enthält eine vollständige Liste aller Schlüsselwörter. Verwendet man mehrere Schlüsselwörter gleichzeitig (wie im nächsten Beispiel), so ist zu beachten, dass man `jurabib` durch ein zusätzliches Klammernpaar anzeigen muss, wo die Schlüsselwort-Liste aufhört und die nächste Option beginnt.

Bsp.
12-5-3

BROX und WALKER
OTTO PALANDT/HEINRICHS, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat={and,smallcaps}}

\cite{aschur} \ \ \cite[Heinrichs][\S\,26]{bgb} \par
```

Durch das Schlüsselwort `dynamic` wird der Bearbeiternamen kursiv ausgegeben und der Name des eigentlichen Autors in der Grundschrift.¹ Bei Werken ohne Bearbeiter wird der Autornamen kursiv gesetzt. Es werden sozusagen diejenigen hervorgehoben, die tatsächlich an der jeweiligen Ausgabe gearbeitet haben.

Bsp.
12-5-4

Brox/Walker
Otto Palandt/*Heinrichs*, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat=dynamic}

\cite{aschur} \ \ \cite[Heinrichs][\S\,26]{bgb} \par
```

Wie im nächsten Beispiel zu sehen, lassen sich die Schlüsselwörter `and`, `dynamic` und `year` miteinander kombinieren, während `smallcaps` und `italic` miteinander konkurrieren, wobei die letzte Einstellung gewinnt.

¹Die verwendeten Schriften lassen sich durch die Parameter `\jbactualauthorfont` und `\jbactualauthorfontifannotator` anpassen.

Brox und Walker (2003)
 Otto Palandt (2003)/Heinrichs, § 26

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{authorformat={and,smallcaps,year,italic}}
\cite{aschur} \\\cite[Heinrichs][\S\,26]{bgb} \par
```

Bsp.
12-5-5

Die von BibTeX zurückgegebenen Daten sind sehr detailliert und in einzelne Felder aufgeteilt, auf deren Inhalt man mit dem Befehl `\citefield` zugreifen kann.

```
\citefield[nach]{feld}{schlüssel}
```

Bei dem Argument *feld* handelt es sich um eines der folgenden Felder des BibTeX-Datenbankeintrags, auf den mit dem Argument *schlüssel* verwiesen wird: *author*, *shortauthor*, *title*, *shorttitle*, *url* oder *year*. Außerdem kommt noch *apy* („address-publisher-year“, eine Kombination aus Adresse, Verlag und Erscheinungsjahr) in Frage.

Bei den meisten Feldern ist es fraglich, ob mehr als ein *schlüssel* sinnvoll ist. Selbst bei `\cite` sind mehrfache Schlüssel selten nützlich, außer wenn keine optionalen Argumente vorhanden sind.

BROX, HANS/WALKER, WOLF-DIETRICH
 BSchuR, § 53
 Reading, MA, USA: Addison-Wesley Long-
 man, 1997
 Allgemeines Schuldrecht; Besonderes Schuld-
 recht

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{authorformat=smallcaps}
\citefield{author}{aschur} \\\
\citefield[\S\,53]{shorttitle}{bschur} \\\
\citefield{apy}{LGC97} \\\
\citefield{title}{aschur,bschur}
```

Bsp.
12-5-6

Quellenverweise mit Kurz- und Langtiteln

Wie bereits erwähnt, enthalten die voreingestellten Quellenverweise des jurabib-Paketes nicht den Titel der zitierten Werke. Davon ausgenommen sind mehrere Werke vom gleichen Autor, die sich dann nur anhand des Titels unterscheiden lassen. Dieses Verhalten lässt sich auf verschiedene Weise ändern. Zunächst wird jedoch der verwendete „Titel“ noch einmal genauer betrachtet.

Brox/Walker: Allgemeines Schuldrecht
 Brox/Walker: BSchuR
 Knuth: The TeXbook
 Knuth: TUGboat 10 [1989]

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\cite{aschur} \\\cite{bschur} \\\[2pt]
\cite{Knuth-CT-a} \\\cite{Knuth:TB10-1}
```

Bsp.
12-5-7

Vergleicht man die ersten beiden Zeilen des letzten Beispiels mit den BibTeX-Datenbankdateien in Abbildung 12.3 auf Seite 742, dann sieht man, dass soweit vorhanden immer das Feld *shorttitle* verwendet wird; ansonsten wird auf das Feld *title* zurückgegriffen. Das jurabib-Paket gibt in diesem Falle sogar eine Warnung aus: „shorttitle for aschur is missing – replacing with title“. Einträge der Publikationstypen *article* oder *periodical* werden anders gehandhabt. Dort wird ein fehlendes Feld durch den Namen, die Nummer und das Erscheinungsjahr der Zeitschrift ersetzt, so dass sich im Beispiel „TUGboat 10 [1989]“ ergab.

```
\citetitle[nach]{schlüssel} \citetitle[bearbeiter][nach]{schlüssel}
\cite*[nach]{schlüssel}    \cite*[bearbeiter][nach]{schlüssel}
```

Um einen Quellenverweis mit Titel zu erzwingen, kann man anstelle von `\cite` den Befehl `\citetitle` verwenden. Mit dem Befehl `\cite*` wird der Titel weggelassen. Man sollte jedoch beachten, dass letzterer Befehl sehr leicht zu mehrdeutigen Verweisen führen kann, wie das nächste Beispiel zeigt.

Bsp.
12-5-8

Baumbach et al.: ZPO, Brox/Walker und Brox/Walker sind drei verschiedene Bücher, oder doch nicht?

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\citetitle{zpo}, \cite*{aschur} und \cite*{bschur}
sind drei verschiedene Bücher, oder doch nicht?
```

Man beachte zudem, dass der Befehl `\cite*` hier eine ganz andere Bedeutung hat als im `natbib`-Paket, wo er eine vollständige Liste aller Autoren eines Werkes ausgibt. Wenn man je nach Aufgabenstellung zwischen den beiden Paketen hin- und herwechselt, sollte man diesen Befehl vielleicht besser ganz vermeiden.

```
\citetitleonly[nach]{schlüssel}
```

Es ist auch möglich, nur den Titel auszugeben, falls gewünscht durch einen *nach*-Text ergänzt.

Bsp.
12-5-9

ZPO, § 13

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\citetitleonly[\S,13]{zpo}
```

Mithilfe der Option `titleformat` und dem Schlüsselwort `all` lässt sich das Kurztitel-Zitierschema als Voreinstellung festlegen. Genau wie `authorformat` kann man auch dieser Option mehrere Schlüsselwörter zuweisen. Das Schlüsselwort `colonsep`, das als Voreinstellung für alle Beispiele benutzt wurde, ist bereits bekannt. Im nächsten Beispiel wird es mit `commasep` überschrieben und die Titel werden kursiv gesetzt (Schlüsselwort *italic*).

*Kurztitelverweise
automatisch erzeugen*

Brox/Walker, *Allgemeines Schuldrecht*
Brox/Walker, *BSchuR*, § 123
Otto Palandt/Heinrichs, *BGB*
Knuth, *TUGboat 10* [1989]

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{titleformat={all,commasep,italic}}
\cite{aschur} \ \ \cite[\S,123]{bschur} \ \
\cite[Heinrichs]{}{bgb} \ \ \cite{Knuth:TB10-1}
```

Bsp.
12-5-10

```
\citetitlefortype{BibTeX-typen-liste}
\citenotitlefortype{BibTeX-typen-liste}
```

Anstatt alle Werke mit Titel zu zitieren kann man auch festlegen, dass für bestimmte BibTeX-Publikationstypen Kurztitel verwendet werden. Der Befehl

```
\citetitlefortype{article,book>manual}
```

legt beispielsweise fest, dass diese drei Typen mit Titel zitiert werden und alle anderen ohne, sofern der Autor nicht mit mehreren Werken vertreten ist. Da eine solche Liste schnell recht umfangreich werden kann, lässt sich alternativ auch bestimmen, dass alle Werke mit Titel zitiert werden (über `titleformat`)

und dann gibt man die Publikationstypen an, auf die ohne Titel verwiesen werden soll. Dies geschieht im nächsten Beispiel für den Publikationstyp book (Buch). Auf das Buch von Knuth wird jedoch trotzdem mit Titel verwiesen, da zusätzlich aus einem seiner Artikel zitiert wird.

Brox/Walker	<code>\usepackage{jurabib}</code>	<code>\bibliographystyle{jurabib}</code>
Goossens/Rahtz	<code>\jurabibsetup{titleformat=all}</code>	<code>\citenotitlefortype{book}</code>
Knuth: The $\TeX$ book	<code>\cite{bschur}</code>	<code>\cite{LWC99}</code>
Knuth: TUGboat 10 [1989]	<code>\cite{Knuth-CT-a}</code>	<code>\cite{Knuth:TB10-1}</code>

Bsp.
12-5-11

Automatisches Indexieren von Quellenverweisen

Autorennamen in Quellenverweisen kann man durch die `authorformat`-Option mit dem Schlüsselwort `indexed` in den Index aufnehmen. Entsprechend der Vorgaben gilt dies nur für im Text zitierte Autoren und nicht für solche, die lediglich im Literaturverzeichnis erscheinen. Mithilfe des Befehls `\jbindexbib` in der Präambel oder einer Konfigurationsdatei lässt sich dieses Verhalten ändern. Zum Formatieren der Indexeinträge dient der Befehl `\jbauthorindexfont`. Die Anweisung

```
\renewcommand\jbauthorindexfont[1]{\textit{#1}}
```

setzt die Autorennamen im Index kursiv.

Anstatt die Namen der Autoren in den Hauptindex aufzunehmen, kann man auch ein eigenes Autorenverzeichnis erstellen. Dies geschieht mithilfe des `index`-Paketes (siehe Abschnitt 11.4.3) und einer Konstruktion wie der folgenden in der Präambel:

```
\usepackage{index}
\newindex{default}{idx}{ind}{Index}           % der Hauptindex
\newindex{authors}{adx}{and}{Autorenverzeichnis}
\renewcommand\jbindextype{authors}
```

Später bestimmt man mit dem Befehl `\printindex[authors]`, wo im Dokument das Autorenverzeichnis ausgegeben werden soll.

Noch speziellere Verzeichnisse, wie sie für einige Arten juristischer Werke erforderlich sind (z.B. „Verzeichnis der Fälle“, „Verzeichnis der Gesetze“), werden nicht unterstützt. Wer diese benötigt, der sollte anstelle von jurabib das camel-Paket in Betracht ziehen.

Verwenden der natbib-Semantik für Quellenverweise

Das optionale Argument *bearbeiter* ist nur in juristischen Werken hilfreich. In anderen Disziplinen wird häufiger ein einleitender *vor*-Text benötigt (z.B. „vergleiche...“). Um diesem Bedarf gerecht zu werden, lässt sich die Bedeutung der optionalen Argumente ändern, indem man das Paket mit der Option *see* lädt.

`\cite[vor] [nach] {schlüssel}` (mit Option see)

Die Option ersetzt das vorgegebene optionale Argument *bearbeiter* durch das Argument *vor*, wenn zwei Argumente verwendet werden. Damit verfügt der

\cite-Befehl dann über die gleiche Syntax und Semantik wie im natbib-Paket.

Bsp.
12-5-12

	<code>\usepackage[see,round]{jurabib}</code>	
	<code>\bibliographystyle{jurabib}</code>	
(Goossens/Rahtz/Mittelbach)	<code>\cite{LGC97}</code>	<code>\\</code>
(Goossens/Rahtz/Mittelbach, Kap. 2)	<code>\cite[Kap.~2]{LGC97}</code>	<code>\\[3pt]</code>
(vergleiche Goossens/Rahtz/Mittelbach)	<code>\cite[vergleiche][]{LGC97}</code>	<code>\\</code>
(siehe Goossens/Rahtz/Mittelbach, Kap. 2)	<code>\cite[siehe][Kap.~2]{LGC97}</code>	

Dieses Werk wurde als ... zitiert

Bei einem Zitierschema mit Kurztiteln (z.B. durch die Option `titleformat` mit dem Schlüsselwort `all`) kann es hilfreich sein, dem Leser eine Übersicht der vollständigen Titel und ihrer zugehörigen Kurzformen anzubieten. Normalerweise geschieht dies, indem man in der Bibliographie am Ende jeder Quelle den jeweiligen Kurztitel in Klammern anführt. Das `jurabib`-Paket unterstützt diese Konvention mittels der Option `howcited`. Mithilfe ihrer verfügbaren Schlüsselwörter lässt sich das Verfahren auf verschiedene Weise leicht abändern. Das Schlüsselwort `all` beispielsweise weist das Paket an, alle Einträge in der Bibliographie um die „zitiert als“-Daten zu ergänzen. Man kann also das Beispiel 12-5-10 auf Seite 745 um folgende Zeile erweitern:

```
\jurabibsetup{howcited=all}
```

Dann erhält man das folgende Literaturverzeichnis. Man beachte, dass der Kurztitel genauso formatiert ist, wie er auch im Quellenverweis erscheint.

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27. Auflage. München, 2002 (zitiert: Brox/Walker, *BSchuR*)

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29. Auflage. München, 2003 (zitiert: Brox/Walker, *Allgemeines Schuldrecht*)

Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207 (zitiert: Knuth, *TUGboat 10 [1989]*)

Bsp.
12-5-13

Palandt, Otto: Bürgerliches Gesetzbuch. 62. Auflage. München: Beck Juristischer Verlag, 2003 (zitiert: Otto Palandt, *BGB*)

Normalerweise ist es jedoch nicht erforderlich, bei allen Quellen anzugeben, wie diese zitiert werden. Bei Artikeln besteht der Kurztitel immer aus „Auturname, Zeitschrift, Band und Jahr“. Wird ein Werk nur mit vollständigem Titel zitiert (etwa weil kein `shorttitle`-Feld vorhanden ist), oder ein bestimmter Autor mit nur einem Werk, dann kann der Leser im Allgemeinen den entsprechenden Eintrag ohne weitere Hilfe finden. Diese eingeschränkte Art von „Rückverweisen“ ermöglicht das `jurabib`-Paket durch die Schlüsselwörter `compare`, `multiple` und `normal`.

Durch `compare` wird nur dann ein Rückverweis erzeugt, wenn ein `shorttitle`-Feld vorhanden ist und dessen Inhalt von dem des Feldes `title` abweicht. Für Beispiel 12-5-13 würde dies bedeuten, dass nur der erste und der letzte Eintrag Rückverweise enthielten.

Verwendet man stattdessen `multiple`, dann erfolgen Rückverweise für alle Autoren, die mit mehreren Werken zitiert werden, es sei denn, es handelt sich um Quellenverweise auf Artikel. Im vorigen Beispiel erhielten so die ersten beiden Einträge Rückverweise. Ein Verweis auf Knuth-CT-a würde ebenfalls einen Rückverweis nach sich ziehen, auf den Artikel von Knuth in *TUGboat* jedoch nicht.

Die beiden Schlüsselwörter lassen sich kombinieren. In diesem Fall werden sowohl Einträge von Autoren mit mehreren Werken, als auch Einträge mit vom Haupttitel abweichenden Kurztiteln, um Rückverweise ergänzt.

Das letzte Schlüsselwort dieser Gruppe ist `normal` (das auch verwendet wird, wenn man der Option kein Schlüsselwort mitgibt). Dieses Schlüsselwort funktioniert etwas anders als die anderen: Es benötigt Unterstützung von der BibTeX-Datenbank. Durch dieses Schlüsselwort erhalten ausschließlich solche Einträge einen Rückverweis, die über das BibTeX-Feld `howcited` verfügen. Dieses Feld kann zwei Arten von Werten enthalten. Mit dem Wert „1“ zeigt der Rückverweis genau den im Text erscheinenden Quellenverweis an. Bei jedem anderen Wert wird als Rückverweis der tatsächliche Inhalt des `howcited`-Feldes ausgegeben, einschließlich etwaig vorhandener Formatierungen.

Der Text um den Rückverweis lässt sich durch Umdefinieren der Befehle `\howcitedprefix` und `\howcitedsuffix` anpassen. Zusätzlich kann man mit `\bibnotcited` (per Voreinstellung leer) angeben, was mit Einträgen geschehen soll, die mit `\nocite` hinzugefügt wurden. Da diese Befehle Texte enthalten, die sich je nach Hauptsprache des Dokumentes ändern sollten, werden sie mit einem besonderen Verfahren (`\AddTo`) umdefiniert, dass auf Seite 759 beschrieben wird.

...Brox/Walker: BSchuR ... Knuth ...

Literatur

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27. Auflage. München, 2002 (zitiert als Brox/Walker: BSchuR).

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29. Auflage. München, 2003 (nicht zitiert).

Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207 (zitiert als Knuth).

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{howcited=all}
\AddTo\bibsall{%
  \renewcommand\howcitedprefix
    { (zitiert als )}%
  \renewcommand\howcitedsuffix{.}%
  \renewcommand\bibnotcited
    { (nicht zitiert).}%
\nocite{aschur}
\ldots \cite{bschur} \ldots
\cite{Knuth:TB10-1} \ldots
\bibliography{jura,tex}
```

Bsp.
12-5-14

Ausführliche Quellenangaben im Fließtext

Während das `natbib`-Paket zum Zitieren mit vollständigen Quellenangaben im Fließtext ein weiteres Paket und einige zusätzliche Vorbereitungen erfordert, ist dieses Zitiervorgehen im `jurabib`-Paket bereits vollständig integriert. Der vollständige Eintrag kann bei ein oder mehreren einzelnen, bei allen oder automatisch beim ersten Quellenverweis auf ein Werk im Dokument aufgeführt werden. Dieses Verfahren wird am häufigsten in Fußnoten verwendet. Auf

Seite 752 wird näher beschrieben, wie man Quellenverweise automatisch als Fußnoten setzt.

```
\fullcite[nach]{schlüssel} \fullcite[bearbeiter][nach]{schlüssel}
```

Dieser Befehl funktioniert wie `\cite`, zeigt jedoch die vollständigen bibliographischen Daten an. Ein etwaiger *bearbeiter* wird vor dem Verweis aufgeführt, genauso als wäre die Option `annotatorfirstsep=in` festgelegt.

Man vergleiche das nächste Beispiel mit Beispiel 12-3-27 von Seite 736. Das Schlüsselwort `citationreversed` ordnet den Autornamen so an, dass der Nachname als letztes erscheint (im Literaturverzeichnis wird er zuerst genannt). Verwandte Schlüsselwörter sind `allreversed` (der Nachname erscheint in Text und Bibliographie nachgeordnet) und `firstnotreversed` (der Nachname des ersten Autors erscheint an erster Stelle, bei weiteren Autoren des gleichen Werkes jedoch nachgeordnet).

Nähere Einzelheiten in Donald E. Knuth: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896–3207. Allgemeine Informationen finden sich in Donald E. Knuth: The TeXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0.

Wie in Knuth (1989) gezeigt ...

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{authorformat=citationreversed}
\raggedright \setlength\parindent{12pt}
```

Nähere Einzelheiten in `\fullcite{Knuth:TB10-1}`. Allgemeine Informationen finden sich in `\fullcite{Knuth-CT-a}`.

Wie in `\cite{Knuth:TB10-1}` gezeigt \ldots

Der `\cite`-Befehl erzeugt automatisch vollständige Quellenangaben, wenn die Option `citefull` mit einem der folgenden Schlüsselwörter verwendet wird: Das Schlüsselwort `all` zitiert alle Verweise mit vollständigen Angaben; `first` zitiert ein Werk beim ersten Verweis vollständig, danach verkürzt; `chapter` funktioniert wie `first`, beginnt jedoch in jedem Kapitel neu; und `section` entspricht `chapter` allerdings auf Abschnittsebene. Für alle Einstellungen gilt die Option `annotatorfirstsep=in`, wie im zweiten Verweis des Beispiels zu sehen ist. Wurde eine dieser Einstellungen in die Konfigurationsdatei aufgenommen, dann lässt sie sich mit dem Schlüsselwort `false` für das aktuelle Dokument abschalten.

*Automatisch
ausführliche Verweise
im Fließtext*

Siehe Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59. Auflage. München, 2002 ...

Wie in Heinrichs in: Baumbach et al., § 216 gezeigt, ist die Interpretation ...

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{citefull=first}
```

Siehe `\cite{zpo}` \ldots

Wie in `\cite[Heinrichs][\S,216]{zpo}` gezeigt, ist die Interpretation \ldots

```
\citefullfirstfortype{BIBTEX-typen-liste}
```

Eine noch genauere Steuerung lässt sich erzielen, indem man die BibTeX-Publikationstypen angibt, für die beim ersten Vorkommen vollständige Quel-

Bsp.
12-5-15

Bsp.
12-5-16

lenangaben erscheinen sollen. Im folgenden Beispiel (das ansonsten Beispiel 12-5-15 entspricht), wird festgelegt, dass nur Einträge des Typs `article` auf diese Weise zu zitieren sind.

Nähere Einzelheiten in Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896-3207. Allgemeine Informationen finden sich in Knuth: The $\TeX$ book.

Wie in Knuth: TUGboat 10 [1989] gezeigt, kann man leicht ...

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{citefull=first}
\citefullfirstfortype{article}
```

Nähere Einzelheiten in `\cite{Knuth:TB10-1}`.
Allgemeine Informationen finden sich in `\cite{Knuth-CT-a}`.

Wie in `\cite{Knuth:TB10-1}` gezeigt, kann man leicht `\ldots`

Bsp.
12-5-17

```
\nextciteshort{schlüssel-liste} \nextcitefull{schlüssel-liste}
\nextcitereset{schlüssel-liste} \nextcitenotitle{schlüssel-liste}
```

Manchmal ist es nicht angebracht, ein Werk bei seiner ersten Erwähnung mit vollständigen Quellenangaben zu zitieren, wie etwa in einer Kurzdarstellung oder einem Vorwort. Andererseits soll eine andere Quelle vielleicht noch einmal ausführlich angegeben werden, obwohl sie bereits zuvor aufgeführt wurde. Zu diesem Zweck gibt es vier Befehle, die festlegen, wie einzelne Quellen ab einer bestimmten Stelle im Dokument weiter zitiert werden.¹

Mit `\nextciteshort` werden alle Quellenverweise der „schlüssel-liste“ ab diesem Punkt im Kurztitelformat gesetzt (wie die Zeilen A,B und D im Beispiel). Durch `\nextcitereset` erscheinen die Quellenverweise (wieder) in ihrer normalen Form. Der nächste Verweis ist also wieder ausführlich, sofern die Quelle zuvor noch nicht zitiert wurde (Zeilen C und F), und ansonsten werden die Verweise im Kurztitelformat gesetzt (Zeile E). Mithilfe von `\nextcitefull` werden ab der entsprechenden Stelle ausführliche Quellenverweise erzwungen (Zeile G). Durch `\nextcitenotitle` erhält man nur noch den bzw. die Namen der Autoren, selbst wenn sich dadurch mehrdeutige Verweise ergeben.

```
\usepackage[citefull=first]{jurabib}
\bibliographystyle{jurabib}

\nextciteshort{Knuth-CT-a,Knuth:TB10-1}
A) \cite{Knuth-CT-a}           \ B) \cite{Knuth:TB10-1}      \
\nextcitereset{Knuth-CT-a}
C) \cite{Knuth-CT-a}           \ D) \cite{Knuth:TB10-1}      \
\nextcitereset{Knuth-CT-a,Knuth:TB10-1}
E) \cite{Knuth-CT-a}           \ F) \cite{Knuth:TB10-1}      \
\nextcitefull{Knuth-CT-a}
\nextcitenotitle{Knuth:TB10-1}
G) \cite{Knuth-CT-a}           \ H) \cite{Knuth:TB10-1}
```

¹Die Namen der Befehle legen nahe, dass sich diese jeweils auf den „nächsten“ Verweis beziehen. Tatsächlich gelten sie aber für alle weiteren Quellenverweise, bis sie überschrieben werden.

- A) Knuth: The T_EXbook
 B) Knuth: TUGboat 10 [1989]
 C) Knuth, Donald E.: The T_EXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0
 D) Knuth: TUGboat 10 [1989]
 E) Knuth: The T_EXbook
 F) Knuth, Donald E.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896–3207
 G) Knuth, Donald E.: The T_EXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0
 H) Knuth

Bsp.
12-5-18

Wenn man bereits im Fließtext vollständige Quellenangaben verwendet, muss man die Quellen nicht unbedingt noch einmal in einer Bibliographie oder einem Quellenverzeichnis zusammenstellen. Man kann zum Beispiel alle Quellenverweise im Fließtext setzen und im Literaturverzeichnis nur empfohlene weiterführende oder sonstige Sekundärliteratur auflisten.

`\citeswithoutentry{schlüssel-liste}`

Diese Deklaration enthält eine Liste aller Quellen, die nicht in der Bibliographie erscheinen sollen, obwohl sie im Fließtext zitiert werden. Die *schlüssel-liste* ist eine durch Kommas unterteilte Liste ohne Leerzeichen. Der Befehl kann beliebig oft wiederholt werden. Man kann ihn als Gegenteil von `\nocite` betrachten. Beide Befehle werden im nächsten Beispiel verwendet.

Das wird erklärt in Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29. Auflage. München, 2003.

Wie zu sehen in Brox/Walker...

Ausgewählte weiterführende Literatur

Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59. Auflage. München, 2002

Bsp.
12-5-19

```
\usepackage{jurabib}
\addto\captionsngerman{%
  \renewcommand\refname{Ausgewählte
    weiterführende Literatur}}
\bibliographystyle{jurabib}
\citeswithoutentry{aschur}
\jurabibsetup{citefull=first}
```

Das wird erklärt in `\cite{aschur}`.

```
Wie zu sehen in \cite{aschur}\ldots
\nocite{zpo}
\bibliography{jura}
```

Der Befehl `\citeswithoutentry` verhindert, dass einzelne Werke in der Bibliographie erscheinen. Mit seiner Hilfe kann man jedoch nicht alle Einträge unterdrücken, da man ansonsten eine leere Liste erhält, die nur aus einer Überschrift besteht. Wenn man vollständig auf ein Literaturverzeichnis verzichten möchte, verwendet man `\nobibliography` anstelle des üblichen `\bibliography`-Befehls. Dieser Befehl liest die von B_BT_EX erzeugte .bbl-Datei um die Quellenverweise zu ermöglichen, gibt aber kein gesetztes Ergebnis aus. Man muss immer noch jurabib als B_BT_EX-Stil angeben und B_BT_EX ganz normal ausführen.

Die Bibliographie ganz unterdrücken

Quellenverweise als Fuß- oder Endnoten

Alle bisher vorgestellten Befehle für Quellenverweise verfügen über Varianten, die Fußnotenverweise oder, in Kombination mit dem Paket `endnotes`, Endnotenverweise erzeugen. Man erhält diese Varianten, indem man dem Befehlsnamen einfach `foot` voranstellt (also z.B. `\footcite` anstelle von `\cite`, `\footcitetitle` anstelle von `\citetitle` usw.). Dadurch kann man Fußnoten- und andere Verweise nach Bedarf miteinander mischen.

Die Fußnotenverweise des `jurabib`-Paketes sind herkömmliche Fußnoten, so dass sich ihr Layout je nach Wunsch auch durch Laden des Paket `footmisc` umgestalten lässt.

...um $\text{\LaTeX}$ im Web zu verwenden.* In Goossens/Rahtz wird zudem besprochen, wie man PDF-

*Goossens, Michel/Rahtz, Sebastian: The $\text{\LaTeX}$ Web companion: integrating $\text{\TeX}$, HTML, and XML. Reading, MA, USA: Addison-Wesley Longman, 1999, Tools and Techniques for Computer Typesetting, ISBN 0-201-43311-7.

```
\usepackage[ragged,symbol]{footmisc}
\usepackage{jurabib}
\bibliographystyle{jurabib}
```

\ldots um $\text{\LaTeX}$ im Web zu verwenden. `\footfullcite{LWC99}` In `\cite{LWC99}` wird zudem besprochen, wie man PDF- und HTML-Dateien erzeugt.

Bsp.
12-5-20

*Quellenverweise
automatisch als
Fußnoten setzen*

Um alle Quellenverweise als Fußnoten zu setzen, bedient man sich der Option `super`. In diesem Fall verwendet das `jurabib`-Paket automatisch die `\foot..`-Varianten, so dass der Befehl `\cite` zu `\footcite` wird usw., wie im nächsten Beispiel zu sehen ist. Dort wird auch die Option `citefull=first` verwendet, so dass die erste Fußnote derjenigen im vorigen Beispiel gleicht. (Aus Platzgründen wird nur die zweite Seite angezeigt. Aufgrund der lächerlich geringen Höhe der Beispielseite erscheint dort noch die letzte Zeile der Fußnote von der ersten Seite.) Die anderen beiden Quellenverweise werden dann automatisch verkürzt, wobei der dritte durch die Option `ibidem`, die auf der gegenüberliegenden Seite erläutert wird, noch kürzer ist.

Außerdem wird die Option `lookat` benutzt, um den Rückverweis auf die frühere Fußnote mit den ausführlichen Quellenangaben vorzunehmen. Diese Option ist nur zulässig, wenn man gleichzeitig die Option `citefull` verwendet und alle ersten Quellenverweise in Fußnoten erfolgen, da sie sich auf eine „Nummer“ beziehen muss.

Man muss darauf achten, ein Fußnotenformat zu verwenden, das eindeutige Nummern erzeugt. Wenn die Fußnoten beispielsweise je Kapitel oder Seite nummeriert werden, sind die Verweise entsprechend mehrdeutig. Dieses Problem lässt sich jedoch durch Laden des Paket `varioref` lösen, das die Rückverweise mit Seitennummern erzeugt. Wird das `varioref`-Paket aus anderen Gründen geladen und man wünscht an dieser Stelle keine Verweise mit Seitenzahlen, dann kann man diese mit dem Befehl `\jbignorevarioref` unterdrücken. Werden die Fußnoten nach Kapiteln nummeriert, so kann man auch mithilfe der `\labelformat`-Deklaration des `varioref`-Paketes angeben, zu welchem Kapitel die jeweilige Fußnote gehört:

```
\labelformat{footnote}{\thechapter--\#1}
```

Die Option `lookat` lässt sich besonders sinnvoll mit dem Befehl `\nobibliography` kombinieren, wodurch alle Literaturhinweise in Fußnoten erfolgen, ohne ein zusammenfassendes Literaturverzeichnis.

Es wird zudem besprochen, wie man PDF-² und HTML-Dateien³ er-

Techniques for Computer Typesetting, ISBN 0-201-43311-7.

²Goossens/Rahtz (Anm. 1), Kap. 2.

³A. a. O., Kap. 3-4.

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,citefull=first,ibidem,lookat}
\ldots um \LaTeX{} im Web zu verwenden.\cite{LWC99}
\newpage % Die nächste Seite wird gezeigt:
Es wird zudem besprochen, wie man
PDF-\cite[Kap.~2]{LWC99} und
HTML-Dateien\cite[Kap.~3--4]{LWC99} erzeugt.
```

Bsp.
12-5-21

Mithilfe der Befehle `\lookatprefix` und `\lookatsuffix` kann man das Erscheinungsbild der Rückverweise anpassen. Beide sind sprachspezifisch, so dass man hier die `\AddTo`-Deklaration verwenden muss (siehe Seite 759). Im nächsten Beispiel wird ein in juristischen Quellenverweisen [21] gängiges Format eingerichtet.

Es wird zudem besprochen, wie man PDF-² und HTML-Dateien³ er-

Techniques for Computer Typesetting, ISBN 0-201-43311-7.

²Goossens/Rahtz, *supra* note 1, Kap. 2.

³Goossens/Rahtz, *supra* note 1, Kap. 4.

```
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,citefull=first,lookat}
\AddTo\bibsall{\renewcommand\lookatprefix
{, \emph{supra} note }
\renewcommand\lookatsuffix{}}
\ldots um \LaTeX{} im Web zu verwenden.\cite{LWC99}
\newpage % Die nächste Seite wird gezeigt:
Es wird zudem besprochen, wie man
PDF-\cite[Kap.~2]{LWC99} und
HTML-Dateien\cite[Kap.~3--4]{LWC99} erzeugt.
```

Bsp.
12-5-22

Lädt man das Paket `endnotes` mit ähnlichen Einstellungen wie denjenigen im vorigen Beispiel, so lassen sich dadurch alle Quellenverweise in Endnoten umwandeln. Wie man sieht, wird den Endnoten entsprechend der Voreinstellungen kein abschließender Punkt angefügt. Wer einen Punkt wünscht, kann diesen über die Option `dotafter` mit dem Schlüsselwort `endnote` einstellen.

...mit Graphiken kombinieren.¹ Auch das Setzen von Noten² und Spielen³ wird besprochen.

Notes

¹Goossens, Michel/Rahtz, Sebastian/Mittelbach, Frank: The $\LaTeX$ Graphics Companion: Illustrating Documents with $\TeX$ and PostScript. Reading, MA, USA: Addison-Wesley Longman, 1997, Tools and Techniques for Computer Typesetting, ISBN 0-201-85469-4

²Goossens/Rahtz/Mittelbach (Anm. 1), Kap. 7

³Goossens/Rahtz/Mittelbach (Anm. 1), Kap. 8

```
\usepackage{jurabib,endnotes}
\bibliographystyle{jurabib}
\jurabibsetup{citefull=first,%
super,lookat}
\ldots mit Graphiken
kombinieren.\cite{LGC97} Auch das
Setzen von Noten\cite[Kap.~7]{LGC97}
und Spielen\cite[Kap.~8]{LGC97}
wird besprochen.
\theendnotes
```

Bsp.
12-5-23

Ibidem – Am angegebenen Ort

In manchen Disziplinen ist es üblich, einen Verweis auf eine soeben zitierte Quelle durch das lateinische Wort „ibidem“ (abgekürzt als „ibid.“ oder „ib.“) auszudrücken. Das Paket `jurabib` unterstützt diese Konventionen durch die Option `ibidem` in mehreren Abwandlungen. Sie muss mit Quellenverweisen im Fußnotenformat kombiniert werden (also z.B. mit `\footcite`-Befehlen bzw. der `super`-Option). Dieser Verweis entspricht im Deutschen dem Ausdruck „ebenda“, bzw. der Abkürzung „ebd.“. Das `jurabib`-Paket verwendet

allerdings in der sprachspezifischen Anpassung „A.a.O.“ (Am angegebenen Ort), was eher dem Lateinischen „*loco citato*“ entspricht.

Wird die Option `ibidem` ohne Wert verwendet (gleichbedeutend mit dem Schlüsselwort `strict`), so bewirkt sie folgendes Verhalten: Quellenverweise, die sich auf die gleiche Quelle beziehen, wie ihr direkter Vorgänger auf der gleichen Seite, werden durch „Ibid.“ (bzw. „A.a.O.“ im Deutschen) ersetzt, wobei ein etwaiger *nach*-Text erhalten bleibt. Dieses Verfahren wird im nächsten Beispiel deutlich: Der erste Verweis erfolgt als Kurztitel; der zweite ist mit diesem identisch, so dass „A.a.O.“ ohne *nach*-Text erscheint. Der dritte und vierte Verweis beziehen sich auf andere Teile des gleichen Werkes, so dass hier auch der *nach*-Text erhalten bleibt. Der fünfte Quellenverweis bezieht sich auf eine andere Publikation des gleichen Autors und somit erfolgt ein weiterer Kurztitelverweis. Der sechste Verweis bezieht sich wieder auf die gleiche Publikation. Trotzdem wird der Kurztitel wiederholt, da er auf einer neuen Seite erscheint. Der siebte und achte Verweis gelten wieder dem ersten Werk, so dass erneut ein Kurztitel genannt und danach noch einmal „A.a.O.“ mit *nach*-Text verwendet wird.

Text ¹ Text ^{2,3} mehr Text ^{4,5}	
¹ Brox/Walker: BSchuR, § 7.	
² A. a. O.	
³ A. a. O., § 16.	
⁴ A. a. O., § 7.	
⁵ Brox/Walker: Allgemei- nes Schuldrecht.	

Text ^{6,7} Text ⁸	
⁶ Brox/Walker: Allgemei- nes Schuldrecht, § 3.	
⁷ Brox/Walker: BSchuR.	
⁸ A. a. O., § 15.	

```
\usepackage[marginal,multiple]{footmisc}
\usepackage[super,ibidem]{jurabib}
\bibliographystyle{jurabib}

Text \cite[\S,7]{bschur}
Text \cite[\S,7]{bschur}
      \cite[\S,16]{bschur} mehr
Text \cite[\S,7]{bschur}
\cite{aschur} \newpage % <---
Text \cite[\S,3]{aschur}
      \cite{bschur}
Text \cite[\S,15]{bschur}
```

Bsp.
12-5-24

Setzt man ein Dokument mit der Klassenoption `twoside` doppelseitig, dann kann man das Schlüsselwort `strictdoublepage` verwenden. Es bedeutet, dass „A.a.O.“ auch über eine Seite hinaus benutzt werden darf, solange der vorherige Quellenverweis noch sichtbar ist (sich also auf der gleichen Doppelseite befindet). Eine Wiederholung von Beispiel 12-5-24 mit diesen Einstellungen ändert den sechsten Verweis in „A.a.O., § 3“.

Die Option `ibidem` erzeugt normalerweise viele sehr kurze Fußnoten, so dass sie sich sehr platz sparend mit der Option `para` des Paketes `footmisc` kombinieren lässt (wenn auch mit hässlichen Löchern im Beispiel). Zusätzlich wird die Option `perpage` hinzugefügt, damit die Fußnotennummern klein bleiben. Man beachte jedoch, dass die `lookat`-Option dann nicht mehr verwendet werden kann, da die Fußnotennummern nicht mehr eindeutig sind.

Text ¹ mehr Text ^{2,3} Text ^{4,5}	
¹ Brox/Walker: BSchuR, § 7.	² A. a. O.
³ A. a. O., § 16.	
⁴ A. a. O., § 7.	⁵ Brox/ Walker: Allgemeines Schuldrecht.

Text ^{1,2} Text ³	
¹ A. a. O., § 3.	² Brox/ Walker: BSchuR.
³ A. a. O., § 15.	

```
\usepackage[para,multiple,perpage]{footmisc}
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=strictdoublepage}

Text \cite[\S,7]{bschur} mehr Text
\cite[\S,7]{bschur} \cite[\S,16]{bschur}
Text \cite[\S,7]{bschur} \cite{aschur}
\newpage Text \cite[\S,3]{aschur}
\cite{bschur} Text \cite[\S,15]{bschur}
```

Bsp.
12-5-25

Man kann die Seitenbegrenzung auch völlig außer Acht lassen, indem man das Schlüsselwort `nostrict` verwendet. Dies kann dem Leser die Identifikation der Quelle jedoch erheblich erschweren, da „A.a.O.“ und der Quellenverweis, auf den es sich bezieht, beliebig weit auseinander liegen können. Falls erforderlich kann man den `ibidem`-Mechanismus für den nächsten Verweis deaktivieren, indem man diesem `\noibidem` voranstellt.

Bsp.
12-5-26

Eine Seite ohne Quellenverweis.	Diese Seite enthält Verweise. ² Oder so? ³
	² A. a. O. ³ Brox/Walker.

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=nostrict}

\ldots \fullcite{bschur} \ldots
\newpage % vorige Seite wird nicht gezeigt
        Eine Seite ohne Quellenverweis.
\newpage Diese Seite enthält
        Verweise.\cite{bschur}
        Oder so? \noibidem\cite{bschur}
```

Mit einem „Ibid./A.a.O“-Verweis ohne weiteren Zusatz kann man nur auf die unmittelbar zuvor zitierte Quelle verweisen. Durch häufiges Wechseln der Quellen werden daher meistens Kurztitelverweise eingefügt. Das ändert sich, wenn man die Option `ibidem` mit dem Schlüsselwort `name` verwendet (welche automatisch die Option `citefull=first` impliziert). In diesem Falle wird „A.a.O.“ mit dem vollständigen Namen des Autors kombiniert, so dass auch ein Bezug auf einen früheren, nicht direkt vorhergehenden Verweis möglich wird. Um nur die Nachnamen der Autoren zu erhalten, fügt man die Option `authorformat` mit dem Schlüsselwort `reducedifibidem` hinzu. Ihre Wirkung wird im nächsten Beispiel deutlich, in dem abwechselnd die Quellen `bschur` und `zpo` zitiert werden. Als weitere Variante kann man immer den Namen und Kurztitel verwenden, außer beim ersten Verweis auf eine Quelle. Dieses Format lässt sich mit dem Schlüsselwort `name&title` festlegen.

Wenn der gleiche Autor mit mehreren Werken zitiert wird, dann kann die `ibidem`-Option mit dem Schlüsselwort `name` sehr leicht zu mehrdeutigen Quellenverweisen führen. In solchen Fällen schaltet das `jurabib`-Paket automatisch auf die im Folgenden beschriebene `name&title&auto`-Methode um.

Bsp.
12-5-27

- text¹ text^{2,3} text^{4,5} text⁶
- ¹ Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27. Auflage. München, 2002, § 7.
 - ² Brox/Walker, a. a. O., § 8.
 - ³ Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59. Auflage. München, 2002, § 16.
 - ⁴ Brox/Walker, a. a. O., § 7.
 - ⁵ Baumbach et al., a. a. O.
 - ⁶ Baumbach et al., a. a. O., § 3.

```
\usepackage[marginal,ragged,multiple]{footmisc}
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=name}
\jurabibsetup{authorformat=reducedifibidem}

text \cite[\S\,7]{bschur} text
\cite[\S\,8]{bschur} \cite[\S\,16]{zpo}
text \cite[\S\,7]{bschur} \cite{zpo}
text \cite[\S\,3]{zpo}
```

Das Schlüsselwort `name&title&auto` (explizit oder implizit) bewirkt Folgendes: Der erste Verweis auf eine Veröffentlichung erzeugt einen vollständigen Eintrag (Quellenverweis 5 im nächsten Beispiel). Bei wiederholten Verweisen auf eindeutige Werke wird nur der Name des Autors (bzw. der Autoren) genannt (Verweis 8). Bei mehrdeutigen Quellenangaben gilt dieses Verfahren

nur für unmittelbar folgende Verweise (Quellenverweis 4). Liegen jedoch andere Quellenverweise dazwischen, so werden immer Autor(en) und Kurztitel angezeigt (Verweise 3, 6 und 7).

Text³ Text^{4,5} Text^{6,7} Text⁸

- ³ Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht, a. a. O., § 7.
⁴ Brox, Hans/Walker, Wolf-Dietrich, a. a. O., § 8.
⁵ Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59. Auflage. München, 2002, § 16.
⁶ Brox, Hans/Walker, Wolf-Dietrich: BSchUR, a. a. O., § 7.
⁷ Brox, Hans/Walker, Wolf-Dietrich: Allgemeines

```
\usepackage[marginal,ragged,multiple]{footmisc}
\usepackage{jurabib} \bibliographystyle{jurabib}
\jurabibsetup{super,ibidem=name&title&auto}

Ausführliche Verweise: \cite{aschur}
\cite{bschur} links nicht angezeigt!

\newpage
Text \cite[\S\,7]{aschur} Text
\cite[\S\,8]{aschur} \cite[\S\,16]{zpo}
Text \cite[\S\,7]{bschur} \cite{aschur}
Text \cite[\S\,3]{zpo}
```

Bsp.
12-5-28

In manchen Disziplinen ist es auch üblich, einen gleich bleibenden Autor in aufeinander folgenden Quellenverweisen mit dem lateinischen Wort „Idem“ (derselbe, abgekürzt ders.) anzugeben. Dieses Verfahren wird durch die Option `idem` unterstützt, die sich in Kombination mit den Schlüsselwörtern `strict`, `strictdoublepage` und `nostrict` entsprechend der `ibidem`-Option verhält. Wie im nächsten Beispiel zu sehen, lassen sich auch beide Optionen miteinander kombinieren. Je nach verwendeten Schlüsselwörtern erhält man unterschiedliche Quellenverweise: Einige verwenden „Idem, ibid.“ bzw. im Deutschen „Ders., a.a.O.“; nach dem Seitenumbruch wird „Ders.“ aufgrund der Option `strict` unterdrückt; in den letzten drei Zitaten erscheint es erneut (selbst bei den ausführlichen Quellenangaben), da sich diese alle auf unterschiedliche Werke von Donald Knuth beziehen.

... Text¹ Text²
Text^{3,4}...

- ¹Knuth, Donald E.: The $\TeX$ book. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0-201-13447-0.
²Ders., a. a. O., S. 22.
³Leunen, Mary-Claire van: A handbook for scholars. Walton Street, Oxford OX2 6DP, UK: Oxford University Press, 92.
⁴Ders., a. a. O.

... Text⁵ Text⁶ Text⁷
Text^{8,9}...

- ⁵Leunen, Mary-Claire van, a. a. O.
⁶Ders., a. a. O., S. 16.
⁷Knuth, Donald E.: The $\TeX$ book, a. a. O., S. 308.
⁸Ders.: Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, ISSN 0896-3207.
⁹Ders.: The $\TeX$ book, a. a. O., S. 80.

```
\usepackage[flushmargin,%
multiple]{footmisc}
\usepackage[super,ibidem=strict,%
ibidem=name]{jurabib}
\bibliographystyle{jurabib}

\ldots Text \cite{Knuth-CT-a}
Text \cite[S.~22]{Knuth-CT-a} \\\
Text \cite{vLeunen:92}
\cite{vLeunen:92}\ldots

\newpage % <--
\ldots Text \cite{vLeunen:92}
Text \cite[S.~16]{vLeunen:92}
Text \cite[S.~308]{Knuth-CT-a} \\\
Text \cite{Knuth:TB10-1}
\cite[S.~80]{Knuth-CT-a}\ldots
```

Bsp.
12-5-29

Man muss sich hier fragen, ob diese Art der Querverweise den Lesern wirklich dienlich ist. Butcher [29] spricht sich beispielsweise dagegen aus. Möglicherweise hat man auch gar keine Wahl, da dieses Verfahren gefordert wird. Man sollte jedoch beachten, dass zwei Verweise im letzten Beispiel tatsächlich falsch sind: Bei van Leunen handelt es sich um eine Autorin, so dass die korrekte lateinische Bezeichnung „Eadem“ und nicht „Idem“ bzw. die deutsche „Dies.“ und nicht „Ders.“ lauten müsste (eine Unterscheidung, die in einigen

Handbüchern zur Gestaltung von Texten jedoch nicht vorgesehen ist). Soweit erforderlich ermöglicht das jurabib-Paket sogar eine so detaillierte Anpassung der Quellenverweise (siehe Seite 760).

Es gibt noch eine weitere Konvention für wiederholte Quellenverweise, die jedoch immer seltener wird: Um anzuzeigen, dass sich ein Quellenverweis auf eine frühere Quellenangabe bezieht, wird er mit *op. cit.* gekennzeichnet (für *opere citato*, „am angegebenen Ort“). Dieses Verfahren wird durch die Option `opcit` unterstützt und erzeugt in der Voreinstellung die lateinische Abkürzung. Der entsprechende Quellenverweis sollte „in der Nähe“ sein, so dass der Leser ihn auch finden kann. Aus diesem Grunde bietet jurabib hier in Analogie zur Option `citefull` die Schlüsselwörter `chapter` und `section` an.

...Text¹ Text² Text³ weiterer Text^{4,5}

¹Knuth, Donald E.: The T_EXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0-201-13447-0.

²Ders., *op. cit.*, S. 22.

³Free Software Foundation: GNU Make, A Program for Directing Recompilation. 2000.

⁴Knuth, *op. cit.*

⁵Free Software Foundation, *op. cit.*

```
\usepackage[multiple]{footmisc}
\usepackage[super,idem=strict,%
  citefull=first,opcit]{jurabib}
\bibliographystyle{jurabib}

\ldots Text \cite{Knuth-CT-a} Text
\cite[S.~22]{Knuth-CT-a} Text
\cite{GNUmake} weiterer Text
\cite{Knuth-CT-a}\cite{GNUmake}
```

Bsp.
12-5-30

In juristischen Zitaten [21] ist es üblich mit dem Wort „*supra*“ auf eine zuvor erfolgte Quellenangabe zu verweisen (entspricht dem deutschen „siehe oben“). Diese Art von Verweisen erhält man, wenn man den Befehl `\opcit`, der den erzeugten Text enthält, folgendermaßen ändert:

```
\renewcommand\opcit{\textit{supra}}
```

Ansonsten kann man auch die in Beispiel 12-5-22 auf Seite 753 gezeigte Methode verwenden.

Querverweise auf Quellen

B_IB_TE_X unterstützt die Idee von Querverweisen zwischen bibliographischen Einträgen durch das Feld `crossref`. So kann beispielsweise eine Publikation des Typs `inproceedings` auf den Tagungsband verweisen, in dem sie erscheint. Je nach Anzahl der Verweise auf einen solchen Sammelband entscheidet B_IB_TE_X dann, ob dieser einen eigenen Eintrag erhält oder die Quellenangaben in jeden entsprechenden `inproceedings`-Eintrag aufgenommen werden. Abschnitt 13.2.5 enthält nähere Einzelheiten hierzu.

Sofern B_IB_TE_X für die Quellen, auf die ein Querverweis erfolgt, eigene Einträge erzeugt, stellt sich die Frage, wie ein ausführlicher Quellenverweis auf diese Werke über `\fullcite` oder `\footfullcite` im Text erscheinen soll. Zu diesem Zweck bietet das jurabib-Paket drei Schlüsselwörter für die Option `crossref`: Mit dem Schlüsselwort `normal` (voreingestellt) werden Querverweise aus den Feldern `author/editor` und `title` zusammengestellt (bzw. `shortauthor` und `shorttitle` soweit vorhanden); durch das Schlüsselwort `short` werden nur die Felder `author` oder `editor` verwendet, solange dadurch keine Mehrdeutigkeit entsteht; und das Schlüsselwort `long` setzt Querverweise in ausführlicher Form. Im nächsten Beispiel wird die voreingestellte

Methode gezeigt, wobei die Herausgeber und der Kurztitel vom jurabib-Paket ausgewählt wurden.

Mittelbach, Frank/Rowley, Chris: The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In Vanoirbeek/Coray: EP92

Southall, Richard: Presentation Rules and Rules of Composition in the Formatting of Complex Text. In Vanoirbeek/Coray: EP92

Mittelbach/Rowley

```
\usepackage{jurabib}
\jurabibsetup{citefull=first,
              crossref=normal}
\bibliographystyle{jurabib}
\cite{MR-PQ}      \par
\cite{Southall}   \par
\cite{MR-PQ}
```

Bsp.
12-5-31

Jedes der drei Schlüsselwörter lässt sich zudem mit dem Schlüsselwort `dynamic` kombinieren, wodurch der erste Querverweis auf die Quelle in längerer und alle weiteren in verkürzter Form erfolgen. Im nächsten Beispiel wird es mit `long` kombiniert. Dadurch erscheint der Verweis auf Vanoirbeek/Coray beim ersten Mal in ausführlicher Form und beim zweiten Mal als Kurztitel.

Frank Mittelbach/Chris Rowley: The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In Christine Vanoirbeek/Giovanni Coray (Hrsg.): EP92—Proceedings of Electronic Publishing, '92. Cambridge: Cambridge University Press, 1992

Richard Southall: Presentation Rules and Rules of Composition in the Formatting of Complex Text. In Vanoirbeek/Coray: EP92

```
\usepackage{jurabib}
\jurabibsetup{citefull=first,
              authorformat=
                citationreversed,
                crossref={dynamic,long}}
\bibliographystyle{jurabib}
\cite{MR-PQ}      \par
\cite{Southall}
```

Bsp.
12-5-32

Unterstützung für das Autor-Jahr-Zitierschema

Wie bereits erwähnt unterstützt `jurabib` die Befehle `\citet` und `\citep`, wie sie mit dem `natbib`-Paket eingeführt wurden. Ebenso verfügt es über die Befehle `\citealt`, `\citealp`, `\citeauthor`, `\citeyear` und `\citeyearpar`. Aus den Befehlsformen, die sich auch für Fußnotenverweise eignen, lassen sich durch das Präfix `foot` vor dem Befehlsnamen entsprechende Varianten bilden (also z.B. `\footcitet`). Die im `natbib`-Paket enthaltenen Sternformen sind jedoch nicht verfügbar.

Goossens/Rahtz (1999)
Goossens/Rahtz (1999, Kap. 2)
siehe Goossens/Rahtz (1999, Kap. 2)
nur Vortext: siehe Goossens/Rahtz (1999)

(Goossens/Rahtz, 1999)
(Goossens/Rahtz, 1999, Kap. 2)
(siehe Goossens/Rahtz, 1999, Kap. 2)
nur Vortext: (siehe Goossens/Rahtz, 1999)

Knuth, 1986

Knuth
(1986)

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\citet{LWC99}          \\\
\citet[Kap.~2]{LWC99}  \\\
\citet[siehe][Kap.~2]{LWC99} \\\
nur Vortext: \citet[siehe][{}]{LWC99}\[5pt]
\citep{LWC99}          \\\
\citep[Kap.~2]{LWC99}  \\\
\citep[siehe][Kap.~2]{LWC99} \\\
nur Vortext: \citep[siehe][{}]{LWC99}\[5pt]
\citealp{Knuth-CT-a}    \\\
\citeauthor{Knuth-CT-a} \\\
\citeyearpar{Knuth-CT-a}
```

Bsp.
12-5-33

Durch die Einstellung `authorformat=year` werden Autor-Jahr und Kurztitel-Zitierschema, wie bereits in Beispiel 12-5-5 vorgestellt, miteinander kombiniert. Die Formatierung des Erscheinungsjahres lässt sich mit `\jbcitationyearformat` verändern, und die Position des Jahres kann mit `\jbyearaftertitle` hinter einen vorhandenen Titel verschoben werden.

```
\usepackage{jurabib}
\bibliographystyle{jurabib}
\jurabibsetup{authorformat=year,annotatorformat=italic}
\renewcommand\jbcitationyearformat[1]{\oldstylenums{#1}}
\jbyearaftertitle
```

Brox/Walker 2003	<code>\cite{aschur}</code>	<code>\\</code>
Otto Palandt/ <i>Heinrichs</i> 2003, § 26	<code>\cite[Heinrichs][\S,26]{bgb}</code>	<code>\\</code>
Baumbach et al. 2002	<code>\cite{zpo}</code>	

Bsp.
12-5-34

Sprachunterstützung

Die meisten automatisch erzeugten Texte eines Eintrags im Literaturverzeichnis oder eines Quellenverweises sind sprachspezifisch; sie richten sich nach der Hauptsprache des Dokumentes. Zu diesem Zweck arbeitet das `jurabib`-Paket mit dem `babel`-Paket zusammen. Je nach der Hauptsprache eines Dokumentes (die durch die letzte Option des `babel`-Paketes festgelegt wird) lädt `jurabib` eine besondere Sprachdefinitionsdatei (mit der Erweiterung `.ldf`). Diese enthält Definitionen für alle möglichen Befehle, die Texte in Quellenverweisen und im Literaturverzeichnis erzeugen. Zurzeit werden annähernd zehn Sprachen unterstützt. Die Sprachdateien, wie zum Beispiel `enjjbib.ldf` für die englische Sprache, sind eine gute Quelle für nähere Informationen zu den einzelnen Anpassungsmöglichkeiten. Mithilfe der `jurabib`-Deklaration `\AddTo` lassen sich Befehle solcher Dateien für eine bestimmte oder für alle Sprachen ändern.

<code>\AddTo\bibsa11{code}</code>	<code>\AddTo\bibs{sprache}{code}</code>

Die Deklaration `\AddTo` verfügt über zwei Argumente: einen Befehlsnamen, der alle sprachspezifischen Definitionen für eine Sprache enthält und den entsprechenden `code`, der diesem Parameter hinzugefügt werden soll.¹ Das erste Argument ist entweder `\bibsa11`, dann gilt der `code` für alle Sprachen, oder `\bibs{sprache}` (z.B. `\bibsgerman`), dann gilt er nur für die genannte Sprache.² In Beispiel 12-5-14 auf Seite 748 und Beispiel 12-5-22 auf Seite 753 wurde die Darstellung der Rückverweise für alle Sprachen mit `\AddTo` geändert, indem `\bibsa11` geänderte Definitionen hinzugefügt wurden.

¹Das `babel`-Paket verwendet einen ähnlichen Mechanismus für seine `\addto`-Deklaration.

²Leider verwendet das `jurabib`-Paket nicht genau das gleiche Verfahren wie `babel`. Gibt man im `babel`-Paket `ngerman` an, um auf die Trennmuster der neuen deutschen Rechtschreibung umzuschalten, dann wird dies `german` zugeordnet, und man muss `\bibsgerman` aktualisieren. Verwendet man einen der Dialekte (wie z.B. `austrian`), dann wird dieser von `jurabib` nicht erkannt, das Paket gibt eine Warnung aus und schaltet dann wieder auf `english` um. In diesem Falle muss man die Definitionen mit `\bibsa11` ändern.

Im nächsten Beispiel wird die Zeichenfolge „Ibid.“ für deutsche Texte ver­ kürzt. Die Vorgaben für die anderen Sprachen bleiben hier unverändert.

Text ¹ und ² oder ³ und mehr Text. ⁴	<pre>\usepackage[super,ibidem,titleformat=all]{jurabib} \AddTo\bibsgerman{\renewcommand\ibidemname{Ib.}}% \renewcommand\ibidemmidname{ib.}} \bibliographystyle{jurabib} Text\cite{vLeunen:92} und\cite{vLeunen:92} \jurabibsetup{ibidem=name} % Konventionsänderung oder\cite{Knuth-CT-a} und mehr Text.\cite{Knuth-CT-a}</pre>	Bsp. 12-5-35
¹ van Leunen: A handbook for scholars.		
² Ib.		
³ Knuth, Donald E.: The T _E Xbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ISBN 0–201–13447–0.		
⁴ Knuth, Donald E., ib.		

Während einige Texte im gesamten Literaturverzeichnis gleich bleiben sollten – wie etwa die Bezeichnung „(Hrsg.)“ für den Herausgeber (`\editorname`) – sind bestimmte andere Aspekte, insbesondere die Silbentrennung, von der Sprache des aktuellen Eintrags abhängig. So sollte der Titel eines deutschen Buches auch mithilfe der deutschen Trennmuster getrennt werden, ungeachtet der Hauptsprache des Dokumentes. Dieses Verfahren wird von jurabib durch das zusätzliche Feld `language` in der B_{IB}T_EX-Datenbankdatei unterstützt. Enthält ein Eintrag dieses Feld, dann geht jurabib davon aus, dass der Titel in der angegebenen Sprache gesetzt werden soll. Daher werden in diesem Falle die Trennmuster für die jeweilige Sprache angewendet, soweit sie verfügbar (also im Format geladen) sind. Beispielsweise erscheint der letzte Querverweis von Beispiel 12-5-6 auf Seite 744 bei geladenem babel-Paket auch dann mit korrekter Silbentrennung, wenn die Dokumentensprache Englisch ist:

Allgemeines Schuldrecht; Besonderes Schuldrecht	<pre>\usepackage[ngerman,english]{babel} \usepackage{jurabib} \bibliographystyle{jurabib} \citefield{title}{aschur,bschur}</pre>	Bsp. 12-5-36

Unterscheiden von Autoren nach Geschlecht

Zuvor wurde bereits erwähnt, dass „Eadem“ die weibliche Form von „Idem“ ist. In der deutschen Sprache gibt es „Derselbe“ (Maskulinum), „Dieselbe“ (Femininum), „Dasselbe“ (Neutrum) und „Dieselben“ (Plural). Das jurabib-Paket unterscheidet das Geschlecht der Autoren mithilfe des B_{IB}T_EX-Feldes `gender`, das eine zweibuchstabige Abkürzung enthält.

	Bedeutung	Im Verweis	In der Bibliographie
sf	Singular, Femininum	<code>\idemSfname, \idemsfname</code>	<code>\bibidemSfname, \bibidemsfname</code>
sm	Singular, Maskulinum	<code>\idemSmname, \idemsname</code>	<code>\bibidemSmname, \bibidemsname</code>
pf	Plural, Femininum	<code>\idemPfname, \idempfname</code>	<code>\bibidemPfname, \bibidempfname</code>
pm	Plural, Maskulinum	<code>\idemPmname, \idempmname</code>	<code>\bibidemPmname, \bibidempmname</code>
sn	Singular, Neutrum	<code>\idemSnname, \idemsname</code>	<code>\bibidemSnname, \bibidemsname</code>
pn	Plural, Neutrum	<code>\idemPnname, \idempnname</code>	<code>\bibidemPnname, \bibidempnname</code>

Tabelle 12.1: Geschlechtsangaben in jurabib (gender-Feld)

In Tabelle 12.1 auf der gegenüberliegenden Seite sind mögliche Werte aufgeführt, sowie die Befehle, welche die „Idem“-Texte enthalten. Befehle, die mit einem Großbuchstaben beginnen, stehen am Satzanfang, die anderen mitten im Satz. Diejenigen, die mit `\bibidem..` anfangen, werden in der Bibliographie verwendet, wenn die Option `bibformat` mit dem Schlüsselwort `ibidem` angegeben ist. Da diese Funktion ressourcenintensiv ist, wird sie nicht per Voreinstellung aktiviert, sondern muss explizit angefordert werden. Um also für weibliche Autoren den Ausdruck „Dies.“ zu erhalten, müssen Werte für `\idemSfname` und `\idemsfname`, sowie die Option `lookforgender` festgelegt werden.

Text¹ und² oder³ und mehr Text.⁴

¹van Leunen: A handbook for scholars.

²Dies.: A handbook for scholars

³Knuth: The T_EXbook.

⁴Ders.: The T_EXbook

```
\usepackage[super, idem=strict, titleformat=all,
lookforgender=true]{jurabib}
\AddTo\bibsngerman{\renewcommand\idemSfname{Dies.}%
\renewcommand\idemsfname{dies.}}
\bibliographystyle{jurabib}
Text\cite{vLeunen:92} und\cite{vLeunen:92}
oder\cite{Knuth-CT-a} und mehr
Text.\cite{Knuth-CT-a}
```

Bsp.
12-5-37

Weitere Anpassungen für eingebettete Quellenverweise

Autoren- und Titelangaben werden, wie bereits besprochen, größtenteils mithilfe der Befehle `authorformat` und `titleformat` formatiert. Außerdem gibt es noch einige weitere Optionen und Befehle, die bisher noch nicht erwähnt wurden.

Soll der ganze Quellenverweis in Klammern stehen, so kann man dies einfach durch die Optionen `round` bzw. `square` festlegen.

Mithilfe der Option `superscriptedition` erscheinen Angaben zu der zitierten Ausgabe des jeweiligen Werkes als Hochstellung nach dem Kurztitel. Durch das Schlüsselwort `all` gilt diese Darstellungsform für alle Kurztitelverweise, mit `commented` nur für kommentierte Werke des Publikationstyps `commented` und `multiple` wählt diese Form nur für Veröffentlichungen, die in verschiedenen Ausgaben zitiert werden. Die beiden letzten Optionen sind hauptsächlich für juristische Abhandlungen gedacht.

[Baumbach et al.: ZPO ⁵⁹]	<code>\usepackage{jurabib} \bibliographystyle{jurabib}</code>
[Brox/Walker ²⁷ , § 3]	<code>\jurabibsetup{square, superscriptedition={all}}</code>
[Otto Palandt/Heinrichs ⁶²]	<code>\citetitle{zpo}\cite[\S, 3]{bschur}\cite[Heinrichs] [] {bgb}</code>

Bsp.
12-5-38

Man kann auch in der B_BT_EX-Datenbank für jeden Eintrag einzeln festlegen, ob die Ausgabe als Hochstellung erscheinen soll, indem man das B_BT_EX-Feld `ssedition` auf den Wert 1 setzt und dann die Option `superscriptedition` mit dem Schlüsselwort `switch` verwendet.

Die Option `authorformat=and` trennt die Autorennamen durch Kommas sowie das Wort „and“, genauer gesagt den Inhalt von `\andname`, einem Befehl der je nach Sprache einen anderen Text ausgibt. Auf diese Weise lässt sich jedoch die Separierung zwischen zweitem und drittem Autorennamen nicht gezielt einstellen. Solche detaillierten Einstellungen sind nur durch Umdefinieren folgender Befehle möglich: `\jbbtasep` für die Trennung zwischen

genau zwei Autoren (**b**etween two authors **s**eparation), `\jbbfsasep` für die Trennung zwischen erstem und zweitem Autor (**b**etween first and second authors) und `\jbbstasep` für die Trennung zwischen zweitem und drittem Autor (**b**etween second and third authors **s**eparation).¹

(Brox und Walker)

(Goossens, Rahtz & Mittelbach)

```
\usepackage[round]{jurabib} \bibliographystyle{jurabib}
\renewcommand\jbbtasep{ und } % genau zwei
\renewcommand\jbbfsasep{, } \renewcommand\jbbstasep{ \& }
\cite{aschur} \ \cite{LGC97}
```

Bsp.
12-5-39

Anstelle der durch die Optionen vorgegebenen Schriften für Autorennamen und Kurztitel möchte man vielleicht manuell selbst andere Einstellungen festlegen. Dazu stehen die Befehle `\jbauthorfont`, `\jbannotatorfont`, `\jbactualauthorfont`, `\jbauthorfontifannotator` und `\jbttitlefont` bereit, die alle über ein Argument verfügen.

Anpassen des Layouts der Bibliographie

Die Formatierung des Literaturverzeichnisses mit Standard- $\text{\LaTeX}$ oder mithilfe des `natbib`-Paketes wird weitgehend durch den verwendeten $\text{\BibTeX}$ -Stil gesteuert, oder durch die Formatierungsanweisungen des Anwenders, wenn das Verzeichnis manuell erzeugt wird. Ein Quellenverweis auf den Eintrag Knuth-CT-a aus der Beispieldatenbank würde beispielsweise durch den `natbib`-Stil `plainnat` folgendermaßen gestaltet:

Donald~E. Knuth.

```
\newblock {\em The {\TeX}book}, volume~A of {\em Computers and Typesetting}.
\newblock Ad{\-d}i{\-s}on-Wes{\-l}ey, Reading, MA, USA, 1986.
```

Das bedeutet, dass Formatierungsentscheidungen, wie etwa das Hervorheben des Buchtitels und der Reihe und die Darstellung des Feldes „volume“, durch die $\text{\BibTeX}$ -Stildatei getroffen werden.

Die $\text{\BibTeX}$ -Stile des `jurabib`-Paketes folgen einem völlig anderen Ansatz: Sie erzeugen sehr strukturierte Ausgaben, die aus einer Vielzahl von $\text{\LaTeX}$ -Befehlen bestehen, so dass sich die letztendliche Formatierung (so wie in einem gewissem Maße auch die Reihenfolge der Elemente) immer noch auf $\text{\LaTeX}$ -Ebene optimieren lässt. Sie müssen sogar auf $\text{\LaTeX}$ -Ebene angepasst werden, wenn man mit der Formatierung anhand der vorgegebenen Definitionen nicht zufrieden ist. Der im letzten Beispiel erzeugte Quellenverweis etwa würde mit einem $\text{\BibTeX}$ -Stil des `jurabib`-Paketes folgenden Eintrag erzeugen:

```
\jbbibargs {\bibnf {Knuth} {Donald~E.} {D.~E.} {} {}} {Donald~E. Knuth} {au}
{\bibtfnt {The {\TeX}book}\bibatsep\ \volumeformat {A} Computers and
Typesetting\bibatsep\ \apyformat {Reading, MA, USA\bpubaddr {}}
Ad{\-d}i{\-s}on-Wes{\-l}ey\bibbdsep {} 1986} \jbPages{ix + 483}\jbisbn
{0--201--13447--0}} {\bibhowcited} \jbdoitem \bibAnnoteFile {Knuth-CT-a}
```

Die meisten der aufgeführten Befehle sind noch weiter untergliedert. Der Befehl `\bibnf` zum Beispiel verfügt über fünf Argumente (den verschie-

¹Weitere Möglichkeiten sind nicht erforderlich, da das Paket `jurabib` automatisch alle etwaigen weiteren Autoren als „et al.“ aufführt.

denen Bestandteilen eines Autorennamens) und gibt diese, je nachdem welche davon nicht leer sind, zur weiteren Verarbeitung an Befehle wie `\jbnfIndNoVonNoJr` weiter (Name ohne „von“- und „Junior“-Elemente). Daher kann man an vielen Stellen in diesen Prozess eingreifen und so verschiedensten Anforderungen gerecht werden, auch wenn die Anpassung des Layouts dadurch relativ komplex wird. Aus diesem Grunde werden hier nur die wichtigsten Möglichkeiten der Anpassung aufgezeigt. Weitere Steuerungsmöglichkeiten sind in der Dokumentation des Paketes erklärt.

Entsprechend der vorgegebenen Einstellungen wird das Literaturverzeichnis weitgehend unabhängig von den Quellenverweisen formatiert. Durch die Option `authorformat=italic` werden die Autorennamen im Text kursiv gesetzt, im Literaturverzeichnis ändert sich jedoch nichts. Dies lässt sich am einfachsten mithilfe der Option `biblikecite` ändern, die bewirkt, dass Formatangaben für Quellenverweise soweit möglich auch für die Bibliographie gelten. Sollte dies nicht erwünscht oder nicht ausreichend sein, gibt es noch explizite Formatierungsanweisungen. Diese werden im Folgenden besprochen.

Die für einen Bibliographieeintrag verwendeten Schriften werden mit den folgenden Befehlen gesteuert: `\biblnfont` und `\bibfnfont` formatieren Nach- bzw. Vornamen des Autors und `\bibelnfont` sowie `\bibefnfont` formatieren Nach- bzw. Vornamen des Herausgebers, soweit vorhanden. Der Befehl `\bibtfont` wird für Buchtitel verwendet, `\bibbtfont` für Titel von Essays (also von Einträgen, die über das `\BIBTEX`-Feld `booktitle` verfügen) und `\bibjtfont` für Titel bzw. Namen von Zeitschriften. Die Schrift für Titel von Zeitschriftenartikeln wird mit `\bibapifont` eingestellt. Alle diese Befehle erhalten den zu bearbeitenden Text als Argument. Ihre geänderten Definitionen müssen also auch ein Argument verwenden, oder aber, wie im nächsten Beispiel, einen `\text{...}`-Fontbefehl, in dem das Argument implizit übernommen wird.

KNUTH, DONALD E.: <i>The T_EXbook</i>. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0 KNUTH, DONALD E.: <i>“Typesetting Concrete Mathematics”</i>. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207	<pre>\usepackage{jurabib} \bibliographystyle{jurabib} \newcommand\biblnfont{\MakeUppercase} \newcommand\bibfnfont{\textsc} \newcommand\bibtfont {\textsf} \newcommand\bibapifont[1]{\textit{'#1'}} \nocite{Knuth-CT-a,Knuth:TB10-1} \bibliography{tex}</pre>

Bsp.
12-5-40

Die Interpunktion zur Unterteilung eines Eintrags lässt sich über einen weiteren Satz von Befehlen steuern: Der Befehl `\bibansep` setzt das Satzzeichen und den Leerraum nach dem Autornamen, `\bibeansp` hinter dem Namen des Herausgebers, `\bibatsep` erzeugt die Zeichensetzung nach dem Titel (das Leerzeichen ist bereits vorhanden!) und `\bibbdsep` legt das Satzzeichen vor dem Erscheinungsjahr fest. Mit dem Befehl `\bibjtsep` wird das Trennzeichen für den Zeitschriftentitel festgelegt. Für die anderen Elemente gibt es entsprechende Befehle.¹ Im nächsten Beispiel wird mithilfe dieser Befehle der Doppelpunkt nach dem Autornamen entfernt, ein Semikolon hinter

¹Dieser Bereich des `jurabib`-Paketes ist in seinen Namenskonventionen und dem Verhalten der Befehle nicht ganz konsistent. Aber das wird sich ja vielleicht noch ändern.

dem Titel eingefügt, das Komma vor dem Erscheinungsjahr entfernt und vor den Namen der Zeitschrift das Wort „in“ gesetzt. Durch die Option `dotafter` mit dem Schlüsselwort `bibentry` wird jeder Eintrag des Literaturverzeichnisses mit einem Punkt beendet.

Knuth, Donald E. Typesetting Concrete Mathematics; in TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207.

Mittelbach, Frank/Rowley, Chris The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In **Vanoirbeek/Coray** EP92, 261–273.

Vanoirbeek, Christine/Coray, Giovanni (Hrsg.) EP92—Proceedings of Electronic Publishing, '92; Cambridge: Cambridge University Press 1992.

```
\usepackage[dotafter=
  bibentry]{jurabib}
\bibliographystyle{jurabib}
\renewcommand\bibtsep{in }
\renewcommand\bibansep{ }
\renewcommand\bibatsep{;}
\renewcommand\bibbdsep{}
\nocite{Knuth:TB10-1,MR-PQ}
\bibliography{tex}
```

Bsp.
12-5-41

Wie bereits erläutert, lassen sich die Trennzeichen zwischen den Namen verschiedener Autoren in einem Quellenverweis mithilfe der Option `authorformat` und verschiedenen Schlüsselwörtern anpassen. Mit Ausnahme des Schlüsselwortes `allreversed` haben diese jedoch keine Auswirkungen auf das Literaturverzeichnis. Um die Stile an dieser Stelle zu ändern, muss man die Befehle `\bibbtasep`, `\bibbfsasep` und `\bibbstasep` undefinieren. Die Befehlsnamen folgen den gleichen Konventionen wie die entsprechenden Befehle für Quellenverweise. Mithilfe der ähnlichen Befehlsreihe `\bibbtsep`, `\bibbfsesep` und `\bibbstesep` lassen sich die Trennzeichen zwischen verschiedenen Herausgebernamen in einem Eintrag festlegen.

Hans Brox und Wolf-Dietrich Walker: Allgemeines Schuldrecht. 29. Auflage. München, 2003

Michel Goossens, Sebastian Rahtz und Frank Mittelbach: The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript. Reading, MA, USA: Addison-Wesley Longman, 1997, Tools and Techniques for Computer Typesetting, xxi + 554, ISBN 0–201–85469–4

```
\usepackage
  [authorformat=allreversed]
  {jurabib}
\bibliographystyle{jurabib}
\renewcommand\bibbtasep{ und }
\renewcommand\bibbfsasep{, }
\renewcommand\bibbstasep{ und }
\nocite{aschur,LGC97}
\bibliography{tex,jura}
```

Bsp.
12-5-42

Das generelle Layout der Bibliographie anpassen

Mit der zentralen Option `bibformat` lässt sich das generelle Erscheinungsbild der Bibliographie beeinflussen. Sie kann als Wert eine Reihe von Schlüsselwörtern annehmen. Durch das Schlüsselwort `nohang` wird der vorgegebene Einzug (von 2.5em) ab der zweiten Zeile eines Bibliographieeintrags unterdrückt. Ansonsten kann man den Einzug auch, wie im nächsten Beispiel, explizit durch Ändern des Dimensionsparameters `\jbbibhang` einstellen. Dort kommen auch die Schlüsselwörter `compress` (weniger Leerraum um Einträge) und `raggedright` (Einträge ohne Randausgleich) zum Einsatz.

Um eine bessere Qualität zu erzielen, insbesondere wenn nur wenig Platz zur Verfügung steht, kann man auch das Paket `ragged2e` laden. Man beachte, wie die Option `newcommands` den voreingestellten `\raggedright`-Befehl (der

vom jurabib-Paket verwendet wird) mit `\RaggedRight` überschreibt.

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29. Auflage. München, 2003

Baumbach, Adolf et al.: Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59. Auflage. München, 2002

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27. Auflage. München, 2002

```
\usepackage[newcommands]{ragged2e}
\usepackage[bibformat={compress,%
                        raggedright}]
                        {jurabib}
\bibliographystyle{jurunsrt}
\setlength\jbbibhang{1pc}
\nocite{aschur,zpo,bschur}
\bibliography{jura}
```

Bsp.
12-5-43

Durch das Schlüsselwort `tabular` wird die Bibliographie als zweispaltige Tabelle gesetzt. Dabei enthält die linke Spalte die Autoren Daten und die rechte Spalte den Rest des Eintrags. Die linke Spalte ist auf ein Drittel der Textbreite `\textwidth` voreingestellt und beide Spalten werden im Flattersatz gesetzt. Die Voreinstellungen lassen sich ändern, indem man, wie im nächsten Beispiel, einige Befehle umdefiniert. Die Breite der rechten Spalte ist folgendermaßen festgelegt:

```
\renewcommand\bibrightrightcolumn
{\textwidth-\bibleftcolumn-\bibcolumnsep}
```

Normalerweise reicht es `\bibleftcolumn` und/oder `\bibcolumnsep` zu ändern. Das Paket `calc` wird vom `jurabib`-Paket automatisch geladen, so dass man es für Maßangaben nutzen kann.

**Brox, Hans/
Walker,
Wolf-Dietrich** Allgemeines Schuldrecht.
29. Auflage. München, 2003

**Knuth,
Donald E.** Typesetting Concrete Mathematics. TUGboat, 10 April
1989, Nr. 1, 31–36, ISSN
0896–3207

**Free Software
Foundation** GNU Make, A Program for Directing Recompilation. 2000

```
\usepackage[bibformat=tabular]{jurabib}
\bibliographystyle{jurabib}
\renewcommand\bibleftcolumn{5pc}
\renewcommand\bibcolumnsep{8pt}
\renewcommand\bibleftcolumnadjust
{\raggedright}
\renewcommand\bibrightrightcolumnadjust{}
\nocite{aschur,Knuth:TB10-1}
\nocite{GNUmake}
\bibliography{tex,jura}
```

Bsp.
12-5-44

Verwendet man das Schlüsselwort `numbered`, so werden die Einträge in der Bibliographie nummeriert, auch wenn die eigentlichen Quellenverweise im Text nach dem Zitierschema Autor-Jahr oder Kurztitel erscheinen. Zurzeit ist es nicht möglich, auf diese Nummern zu verweisen.

Die hauseigenen Vorlagen einiger Verlage lassen den Autornamen weg (oder ersetzen ihn durch einen Bindestrich oder ein anderes Zeichen), wenn der Autor mit mehreren Werken aufgeführt ist. Diese Schreibweise wird durch das Schlüsselwort `ibidem` unterstützt, das per Vorgabe den Text „Idem“ erzeugt, oder genauer gesagt das Ergebnis des Befehls `\bibidemSname`. Mit dem Befehl `\jbuseidemhrule` fügt man stattdessen eine Linie fester Länge ein. Durch Umdefinieren des Befehls `\bibauthormultiple` kann man auch noch andere Darstellungsformen erzeugen. Beide Möglichkeiten werden im

nächsten Beispiel gezeigt. Das jurabib-Paket stellt automatisch fest, ob ein Eintrag am Seitenanfang erscheint, und wiederholt in diesem Falle den oder die Autorennamen. Dieser Mechanismus kann einige zusätzliche L^AT_EX-Durchläufe erfordern, bis das Dokument ohne die Meldung „Rerun to get...“ kompiliert.

Brox, Hans/Walker, Wolf-Dietrich: Besonderes Schuldrecht. 27. Auflage. München, 2002

—— Allgemeines Schuldrecht. 29. Auflage. München, 2003

Knuth, Donald E.: The T_EXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

—— Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896–3207

```
\usepackage[bibformat=ibidem]
{jurabib}
\bibliographystyle{jurabib}
% Standardlinie verwenden:
\jbuseidemhrule
% Generisch geänderte Definition
% anstelle der Standardlinie:
%\renewcommand\bibauthormultiple
% {[gleiches Namenssymbol]}
\nocite{aschur,bschur}
\nocite{Knuth-CT-a,Knuth:TB10-1}
\bibliography{tex,jura}
```

Bsp.
12-5-45

Über das Schlüsselwort `ibidemalt` erhält man eine Variante des Bibliographiestils mit nach Autoren sortierten Werken. Diese Einstellung impliziert automatisch das Schlüsselwort `compress`.

Baumbach, Adolf et al.:

- ▷ Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen. 59. Auflage. München, 2002

Brox, Hans/Walker, Wolf-Dietrich:

- ▷ Besonderes Schuldrecht. 27. Auflage. München, 2002
- ▷ Allgemeines Schuldrecht. 29. Auflage. München, 2003

Palandt, Otto:

- ▷ Bürgerliches Gesetzbuch. 62. Auflage. München: Beck Juristischer Verlag, 2003

```
\usepackage{jurabib}
\jurabibsetup
{bibformat=ibidemalt}
\bibliographystyle{jurabib}
\nocite{aschur,bschur,zpo,bgb}
\bibliography{jura}
```

Bsp.
12-5-46

Kommentierte Bibliographien

Für ein kommentiertes Literaturverzeichnis steht die Option `annotate` zur Verfügung. Verfügt der jeweilige B^IB^IX-Eintrag über ein `annotate`-Feld, dann wird der enthaltene Text nach dem Eintrag gesetzt und mit dem Befehl `\jbannoteformat` formatiert. Dieser ist auf `\small` voreingestellt. Ist kein `annotate`-Feld vorhanden, so sucht das `jurabib`-Paket nach einer Datei mit der Erweiterung `.tex` und dem Schlüssel des Eintrags als Dateiname. Sofern eine solche Datei existiert, wird ihr Inhalt als Kommentar eingefügt.

Knuth, Donald E.: The T_EXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

Das maßgebliche Handbuch zum T_EX-Programm, von dessen Entwickler verfasst.

```
\begin{filecontents}{Knuth-CT-a.tex}
Das maßgebliche Handbuch zum \TeX{-}Programm,
von dessen Entwickler verfasst.
\end{filecontents}
\usepackage[annotate]{jurabib}
\bibliographystyle{jurabib}
\renewcommand\jbannoteformat[1]
{{\footnotesize\begin{quote}#1\end{quote}}}}
\nocite{Knuth-CT-a} \bibliography{tex}
```

Bsp.
12-5-47

Da es störend ist, sein Arbeitsverzeichnis mit derart vielen Dateien (eine je Kommentar) voll zu stopfen, verfügt das jurabib-Paket über eine Suchpfad-Deklaration, ähnlich dem Befehl `\graphicspath` des graphics-Paketes. Durch die Anweisung

```
\bibAnnotePath{./books}{./articles}}
```

werden die Kommentardateien folglich in den Unterverzeichnissen `books` und `articles` des aktuellen Verzeichnisses gesucht.

Externe Konfigurationsdateien

Das jurabib-Paket lässt sich auf zwei Ebenen anpassen: durch die Angabe von Optionen oder, für eine genauere Steuerung, durch Umdefinieren bestimmter Parameter bzw. Ausführen von Befehlen. Im vorigen Abschnitt wurden bereits eine Reihe von Paketoptionen mit ihren Schlüsselwörtern vorgestellt. Dabei handelt es sich aber um weniger als ein Drittel der verfügbaren Möglichkeiten. In der Standardkonfigurationsdatei `jurabib.cfg` befindet sich eine `\jurabibsetup`-Deklaration, die *alle* Optionen mit all ihren Schlüsselwörtern auflistet – insgesamt nahezu 100 Möglichkeiten. Sie sind alle auskommentiert, so dass man seine eigene Konfigurationsdatei erstellen kann, indem man die vorgegebene kopiert und dann die Kommentarzeichen der Optionen entfernt, die man ausführen möchte. Speichert man diese Konfiguration in einer Datei mit der Erweiterung `.cfg`, dann kann man diese mithilfe der `config`-Option anstelle der vorgegebenen Konfiguration laden. Das Beispiel

```
\usepackage[config=law]{jurabib}
```

lädt die Konfigurationsdatei `law.cfg`, die eine `\jurabibsetup`-Deklaration und möglicherweise einige zusätzliche Anpassungsbefehle enthalten sollte. Eine solche Datei könnte z.B. Folgendes beinhalten:

```
\jurabibsetup{lookat,opcit,commabeforerest,titleformat=colonsep}
\renewcommand\opcit{\textit{supra}}
```

Außerdem könnte sie einige weitere Initialisierungen vornehmen, um Quellenverweise auf juristische Werke einzurichten. Wie bereits erwähnt lassen sich solche in einer Datei gespeicherten Vorgaben durch das Laden weiterer Optionen oder eine `\jurabibsetup`-Deklaration in der Präambel des Dokumentes überschreiben.

B₁B₂TEX-Stile für jurabib

Mit dem jurabib-Paket werden die vier B₁B₂TEX-Stildateien `jurabib`, `jureco`, `jurunsrt` und `jox` verteilt. Sie unterscheiden sich nur geringfügig voneinander: `jureco` erzeugt eine etwas kompaktere Bibliographie, die einige Daten auslässt, während `jurunsrt` dem Stil `jurabib` entspricht, jedoch ohne Sortierung, so dass die Quellen in der gleichen Reihenfolge erscheinen wie ihre Verweise im Dokument. Der Stil `jox` erzeugt Quelleneinträge im „Oxford-Stil“. Da

das jurabib-Paket besonders strukturierte `\bibitem`-Befehle erfordert, können bisher nur die vier genannten Stile mit dem Paket zusammen verwendet werden.

Alle vier Stile halten eine Reihe zusätzlicher BibTeX-Publikationstypen bereit, sowie einige zusätzliche Felder für existierende Typen. Zusätzliche Felder in einer BibTeX-Datenbank sind normalerweise kein Problem, da BibTeX alle ihm unbekannten Felder einfach ignoriert. Daher lässt sich eine solche Datenbank auch mit anderen BibTeX-Stilen verwenden, die nicht mit diesen Feldern arbeiten. Zusätzliche Publikationstypen sind ein etwas anderes Thema, da man nur mit dem jurabib-Paket auf diese verweisen kann.

Zusätzliche BibTeX-Publikationstypen

Bei den zusätzlichen Eintragstypen handelt es sich um `www` für Verweise auf eine URL, `periodical` für Periodika, die nicht nach Erscheinungsjahr sondern nach Nummer des Bandes zitiert werden und `commented` für Kommentare bei juristischen Werken.

Die BibTeX-Standardfelder sind in Tabelle 13.2 auf Seite 792 beschrieben. Durch die BibTeX-Stile des jurabib-Paketes sind folgende zusätzliche Felder verfügbar:

annotate Ein Kommentar, der gesetzt wird, wenn man jurabib mit der Option `annotate` verwendet. Nähere Informationen hierzu befinden sich auf Seite 766.

booktitleaddon Zusätzliche Daten, die nach dem `booktitle`-Text einer Sammlung erscheinen sollen.

dissyear Erscheinungsjahr einer Dissertations- oder Habilitationsschrift oder eines ähnlichen Werkes, sofern dieses zusätzlich als Buch veröffentlicht wurde (möglicherweise in einem anderen Jahr).

editortype Funktion der im Feld `editor` aufgeführten Person (soweit sie nicht wirklich der „Herausgeber“ ist).

flanguage Sprache des Originals, im Falle eines übersetzten Werkes.

founder Bei juristischen Werken der ursprüngliche Begründer einer Publikation (im Gegensatz zum Herausgeber). Auf den Namen folgt der Ersetzungstext des Befehls `\foundername`, dessen Voreinstellung „□ (Begr.)“ ist.

gender Geschlecht des oder der Autoren. Das jurabib-Paket verwendet dieses Feld, um die richtige Entsprechung für den Begriff „Idem“ in der aktuellen Sprache zu finden (vgl. Seite 760).

howcited Text für den Rückverweis bzw. 1, damit ein normaler Rückverweis erzeugt wird. Dieses Feld wird ausgewertet, wenn man die Option `howcited` mit dem Schlüsselwort `normal` verwendet; siehe Seite 747.

oaddress/opublisher/oyear Daten zur Erstausgabe eines Werkes.

shortauthor Text, der in einem Kurztitelverweis als Autorenangabe erscheinen soll. Entsprechend der Voreinstellung verwendet jurabib automatisch den (oder die) letzten Autoren- oder Herausgebernamen.

- shorttitle** Text, der in einem Kurztitelverweis als Titelangabe erscheinen soll. Fehlt dieses Feld, so wird das gesamte Feld **title** verwendet.
- sortkey** Zeichenfolge, nach der in außergewöhnlichen Fällen sortiert werden soll. Um „von Bismarck, Otto“ unter B einzusortieren, kann man **sortkey**="Bismarck, Otto von" angeben.
- ssedition** Schalter, der festlegt, dass bei diesem Eintrag die Editionsdaten als Hochstellung erscheinen sollen. Diese Darstellungsform erfordert die Option **superscriptedition** mit dem Schlüsselwort **switch** (vgl. Seite 761).
- titleaddon** Zusatzinformationen, die nach einem Titel, jedoch z.B. nicht bei der Darstellung als Kurztitel, erscheinen sollen.
- totalpages** Gesamtzahl der Seiten in einer Veröffentlichung. Sofern das Feld vorhanden ist, folgt auf diese Angabe der Ersetzungstext des sprachspezifischen Befehls **\bibtotalpagesname**.
- translator** Übersetzer des Werkes.
- updated** Datum der letzten Aktualisierung einer Loseblattsammlung oder eines ähnlichen Werkes. Das Feld ist nur für den **BIBTEX**-Publikationstyp **commented** verfügbar. Als Voreinstellung wird der Text „last update *datum*“ (im Deutschen „Stand:“) erzeugt. Dieser lässt sich mithilfe der Befehle **\updatenname** und **\updatesep** anpassen.
- urldate** Datum, zu dem eine URL als aktuell bekannt war. Das **jurabib**-Paket ist so voreingestellt, dass es dieses Feld, sofern vorhanden, als „visited on *datum*“ (im Deutschen „Zugriff am“) ausgibt. Dieser Text lässt sich durch Umdefinieren des Befehls **\urldatecomment** ändern.
- url** Eine URL, die sich auf die aktuelle Veröffentlichung bezieht. Das Feld ist für den Publikationstyp **www** obligatorisch und ansonsten optional.
- volumetitle** Der Titel eines Bandes, der in der Darstellung der Bandnummer folgt. Dieses Feld ist für die Publikationstypen **book**, **commented**, **incollection** und **inbook** verfügbar.

12.5.2 camel – Unterstützung für Gesetzestexte

Für jeden, der den Konventionen zum Zitieren (angloamerikanischer) juristischer Werke folgen muss, kann vielleicht auch die „Bibliographie-Engine“ [15, 16] **camel** von Frank Bennett, Jr. aus dem Jahre 1997 von Interesse sein. Das Paket setzt die im *Blue Book* [21] (allerdings in einer früheren Edition) aufgeführten Konventionen um und bietet besondere Funktionen, wie etwa nach Klassen sortierte Quellenverweise. Mit seiner Hilfe lassen sich Verzeichnisse von Fällen, von Gesetzen und vieles mehr erzeugen. Da das **camel**-Paket jedoch zurzeit nicht weiterentwickelt wird (Freiwillige sind willkommen), muss man einige Ecken und Kanten sozusagen als besondere Eigenheiten der Software in Kauf nehmen. Da es derzeit im Wesentlichen nur für englischsprachige Texte geeignet ist, sind die nachfolgenden Beispiele auch in dieser Sprache gehalten.

Anders als die bisher beschriebenen Pakete verwendet `camel` einen eigenen Satz an Befehlen: Quellenverweise erfolgen mit dem Befehl `\source` anstelle von `\cite`, Literaturdatenbanken werden mit `\citationdata` anstatt mit `\bibliography` erstellt, die Konventionen für Quellenangaben legt `\citationstyle` anstelle von `\bibliographystyle` fest und die Ausgabe von Literaturverzeichnissen erfolgt mit `\printbibliography` als zweitem Teil der Funktion des `\bibliography`-Befehls.

Das nächste Beispiel zeigt die praktische Anwendung dieser Befehle. Der Befehl `\source` verfügt über ein optionales erstes Argument, mit dem sich die Art des Quellenverweises angeben lässt (z.B. „f“ für ausführliche Verweise, „t“ für Verweise ohne Titel und „a“ ohne Autorennamen). Mit einem zweiten optionalen Argument nach dem *schlüssel* kann man Seitenzahlen im Quellenverweis angeben.

Als interessante Funktion erkennt das Paket einige Schlüsselwörter zwischen Quellenverweisen, wie z.B. „see-also“ und „cited-in“ im nächsten Beispiel. Entsprechend werden diese Quellenverweise dann als zusammengehörig betrachtet und automatisch in die gleiche Fußnote gesetzt.

...text ¹ ... somewhat later ... ²

References

D. E. KNUTH, THE T_EXBOOK (Computers and Typesetting, 1986).

¹ D. E. KNUTH, (Computers and Typesetting, 1986); *see also* Knuth, TUGBOAT, v. 10, n. 1, p. 31 (1989).

² H. BROX AND W.-D. WALKER, BESONDERES SCHULDRECHT 24, 130, 216 (27. ed. 2002) *cited in* ZIVILPROZESSORDNUNG MIT RICHTSVERFASSUNGSGESETZ UND ANDEREN NEBENGESETZEN (59. neubearb. ed. 2002).

```
\usepackage{camel}
\forcefootnotes
\citationstyle{law}
\citationdata{jura,tex}

\ldots text \source[t]{Knuth-CT-a}
see-also \source[f]{Knuth:TB10-1}
\ldots\ somewhat later \ldots
\source[f]{bschur}[24,130,216]
cited-in \source[a]{zpo}

\printbibliography[labels=false]{all}
```

Bsp.
12-5-48

Als weitere möglicherweise interessante Funktion kann das Paket mithilfe der `\citationsubject`-Deklaration thematische Bibliographien erzeugen.

...text ¹ ... somewhat later. ... ²

Law

[1] H. BROX AND W.-D. WALKER, BESONDERES SCHULDRECHT (27. ed. 2002)

Typography

[1] D. E. KNUTH, THE T_EXBOOK (Computers and Typesetting, 1986)

[2] Knuth, *Typesetting Concrete Mathematics*, TUGBOAT, v. 10, n. 1, p. 31 (1989)

¹ THE T_EXBOOK (Computers and Typesetting, 1986); *see also* *Typesetting Concrete Mathematics*, TUGBOAT, v. 10, n. 1, p. 31 (1989).

² H. BROX AND W.-D. WALKER, (27. ed. 2002).

```
\usepackage{camel}
\citationsubject[o=tts,i=tbt]
{tex}{Typography}
\citationsubject[o=lts,i=ltb]
{jur}{Law}
\forcefootnotes
\citationstyle{law}
\citationdata{jura,tex}

\ldots text
\source[a,s=tex]{Knuth-CT-a}
see-also \source[f,s=tex]
{Knuth:TB10-1}
\ldots somewhat later\ldots
\source[t,s=jur]{bschur}
\printbibliography{jur}
\printbibliography{tex}
```

Bsp.
12-5-49

Die Quellenverweisdaten werden in externe Dateien geschrieben (die Dateierweiterung wird der `\citationsubject`-Deklaration mit `o=` übergeben). Diese Dateien müssen mit *MakeIndex* verarbeitet werden:

```
makeindex -s camel.ist -o <jobname>.ttb <jobname>.tts
makeindex -s camel.ist -o <jobname>.ltb <jobname>.lts
```

Die Ergebnisse werden dann im nächsten L^AT_EX-Lauf wieder eingelesen (Argument `i=`).

12.6 Mehrere Bibliographien in einem Dokument

In großen Dokumenten mit mehreren voneinander unabhängigen Abschnitten, in Konferenzberichten, die viele verschiedene Artikel enthalten, oder in einem Buch mit separaten Teilen, die von verschiedenen Autoren geschrieben sind, ist es manchmal erforderlich, dass jede der Einheiten über ein separates Literaturverzeichnis verfügt. In einem solchen Fall beschränken sich die Quellenverweise jeweils auf den Teil des Dokumentes, zu dem die jeweilige Bibliographie gehört.

Außerdem kann auch Bedarf an mehreren parallelen Literaturverzeichnissen bestehen, etwa für Primär- und für Sekundärliteratur. In diesem Fall muss man von jeder Stelle im Dokument auf Werke in zwei verschiedenen Bibliographien verweisen können.

Beide Anforderungen lassen sich automatisch erfüllen, wenn die Bibliographien nur unterschiedliche Werke enthalten¹ und man bereit ist, diese manuell, ohne die Hilfe von B_BT_EX, in mehreren `thebibliography`-Umgebungen zu erstellen. In diesem Falle stellen die `\bibitem`-Befehle in der jeweiligen Umgebung die passenden Querverweisdaten für die `\cite`-Befehle (oder ihre Varianten) an jeder Stelle im Dokument zur Verfügung. Dabei darf kein Werk (genauer gesagt kein Verweisschlüssel) in mehreren Bibliographien vorkommen, da dies zu der Warnung „multiply defined labels“ (mehrfach definierte Label, siehe Seite 972) und zu falschen Querverweisen führen würde. Man könnte solche Dubletten natürlich manuell korrigieren, indem man ihnen einen anderen Schlüssel zuweist.

Ohne B_BT_EX arbeiten zu müssen, zieht einige Folgen nach sich. Zunächst einmal wird es schwieriger, alle bibliographischen Einträge einheitlich zu gestalten (was B_BT_EX normalerweise automatisch übernimmt). Zum anderen wird die Verwendung der Autor-Jahr- oder Kurztitel-Zitierschemata erschwert (da das `natbib`-Paket ein besonders strukturiertes optionales `\bibitem`-Argument benötigt) oder sogar unmöglich (da sich die Struktur, die das `jurabib`-Paket erfordert, nicht für eine manuelle Erarbeitung eignet). Die für das jeweilige Paket benötigte `\bibitem`-Syntax wurde in Abschnitt 12.3 besprochen.

Damit man B_BT_EX auch für diese Aufgabe verwenden kann, musste ein Weg gefunden werden, mehrere `.bb1`-Dateien aus einem Dokument zu erzeugen. Wie in Abschnitt 12.1.3 erläutert, erfolgt die Zusammenarbeit

¹Überschneidungen können z.B. vorkommen, wenn man Konferenzberichte zusammenstellt und dabei jeder Artikel über ein eigenes Literaturverzeichnis verfügt.

	Bibliographien je Einheit			
	chapterbib	bibunits	bibtopic	multibib
Kapitelweise Bibliographie	x	x	x	n/a
Bibliographie je anderer Einheit	Eingeschränkt	x	x	n/a
Übergehen bestimmter Verweise	x	Eingeschränkt	Fehler	n/a
Zusätzliche Gesamtbibliographie	Aufwand	x	Nein	n/a
Gruppierte Bibliographien	x	Nein	Nein	n/a
Mehrere globale Bibliographien	Nein	Nein	x	x
Mehrere Bibliographien je Einheit	Nein	Nein	x	Nein
cite kompatibel	x	x	x	x
jurabib kompatibel	x	x	Eingeschränkt	x
natbib kompatibel	x	x	x	x
Unterstützt nicht sortierende BibTeX-Stile	x	x	Nein	x
Arbeitet mit .bib-Standarddateien	x	x	Nein	x
	chapterbib	bibunits	bibtopic	multibib
	Je Thema			

Blaue Einträge verweisen auf Funktionen (oder ihr Fehlen), die eine bestimmte Wahl vorschreiben können.

Tabelle 12.2: Vergleich der Pakete für mehrfache Bibliographien

mit BibTeX normalerweise folgendermaßen: Jeder Quellenverweisbefehl (z.B. \cite) schreibt seine *schlüssel-liste* als \citation-Befehl in die .aux-Datei. Auch die \bibliography- und \bibliographystyle-Befehle kopieren einfach ihre Argumente in diese Hilfsdatei. Dann liest BibTeX die zentrale .aux-Datei (und gegebenenfalls die mit \include eingeschlossenen Dateien) und durchsucht diese nach den zuvor genannten Befehlen. Aus diesen Daten erzeugt BibTeX eine einzige .bbl-Datei. Damit BibTeX in den zuvor beschriebenen Situationen funktioniert, müssen vier Hürden genommen werden:

1. Für jede Bibliographie muss eine eigene .aux-Datei als Eingabe für BibTeX erzeugt werden.
2. Jeder Quellenverweisbefehl muss seine Daten in die richtige .aux-Datei schreiben, damit BibTeX bei der Verarbeitung der .aux-Datei die entsprechenden bibliographischen Daten nur in die passende .bbl-Datei schreibt und nicht in die anderen.
3. Die erhaltenen .bbl-Dateien müssen an der richtigen Stelle wieder in L^AT_EX eingelesen werden.
4. Quellenverweise in Gliederungs- oder \caption-Befehlen müssen übergangen werden. Ein Quellenverweis an einer solchen Stelle würde später im Inhalts- oder Abbildungsverzeichnis erscheinen, wo er von L^AT_EX durch den veränderten Kontext nicht mehr richtig aufgelöst werden kann.

Die in diesem Abschnitt beschriebenen Pakete chapterbib, bibunits, bibtopic und multibib lösen die genannten Probleme auf unterschiedliche Weise. Sie haben alle ihre Vor- und Nachteile. In Tabelle 12.2 erfolgt ein kurzer

Vergleich der Pakete, wobei **blaue** Einträge auf Funktionen (oder ihr Fehlen) hinweisen, die eine bestimmte Wahl vorschreiben können, wenn man etwa nach einer Lösung für Bibliographien je einer bestimmten Gliederungseinheit, je Thema oder nach einer Kombination aus beiden sucht.

12.6.1 chapterbib – Eine Bibliographie je \include-Datei

Das chapterbib-Paket von Donald Arseneau, das auf vorhergehenden Arbeiten von Niel Kempson beruht, ermöglicht mehrere Bibliographien in einem L^AT_EX-Dokument. Dabei darf ein und dieselbe Quelle parallel in mehreren Bibliographien erscheinen.

Es löst das Problem verschiedener .aux-Dateien für B^BT_EX mithilfe des \include-Mechanismus von L^AT_EX. Je eingebundener Datei kann man eine Bibliographie erzeugen. Mit dem Paket lassen sich z.B. kapitelweise Literaturverzeichnisse für ein Dokument erstellen (daher der Name), wobei jedes Kapitel in einer eigenen Datei gespeichert ist, die mit dem \include-Befehl eingebunden wird. Dieser Ansatz unterliegt den folgenden Einschränkungen:

- Jede \include-Datei benötigt einen eigenen \bibliography-Befehl. Die im Argument aufgeführten Datenbankdateien können natürlich in jeder Datei unterschiedlich sein. Nicht so nahe liegend ist, dass jede Datei auch einen \bibliographystyle-Befehl enthalten muss, der aus Gründen der Einheitlichkeit *immer auf den gleichen Stil* verweisen sollte (Beispiel 12-6-1 auf der nächsten Seite zeigt, dass auch unterschiedliche Stile möglich sind).
- Eingebundene Dateien ohne \bibliography-Befehl dürfen keine Quellenverweisbefehle enthalten, da diese nicht aufgelöst werden können.
- Auch Verweisbefehle außerhalb der \include-Dateien können nicht aufgelöst werden, es sei denn, man verwendet auf dieser Ebene eine thebibliography-Umgebung (dies gilt nicht für Verweise im Inhaltsverzeichnis, wie im Folgenden noch erläutert wird). Ohne besondere Vorsichtsmaßnahmen muss diese Umgebung manuell erstellt werden. Verwendet man nämlich B^BT_EX für die .aux-Datei des Gesamtdokumentes, so erhält man Fehlermeldungen, da B^BT_EX bei der Verarbeitung der eingebundenen .aux-Dateien auf mehrfache \bibdata- und \bibstyle-Befehle stößt. Außerdem werden dann *alle* Quellenverweise aus *allen* \include-Dateien hinzugefügt, was sicherlich nicht erwünscht ist. Wer eine zusammenhängende Bibliographie für das ganze Dokument erzeugen möchte, kann dies mithilfe der Option rootbib erreichen. Dazu muss man diese Option jedoch in verschiedenen Phasen entfernen und wieder hinzufügen. Die Paketdokumentation gibt näheren Aufschluss über dieses Verfahren.
- Einheiten mit eigener Bibliographie beginnen immer auf einer neuen Seite (aufgrund des \include-Befehls). Für Fälle, in denen ein solches Erscheinungsbild ungeeignet ist, bietet chapterbib den Befehl \cbinput sowie die Umgebung cbunit. Beide sind in der Paketdokumentation beschrieben. Sofern man die Option gather nicht benötigt, empfiehlt sich in solchen Fällen eher das bibunits-Paket.

Die Umgebung `thebibliography` ist so voreingestellt, dass sie eine nicht nummerierte Überschrift der höchsten Gliederungsebene der gegebenen Dokumentenklasse erzeugt (in der Klasse `book` wäre dies z.B. `\chapter*`). Für Bibliographien, die sich nur auf einen Teil des Dokumentes beziehen, kann dies jedoch die falsche Ebene sein. Mithilfe der Option `sectionbib`¹ kann man in solchen Fällen `\section*`-Überschriften für Bibliographien erzwingen.

Im folgenden Beispiel werden die `\include`-Dateien `art-1.tex` und `art-2.tex` in `filecontents`-Umgebungen aufgeführt, so dass sich das Beispiel für dieses Buch automatisch erzeugen lässt. In einem echten Dokument würde es sich um verschiedene Dateien im Dateisystem des jeweiligen Computers handeln. Außerdem wird der Zähler `chapter` mithilfe des `\stepcounter`-Befehls erhöht, um große Kapitelüberschriften im Beispiel zu vermeiden. Man beachte, dass beide eingebundenen Dateien auf eine Veröffentlichung mit dem Schlüssel `Knuth-CT-a` verweisen. Sie werden tatsächlich als unterschiedliche Schlüssel behandelt, da sich der eine auf eine Veröffentlichung aus `art-1.bbl` und der andere aus `art-2.bbl` bezieht.

...siehe [Knu86] ...

Literaturverzeichnis

- [Knu86] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

...siehe [2] und [1] ...

Literaturverzeichnis

- [1] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht*. München, 27. edition, 2002.
- [2] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
\begin{filecontents}{art-1.tex}
\stepcounter{chapter} \ldots
  siehe \cite{Knuth-CT-a} \ldots
\bibliographystyle{alpha}
\bibliography{tex}
\end{filecontents}
\begin{filecontents}{art-2.tex}
\stepcounter{chapter}
\ldots siehe \cite{Knuth-CT-a}
  und \cite{bschur} \ldots
\bibliographystyle{plain}
\bibliography{tex,jura}
\end{filecontents}
\usepackage[sectionbib]
  {chapterbib}

\include{art-1}
\include{art-2}
```

Bsp.
12-6-1

Um ein Gesamtverzeichnis der zitierten Literatur zu erhalten (z.B. am Ende des Dokumentes), verwendet man die Option `gather` und platziert den Befehl `\bibliography` an der Stelle im Dokument, wo das kombinierte Literaturverzeichnis erscheinen soll. Sein Argument kann in diesem Falle leer bleiben, da es nicht für den Austausch mit `LaTeX` benötigt wird.

Vielleicht möchte man auch anstelle von `gather` die Option `duplicate` verwenden. Sie erzeugt „Kapitelbibliographien“ und zusätzlich das kombinierte Verzeichnis. Beide Optionen funktionieren nur in Dokumentenklassen, die über einen `\chapter`-Befehl verfügen. Die Überschriften, die von beiden Optionen erzeugt werden, lassen sich durch Umdefinieren des Befehls `\StartFinalBibs` anpassen, der dort ausgeführt wird, wo sich der `\bibliography`-Befehl der höchsten Ebene befindet. Im nächsten Beispiel erzeugt er eine nicht nummerierte Kapitelüberschrift, richtet mit dem Be-

¹Werden sowohl `chapterbib` als auch `natbib` eingesetzt, dann sollte man stattdessen die `sectionbib`-Option des `natbib`-Paketes verwenden!

fehl `\chaptermark` einen lebenden Kolumnentitel ein und definiert dann den `\bibname`-Befehl um, der den Text für die Überschriften der jeweiligen Teilbibliographien bereitstellt. Wie man sieht, werden die Teilbibliographien mit dem Befehl `\thechapter` nummeriert. Dieses Verfahren funktioniert daher nur, wenn alle Kapitel Bibliographien enthalten, da ansonsten die Nummerierung nicht mehr stimmt.

Sofern die Gesamtbibliographie nicht am Ende des Dokumentes erscheint, muss man sichergehen, dass `\bibname` anschließend richtig zurückgesetzt wird. Ansonsten erbt jede weitere Bibliographie in einer eingebundenen Datei die geänderte Definition.

In Dokumenten, bei denen `\section` die höchste Gliederungsebene ist, lässt sich `\StartFinalBibs` auf ähnliche Weise umdefinieren. Hier muss man anstelle von `\bibname` den Befehl `\refname` verwenden, da dieser in den Klassen benutzt wird, die sich von der Dokumentenklasse `article` ableiten.

Quellen nach Artikel

Artikel 1

[Knu86] Donald E. Knuth. *The T_EXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

Artikel 2

[1] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht*. München, 27. edition, 2002.

[2] Donald E. Knuth. *The T_EXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
% eingebundene Dateien wie
% im vorigen Beispiel
\usepackage
    [gather,sectionbib]
    {chapterbib}
\renewcommand\StartFinalBibs
    {\chapter*
     {Quellen nach Artikel}%
    \chaptermark
     {Quellen nach Artikel}%
    \renewcommand\bibname
     {Artikel-\thechapter}}
\include{art-1}
\include{art-2}
\bibliography{}
```

Bsp.
12-6-2

Quellenverweise in Gliederungs- oder `\caption`-Befehlen erscheinen letztlich in irgendeiner Art von Verzeichnis (also auf der höchsten Ebene). Das Paket `chapterbib` löst diese jedoch korrekt auf, indem es zusätzlichen Code in die `.toc`-, `.lof`- und `.lot`-Dateien einfügt, damit der `\cite`-Befehl bestimmen kann, zu welcher Bibliographie die Verweise gehören. Wer jedoch, wie in Abschnitt 2.3.4 beschrieben, zusätzliche Verzeichnislisten eingerichtet hat, muss Quellenverweise, die in diesen Listen erscheinen würden, vermeiden, da `chapterbib` diese nicht findet.

Einige B_T_EX-Stile verwenden in den erzeugten `.bbl`-Dateien leider `\newcommand`-Deklarationen anstelle des Befehls `\providecommand`, so dass diese Dateien nicht mehrmals geladen werden dürfen. Sollte es aus diesem Grund zu der Fehlermeldung „Command `(name)` already defined“ kommen, dann kann man einfach die `\bibliography`-Befehle und ihre Argumente in Klammern setzen. Man schreibt zum Beispiel:

```
{\bibliography{tex,jura}}
```

Das `chapterbib`-Paket ist mit den meisten bisher in diesem Kapitel beschriebenen Paketen für Quellenverweise kompatibel. Bei einer Kombination mit dem `babel`-Paket sollte das `chapterbib`-Paket zuerst geladen werden.

 „Command already defined“-Fehler

12.6.2 bibunits – Bibliographien für beliebige Einheiten

Das Paket `bibunits` von Thorsten Hansen (auf Grundlage früherer Arbeiten von José Alberto Fernández) erzeugt getrennte Bibliographien für verschiedene Einheiten (Teile) des Textes (Kapitel, Abschnitte oder `bibunit`-Umgebungen). Das Paket verteilt die Quellenverweise der jeweiligen Textteile in getrennte Dateien, die dann von `LaTeX` verarbeitet werden. Im Dokument kann zudem eine globale Bibliographie erscheinen und die Quellen können in beiden gleichzeitig aufgeführt werden.

Ein Weg, um die Einheiten festzulegen, die jeweils ein eigenes Literaturverzeichnis erhalten sollen, besteht darin, sie in eine `bibunit`-Umgebung zu setzen.

```
\begin{bibunit}[stil] ... \putbib[datei-liste] ... \end{bibunit}
```

Das optionale Argument `stil` legt für die Bibliographie einen anderen Stil fest, als er möglicherweise bereits vorgegeben wurde (wie im Folgenden beschrieben). Die Bibliographie wird mit dem Befehl `\putbib` anstelle des `\bibliography`-Befehls platziert. Wie man am Beispiel sehen kann, darf sie überall in dem jeweiligen Textteil erscheinen. Das optionale Argument `datei-liste` enthält eine durch Kommas unterteilte Liste von `LaTeX`-Datenbankdateien. Auch hier kann eine Vorgabe festgelegt werden. Mithilfe des Befehls `\defaultbibliographystyle` lässt sich ein bestimmter `LaTeX`-Stil voreinstellen, ansonsten wird `plain` vorgegeben. Genauso lässt sich mit dem Befehl `\defaultbibliography` eine vorgegebene Liste von `LaTeX`-Datenbanken definieren. Andernfalls wird die Datei `\jobname.bib` gesucht. Die Deklarationen für die Voreinstellungen müssen nach `\begin{document}` stehen, um in Kraft zu treten.

Voreinstellungen
festlegen

1 Die Erste

[1] wurde verwendet um [2] zu erzeugen.

Literatur

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.

[2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Eine weitere

Literatur

[1] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht*. München, 29. edition, 2003.

Wie von [1] beschrieben ...

```
\usepackage{bibunits}
\defaultbibliographystyle{plain}

\section{Die Erste}
\begin{bibunit}[plain]
\cite{GNUMake} wurde verwendet um
\cite{Knuth:TB10-1} zu erzeugen. \putbib[tex]
\end{bibunit}
\section{Eine weitere}
\begin{bibunit}[plain]
\putbib[jura]
Wie von \cite{aschur} beschrieben \ldots
\end{bibunit}
```

Bsp.
12-6-3

Für jede Einheit schreibt `bibunits` die Zitierschlüssel (für den Austausch mit `LaTeX`) in die Datei `bu<num>.aux`, wobei `<num>` eine ganze Zahl ist, beginnend mit 1. Um also die erforderlichen Literaturverzeichnisse zu erzeugen,

muss man die Dateien bu1, bu2 usw. mit $\text{\LaTeX}$ verarbeiten. Daraus folgt, dass man normalerweise nicht mehr als ein Dokument, welches das bibunits-Paket verwendet, in einem Verzeichnis verarbeiten kann, da ansonsten die Hilfsdateien überschrieben werden.¹

Nach Erzeugen der Bibliographien muss man $\text{\LaTeX}$ noch mindestens zweimal laufen lassen um die neuen Querverweise aufzulösen. Man beachte, dass ältere Versionen des Paketes *nicht* warnend darauf hinweisen, dass noch weitere Läufe erforderlich sind.

Eine zusätzliche globale Bibliographie, neben denjenigen der einzelnen Einheiten, lässt sich wie üblich mit den Befehlen `\bibliography` und `\bibliographystyle` erzeugen. Außerhalb der bibunit-Umgebungen sollten die Standardbefehle für Quellenverweise auf die globale Bibliographie verwendet werden. Innerhalb der bibunit-Umgebungen kann man mithilfe der Befehle `\cite*` und `\nocite*`, anstelle von `\cite` und `\nocite`, einen Quellenverweis auf sowohl die lokale als auch die globale Bibliographie erzeugen. Dabei gibt es jedoch einige Einschränkungen. Bei gleichzeitig geladenem natbib-Paket wird der Befehl `\cite*` gemäß seiner Definition durch dieses Paket verarbeitet und kann somit nicht für einen globalen Quellenverweis verwendet werden. In diesem Falle bietet sich `\nocite` zum Gebrauch außerhalb der jeweiligen Einheit an. Außerdem sollte man auf numerische Label für Quellenverweise verzichten, da sie mit großer Wahrscheinlichkeit zu mehrdeutigen Kennungen in der globalen Bibliographie führen, wie auch das nächste Beispiel zeigt. Ein $\text{\LaTeX}$ -Stil wie alpha ist hier die bessere Wahl.

1 Die Erste

[1] wurde verwendet um
[2] zu erzeugen.

Literatur

- [1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Re-compilation*, 2000.
- [2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Eine weitere

Nachzulesen in [1] ...

Literatur

- [1] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

Globale Verweise

- [1] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

```
\usepackage{bibunits}
\section{Die Erste}
\begin{bibunit}[plain]
\cite{GNUmake} wurde
verwendet um
\cite*{Knuth:TB10-1} zu
erzeugen. \putbib[tex]
\end{bibunit}
\section{Eine weitere}
\begin{bibunit}[plain]
Nachzulesen in
\cite*{Knuth:TB10-1}
\ldots \putbib[tex]
\end{bibunit}
\renewcommand\refname
{Globale Verweise}
\bibliographystyle{plain}
\bibliography{tex}
```

Anstatt überall im Dokument `\cite*` zu verwenden, kann man besser die Paketoption `globalciterecopy` einstellen. Dadurch werden alle lokalen Quellenverweise zusätzlich in die globale Bibliographie kopiert.

¹Falls erforderlich, kann man das Paket auch anweisen, verschiedene Namen zu verwenden (siehe Paketdokumentation).

Die Textabschnitte, die eine eigene Bibliographie erhalten sollen, lassen sich nicht nur mithilfe von `bibunit`-Umgebungen festlegen. Stattdessen kann man auch angeben, in welcher Gliederungsebene automatisch ein eigenes Literaturverzeichnis erzeugt werden soll.

```
\bibliographyunit[einheit]
```

Dieser Befehl bestimmt, für welche Teile eines Dokumentes eigene Literaturhinweise erzeugt werden müssen: Der Wert `\chapter` im Argument *einheit* sorgt z.B. für kapitelweise Bibliographien, während `\section` Bibliographien pro Abschnitt erzeugt. Ohne optionales Argument deaktiviert der Befehl `\bibliographyunit` alle weiteren Bereiche für eigene Literaturverzeichnisse. Solange `\bibliographyunit` aktiv ist, geben die Befehle `\bibliographystyle` und `\bibliography` vor, welche BibTeX-Dateien und welcher Stil für die globale sowie die einzelnen Bibliographien zu verwenden sind. Einstellungen, die nur für lokale Literaturhinweise gelten sollen, lassen sich mit `\bibliography*` und `\bibliographystyle*` vornehmen. Diese Deklarationen müssen hinter `\begin{document}` stehen und dürfen *nicht* in der Präambel vorkommen.

Dieser Ansatz hat jedoch einen Nachteil: Die herkömmliche Definition der Umgebung `thebibliography`, welche die Literaturverzeichnisse umgibt, erzeugt eine Überschrift der höchsten Gliederungsebene. Verwendet man nun `\chapter`-Einheiten in einem Dokument der Klasse `report`, dann beendet die Überschrift jener Umgebung die Einheit und folglich erhält man nicht aufgelöste Verweise, wie im nächsten Beispiel zu sehen ist (dort werden `\section`-Einheiten in einem Dokument der Klasse `article` benutzt).

Nicht aufgelöste
Literaturhinweise



1 Die Erste

[?] wurde verwendet um
[?] zu erzeugen.

Literatur

- [1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Re-compilation*, 2000.
- [2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Eine weitere

Wie nachzulesen in [?] ...

Literatur

- [1] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht*. München, 29. edition, 2003.

```
\usepackage{bibunits}
\bibliographyunit[\section]
\bibliographystyle*{plain}
\bibliography*{tex,jura}
\section{Die Erste}
  \cite{GNUMake} wurde
  verwendet um
  \cite{Knuth:TB10-1}
  zu erzeugen. \putbib
\section{Eine weitere}
  Wie nachzulesen in
  \cite{aschur} \ldots
\putbib
```

Bsp.
12-6-5

Dieses Problem lässt sich lösen, indem man eine eigene Definition für die `thebibliography`-Umgebung nutzt, die mit einer anderen Gliederungsebene arbeitet als die `\bibliographyunit`-Deklaration. Oder man verwendet die Option `sectionbib` (mit `\section*` als Überschrift für die

thebibliography-Umgebung) oder subsectionbib (mit \subsection*) um die thebibliography-Umgebung entsprechend zu verändern.

1 Die Erste

[1] wurde verwendet um [2] zu erzeugen.

Literatur

[1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Re-compilation*, 2000.

[2] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.

2 Eine weitere

Wie nachzulesen in [1] ...

Literatur

[1] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht*.

```
\usepackage[subsectionbib]{bibunits}

\bibliographyunit[\section]
\bibliographystyle*{plain}
\bibliography*{tex,jura}
\section{Die Erste}
\cite{GNUmake} wurde
verwendet um
\cite{Knuth:TB10-1}
zu erzeugen.
\putbib
\section{Eine weitere}
Wie nachzulesen in
\cite{aschur} \ldots
\putbib
```

Bsp.
12-6-6

Man beachte, dass die mit dem Befehl \bibliographyunit festgelegte Gliederungsstufe eine andere sein muss, als die für die Option angegebene. Im letzten Beispiel war die Einheit \section, so dass subsectionbib für die Option verwendet wurde.

Für den Umgang mit Quellenverweisen in Überschriften und Legenden (siehe Seite 772) bietet das Paket die Option labelstoglobalaux. Diese hat jedoch den Nebeneffekt, dass solche Literaturhinweise in der globalen Bibliographie erscheinen und numerische Verweisschemata sehr wahrscheinlich falsche Label erzeugen. Die Paketdokumentation enthält nähere Erläuterungen zu diesem Punkt.

12.6.3 bibtopic – Nach Themen sortierte Literaturhinweise

Im Gegensatz zu chapterbib und bibunits, die Quellenverweise für einzelne Bereiche eines Dokumentes zusammenstellen, kombiniert das Paket bibtopic Literaturhinweise nach Themen. Es stammt von Stefan Ulrich, der bei der Entwicklung auf frühere Arbeiten von Pierre Basso zurückgegriffen hat. Mithilfe dieses Paketes kann man z.B. ein primäres Quellenverzeichnis und eine Liste weiterführender Literatur anbieten, oder alle Verweise auf Bücher getrennt von Verweisen auf Artikel aufführen.

Innerhalb des Dokumentes werden alle Quellenverweise mit \cite, \nocite und deren Varianten erzeugt (sofern natbib oder ähnliche Pakete geladen sind). Die Einteilung nach Themen wird also zu einem späteren Zeitpunkt vorgenommen. Um thematische Literaturverzeichnisse zu erzeugen, müssen die bibliographischen Einträge zu einem Thema in einer eigenen BibTeX-Datenbankdatei stehen (z.B. eine Datei für Primär- und eine für Sekundärliteratur). Die Bibliographien werden dann mithilfe mehrerer btSect-Umgebungen erzeugt. In Kapitel 13 werden verschiedene Verfahren zum Erzeugen von Datenbankdateien beschrieben. So kann man z.B. mithilfe des Programms bibtool nach bestimmten Kriterien Quellenangaben aus größeren BibTeX-Datenbanksammlungen extrahieren.

```
\begin{btSect}[stil]{datei-liste}
```

Die Umgebung `btSect` erzeugt ein Literaturverzeichnis für alle Quellenverweise des gesamten Dokumentes, sofern sich diese auf Einträge in den $\text{\BibTeX}$ -Datenbankdateien beziehen, die in der, durch Kommas unterteilten, Liste des Argumentes `datei-liste` aufgeführt sind. Das optionale Argument `stil` legt den $\text{\BibTeX}$ -Stil für die aktuelle Bibliographie fest. Ohne dieses Argument wird der zuvor durch die `\bibliographystyle`-Deklaration bestimmte Stil verwendet. Ist keine solche Deklaration erfolgt, so wird der $\text{\BibTeX}$ -Stil `plain` als Voreinstellung benutzt.

Die Umgebung selbst erzeugt keine Überschrift, es sei denn, das Paket wurde mit der Option `printheadings` geladen. Normalerweise muss man mithilfe von `\section*` oder einem ähnlichen Befehl eine eigene Überschrift hinzufügen.

```
\btPrintCited    \btPrintNotCited    \btPrintAll
```

Innerhalb einer `btSect`-Umgebung lässt sich mit einem der hier aufgeführten Befehle definieren, welche der Literaturangaben aus den Datenbanken, die im Argument `datei-liste` aufgeführt sind, übernommen werden. Der Befehl `\btPrintCited` gibt alle Quellen aus den genannten Datenbanken aus, auf die irgendwo im Dokument verwiesen wird, `\btPrintNotCited` druckt die nicht zitierten Quellen und `\btPrintAll` gibt alle Einträge der $\text{\BibTeX}$ -Datenbankdateien aus.

Das folgende Beispiel demonstriert das zugrunde liegende Konzept anhand von zwei Themen: „ $\text{\TeX}$ “- und „juristische“ Literatur. Die erste Bibliographie bedient sich des vorgegebenen Stils `plain`, während für die zweite ausdrücklich der $\text{\BibTeX}$ -Stil `abbrv` festgelegt wird. (Diese Kombination soll lediglich als Beispiel dienen – das Mischen von Stilen ist normalerweise keine gute Idee.) Wie man sieht, nummeriert `bibtopic` automatisch die Einträge aller Literaturverzeichnisse durch, wenn man einen numerischen $\text{\BibTeX}$ -Stil gewählt hat. Dadurch wird sichergestellt, dass die Verweise im Dokument eindeutig sind.

Bisher wurden die Quellen [3], [2] und [1] betrachtet.

Juristische Literatur

- [1] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht*. München, 27. edition, 2002.
- [2] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht*. München, 29. edition, 2003.

$\text{\TeX}$ Literatur

- [3] D. E. Knuth. *The $\text{\TeX}$ book*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

```
\usepackage{bibtopic}
```

```
Bisher wurden die Quellen
\cite{Knuth-CT-a}, \cite{aschur}
und \cite{bschur} betrachtet.
\begin{btSect}{jura}
  \section*{Juristische Literatur}
  \btPrintCited
\end{btSect}
\begin{btSect}{abbrv}{tex}
  \section*{\text{\TeX} Literatur}
  \btPrintCited
\end{btSect}
```

Bsp.
12-6-7

Das `bibtopic`-Paket erzeugt für jede `btSect`-Umgebung eine eigene `.aux`-Datei, deren Name per Voreinstellung aus dem Namen des Quelldokumentes

(\jobname) und einer fortlaufenden Nummer gebildet wird. Dieses Namensschema lässt sich durch Undefinieren von \thebtauxfile ändern, wobei man mithilfe des Zählers btauxfile automatisch eine fortlaufende Nummerierung erhält.

Für die Beispiele im Buch wurde folgende abgewandelte Definition verwendet:

```
\renewcommand\thebtauxfile{\jobname+\arabic{btauxfile}}
```

Das bibtopic-Paket ist nicht kompatibel zu chapterbib und bibunits. Es verfügt jedoch über die Umgebung btUnit um die Quellenverweise auf logische Einheiten zu beschränken. Innerhalb solcher Einheiten lässt sich die btSect-Umgebung ganz normal verwenden, so dass auch thematische Literaturverzeichnisse je Kapitel oder Abschnitt möglich sind. In diesem Falle müssen *alle* Quellenverweise innerhalb dieser Bereiche stehen (Quellenverweise in Überschriften oder Legenden werden nicht verarbeitet, so dass man darauf achten muss, dass sie gar nicht erst vorkommen). Das Paket ist so voreingestellt, dass numerische Stile ihre Nummerierung mit jedem Bereich neu beginnen (z.B. bei einem Konferenzbericht mit jedem Artikel). Wer eine durchgehende fortlaufende Nummerierung wünscht, kann dazu die Option unitcntnoreset verwenden.

*Bibliographische
Themen je logischer
Einheit*

Das bibtopic-Paket lässt sich zwar mit den meisten BibTeX-Stilen kombinieren, es gibt jedoch einige Ausnahmen. Die wichtigste darunter ist, dass es bei „nicht sortierenden“ Stilen (wie etwa unsrt) anders reagiert als erwartet. Bei der Kombination mit einem solchen Stil ergibt sich die Reihenfolge im Literaturverzeichnis aus derjenigen in der Datenbank und richtet sich *nicht* nach der Reihenfolge der Quellenverweise im Dokument. Wer letztere Sortierung benötigt, sollte das im nächsten Abschnitt beschriebene multibib-Paket verwenden.

*Probleme mit nicht
sortierenden
BibTeX-Stilen*

Das bibtopic-Paket ist zu den meisten anderen Paketen kompatibel, die den Quellenverweismechanismus erweitern, einschließlich cite, natbib und jurabib. Es gibt allerdings einige Einschränkungen hinsichtlich der Erstellung der Literaturverzeichnisse. So können etwa Befehle zur Veränderung des Layouts, wie sie von natbib und jurabib bereitgestellt werden, fehlschlagen. Die Paketdokumentation enthält nähere Einzelheiten zu diesem Thema.

Bisher wurden Knuth: The T_EXbook und Brox/Walker: Allgemeines Schuldrecht betrachtet.

T_EX Literatur

Knuth, Donald E.: The T_EXbook. Band A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0–201–13447–0

Juristische Literatur

Brox, Hans/Walker, Wolf-Dietrich: Allgemeines Schuldrecht. 29. Auflage. München, 2003

```
\usepackage{bibtopic,jurabib}
\bibliographystyle{jurabib}
```

Bisher wurden \cite{Knuth-CT-a} und \cite{aschur} betrachtet.

```
\begin{btSect}{tex}
  \section*{\TeX{} Literatur}
  \btPrintCited
\end{btSect}
\begin{btSect}{jura}
  \section*{Juristische Literatur}
  \btPrintCited
\end{btSect}
```


12.6.4 multibib – Separate globale Bibliographien

Wie bibtopic erzeugt auch das multibib-Paket von Thorsten Hansen getrennte globale Bibliographien. Während das erste Paket die Literaturverzeichnisse über getrennte BibTeX-Datenbankdateien erzeugt, arbeitet multibib mit unterschiedlichen Quellenverweisbefehlen, um die Verweise verschiedenen Bibliographien zuzuordnen.

Jede Methode hat ihre Vor- und Nachteile. Durch das multibib-Paket wird die Zugehörigkeit der einzelnen Quellenverweise bereits im Quelldokument deutlich. Folglich erfordert das durchgängige Verschieben einer Quelle aus einer Bibliographie in eine andere Änderungen an mehreren Stellen im Dokument. Im Falle des bibtopic-Paketes muss man hier lediglich den entsprechenden Datenbankeintrag aus einer Datei in die andere verschieben. Andererseits erfordert das bibtopic-Paket häufig für jedes neue Dokument eigens zusammengestellte .bib-Dateien, während man für multibib einfach allgemein verfügbare Sammlungen von BibTeX-Datenbankdateien verwenden kann.

Neuere Versionen des multibib-Paketes sind zu den meisten anderen Paketen kompatibel, die den Quellenverweismechanismus erweitern, einschließlich cite, jurabib und natbib. Außerdem verfügt das Paket über eine generelle Schnittstelle, mit deren Hilfe man den Quellenverweisbefehlen beliebige Erweiterungen hinzufügen kann, die dann von multibib erkannt werden.

```
\newcites{typ}{titel}
```

Die Deklaration `\newcites` definiert eine zusätzliche Reihe von Quellenverweisbefehlen für einen neuen Typ von Quellenverweisen. Die Überschrift für das entsprechende neue Literaturverzeichnis ergibt sich aus dem Argument *titel*. Nach erfolgter Deklaration sind vier zusätzliche Befehle verfügbar. Der Befehl `\cite{typ}` erzeugt wie `\cite` einen Quellenverweis im Text und die entsprechende Quelle erscheint im Literaturverzeichnis für diesen neuen *typ*. Ebenso erzeugt `\nocite{typ}` einen Eintrag für die entsprechende Bibliographie, ohne dass ein Quellenverweis im Text erfolgt. Das entsprechende Literaturverzeichnis wird an der Stelle im Text ausgegeben, an welcher der Befehl `\bibliography{typ}` erfolgt und der BibTeX-Stil für dieses Verzeichnis wird durch den Befehl `\bibliographystyle{typ}` festgelegt. Es folgt ein Beispiel.

Eins der Bücher über Graphiken in L^AT_EX ist [1]; Vorschläge zu Quellenverweisen finden sich in [vL92].

Literatur zu L^AT_EX

- [1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997.

Allgemeine Literatur

- [vL92] Mary-Claire van Leunen. *A handbook for scholars*. Oxford University Press, Walton Street, Oxford OX2 6DP, UK, 92.

```
\usepackage{multibib}
\newcites{latex}
{Literatur zu \LaTeX{}}
```

Eins der Bücher über Graphiken in \LaTeX{} ist `\citelatem{LGC97}`; Vorschläge zu Quellenverweisen finden sich in `\cite{vLeunen:92}`.

```
\bibliographystylelatex{plain}
\bibliographylatex{tex}
```

```
\renewcommand\refname
{Allgemeine Literatur}
\bibliographystyle{alpha}
\bibliography{tex}
```

Bsp.
12-6-9

Die `\newcites`-Deklaration kann wiederholt verwendet werden, um noch weitere Typen von Quellenverweisen zu erzeugen. Ihre Anzahl wird nur durch die Menge der Ausgabedateien beschränkt, die $\TeX$ gleichzeitig verwenden kann. Die `.aux`-Datei zur Kommunikation mit $\LaTeX$ heißt `(typ).aux`. Aus diesem Grund muss man die Bezeichnung für das erste Argument `typ` des `\newcites`-Befehls sorgfältig wählen, damit nicht versehentlich andere `.aux`-Dateien überschrieben werden.

Bei numerischen Formaten für Quellenverweise werden die Quellen aller Bibliographien per Vorgabe fortlaufend durchnummeriert, um mehrdeutige Verweise zu vermeiden. Durch die Option `resetlabels` beginnt die Nummerierung in jeder Bibliographie neu.

$\LaTeX$ verfügt über eine Schnittstelle zum Einbinden von Graphiken.¹ Das in $\LaTeX$ vorgegebene Zitierschema ist das numerische.²

Literatur zu $\LaTeX$

- [1] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The $\LaTeX$ Graphics Companion: Illustrating Documents with $\TeX$ and PostScript*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1997.

Allgemeine Literatur

- [2] Mary-Claire van Leunen. *A handbook for scholars*. Oxford University Press, Walton Street, Oxford OX2 6DP, UK, 92.

```
\usepackage[super]{cite}
\usepackage{multibib}
\newcites{latex}
    {Literatur zu \LaTeX{}}

\LaTeX{} verfügt über eine
Schnittstelle zum Einbinden von
Graphiken \citelate{LGC97}. Das in
\LaTeX{} vorgegebene Zitierschema
ist das numerische \cite{vLeunen:92}.

\bibliographystyle{latex{plain}}
\bibliographylate{tex}
\renewcommand\refname
    {Allgemeine Literatur}
\bibliographystyle{plain}
\bibliography{tex}
```

Bsp.
12-6-10

KAPITEL 13

Erzeugen von Literaturverzeichnissen

13.1 Das BibTEX-Programm und seine Varianten	786
13.2 Das BibTEX-Datenbankformat	789
13.3 Online-Bibliographien	802
13.4 Werkzeuge für Bibliographiedatenbanken	803
13.5 Formatieren von Bibliographien mit BibTEX-Stilen	820
13.6 Die BibTEX-Programmiersprache	835

Während das Inhaltsverzeichnis (siehe Abschnitt 2.3) und der Index (siehe Besprechung in Kapitel 11) die Orientierung im Buch erleichtern, eröffnet ein Literaturverzeichnis (auch Bibliographie genannt) dem Leser die Möglichkeit, Quellen zu überprüfen und interessante Themen zu vertiefen. Um dies zu unterstützen, sollten die Quellenangaben exakt sein und mit minimalem Aufwand zum entsprechenden Werk führen.

Es gibt viele unterschiedliche Möglichkeiten, Literaturverzeichnisse zu gestalten, und in verschiedenen wissenschaftlichen Bereichen gibt es präzise Richtlinien für ihre Erstellung. Das Kapitel über Bibliographien in dem Nachschlagewerk *The Chicago Manual of Style* [38] bietet einen interessanten Überblick über die im englischsprachigen Raum üblichen Stile. Normalerweise müssen sich Autoren an die Vorgaben ihrer Verleger halten. Will man ein Buch oder einen Artikel veröffentlichen, so besteht eine wichtige Aufgabe darin, ein diesen Richtlinien entsprechendes Literaturverzeichnis zu erstellen.

Die traditionelle Methode, solche Listen ohne die systematische Hilfe eines Computers zusammenzustellen, birgt folgende Probleme:

- Literaturverweise sind schwer zu vereinheitlichen, insbesondere in einem Dokument mit Beiträgen von mehreren Autoren. Schwierigkeiten ergeben sich z.B. durch die Verwendung von ausgeschriebenen Vornamen gegenüber Kurzformen (mit oder ohne Abkürzungspunkt), die Schreibweise von Titeln in Kursivschrift oder Anführungszeichen, Angaben wie

„Herausgeber“ oder „Hrsg.“ sowie verschiedene Formate für Zeitschriftennummern.

- Ein Literaturverzeichnis, das nach einem bestimmten Schema angelegt ist (z.B. alphabetisch nach Autor und Jahr), lässt sich nur sehr schwer in einen anderen Stil übertragen (wie etwa eine numerische Reihenfolge entsprechend der Quellenverweise), sollte ein Verleger dies wünschen.
- Es ist schwierig, eine große Sammlung bibliographischer Quellen anzulegen, die sich in verschiedenen Dokumenten wiederverwenden lässt.

Kapitel 12 befasste sich hauptsächlich mit Quellenverweisen im Text. Das vorliegende Kapitel konzentriert sich nun auf die Formatierung von Literatur- und Quellenverzeichnissen sowie auf die verschiedenen Möglichkeiten, Quellensammlungen in Form von Datenbanken zu pflegen. Dabei liegt ein wichtiger Schwerpunkt auf dem Programm `BibTeX` von Oren Patashnik, das sehr gut mit `LATEX` zusammenarbeitet.

Zunächst wird dieses Programm mit seinen Varianten vorgestellt, wobei auch jüngere Bestrebungen, einen Nachfolger zu entwickeln, angesprochen werden. Dann folgt eine ausführliche Beschreibung des `BibTeX`-Datenbankformates mit umfassenden Informationen darüber, wie man bibliographische Daten so aufbereitet, dass sie mit `BibTeX` verarbeitet werden können. Wer keine eigenen Literaturdaten sammeln möchte, kann sich zahlreicher Online-Datenbanken bedienen, die solche Daten im `BibTeX`-Format anbieten. Einige dieser Datenbanken werden in Abschnitt 13.3 vorgestellt.

Hat man erst einmal eine `BibTeX`-Datenbank mit Informationen gefüllt, so stellt sich natürlich die Frage, mit welchen Hilfsmitteln sie sich pflegen lässt. Abschnitt 13.4 stellt entsprechende Werkzeuge für jeden Geschmack vor, von Dienstprogrammen auf Kommandozeilenbasis bis hin zu Programmen mit graphischer Benutzeroberfläche für verschiedene Plattformen.

Sobald all diese Fragen geklärt sind, wendet sich Abschnitt 13.5 wieder der setzerischen Tätigkeit zu und untersucht, wie man mithilfe unterschiedlicher `BibTeX`-Stile aus den gleichen Daten verschieden gestaltete Literaturverzeichnisse erzeugen kann. Da es nicht für alle Layoutanforderungen immer gleich eine geeignete Stildatei gibt, wird in Abschnitt 13.5.2 besprochen, wie man mit dem Paket `custom-bib` passende `BibTeX`-Stile erzeugen kann, ohne sie programmieren zu müssen.

Alle Leser, die gerne tiefer in die Geheimnisse der Programmierung von `BibTeX`-Stilen eintauchen möchten (oder dies müssen), finden im letzten Abschnitt nähere Einzelheiten zum Aufbau solcher Stildateien. Dazu gehört auch ein kurzer Überblick über die verfügbaren Befehle und eingebauten Funktionen. Dort wird die generische Dokumentationsdatei für `BibTeX`-Stile `btbxst.doc` erklärt und es wird gezeigt, wie man eine vorhandene Stildatei an bestimmte Vorgaben oder andere Sprachen anpasst.

13.1 Das `BibTeX`-Programm und seine Varianten

Das Programm `BibTeX` von Oren Patashnik wurde als flexible Lösung entwickelt, mit der sich automatisch Literaturverzeichnisse erzeugen lassen, die bestimmten Layoutvorgaben entsprechen müssen. Es entdeckt selbständig, welche Quellenangaben ein `LATEX`-Dokument benötigt (indem es dessen `.aux`-Datei

oder -Dateien liest), wählt die entsprechenden bibliographischen Daten aus einer oder mehreren angegebenen Datenbanken und formatiert diese nach einem im Dokument vorgegebenen Stil. Das Programm erzeugt eine Datei, welche das Literaturverzeichnis im L^AT_EX-Format enthält. Diese wird dann beim nächsten L^AT_EX-Lauf automatisch geladen und verwendet. In Abschnitt 12.1.3 auf Seite 711 wurde die Schnittstelle zwischen den beiden Programmen bereits in einiger Ausführlichkeit besprochen.

Als dieses Buch geschrieben wurde, war BibTeX in der Version 0.99c erhältlich. Wirft man jedoch einen Blick in die erste Ausgabe dieses Buches (die vor zehn Jahren verfasst wurde), so bemerkt man, dass auch dort bereits die Version 0.99c behandelt wurde. Version 0.99a wurde anscheinend schon 1986 veröffentlicht. Anders ausgedrückt ist dieses Programm seit sehr langer Zeit unverändert geblieben. Dementsprechend erfreut sich das BibTeX-Datenbankformat in der L^AT_EX-Welt großer Beliebtheit und viele Benutzer haben im Laufe der Jahre zahlreiche Quelldaten gesammelt. Daher ist es nicht weiter verwunderlich, dass alle Entwicklungen der letzten zehn Jahre stets dieses Format als Standard herangezogen haben.

Dieser Abschnitt gibt einen kurzen Überblick über die Entwicklungen auf diesem Gebiet. Besonders in den letzten Jahren tauchten einige neue Vorhaben auf, es gibt jedoch auch Projekte, die bereits ein paar Jahre alt sind.

13.1.1 bibtex8 – Eine 8-Bit-Implementierung von BibTeX

Seinem Alter und Ursprung entsprechend liegt BibTeX als auf ASCII basierende 7-Bit-Version vor. Es kann zwar sprachspezifische Buchstaben verarbeiten, jedoch sind seine Fähigkeiten in dieser Hinsicht sehr begrenzt. Das Programm BibTeX8 von Niel Kempson und Alejandro Aguilar-Sierra ist eine neue Implementierung von BibTeX, in der man auch Daten zur Sortierreihenfolge angeben kann. Dadurch kann jeder Anwender die eigenen BibTeX-Datenbankeinträge in seiner bevorzugten 8-Bit-Fontkodierung speichern und das inputenc-Paket in seinem L^AT_EX-Dokument verwenden (siehe Abschnitte 7.5.2 und 7.11.3). Angaben zur Sortierreihenfolge für eine bestimmte Kodierung können über die Befehlszeile erfolgen, wie etwa:

```
bibtex8 -c 885911at lb2
```

Die Sortierreihenfolge wird in Dateien mit der Erweiterung .csf gespeichert (im vorigen Beispiel in der Datei 885911at.csf). Die Distribution enthält eine Reihe solcher Dateien für die gängigsten Kodierungen. Der Aufbau dieser Dateien ist gut dokumentiert, so dass man gegebenenfalls fähig sein sollte, eine eigene .csf-Datei zu erstellen. Verwandte Kommandozeilenoptionen sind -7 und -8, die dazu dienen, eine 7-Bit- bzw. 8-Bit-Verarbeitung ohne besondere Sortierreihenfolge zu erzwingen.

Das BibTeX8-Programm verfügt zudem über einen zweiten Satz von Kommandozeilenoptionen, mit deren Hilfe sich die internen Tabellen erweitern lassen. Als 1995 die erste Version des Programms geschrieben wurde, verfügte Standard-BibTeX nur über kleine, festgeschriebene interne Tabellen, die es nicht zuließen, eine Bibliographie mit z.B. mehreren Hundert Quellen zu setzen. Heute verwenden die meisten Installationen großzügigere interne Tabellen (z.B. 5000 Quellen), so dass der größere diesbezügliche Spielraum von

BibTeX8 nur selten benötigt wird. Sollte jedoch irgendein Verarbeitungslauf an diese Grenzen stoßen und eine Meldung wie „Sorry—you’ve exceeded BibTeX’s...“ („Sie haben leider BibTeXs ... Grenzen überschritten“) auslösen, kann man BibTeX8 mit einer passenden Befehlszeile verwenden, um dieses Problem zu umgehen. Informationen über die verfügbaren Optionen erhält man, indem man das Programm ohne weitere Eingaben oder mit der Option `-h` oder `--help` aufruft.

13.1.2 Neuere Entwicklungen

Abgesehen von BibTeX und BibTeX8, die beide schon seit längerer Zeit vorliegen, gibt es einige neuere Entwicklungen, die sich mit dem Erstellen von Bibliographien befassen. In diesem Abschnitt werden drei Projekte vorgestellt, die für die Leser von Interesse sein könnten. Es ist durchaus möglich, dass einzelne Projekte in Zukunft zusammengefasst werden. Daher sollte man diese Liste als Momentaufnahme der Situation 2003 ansehen und als Beweis dafür, dass es wieder ein stärkeres Interesse an einer Weiterentwicklung gibt.

bibulus – Bibliographien mit XML und perl

Bei dem Programm bibulus von Thomas Widmann handelt es sich um einen in perl geschriebenen BibTeX-Ersatz.¹ Es greift nicht auf das Format der BibTeX-Datenbankdateien zurück, sondern verwendet Einträge im XML-Format und verfügt über eine eigene Dokumententyp-Definition (`bibulus.dtd`). Dadurch lassen sich die bibliographischen Daten mit jeder XML-fähigen Anwendung verarbeiten. Damit man existierende `.bib`-Dateien weiterverwenden kann, bietet das Programm ein eigenes Werkzeug zur Konvertierung von BibTeX-Datenbanken in das XML-Format.

Das bibulus-Programm verwendet intern Unicode und ist damit mehrsprachig. Gleichzeitig kann es in anderen Kodierungen ausgegebene Daten lesen und schreiben. Die vom Programm erzeugten Texte wurden in viele verschiedene Sprachen übersetzt. Die aktuelle bibulus-Version unterstützt über ein Dutzend Sprachen.

Aus Sicht des Programms stellt L^AT_EX nur eine von vielen verschiedenen Ausgabemöglichkeiten dar. Die weiteren Formate reichen von einfachem Text über HTML zu Eingabeformaten für andere Programme, die Literaturdaten verarbeiten.

Wie bei den anderen beiden Programmen, die im Folgenden noch beschrieben werden, handelt es sich bei bibulus um ein in der Entwicklung befindliches Projekt. Es ist über die Webadresse <http://www.nongnu.org/bibulus> erhältlich, unter der man auch Näheres über das Projekt erfährt.

BibTeX++ – Ein BibTeX-Nachfolger in Java

Das BibTeX++-Projekt ist eine auf Java basierende Implementierung eines Programms zum Verwalten von Literaturdaten. Es wurde von Emmanuel Donin de Rosière geschrieben, und zwar im Rahmen einer Diplomarbeit [42], die

¹Für die Installation und Verwendung des Programms ist mindestens die perl-Version 5.8 erforderlich.

von Ronan Keryell betreut wurde. Da es als BibTeX-Nachfolger gedacht ist, eignet es sich natürlich sehr gut für eine Zusammenarbeit mit L^AT_EX. Es akzeptiert jedoch auch andere Bibliographieformate und unterschiedliche Formatierungssprachen und kann Daten für verschiedene Satzsysteme erzeugen. Das Programm ist in eine webbasierte Umgebung eingebettet, so dass es fehlende Informationen von verschiedenen Internetquellen abrufen kann. BibTeX++ bedient sich des Baukastenprinzips (Plugin-Konzept). Man kann seinen Funktionsumfang also dynamisch erweitern, etwa um besondere Formatierungskonventionen zu unterstützen oder um Daten für andere Satzprogramme auszugeben.

Bereits vorhandene BibTeX-Stildateien lassen sich in BibTeX++-Stile konvertieren. Dies geschieht mithilfe eines Übersetzungsprogramms, das ebenfalls im Rahmen des Projektes entwickelt wurde. Das Ergebnis lässt sich mittels der BibTeX++-Konzepte noch weiter anpassen, so dass die Entwicklung eines neuen Stils erleichtert wird.

Die Webseite des Projektes hat die Internetadresse <http://bibtex.enstb.org>. Dort befinden sich auch ein CVS-Repository (Versionsverwaltung) sowie kompilierte Binärdateien und weitere Informationen.

MBibTeX – Ein mehrsprachiger Nachfolger von BibTeX

Das Programm MBibTeX von Jean-Michel Hufflen ist eine neue Implementierung und Erweiterung von BibTeX mit Schwerpunkt auf der Mehrsprachfähigkeit. Eine erste Version wurde in 2001 veröffentlicht. Der Autor war jedoch der Ansicht, dass der zu jener Zeit verfolgte Ansatz für die typographischen Konventionen einiger Sprachen nicht dienlich sei. Er verfasste einen Fragebogen, um sich ein klareres Bild davon zu verschaffen, welche Probleme und Konventionen im Bereich bibliographischer Daten in verschiedenen europäischen Ländern vorliegen. Daraufhin wurde eine neue Ausarbeitung in Angriff genommen, deren erste Ergebnisse 2003 auf verschiedenen Tagungen zu sehen waren.

Das aktuelle Release (v1.3) verwendet eine Formatierungssprache namens *nbst*, um das Layout festzulegen und Anweisungen zu formatieren. Diese Sprache ist sehr stark an XSLT, die Sprache zum Verändern und Verarbeiten von XML-Dokumenten, angelehnt; sie sind jedoch nicht identisch.

Die Webseite des Projektes hat die Internetadresse <http://lifc.univ-fcomte.fr/~hufflen/texts/mlbibtex/mlbibtex/mlbibtex.html>. Dort sind auch weitere Informationen zu finden.

13.2 Das BibTeX-Datenbankformat

Eine BibTeX-Datenbank ist eine reine Textdatei (ASCII) mit bibliographischen Einträgen, die aus Schlüssel-Wert-Paaren zusammengesetzt sind. Abbildung 12.2 auf Seite 714 zeigt eine typische Datenbank dieser Art. In diesem Abschnitt wird die zulässige Syntax der Einträge näher betrachtet (siehe auch [137]).

Jeder Eintrag in einer BibTeX-Datenbank besteht aus drei Hauptteilen: einer *typ*-Bezeichnung gefolgt von einem *schlüssel* und schließlich den *daten* des Eintrags selbst. Der *typ* gibt die allgemeine Art, den Publikationstyp, des

Eintrags an, also ob es sich um einen Artikel, ein Buch oder eine anderweitige Veröffentlichung handelt. Der *schlüssel* dient als Schnittstelle zu $\LaTeX$. Mithilfe dieser Zeichenfolge im Argument eines `\cite`-Befehls wird der Verweis auf den entsprechenden Eintrag hergestellt. Der *daten*-Teil besteht aus einer Reihe von *Feldern* (je nach *typ*), die eine der beiden Formen haben dürfen, die im folgenden generischen Format und Beispiel zu sehen sind.

```
@type_specifier{schlüssel,    @book{lamport86,
    feld_1 = "wert_1",        author = "Leslie Lamport",
    feld_2 = {wert_2},        title  = "{\LaTeX{}} A Document
    . . .                    Preparation system",
    feld_n = {wert_n}         publisher = {Addison-Wesley},
}                             year    = 1986    }
```

Als Trennzeichen dient das Komma. Leerzeichen vor und nach dem Gleichheitszeichen und dem Komma werden ignoriert. Der Textteil des Feldes, der in Anführungszeichen oder geschweiften Klammern steht, darf jede beliebige Zeichenfolge enthalten, wobei Klammern jedoch immer geschlossen werden müssen. Diese Anführungszeichen oder Klammern dürfen entfallen, wenn eine Zeichenfolge ausschließlich aus Ziffern besteht. Dies gilt z.B. für das Feld *year* (Jahr) im vorigen Beispiel. Man beachte, dass das $\LaTeX$ -Kommentarzeichen % in .bib-Datenbankdateien nicht als Kommentarzeichen gilt. Stattdessen wird alles außerhalb eines Eintrags als Kommentar betrachtet, solange kein @-Zeichen darin enthalten ist. In letzterem Falle interpretiert BibTeX den Text fälschlicherweise als Beginn eines neuen Eintrags.

BibTeX *ignoriert* die Groß-/Kleinschreibung der Zeichen im Publikationstyp, im Schlüssel und in den Feldnamen. Trotzdem erfordert der Schlüssel einige Sorgfalt. $\LaTeX$ unterscheidet die Schlüssel im Argument eines `\cite`-Befehls sehr wohl nach Groß- oder Kleinschreibung, so dass der Schlüssel des jeweiligen Literatureintrags genau mit demjenigen in der $\LaTeX$ -Datei übereinstimmen muss (siehe Abschnitt 12.2.1).

13.2.1 Publikationstypen und Felder

Wie zuvor besprochen, muss man jeden bibliographischen Eintrag einer bestimmten Klasse zuordnen, zu der wiederum bestimmte Felder mit Informationen gehören.

Zunächst muss man entscheiden, welcher Publikationstyp vorliegt. Auch wenn ein festes Klassifizierungsschema niemals vollständig sein kann, ist es doch mit ein wenig Kreativität möglich, selbst die etwas abstruseren Arten von Publikationen mit BibTeX zu verarbeiten. Im Falle nicht standardisierter Publikationstypen sollte man den Warnungen von BibTeX allerdings nicht allzu viel Bedeutung beimessen (siehe unten).

Die meisten BibTeX-Stile verfügen mindestens über die 13 Standardpublikationstypen, die in Tabelle 13.1 auf der nächsten Seite aufgeführt sind. Für die unterschiedlichen Arten von Veröffentlichungen sind verschiedene Arten von Daten erforderlich. Ein Verweis auf einen Zeitschriftenartikel enthält wahrscheinlich den Jahrgang und die Ausgabennummer der Zeitschrift, was für

article	Zeitungs- oder Zeitschriftenartikel. <i>Erforderlich:</i> author, title, journal, year. <i>Optional:</i> volume, number, pages, month, note.
book	Buch mit einem expliziten Verlag. <i>Erforderlich:</i> author oder editor, title, publisher, year. <i>Optional:</i> volume oder number, series, address, edition, month, note.
booklet	Gebundenes Druckwerk ohne namentlich aufgeführten Verleger oder Sponsor. <i>Erforderlich:</i> title. <i>Optional:</i> author, howpublished, address, month, year, note.
inbook	Teil eines Buches, z.B. ein Kapitel, Abschnitt oder ein Seitenbereich. <i>Erforderlich:</i> author oder editor, title, chapter und/oder pages, publisher, year. <i>Optional:</i> volume oder number, series, type, address, edition, month, note.
incollection	Teil eines Buches mit eigenem Titel. <i>Erforderlich:</i> author, title, booktitle, publisher, year. <i>Optional:</i> editor, volume oder number, series, type, chapter, pages, address, edition, month, note.
inproceedings	Artikel in einem Konferenzbericht. <i>Erforderlich:</i> author, title, booktitle, year. <i>Optional:</i> editor, volume oder number, series, pages, address, month, organization, publisher, note.
manual	Technische Dokumentation. <i>Erforderlich:</i> title. <i>Optional:</i> author, organization, address, edition, month, year, note.
mastersthesis	Diplomarbeit. <i>Erforderlich:</i> author, title, school, year. <i>Optional:</i> type, address, month, note.
misc	Freier Eintrag, wenn nichts anderes passt. Sind alle optionalen Felder leer (wenn also der ganze Eintrag leer ist oder nur aus ignorierten Feldern besteht), so wird eine Warnung ausgegeben. <i>Erforderlich:</i> keine. <i>Optional:</i> author, title, howpublished, month, year, note.
phdthesis	Doktorarbeit. <i>Erforderlich:</i> author, title, school, year. <i>Optional:</i> type, address, month, note.
proceedings	Konferenzbericht. <i>Erforderlich:</i> title, year. <i>Optional:</i> editor, volume oder number, series, address, publisher, note, month, organization.
techreport	Von einer Hochschule oder einer anderen Institution veröffentlichter Bericht; normalerweise ein nummerierter Band einer Reihe. <i>Erforderlich:</i> author, title, institution, year. <i>Optional:</i> type, number, address, month, note.
unpublished	Dokument mit Autor und Titel, das nicht formell veröffentlicht wurde. <i>Erforderlich:</i> author, title, note. <i>Optional:</i> month, year.

Tabelle 13.1: Standardpublikationstypen von BibTeX

<code>address</code>	Normalerweise die Anschrift des Verlages (<code>publisher</code>) oder einer entsprechenden Einrichtung; bei großen Verlagshäusern reicht die Angabe der Stadt. Bei weniger bekannten Verlagen kann die vollständige Adresse für den Leser hilfreich sein.
<code>annote</code>	Kommentar; wird nicht von den Standard-Bibliographiestilen verwendet, wohl aber von anderen, die ein Literaturverzeichnis mit Kommentaren erzeugen (wie z.B. <code>annote</code>). Das Feld beginnt einen neuen Satz. Folglich sollte das erste Wort großgeschrieben werden.
<code>author</code>	Autorname(n) im BibTeX-Namensformat (siehe Abschnitt 13.2.2).
<code>booktitle</code>	Haupttitel bei <code>incollection</code> oder <code>inproceedings</code> .
<code>chapter</code>	Nummer des Kapitels (oder Abschnitts oder der entsprechenden Passage).
<code>crossref</code>	Datenbankschlüssel des Eintrags, auf den ein Querverweis erfolgt (Abschnitt 13.2.5).
<code>edition</code>	Auflage eines Buches (z.B. „Zweite“); sie sollte als Zahlwort angegeben werden und, wie im obigen Beispiel, mit einem Großbuchstaben beginnen. Soweit erforderlich, wandeln die Standardstile diesen in einen Kleinbuchstaben um.
<code>editor</code>	Herausgebername(n) im BibTeX-Namensformat; sofern zusätzlich das Feld <code>author</code> vorhanden ist, enthält das Feld <code>editor</code> den Herausgeber des Buches oder der Sammlung, in welcher die Quelle erscheint.
<code>howpublished</code>	Art der Veröffentlichung für ungewöhnliche Werke.
<code>institution</code>	Institution, die einen technischen Bericht finanziert.
<code>journal</code>	Name der Zeitung oder Zeitschrift; für viele Zeitungen bzw. Zeitschriften existieren Abkürzungen (Abschnitt 13.2.3).

Tabelle 13.2: Standardeingabefelder in BibTeX

ein Buch normalerweise nicht von Bedeutung ist. Daher verfügen unterschiedliche Datenbankarten über unterschiedliche Felder. Die Felder jedes Eintrags sind in drei Klassen unterteilt:

Erforderlich Das Fehlen eines solchen Feldes erzeugt eine Warnung und führt möglicherweise zu einem schlecht formatierten Eintrag im Literaturverzeichnis. Wenn ein erforderliches Feld keinen Sinn ergibt, liegt vermutlich der falsche Publikationstyp vor. Sind die geforderten Daten sinnvoll, aber z.B. schon in einem anderen Feld enthalten, so kann man die Warnung in den meisten Fällen einfach ignorieren.

Optional Die Daten dieses Feldes werden verwendet, soweit sie vorhanden sind. Sie können jedoch auch weggelassen werden, ohne dass Formatierungsprobleme auftreten. Optionale Felder sollen dem Leser ergänzende Informationen bieten.

Ignoriert Das Feld wird ignoriert. BibTeX ignoriert jedes Feld, das nicht erforderlich oder optional ist, so dass man `.bib`-Dateieinträge um beliebige Felder erweitern kann. Es ist sinnvoll, alle für eine Quelle relevanten Informationen in den jeweiligen Eintrag in der `.bib`-Datei aufzunehmen, selbst wenn einige Teile davon niemals im Literaturverzeichnis erscheinen werden. So kann man z.B. die Zusammenfassung eines Berichtes im Feld `abstract` des `.bib`-Dateieintrags speichern. Die `.bib`-Datei eignet sich wahrscheinlich ebenso gut für das Ablegen einer Zusammenfassung wie jeder andere Ort. Es gibt sogar BibTeX-Stile, die ausgewählte Zusammenfassungen drucken (wie etwa der BibTeX-Stil `abstract` aus Tabelle 13.4 auf Seite 821).

key	Wird für die alphabetische Sortierung genutzt sowie zum Erstellen eines Labels, wenn die Angaben zu Verfasser (author) und Herausgeber (editor) fehlen. Dieses Feld sollte nicht mit dem Schlüssel verwechselt werden, der im <code>\cite</code> -Befehl und am Anfang jedes Datenbankeintrags erscheint.
month	Monat, in dem das Werk veröffentlicht wurde oder in dem ein nicht veröffentlichtes Werk geschrieben wurde. Zugunsten der Einheitlichkeit sollte man immer die dreibuchstabigen Standardabkürzungen (wie <code>jan</code> , <code>feb</code> , <code>mar</code> usw.) verwenden (Abschnitt 13.2.3).
note	Jegliche Zusatzinformationen, die für den Leser nützlich sein können.
number	Nummer einer Zeitung oder Zeitschrift, eines Fachberichtes oder eines Werkes aus einer Reihe; die Ausgabe einer Zeitung oder Zeitschrift wird normalerweise durch den Jahrgang und die Nummer gekennzeichnet. Ein Fachbericht hat normalerweise eine Nummer, ebenso wie gelegentlich auch Bücher einer Reihe mit eigenem Titel.
organization	Organisation, die eine Konferenz finanziert oder ein Handbuch (<code>manual</code>) herausgibt.
pages	Eine oder mehrere Seiten oder Seitenbereiche, wie z.B. 42-111 oder 7, 41, 73-97 oder 43+; das Zeichen '+' weist hier darauf hin, dass die Seitenzahlen keinen einfachen Bereich bilden.
publisher	Name des Verlages.
school	Name der Hochschule, an der eine Abhandlung geschrieben wurde.
series	Name einer Buchreihe oder eines Satzes zusammengehöriger Bücher; bei einem Bucheintrag gibt <code>title</code> den Buchtitel an und das optionale Feld <code>series</code> den Namen der Reihe oder den Namen des mehrbändigen Satzes, in dem das Buch erschienen ist.
title	Der Titel des Werkes; die Schreibweise wird in Abschnitt 13.2.2 erläutert.
type	Die Art eines Fachberichtes (z.B. „Forschungsbericht“); der Name wird dann anstelle des vorgegebenen Textes „Technical Report“ verwendet. Für den Publikationstyp <code>phdthesis</code> könnte man folgendermaßen den Begriff „Doktorarbeit“ einsetzen: <code>type = "Doktorarbeit"</code> . Für die Publikationstypen <code>inbook</code> und <code>incollection</code> kann man ebenso den vom <code>chapter</code> -Feld erzeugten Text beeinflussen, etwa durch <code>type = Abschnitt</code> .
volume	Der Jahrgang einer Zeitung/Zeitschrift oder die Bandnummer einer mehrbändigen Ausgabe.
year	Erscheinungsjahr oder, bei nicht veröffentlichten Werken, das Entstehungsjahr; es sollte in der Regel aus vier Ziffern bestehen, wie etwa 2005, wobei die Standardstile auch <code>year</code> -Angaben verarbeiten können, deren letzte vier Zeichen (ohne Satzzeichen) Ziffern sind, wie z.B. „um 1600“.

Tabelle 13.2: Standardeingabefelder in BibTeX (Forts.)

Tabelle 13.1 auf Seite 791 beschreibt die Standardpublikationstypen mit ihren erforderlichen und optionalen Feldern, wie sie von den Standard-Bibliographiestilen verwendet werden. Die Felder jedes Typs (erforderliche und optionale) sind in der Reihenfolge aufgelistet, in der sie normalerweise in einem Eintrag erscheinen. Einige Publikationstypen können jedoch etwas von dieser Reihenfolge abweichen, je nachdem welche Felder fehlen. Die Bedeutung der einzelnen Felder wird in Tabelle 13.2 auf dieser Doppelseite erklärt. Vom Standard abweichende Bibliographiestile ignorieren möglicherweise einige der optionalen Felder, wenn die ein Literaturverzeichnis erzeugen. Oder Sie verwenden zusätzliche, wie etwa `isbn` (vergleiche die Beispiele ab Seite 823).

Man beachte: Die in einer .bib-Datei verwendeten Bezeichnungen der Publikationstypen werden durch das @-Zeichen eingeleitet.

Die meisten BibTeX-Stildateien sortieren die bibliographischen Einträge. Dazu erzeugen sie intern einen Sortierschlüssel aus dem Namen des Verfassers oder Herausgebers, dem Erscheinungsdatum, dem Titel und weiteren Daten. Einträge mit gleichem Sortierschlüssel erscheinen in der Reihenfolge, in der sie im Text zitiert werden.

Der Verfasser wird normalerweise im Feld `author` angegeben, einige Formate verwenden jedoch die Felder `editor` oder `organization`. Außer den in Tabelle 13.1 aufgeführten Feldern verfügt jeder Eintrag zusätzlich über ein optionales Schlüsselfeld `key`, das manche Stile für die alphabetische Sortierung, für Querverweise oder zum Erstellen eines \bibitem-Labels heranziehen. Daher sollte jeder Eintrag ohne Autorenangabe ein solches `key`-Feld enthalten. Manche Stile ersetzen auch den automatisch erzeugten, internen Sortierschlüssel durch das Feld `key`.¹ In folgenden Fällen ist das Feld `key` nützlich:

```
organization = "The Association for Computing Machinery",
key = "ACM"
```

Ohne das Feld `key` erzeugt der BibTeX-Stil `alpha` das Label aus den ersten drei Buchstaben des Feldes `organization`. Auch wenn dabei der Artikel „The“ unberücksichtigt bleibt, entsteht in diesem Falle noch immer ein eher nichts sagendes Label, wie „[Ass86]“. Das `key`-Feld liefert dagegen die aussagekräftigere Angabe „[ACM86]“.

Als Nächstes folgt eine nähere Betrachtung der Felder, die den BibTeX-Standardstilen bekannt sind. Diese „Standardfelder“ sind in Tabelle 13.2 auf Seite 792 aufgeführt. Verwendet man einen der erweiterten und vom Standard abweichenden Stile aus Tabelle 13.4 auf Seite 821, so können weitere Felder wie etwa `abstract` erforderlich sein. Da unbekannte Felder von den BibTeX-Stilen einfach ignoriert werden, kann man sie nutzen, um einem Eintrag „Anmerkungen“ hinzuzufügen. Dazu reicht es aus, die zu ignorierenden Daten in geschweifte Klammern zu setzen, die einem Feldnamen (mit =-Zeichen) folgen, der von dem jeweiligen BibTeX-Stil nicht erkannt wird.

Genau wie die Namen der Publikationstypen in Tabelle 13.1 auf Seite 791 sollten auch die Feldnamen in ihrem weitesten Sinne interpretiert werden, damit sie sich für möglichst viele Zwecke verwenden lassen. Man sollte zudem nicht vergessen, dass ein geschickter Umgang mit dem Feld `note` selbst kompliziertere Fälle lösen kann.

13.2.2 Der Textteil der Felder

Der Textteil eines Feldes innerhalb eines BibTeX-Eintrags steht in Anführungszeichen oder geschweiften Klammern. Teile eines solchen Textes können darüber hinaus noch einmal selbst in geschweiften Klammern stehen und werden dann als *von Klammern umgeben* bezeichnet.

¹Einige BibTeX-Stile (z.B. `jurabib`) nutzen stattdessen das Feld `sortkey`.

Aufbau des Namens

Die Felder `author` und `editor` enthalten jeweils eine Namensliste. Wie diese Namen später genau gesetzt werden, entscheidet der jeweilige Bibliographiestil. Der Eintrag in der `.bib`-Datenbank teilt BibTeX den Namen mit. Die Namen sollten immer genau so eingegeben werden, wie sie in dem zitierten Werk erscheinen. Das gilt auch, wenn sie in zwei verschiedenen Werken etwas unterschiedlich geschrieben werden, wie z.B.:

```
author = "Donald E. Knuth"          author = "D. E. Knuth"
```

Ist man sicher, dass sich beide Autorenangaben auf dieselbe Person beziehen, so kann man für beide die Form wählen, die der Verfasser bevorzugt (wie etwa Donald E. Knuth). Dabei sollte man jedoch (wie im nachfolgenden Beispiel) stets anmerken, dass in der Originalveröffentlichung eine andere Schreibweise verwendet wurde.

```
author = "D[onald] E. Knuth"
```

BibTeX sortiert diesen Eintrag alphabetisch so ein, als seien die Klammern nicht vorhanden, so dass die Identität des Autors eindeutig zu erkennen ist.

Die meisten Namen können auf eine der beiden folgenden, gleichwertigen Weisen eingegeben werden:

```
"John Chris Smith"          "Smith, John Chris"
"Thomas von Neumann"        "von Neumann, Thomas"
```

Die zweite Form, mit Komma, sollte nur bei Personen verwendet werden, die mehrere großgeschriebene Nachnamen haben, wie z.B.:

```
"Parra Benavides, Miguel"
```

Gibt man "Miguel Parra Benavides" ein, so geht BibTeX davon aus, dass "Parra" der zweite Vorname ist, was in diesem Falle jedoch nicht zutrifft. Werden die anderen Namensteile kleingeschrieben, so entsteht dieses Problem gar nicht erst (z.B. "Johann von Bergen" oder "Pierre de la Porte").

Mehrere zusammengehörige Wörter in einem Namen sollten von Klammern umgeben werden. Wie im nächsten Beispiel zu sehen, behandelt BibTeX den Inhalt der Klammern jeweils als einzelnen Namen.

```
"{Boss and Friends, Inc.}"
```

In diesem Fall wird `Inc.` nicht fälschlicherweise für den Vornamen gehalten.

Generell können BibTeX-Namen aus vier verschiedenen Teilen bestehen, die als `First`, `von`, `Last` und `Jr` bezeichnet werden. Jeder Teil besteht aus einer Liste von Tokens (zusammengehörigen Zeichen) und bis auf `Last` darf jede dieser Listen leer sein. Entsprechend sind die beiden folgenden Einträge unterschiedlich:

```
"von der Schmidt, Alex"          "{von der Schmidt}, Alex"
```

Der erste verfügt über die Teile `von`, `Last` und `First`, während der zweite nur aus den Teilen `First` und `Last` (`von der Schmidt`) besteht. Dadurch kann sich eine unterschiedliche Sortierreihenfolge ergeben.

Der „Junior“-Teil kann zu einem ganz eigenen Problem werden. Die meisten Personen, die ein „Jr.“ im Namen führen, stellen diesem ein Komma voran,

so dass es folgendermaßen eingegeben wird:

```
"Smith, Jr., Robert"
```

Manche Menschen verwenden das Komma jedoch nicht und in diesem Falle wird das „Jr.“ als Bestandteil des Nachnamens betrachtet:

```
"{Lincoln Jr.}, John P."           "John P. {Lincoln Jr.}"
```

Zur Erinnerung, „Miguel Parra Benavides“ sollte folgendermaßen eingegeben werden:

```
"Parra Benavides, Miguel"
```

Der Teil `First` seines Namens besteht aus dem einzelnen Token „Miguel“, der Teil `Last` aus den zwei Tokens „Parra“ und „Benavides“, während die Teile `von` und `Jr` leer sind.

Ein komplexes Beispiel:

```
"Johannes Martinus Albertus van de Groene Heide"
```

Dieser Name besteht aus drei Tokens im `First`-Teil, zwei im `von`-Teil und zwei im `Last`-Teil. `BBTEX` kann erkennen, wo ein Teil aufhört und der andere beginnt, da die Tokens im `von`-Teil kleingeschrieben sind (van de im vorigen Beispiel).

Genauer gesagt, wird der `von`-Teil daran erkannt, dass der erste Buchstabe auf Klammerebene 0 kleingeschrieben ist. Da dabei Klammern von „Spezialzeichen“ (siehe Seite 797) ignoriert werden, kann man durch Einfügen eines Dummy-Spezialzeichens bestimmen, wie `BBTEX` ein Token behandelt. Dazu muss der erste Buchstabe nach dem `TEX`-Befehl in der gewünschten Schreibweise, d.h. Klein- oder Großbuchstaben, erscheinen. In dem Beispiel

```
Maria {\MakeUppercase{d}e La} Cruz
```

interpretiert `BBTEX` das großgeschriebene „De La“ als `von`-Teil, da der erste Buchstabe nach dem Befehl kleingeschrieben ist. Das `BBTEX`-Format erzeugt aus diesen Daten die korrekte Abkürzung M. De La Cruz anstelle der falschen Version M. D. L. Cruz, die erscheint, wenn man diesen Trick nicht anwendet.

`BBTEX` verarbeitet auch Namen mit Bindestrichen fehlerfrei. Ein Eintrag wie der folgende:

```
author = "Maria-Victoria Delgrande",
```

wird durch das `abbrv`-Format zu „M.-V. Delgrande“.

Sind mehrere Autoren angegeben, so sollten die Namen durch das Wort „and“ getrennt sein, das nicht in Klammern stehen darf.

```
author = "Frank Mittelbach and Rowley, Chris"
editor = "{Lion and Noble, Ltd.}"
```

Im vorigen Beispiel werden zwei Autoren genannt, Frank Mittelbach und Chris Rowley, aber nur ein Herausgeber, da das „and“ von Klammern umgeben ist. Ist die Anzahl von Autoren zu lang, um sie *in extenso* anzuführen, so kann man die Namensliste mit der Zeichenfolge „and others“ beenden, die von den Standardstilen in die gebräuchliche Form „et al.“ umgesetzt wird.

Man kann Namen in BibTeX also auf insgesamt drei mögliche Arten angeben (wobei Anführungszeichen und geschweifte Klammern immer erlaubt sind):

"First von Last"	z.B. {Johan van der Winden}
"von Last, First"	z.B. "von der Schmidt, Alexander"
"von Last, Jr, First"	z.B. {de la Porte, Fils, {\`Emile}}

Die erste Form passt fast immer. Sie eignet sich jedoch nicht für Namen mit einem Jr-Teil oder mit einem Last-Teil aus mehreren Tokens und ohne von-Teil.

Das Titelformat

Der Bibliographiestil bestimmt, ob ein Titel großgeschrieben wird. Normalerweise werden Buchtitel in der englischen Sprache großgeschrieben, Artikelüberschriften jedoch klein. Der Titel sollte immer originalgetreu erfasst werden, z.B.:

```
TITLE = "A Manual of Style"
TITLE = "Hyphenation patterns for ancient Greek and Latin"
```

Verschiedene Sprachen und Stile haben ihre eigenen Regeln zur Großschreibung. Um einen Titel anders anzugeben, als es der Bibliographiestil verlangt, muss man die Teile, die nicht verändert werden sollen, in Klammern setzen. Man beachte, dass diese Methode nicht ausreicht, wenn das erste Zeichen nach der linken Klammer ein Backslash ist (wie im nächsten Abschnitt erläutert). Am besten setzt man Wörter immer als Ganzes in Klammern, da L^AT_EX ansonsten beim Setzen des betreffenden Wortes Unterscheidungen oder Ligaturen verloren gehen könnten. Im nächsten Beispiel ist die erste Version der zweiten vorzuziehen:

```
TITLE = "The Towns and Villages of {Belgium}"
TITLE = "The Towns and Villages of {B}elgium"
```

Sonderzeichen und Zeichen mit Akzenten

BibTeX akzeptiert Akzentbuchstaben. Bei einem Eintrag mit zwei Feldern, wie etwa

```
author = "Kurt G{\`o}del",
year = 1931,
```

gibt der Bibliographiestil alpha das Label [Göd31] aus, das wahrscheinlich dem gewünschten Resultat entspricht. Wie das vorige Beispiel zeigt, muss der gesamte Umlaut in Klammern stehen. In diesem Falle sind die Versionen {\`o} und {\`{o}} möglich. Die Klammern dürfen selbst nicht noch einmal in Klammern stehen (außer denjenigen, die das gesamte Feld einschließen). Zudem muss das erste Zeichen innerhalb der Klammern ein Backslash sein. Entsprechend würden die Schreibweisen {G{\`{o}}del} und {G\`{o}del} hier nicht funktionieren.

Diese Funktion dient der Verarbeitung von Akzentbuchstaben und sprachspezifischen Zeichen in $\text{\LaTeX}$. Sie ermöglicht auch benutzerdefinierte „Akzente“. Beim Zählen der Zeichen in einem Label betrachtet $\text{\BibTeX}$ den Klammerausdruck als einen einzelnen Buchstaben. Für $\text{\BibTeX}$ stellt ein Akzentbuchstabe ein spezielles „Sonderzeichen“ dar, das auf der linken Seite auf oberster Ebene mit einer öffnenden geschweiften Klammer beginnt, der unmittelbar ein Backslash folgt, und mit der zugehörigen schließenden geschweiften Klammer endet. Beispielsweise enthält das Feld

```
author = "\OE{le} {\' {E}mile} {Ren\'{e}} van R{\i{j}}den"
```

zwei Sonderzeichen: „ $\backslash' \{E\}mile\}$ “ und „ $\backslash i\{j\}$ “.

Im Allgemeinen verarbeitet $\text{\BibTeX}$ keine $\text{\TeX}$ - oder $\text{\LaTeX}$ -Befehle innerhalb eines Sonderzeichens, wohl aber andere Zeichen. Daher konvertiert ein Stil, der alle Titel in Kleinbuchstaben setzt, den Titel

“The $\backslash \text{\TeX} \text{ BOOK\NOOP}$ Saga” in “The $\backslash \text{\TeX} \text{ book\NOOP}$ saga”

Der Artikel „The“ bleibt großgeschrieben, da es sich um das erste Wort des Titels handelt.

Dieses Sonderzeichensystem hat beim Umgang mit Akzentbuchstaben durchaus seine Vorteile, auch wenn die zusätzlichen Klammern Ligaturen und Unterschneidungen verhindern können. Mithilfe dieses Systems kann man $\text{\BibTeX}$ dazu bringen, die Einträge nach den eigenen Vorstellungen zu sortieren. Allerdings gilt es auch hier einige Dinge zu beachten (vgl. die Erläuterungen zum Befehl $\backslash \text{SortNoop}$ auf Seite 800). Da $\text{\BibTeX}$ ein Sonderzeichen als insgesamt ein Zeichen betrachtet, kann man durch dieses Verfahren auch zusätzliche Zeichen in Labels erzwingen.

13.2.3 Kurzformen in $\text{\BibTeX}$

$\text{\BibTeX}$ -Textfelder lassen sich auch abkürzen. Eine Kurzform ist eine Zeichenfolge von ASCII-Zeichen, die mit einem Buchstaben beginnt und weder ein Leerzeichen noch eines der folgenden 10 Zeichen enthält:

" # % ' () , = { }

Mithilfe des Befehls @string kann man in einer .bib -Datei folgendermaßen eigene Kurzformen definieren:

```
@string{AW          = "Addison--Wesley Publishing Company"}
@STRING{cacm        = "Communications of the ACM"}
@String{pub-AW      = {{Ad-di-son-Wes-ley}}}
@String{pub-AW:adr  = "Reading, MA, USA"}
@String{TUG         = "\TeX{} Users Group"}
@String{TUG:adr     = {Providence, RI, USA}}
```

Diese Kurzformen lassen sich im Textteil der $\text{\BibTeX}$ -Felder verwenden, wobei sie jedoch nicht in Klammern oder Anführungszeichen stehen dürfen.

Geht man von den zuvor definierten Kurzformen aus, so bedeuten die beiden folgenden Angaben im Feld `journal` das gleiche:

```
journal = "Communications of the ACM"
journal = cacm
```

Bei Abkürzungen ist die Groß- oder Kleinschreibung des Namens unerheblich. CACM und cacm werden daher als identische Einträge betrachtet. BibTeX gibt jedoch eine Warnung aus, wenn Groß- und Kleinschreibung miteinander vermischt werden. Auch der `@string`-Befehl selbst kann in Kleinbuchstaben, Großbuchstaben oder gemischt geschrieben werden.

Solche `@string`-Befehle lassen sich überall in einer `.bib`-Datei verwenden, eine Kurzform muss jedoch zunächst definiert sein, bevor man sie benutzen kann. Es empfiehlt sich, alle `@string`-Befehle am Anfang einer `.bib`-Datei zusammenzufassen oder sie in eine gesonderte `.bib`-Datei zu schreiben, die ausschließlich Abkürzungen enthält. Die in der `.bib`-Datei definierten `@string`-Befehle haben Vorrang vor den Definitionen in einer Stildatei.

Mithilfe des Verkettungszeichens `#` lassen sich mehrere Zeichenfolgen (oder mehrere `@string`-Definitionen) aneinander koppeln. Ausgehend von der folgenden Definition

```
@STRING{TUB = {TUGboat }}
```

kann man leicht für unterschiedliche Einträge nahezu identische `journal`-Felder zusammenstellen:

```
@article(tub-98,   journal = TUB # 1998,   ...
@article(tub-99,   journal = TUB # 1999,   ...
@article(tub-00,   journal = TUB # 2000,   ...
```

Die meisten Bibliographiestile verfügen bereits über einige vordefinierte Kurzformen. Üblicherweise sollte es immer dreibuchstabile Abkürzungen für die Monate geben: `jan`, `feb`, `mar` und so weiter. In den BibTeX-Datenbankdateien sollte man immer diese Kurzformen für die Monate verwenden, anstatt sie auszuschreiben. Dadurch ist eine einheitliche Schreibweise innerhalb des Literaturverzeichnisses gewährleistet. Das Tagesdatum ist normalerweise am besten im Feld `month` untergebracht. Hier kann man z.B. sinnvoll eine Verkettung durchführen:

```
month = apr # "~1,"
```

Die meisten BibTeX-Stile enthalten auch bereits Kurzformen für die bekanntesten Zeitschriften des jeweiligen Fachgebietes. Diese lassen sich der Dokumentation zu dem entsprechenden Bibliographiestil entnehmen. Die in Tabelle 13.3 auf der nächsten Seite aufgeführten Zeitschriften sollten in allen Stilen vorhanden sein. Man kann auch sehr einfach eigene Kurzformen für Zeitschriften anlegen, indem man sie in einer eigenen Datenbankdatei mit `@string`-Befehlen definiert und diese Datei dem LaTeX-Befehl `\bibliography` als Argument übergibt.

acmcs	ACM Computing Surveys	jcss	Journal of Computer and System Sciences
acta	Acta Informatica	scp	Science of Computer Programming
cacm	Communications of the ACM	sicomp	SIAM Journal on Computing
ibmjrd	IBM Journal of Research and Development	tocs	ACM Transactions on Computer Systems
ibmsj	IBM Systems Journal	tods	ACM Transactions on Database Systems
ieeese	IEEE Transactions on Software Engineering	tog	ACM Transactions on Graphics
ieeetc	IEEE Transactions on Computers	toms	ACM Transactions on Mathematical Software
ieeetcad	IEEE Transactions on Computer-Aided Design of Integrated Circuits	toois	ACM Transactions on Office Information Systems
ipl	Information Processing Letters	toplas	ACM Transactions on Programming Languages and Systems
jacm	Journal of the ACM	tcs	Theoretical Computer Science

Tabelle 13.3: Vordefinierte Zeitschriftenkürzel in $\text{\LaTeX}$ -Stilen

13.2.4 Die $\text{\LaTeX}$ -Präambel

$\text{\LaTeX}$ verfügt über den Befehl `@preamble`, dessen Syntax der des `@string`-Befehls ähnelt. Er verwendet jedoch kein Gleichheitszeichen, sondern lediglich die Zeichenfolge selbst, wie z.B.:

```
@preamble{ "\providecommand\url[1]{\texttt{#1}}" #
           "\providecommand\SortNoop[1]{}" }
```

Wie das Beispiel zeigt, werden die verschiedenen Befehlsdefinitionen innerhalb des `@preamble`-Befehls durch das `#`-Zeichen miteinander verknüpft. Die Standardstile reichen das Argument des Befehls `@preamble` eins zu eins an die `.bbl`-Datei durch, so dass die Befehlsdefinitionen zur Verfügung stehen, wenn $\text{\LaTeX}$ die Datei liest. Auf diese Weise hinzugefügte $\text{\LaTeX}$ -Befehle müssen mit `\providecommand` anstatt mit `\newcommand` definiert werden. Dafür gibt es zwei Gründe: Zum einen beraubt man sich sonst der Möglichkeit, die Definition im Dokument zu ändern. So könnte die Bibliographie z.B. eine einfache Befehlsdefinition für den Befehl `\url` hinzufügen, die man durch die entsprechende Definition des `url`-Paketes ersetzen möchte. Zum anderen wird die Bibliographie unter Umständen mehrmals gelesen, z.B. wenn man das Paket `chapterbib` verwendet. Dieser Vorgang schlägt fehl, sofern man `\newcommand` benutzt.

Der zweite im letzten Beispiel verwendete Befehl, `\SortNoop`, wurde von Oren Patashnik vorgeschlagen, um $\text{\LaTeX}$ s Sortieralgorithmus in schwierigen Situationen zu steuern. Dieser Algorithmus funktioniert normalerweise ganz ordentlich. Manchmal kann es jedoch sein, dass man die Entscheidungen von $\text{\LaTeX}$ durch einen eigenen Sortierschlüssel übergehen will. Dieser Trick eignet sich insbesondere für Sprachen, die eine von der englischen abweichende Sortierreihenfolge aufweisen, oder wenn man die verschiedenen Bände eines Buches nach dem Datum der Erstveröffentlichung sortieren möchte, unabhängig von dem Datum ihrer Neuerscheinung.

Angenommen, der erste Band eines Buches wurde ursprünglich 1986 veröffentlicht, eine zweite Ausgabe davon erschien 1991, und der zweite Band kam 1990 heraus. In diesem Fall könnte man schreiben:

```
@book{ ... volume=1, year = "{\SortNoop{86}}1991" ...
@book{ ... volume=2, year = "{\SortNoop{90}}1990" ...
```

Entsprechend der Definition des `\SortNoop`-Befehls ignoriert L^AT_EX dessen Argument und druckt nur das in den Feldern angegebene tatsächliche Erscheinungsjahr. BibTeX nimmt den Befehl `\SortNoop` als „Akzent“ wahr, so dass es die Werke nach den Nummern 861991 und 901990 sortiert, und so wie gewünscht Band 1 vor Band 2 platziert.

Man beachte jedoch, dass dieser Trick bei neueren BibTeX-Stilen möglicherweise nicht mehr funktioniert (z.B. bei mit dem `custom-bib`-Paket erzeugten) und dass einige Stile ein eigenes `sortkey`-Feld hinzufügen, das solche Probleme viel eleganter löst.

13.2.5 Querverweise zwischen Einträgen

BibTeX-Einträge können aufeinander verweisen. Angenommen, man gibt im Dokument `\cite{Wood:color}` an und die beiden folgenden Einträge stehen in der Datenbankdatei:

```
@Inbook{Wood:color, author = {Pat Wood}, crossref={Roth:postscript},
title = {PostScript Color Separation}, pages={201--225}}
@Book{Roth:postscript, editor = {Stephen E. Roth}, title =
{{Real World PostScript}}, booktitle = {{Real World PostScript}},
publisher=AW, address=AW:adr, year=1988, ISBN={0-201-06663-7}}
```

Das besondere Feld `crossref` informiert BibTeX, dass der Eintrag `Wood:color` fehlende Felder aus dem Eintrag übernimmt, auf den es verweist: aus `Roth:postscript`. BibTeX fügt die Quelle `Roth:postscript` automatisch in das Literaturverzeichnis ein, wenn eine bestimmte Anzahl von Einträgen (voreingestellt sind 2), die in einem `\cite`- oder `\nocite`-Befehl vorkommen, auf diese verweisen. Das gilt auch dann, wenn `Roth:postscript` selbst in keinem Argument eines solchen Befehls steht. Auf diese Weise erscheint `Roth:postscript` automatisch im Literaturverzeichnis, sobald neben `Wood:color` noch ein weiterer Eintrag einen Querverweis auf diese Quelle enthält.

Der Vorgabewert ist im BibTeX-Programm eingebaut, aber bei moderneren Installationen¹ lässt sich der gewünschte Wert über die Befehlszeile mit `--min-crossrefs` festlegen:

```
bibtex --min-crossrefs=1 12-5-41
```

Beispielsweise wurde die Bibliographie aus Beispiel 12-5-41 von Seite 764 mit diesen Einstellungen erzeugt, damit der entsprechende Konferenzbericht als eigene Quelle aufgeführt wird, obwohl nur ein Querverweis darauf existiert.

¹In BibTeX8 heißt diese Option `-min_crossrefs` oder `-M`.

Will man andererseits vermeiden, dass der Konferenzbericht als eigene Quelle erscheint, ganz gleich wie viele Einträge darauf verweisen, so kann man die Option `--min-crossrefs` einfach auf eine genügend große Zahl setzen (z.B. 500).

Ein Eintrag, auf den Querverweise erfolgen, muss in der Datenbankdatei nach den Einträgen erscheinen, die auf ihn verweisen. Es wäre also günstig, alle Ziele von Querverweisen ans Ende der Datenbank zu setzen. Einträge, auf die ein Querverweis erfolgt, können zudem selbst keine weiteren Querverweise enthalten.

Auch der $\LaTeX$ -Befehl `\cite` lässt sich innerhalb der Felder der $\BibTeX$ -Einträge verwenden. Dies kann sinnvoll sein, wenn man etwa im Hinweisfeld `note` auf eine andere Quelle verweisen möchte:

```
note = "N\"ahere Einzelheiten in Eijkhout~\cite{Eijkhout:1991}"
```

In diesem Falle können jedoch zusätzliche $\LaTeX$ - und $\BibTeX$ -Läufe erforderlich sein, um das Dokument vollständig zu bearbeiten. Dies ist dann der Fall, wenn sich der Verweis, den $\BibTeX$ in die `.bbl`-Datei überträgt, auf den Schlüssel eines Eintrags bezieht, der nicht im Hauptdokument zitiert wurde. Dadurch kann $\LaTeX$ den Bezug im folgenden Lauf nicht auflösen und benötigt einen zusätzlichen $\BibTeX$ - und zwei weitere $\LaTeX$ -Läufe.

13.3 Online-Bibliographien

Wer im Internet sucht, findet zahlreiche Bibliographieeinträge, sowohl für Primär- als auch für Sekundärliteratur, in kostenlosen sowie in kommerziellen Datenbanken. In diesem Abschnitt werden einige kostenlose Datenquellen mit wissenschaftlichen Veröffentlichungen genannt, die bibliographische Daten im $\BibTeX$ - und einigen weiteren Formaten anbieten.

Nelson Beebe verwaltet fast 400 $\BibTeX$ -Datenbanken zu Fachzeitschriften und einzelnen wissenschaftlichen Themen.¹ Die Themen reichen von „Acta Informatica“ und „Ada User Journal“ bis zu „X Journal“ und „X Resource [journal]“. Sie sind alle als `.bib`-Quelldatei, `.html`, `.pdf` und `.ps`-Listings verfügbar.

Die für den $\TeX$ -Bereich wohl interessantesten `.bib`-Datenbanken dieser Sammlung sind die Dateien `texbook2.bib` und `texbook3.bib` (Bücher über $\TeX$, METAFONT und ähnliche Programme), `type.bib` (eine Liste von Artikeln und Büchern über Typographie), `gut.bib` (Inhaltsangaben für die französische Zeitschrift *Cahiers Gutenberg*), `komoedie.bib` (Inhaltsangaben für die deutsche Zeitschrift *Die TeXnische Komödie*), `texgraph.bib` (Quellen, die erläutern, wie man Graphiken mit $\TeX$ verarbeiten kann), `texjourn.bib` (eine Liste von Zeitschriften, die Eingaben im $\TeX$ -Format akzeptieren), `tugboat.bib` (alle in *TUGboat* erschienenen Artikel) und `standard.bib` (Softwarestandards). Die von Nelson Beebe angebotenen Webressourcen

¹Die Bibliographiedatenbanken und die Hilfsprogramme zu ihrer Verwaltung und Bearbeitung finden sich unter der Internetadresse <http://www.math.utah.edu:8080/pub/tex/bib/index-table.html>.

umfassen auch eine Reihe von $\text{\LaTeX}$ -Stilen und viele Kommandozeilenprogramme zum Bearbeiten bibliographischer Daten (wie in Abschnitt 13.4.3 erläutert).

Die Collection of Computer Science Bibliographies (Sammlung von Bibliographien der Computerwissenschaften) von Alf-Christian Achilles enthält mehr als 1,2 Millionen Quellen. Sie ist unter der Internetadresse <http://liinwww.ira.uka.de/bibliography/index.html> sowie auf verschiedenen Spiegelservern zu finden. Die in ihr enthaltenen Daten stammen aus externen Bibliographiesammlungen wie denjenigen, die Nelson Beebe zusammengestellt hat. Einen zusätzlichen Nutzen bietet die Suchfunktion, mit deren Hilfe man nach Autoren, bestimmten Fachgebieten, Themen und anderen Kategorien recherchieren kann. Fast alle Literaturdaten sind im $\text{\LaTeX}$ -Format verfügbar.

Eine weitere interessante Ressource ist CiteSeer, Scientific Literature Digital Library, eine wissenschaftliche Literaturdatenbank, die von Steve Lawrence entwickelt wurde. Sie ist unter der Internetadresse <http://citeseer.nj.nec.com> zu finden. Zum Funktionsumfang gehören so hilfreiche Elemente wie erweiterte Suchfunktionen, Kontextinformationen zu Veröffentlichungen (z.B. themenverwandte Werke), Daten darüber, in welchen anderen Publikationen ein Werk zitiert wurde, statistische Daten über Verweise auf eine Quelle und vieles mehr.

Dies ist nur eine kleine Auswahl der riesigen Menge an Material, die man im Internet findet. Sie können sich als hilfreich erweisen, wenn man sich für wissenschaftliche Arbeiten in den Bereichen Mathematik, Computerwissenschaften und ähnlichen Fachgebieten interessiert.

13.4 Werkzeuge für Bibliographiedatenbanken

Da es sich bei $\text{\LaTeX}$ -Datenbanken um einfache Textdateien handelt, können sie mit jedem Editor be- und verarbeitet werden, der ASCII-Dateien erzeugen kann. Gerade bei großen Sammlungen von $\text{\LaTeX}$ -Einträgen kann sich dieses Verfahren jedoch recht mühsam gestalten und es wird immer schwieriger, bestimmte Informationen zu finden. Aus diesem Grunde begann man damit, spezielle Werkzeuge für diese Aufgaben zu entwickeln. Viele dieser Werkzeuge sind unter der Internetadresse <http://www.tug.org/tex-archive/biblio/bibtex/utils/> zu finden.

In diesem Abschnitt wird eine Auswahl von ihnen beschrieben. Ihre Bandbreite reicht von spezialisierten Kommandozeilenprogrammen bis hin zu Anwendungen mit graphischer Benutzeroberfläche zur Verwaltung von Datenbanken. Die Zahl der Produkte beider Arten wächst ständig, so dass sich eine Überprüfung der verfügbaren Internet-Ressourcen sicher lohnt (z.B. unter <http://bibliographic.openoffice.org/biblio-sw.html>).

13.4.1 biblist – $\text{\LaTeX}$ -Datenbankdateien drucken

Eine sortierte Liste aller Einträge einer $\text{\LaTeX}$ -Datenbank kann das Erstellen von Quellenverweisen oftmals erleichtern. Für diesen Zweck gibt es verschiedene Werkzeuge mit mehr oder weniger gleichem Funktionsumfang, so dass

die Auswahl des einen oder anderen hauptsächlich eine Frage des persönlichen Geschmacks ist. Dieser Abschnitt beschäftigt sich stellvertretend mit einem dieser Werkzeuge, dem biblist-Paket von Joachim Schrod. Das Paket kann eine formatierte Liste der Inhalte auch großer BibTeX-Datenbanken erzeugen. In späteren Abschnitten werden noch weitere Möglichkeiten aufgezeigt.

Zur Verwendung des biblist-Paketes muss man zunächst ein L^AT_EX-Dokument der article-Klasse erstellen. Dabei dürfen auch Optionen und Pakete wie twoside, german oder geometry zum Einsatz kommen. Da die Einträge jedoch niemals über Spalten hinweg umbrochen werden, ist es nicht empfehlenswert, sie etwa mit multicol mehrspaltig zu setzen.

Das Argument des Befehls \bibliography muss die Namen aller BibTeX-Datenbanken enthalten, die man ausgeben möchte. Mithilfe des \bibliographystyle-Befehls kann man einen bestimmten BibTeX-Stil auswählen. Entsprechend der Voreinstellung werden alle bibliographischen Einträge der Datenbanken ausgegeben. Bei Verwendung expliziter \nocite-Befehle (wie im folgenden Beispiel), werden jedoch nur die ausgewählten Datenbankeinträge ausgedruckt. Interne Querverweise mithilfe des crossref-Feldes oder expliziter \cite-Befehle werden durch Kästen um den *schlüssel* angezeigt, anstatt sie aufzulösen.

(28. Juli 2005) tex.bib

Literatur

- MR-PQ
 Frank Mittelbach and Chris Rowley.
 The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography?
 In Vanoirbeek and Coray [EP92], pages 261–273.
- EP92
 Christine Vanoirbeek and Giovanni Coray, editors.
 EP92—*Proceedings of Electronic Publishing*, '92, Cambridge, 1992. Cambridge University Press.

```
\usepackage{biblist}
\bibliographystyle
  {alpha}

\nocite{MR-PQ}
\footnotesize
\bibliography{tex}
```

Bsp.
13-4-1

Zunächst muss man L^AT_EX aufrufen, dann BibTeX und dann noch einmal L^AT_EX. Es sind keine zusätzlichen L^AT_EX-Läufe erforderlich, da die Querverweise aus Platzgründen nicht aufgelöst werden. Entsprechend erfolgen immer Warnungen über nicht aufgelöste Quellenverweise.

13.4.2 bibtools – Ein Werkzeugset

Es sind mehrere Sätze interessanter BibTeX-Werkzeuge im Umlauf. Der erste wurde (hauptsächlich) von David Kotz entwickelt. Seine Werkzeuge sind gesammelt für UNIX-Systeme verfügbar (oder für cygwin unter Windows). Damit die Skripte in der jeweiligen Umgebung funktionieren, muss man möglicherweise die an ihrem Anfang stehenden Pfadangaben anpassen.

aux2bib Mittels einer .aux-Datei erstellt dieses perl-Skript eine portable .bib-Datei, welche nur die für das jeweilige Dokument benötigten Einträge enthält. Das ist z.B. hilfreich, wenn man L^AT_EX-Dateien versenden will. Das Skript nutzt eine eigene BibTeX-Stildatei (subset) zum Extrahieren der benötigten Einträge, so dass nur Standardfelder unterstützt werden.

Bibliography files

../EX/jura

July 13, 2003

References

- [aschur] Hans Brox and Wolf-Dietrich Walker. *Allgemeines Schuldrecht*. München, 29. edition, 2003.
- [bgb] Otto Palandt. *Bürgerliches Gesetzbuch*. Beck Juristischer Verlag, München, 62. edition, 2003.
- [bschur] Hans Brox and Wolf-Dietrich Walker. *Besonderes Schuldrecht*. München, 27. edition, 2002.
- [zpo] Adolf Baumbach, Wolfgang Lauterbach, Jan Albers, and Peter Hartmann. *Zivilprozeßordnung mit Gerichtsverfassungsgesetz und anderen Nebengesetzen*. München, 59. neubearb. edition, 2002.

Abbildung 13.1: Ausgabe des Programms printbib

bibkey Dieses C-Shellskript erstellt mittels der Hilfsprogramme `sed`, `egrep` und `awk` eine Liste aller Einträge, deren Verweisschlüssel eine bestimmte Zeichenfolge enthält.

Verwendung: `bibkey zeichenfolge datei`

Zeichen im Argument *zeichenfolge*, die in regulären Ausdrücken von `sed` oder `egrep` eine besondere Bedeutung haben, müssen mit einem `\`-Zeichen maskiert werden (wie z.B. `\\` für einen Backslash). Groß- oder Kleinschreibung werden bei der Suche ignoriert. Jeder gültige `egrep`-Ausdruck ist erlaubt, so z.B. auch eine Suche nach mehreren Schlüsseln:

`bibkey 'bgb|zpo' jura.bib`

looktex Ruft man dieses C-Shellskript auf, so werden alle Einträge einer BibTeX-Datenbank aufgelistet, die eine gegebene Zeichenfolge enthalten. Es handelt sich um eine verallgemeinerte Form des `bibkey`-Skripts und alle Anmerkungen zu diesem Skript treffen auch hier zu.

makebib Dieses C-Shellskript erstellt aus einem gegebenen Satz von `.bib`-Dateien und einer optionalen Verweisliste eine exportierbare `.bib`-Datei.

Verwendung: `makebib bib-datei(en) [zitierschlüssel]`

Das Ergebnis wird in die Datei `subset.bib` geschrieben. Ohne das Argument *zitierschlüssel* werden alle Quellen der angegebenen *bib-datei(en)* verarbeitet.

printbib Dieses C-Shellskript erzeugt aus einer `.bib`-Datei eine `.dvi`-Datei, die dazu dient, das Erstellen von Quellenverweisen zu vereinfachen. Die Datei ist nach dem Zitierschlüssel sortiert und enthält auch die Felder `keyword` und `abstract`.

Verwendung: `printbib bib-datei(en)`

Es wird eine Datei namens `abstract.dvi` erzeugt, die sich mittels eines `dvi`-Treibers drucken lässt. Abbildung 13.1 zeigt die Ausgabe, die erzeugt

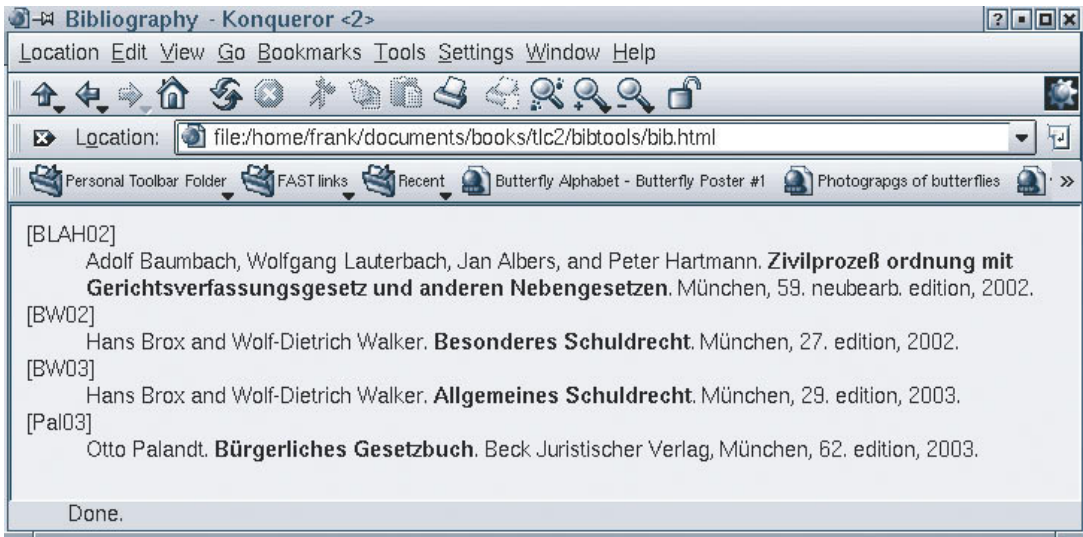


Abbildung 13.2: Ausgabe des Programms bib2html

wird, wenn man die Datenbank `jura.bib` von Seite 742 mit diesem Shellskript verarbeitet.

bib2html Dieses perl-Skript erzeugt eine HTML-Version einer oder mehrerer BibTeX-Datenbankdateien.

Verwendung: `bib2html stil [-o ausgabedatei] bib-datei(en)`

Es stehen verschiedene *stile* zur Auswahl. Abbildung 13.2 wurde mithilfe des Stils `alpha` aus der `jura.bib`-Datenbank erzeugt. Wird keine *ausgabedatei* benannt, dann verwendet das Programm den voreingestellten Namen `bib.html`. Mithilfe der Option `-a` und unter Angabe einer *.aux*-Datei kann man anstelle der gesamten Datenbankeinträge auch eine Bibliographie ausgeben, die nur Einträge enthält, auf die in der entsprechenden Datei verwiesen wird.

Verwendung: `bib2html format [-o ausgabedatei] -a aux-datei`

13.4.3 bibclean usw. – Noch ein Werkzeugset

Eine zweite Sammlung von Werkzeugen, die den Umgang mit BibTeX-Datenbanken erleichtern, wurde von Nelson Beebe entwickelt. Sie werden im Folgenden kurz beschrieben.

bibclean Dieses C-Programm dient als Prettyprinter, zur Syntaxüberprüfung und zur lexikalischen Analyse von BibTeX-Datenbankdateien [13]. Das Programm, das auf UNIX-, Vax/VMS- und Windows-Plattformen funktioniert, verfügt über zahlreiche Optionen. In der Regel genügt jedoch die folgende Eingabe:

`bibclean < bib-datei(en) > ausgabedatei`

Verarbeitet man beispielsweise die Datenbankdatei `tex.bib` mit `bibclean`, so erhält man folgende Fehlermeldung:

```
%% "EX/tex.bib", line 92: Unexpected value in ‘‘year = "1980ff’’’.
```

(Unerwarteter Wert im Feld „year“ = "1980ff".)

bibextract Dieses Programm entnimmt aus einer Liste von `BiBTeX`-Dateien alle Literatureinträge, die einem angegebenen Paar regulärer Ausdrücke entsprechen, und schreibt diese zusammen mit allen `@preamble`- und `@string`-Deklarationen nach `stdout`. Dazu müssen zwei reguläre Ausdrücke angegeben werden: Der erste legt die zu untersuchenden Felder fest (wenn diese Zeichenfolge leer ist, werden alle Felder eines Eintrags untersucht); der zweite dient dazu, nach den Feldinhalten zu bestimmen, welche Bibliographieeinträge ausgegeben werden müssen. Reguläre Ausdrücke sollten nur Kleinbuchstaben enthalten.

Der folgende Befehl z.B. extrahiert alle Einträge, deren Felder irgendwo die Zeichenfolge „PostScript“ enthalten:

```
bibextract "" "postscript" bib-datei(en) > neue-bib-datei
```

Der nächste Befehl extrahiert nur solche Einträge, deren Felder `author` oder `organization` die Zeichenfolge `Adobe` enthalten.

```
bibextract "author|organization" "adobe" bib-datei(en) > neue-bib-datei
```

Man beachte, dass die `.bib`-Dateien möglicherweise zunächst mit `bibclean` bereinigt werden müssen, bevor `bibextract` die richtigen Einträge findet. Die beiden Einträge zum Verfasser „Mittelbach“ sind mit folgendem Befehl zu finden:

```
bibclean tex.bib | bibextract "author" "mittelbach"
```

Der Einsatz von `bibextract` allein würde hier fehlschlagen, da ein Eintrag die Zeile `year={1980ff}` enthält.

citefind und citetags Manchmal muss man die Quellen, auf die in einem Dokument tatsächlich verwiesen wird, erst aus mehreren großen Datenbanken zusammenstellen. Die Bourne-Shellskripts `citefind` und `citetags` lösen diese Aufgabe mithilfe der Werkzeuge `awk` und `sed`.

Zunächst extrahiert `citetags` die `BiBTeX`-Verweisschlüssel aus den Quell- oder `.aux`-Dateien in `LaTeX` und überträgt diese nach `stdout`. Dort werden sie von `citefind` übernommen und das Programm versucht, die entsprechenden Schlüssel in den angegebenen `.bib`-Dateien zu finden. Danach schreibt es die so zusammengestellte neue Bibliographiedatei nach `stdout`, zum Beispiel:

```
citetags *.aux | citefind - bib-datei(en) > ausgabedatei
```

Nelson Beebe hat auch das `showtags`-Paket entwickelt, das eine Literaturliste um Verweisschlüssel ergänzt. Es erfüllt also eine ähnliche Aufgabe,

wie das Paket `biblist` (vgl. Beispiel 13-4-1 auf Seite 804) oder das `printbib`-Programm (vgl. Abbildung 13.1 auf Seite 805).

Literatur

[MR92] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Vanoirbeek and Coray [VC92], pages 261–273.

MR-PQ

```
\usepackage
{showtags}
\bibliographystyle
{is-alpha}
```

[VC92] Christine Vanoirbeek and Giovanni Coray, editors. *EP92—Proceedings of Electronic Publishing*, '92, Cambridge, 1992. Cambridge University Press.

EP92

```
\nocite{MR-PQ}
\footnotesize
\bibliography{tex}
```

Bsp.
13-4-2

13.4.4 bibtool – Ein Multifunktionswerkzeug

Das Programm `bibtool` wurde von Gerd Neugebauer entwickelt, um damit `BiBTeX`-Datenbanken zu bearbeiten. Es vereint viele Funktionen der bereits besprochenen Werkzeuge in einem einzigen Programm und fügt noch einige neue hinzu. Es ist im C-Quellcode erhältlich; man findet jedoch auch vorkompilierte Binärdateien, etwa in der Debian-Distribution. Es wurde bereits auf vielen Rechnerarchitekturen erfolgreich kompiliert, vorausgesetzt sie verfügen über einen geeigneten C-Compiler.

In diesem Abschnitt werden einige der Programmfunktionen vorgestellt. Das zum Programm gehörige Anwenderhandbuch [134] beschreibt einen noch wesentlich größeren Funktionsumfang.

Formatieren, zusammenführen und sortieren

Die einfachste Art das Programm aufzurufen, besteht darin, dass man ihm ein oder mehrere `BiBTeX`-Datenbanken als Argument(e) übergibt. Es dient dann als Prettyprinter und schreibt seine Ergebnisse an `stdout`.¹ Ist die Option `-o datei` angegeben, so wird das Ergebnis in die angegebene `datei` geschrieben. Um beispielsweise die in Abbildung 12.2 auf Seite 714 gezeigte Datenbank mit diesem Programm zu verarbeiten, könnte man folgenden Befehl eingeben:

```
bibtool tex.bib -o new-tex.bib
```

Dadurch erhält man eine ordentlich formatierte Version dieser Datenbank als Datei mit dem Namen `new-tex.bib`. Alle Einträge erscheinen mit akkuratem Einzug, wobei jedes Feld in einer eigenen Zeile steht und die Gleichheitszeichen exakt untereinander angeordnet sind. Hier der unübersichtlichste Eintrag aus der Datei `tex.bib`:

```
@manual{GNUmake,      key = {make},
  title = {{GNU Make}, A Program for Directing
  Recompilation}, organization= "Free
  Software Foundation",address = "Boston,
  Massachusetts",ISBN={1-882114-80-9},year = 2000}
```

¹ Sofern keine Eingabedateien festgelegt werden, liest `bibtool` von `stdin`. Dadurch eignet es sich auch als Filter für eine UNIX-Pipe-Konstruktion, was manchmal sehr praktisch sein kann.

Dieser wird mit folgendem Ergebnis umformatiert:

```
@Manual{      gnumake,
  key         = {make},
  title       = {{GNU Make}, A Program for Directing Recompilation},
  organization = "Free Software Foundation",
  address     = "Boston, Massachusetts",
  isbn        = {1-882114-80-9},
  year       = 2000
}
```

Gibt man mehrere Datenbankdateien an, so werden sie bei der Ausgabe alle zusammengeführt. Sofern gewünscht, kann man sie nach ihren Verweisschlüsseln sortieren (mithilfe der Option `-s` bzw. `-S` für eine umgekehrte Sortierreihenfolge). Stattdessen kann man auch einen eigenen Sortierschlüssel festlegen. Das geschieht mittels der Ressource¹ `sort.format`:

Zusammenführen und Sortieren

```
bibtex -- 'sort.format="%N(author)'" tex.bib jura.bib
```

Dabei ist zu beachten, dass durch die Sortierung eine ungültige Bibliographiedatei entstehen kann: Einträge, auf die mit dem `BIBTEX`-Feld `crossref` verwiesen wird, müssen in der Datenbank hinter den auf sie verweisenden Einträgen erscheinen. Ein Sortiervorgang kann diese Reihenfolge ändern. Im Handbuch ist beschrieben, wie man dieses Problem beim Definieren eines Sortierschlüssels berücksichtigt.

Das Zusammenführen von Datenbanken kann auch zu doppelten Einträgen führen, oder genauer gesagt zu Einträgen, deren `LATEX`-Verweisschlüssel identisch ist. Datenbanken mit solchen Duplikaten erzeugen bei der Verarbeitung mit `BIBTEX` Fehlermeldungen. Durch die Option `-d` werden die Duplikate als Kommentare ausgegeben, so dass `BIBTEX` wieder zufrieden ist. Das kann jedoch auch bedeuten, dass verschiedene Einträge (die fälschlicherweise den gleichen Schlüssel haben) zu einem einzelnen zusammengefasst werden. Daher ist diese Option mit Vorsicht zu genießen.

Doppelte Schlüssel entfernen

Normalisieren und Umschreiben von Einträgen

`BIBTEX` unterstützt als Feldbegrenzungen sowohl Anführungszeichen als auch geschweifte Klammern, so dass deren Vermischung im `GNUMake`-Eintrag durchaus zulässig, wenn vielleicht auch nicht empfehlenswert, ist. Es ist besser, sich an ein Schema zu halten, also entweder immer Klammern oder immer Anführungszeichen zu verwenden. Die folgende Regel dient zum Umschreiben von Einträgen:

```
bibtex -- 'rewrite.rule {"^"\([^#]*\)"$ " {\1}"}' tex.bib
```

Sie verändert alle Feldbegrenzungssymbole zu Klammerpaaren, außer in Fällen, in denen Zeichenfolgen miteinander verkettet werden. Der Beispieleintrag

¹Ressourcen sind Programmdirektiven, denen Werte zugeordnet werden. Diese Zuordnung erfolgt häufig in externen Dateien (wie später noch erklärt wird). In der Befehlszeile werden sie nach der Option `--` angegeben.

wird folgendermaßen ausgegeben:

```
@Manual{          gnumake,
  key             = {make},
  title           = {{GNU Make}, A Program for Directing Recompilation},
  organization    = {Free Software Foundation},
  address         = {Boston, Massachusetts},
  isbn            = {1-882114-80-9},
  year           = 2000
}
```

Leser, die mit regulären Ausdrücken vertraut sind, können die relativ komplexe Regel zum Umschreiben der Einträge wahrscheinlich ohne weiteres verstehen. Derartige reguläre Ausdrücke sind aber auch sehr ausführlich im Programmhandbuch beschrieben.

*Externe
Ressourcendateien*

Regeln zum Umschreiben können (wie auch alle anderen Ressourcendefinitionen) ebenso in einer eigenen Datei (mit der Erweiterung `.rsc`) gespeichert und dann mithilfe der Option `-r` geladen werden. Beispielsweise kann man Begrenzungssymbole in Form von Anführungszeichen mit folgender Befehlszeile entfernen:

```
bibttool -r braces tex.bib
```

Diese lädt die Distributionsdatei `braces.rsc`, welche drei Regeln zum Umschreiben von Einträgen enthält. Sie ähneln der zuvor besprochenen, behandeln dabei jedoch zusätzliche Fälle.

Umschreibungsregeln können auf bestimmte Felder begrenzt werden. Dazu gibt man diese Felder gefolgt von einem `#`-Zeichen vor dem regulären Ausdruck an. Die folgende Regel zum Beispiel schreibt das Erscheinungsjahr um, wenn das Feld nur zwei Ziffern enthält, die möglicherweise in Anführungszeichen oder Klammern gesetzt sind, wobei die erste Ziffer keine null sein darf (da in diesem Falle nicht sicher ist, ob es sich um das Jahr 1902 oder 2002 handelt):

```
rewrite.rule {year # "^\[\"{]?\[([1-9][0-9])\]"}} "$" "19\1"
```

*Semantische
Prüfungen*

Anstelle des Umschreibens von Datenbankeinträgen kann man mithilfe der Ressource `check.rule` auch semantische Überprüfungen vornehmen. Die Befehlszeile

```
check.rule {year # "^\[\"{]?\[([0-9][0-9])\]"}} "\@ \$: year = \1\n"}
```

erzeugt eine Warnung, dass ein `year`-Feld mit verdächtigem Inhalt gefunden wurde, sofern das Feld nur zwei Ziffern enthält. (In der Mitteilung wird das `\@`-Zeichen durch den Publikationstyp und das `\$`-Zeichen durch den Verweisschlüssel ersetzt.) Wenn man dies auf die Beispieldatenbank anwendet, erhält man die Meldung:

```
*** BibTool: Book vleunen:92: year = 92
```

Im Benutzerhandbuch werden auch ausführlichere semantische Prüfungen besprochen.

LaTeX-Datenbanken dürfen auch @string-Deklarationen enthalten, die in den Einträgen als Kurzformen verwendet werden. In bestimmten Fällen sollen diese vielleicht durch die eigentlichen Begriffe ersetzt werden. Dazu dient die folgende Befehlszeile:

@string-
Deklarationen
entfernen

```
bibtex -- 'expand.macros=ON' tex.bib
```

Als Ergebnis ändert sich der Inhalt des Feldes `series` in den Einträgen `lgc97` und `lwc99` von

```
series = ttct
```

zu der Langform

```
series = {Tools and Techniques for Computer Typesetting}
```

Das `bibtex`-Programm erweitert nur die Kurzformen, die in der Datenbankdatei selbst definiert sind – Kurzformen aus LaTeX-Stildateien bleiben unangetastet. Sollen auch diese erweitert werden, so muss man zusätzlich eine `.bib`-Datei laden, in welcher diese explizit als @string-Deklarationen enthalten sind.

Extrahieren von Einträgen

Es gibt verschiedene Möglichkeiten, eine Untermenge von Einträgen aus einer Datenbank auszuwählen. Die Option `-x aux-datei` sucht in der angegebenen `aux-datei` nach `\citation`-Befehlen. Auf deren Grundlage erzeugt sie eine neue `.bib`-Datei, die nur die Einträge enthält, die für das entsprechende Dokument benötigt werden, wie zum Beispiel:

```
bibtex -x 12-1-1.aux -o 12-1-1.bib
```

Es müssen keine Quelldatenbanken angegeben werden, da das Programm auch diese Information aus der `.aux`-Datei erhält. Alle Einträge, auf die Querverweise erfolgen, werden automatisch als erforderlich gekennzeichnet.

Eine weitere Möglichkeit bietet sich mit der Option `-X reg-ausdr`, die alle Einträge extrahiert, deren Verweisschlüssel mit dem regulären Ausdruck `reg-ausdr` übereinstimmt. Zum Beispiel wählt die Befehlszeile

```
bibtex -X '^mr-|^so-' tex.bib
```

die beiden Einträge mit den Verweisschlüsseln `MR-PQ` und `Southall` aus. Das Handbuch enthält nähere Einzelheiten zu regulären Ausdrücken. Mithilfe dieser Ausdrücke lassen sich nur Einträge auswählen, die explizit mit ihnen übereinstimmen. Die Einträge des Beispiels, auf die Querverweise erfolgen, wie etwa `EP92`, werden daher nicht automatisch mit eingeschlossen. Dies lässt sich jedoch erzwingen, indem man der Ressource `select.crossrefs` den Wert `ON` zuweist.

Durch Verwenden weiterer Ressourcen lässt sich die Auswahl noch genauer steuern. Mit der folgenden Befehlszeile etwa, kann man alle Einträge auswählen, in denen Knuth oder Lamport als Verfasser oder Herausgeber erscheinen:

```
bibttool -- 'select={author editor "Knuth\Lamport"}' tex.bib
```

Alle Einträge der Publikationstypen `book` oder `article` findet man mit

```
bibttool -- 'select={@book @article}' tex.bib
```

Und um alle Einträge ohne Erscheinungsjahr zu finden, schreibt man

```
bibttool -- 'select.non={year ".+"}' tex.bib
```

Durch Kombinieren solcher Ressourcendefinitionen in einer Ressourcendatei und durch Übergeben der Ergebnisse von einer `bibttool`-Instanz an die nächste, kann man so beliebig komplizierte Umschreibungs- und Suchmethoden bereitstellen.

Erzeugen des Verweisschlüssels

Wie bereits in Kapitel 12 erklärt wurde, kann man den Verweisschlüssel mit ein paar Einschränkungen frei wählen. Dieser Schlüssel ist die Zeichenfolge, die der `\cite`-Befehl als Argument erhält, um auf einen Eintrag in der Bibliographie zu verweisen. Trotzdem ist es empfehlenswert, sich an ein bestimmtes Schema zu halten, da sich die Schlüssel dann leichter einprägen und es seltener zu doppelten Schlüsseln kommt. Das Programm `bibttool` kann dieses Vorgehen unterstützen, indem es die Schlüssel in einer Datenbank nach einem solchen Schema abändert. Das ist natürlich nur bei Datenbanken sinnvoll, die noch nicht im Gebrauch sind, da $\text{\LaTeX}$ ansonsten die in den Dokumenten angegebenen Schlüssel nicht mehr finden kann.

Mit `-k` und `-K` stehen zwei vordefinierte Optionen zur Verfügung. Beide erzeugen kleingeschriebene Schlüssel, die aus Autorennamen und dem ersten relevanten Wort im Titel (außer „The“ oder ähnlichen Wörtern) bestehen, wobei Befehle und Klammern ignoriert werden. Verarbeitet man also die Datenbank aus Abbildung 12.3 auf Seite 742 mit `bibttool` und sucht dann mithilfe der Befehlszeile

```
bibttool -k jura.bib | grep @
```

nach Zeilen mit `@`-Zeichen (um die Ergebnisliste zu verkleinern), so erhält man das folgende Ergebnis:

```
@Book{      baumbach.lauterbach.ea:zivilproze,
@Book{      brox.walker:allgemeines,
@Book{      brox.walker:besonderes,
@Book{      palandt:bürgerliches,
```

Der etwas seltsame Schlüssel, der auf `:zivilproze` endet, resultiert daraus, dass der Eintrag das Wort `Zivilproze\ssordnung` enthält. Das Programm geht davon aus, dass dieses Wort nach `\ss` endet, das selbst nicht berücksichtigt wird, da es ein Befehl ist. Ebenso wird `\u` im vierten Schlüssel als „u“ wiedergegeben. Durch Laden der Ressourcendatei `tex_def.rsc` lässt sich das Ergebnis erheblich verbessern. Diese Datei verwendet die Ressource `tex.define`, um gängige L^AT_EX-Befehle zu übersetzen, so dass

```
bibttool -r tex_def -k jura.bib | grep @
```

folgende Schlüssel erzeugt:

```
@Book{          baumbach.lauterbach.ea:zivilprozessordnung,
@Book{          brox.walker:allgemeines,
@Book{          brox.walker:besonderes,
@Book{          palandt:buergerliches,
```

Auch andere Programme zum Bearbeiten von Bib_TE_X-Datenbanken haben ähnliche Probleme beim Parsen von Befehlen, die durch Leerzeichen begrenzt sind. Daher ist an solchen Stellen normalerweise die Schreibweise `\ss{}` oder `{\ss}` vorzuziehen. In Abbildung 13.2 auf Seite 806 sieht man z.B., dass `bib2html` sich ebenfalls durch die Schreibweise täuschen ließ und im ersten Eintrag fälschlicherweise ein zusätzliches Leerzeichen eingefügt hat.

Die andere Option zum Erzeugen der Schlüssel (`-K`) funktioniert ähnlich. Sie fügt noch die Initialen der Autoren nach dem Namen ein:

```
@Book{          baumbach.a.lauterbach.w.ea:zivilproze,
@Book{          brox.h.walker.w:allgemeines,
```

Mithilfe der im Handbuch beschriebenen vielfältigen Konfigurationsmöglichkeiten lassen sich weitere Muster festlegen.

13.4.5 pybliographer – Eine erweiterbare Bibliographieverwaltung

Die Scripting-Umgebung `pybliographer` von Frédéric Gobry ist ein Werkzeug zum Verwalten bibliographischer Datenbanken. In der aktuellen Version unterstützt sie die folgenden Formate: Bib_TE_X, ISI (Web of Knowledge), Medline, Ovid und Refer/EndNote, die sie auch ineinander konvertieren kann. Die Anwendung ist in Python geschrieben, so dass sie ohne weiteres für UNIX-Systeme geeignet ist; auf Windows-Rechnern kann sich ihr Einsatz jedoch problematisch gestalten, obwohl es auch Python-Versionen für diese Plattform gibt. Die `pybliographer`-Website ist unter der Internetadresse <http://pybliographer.org> zu finden.

Das graphische Frontend für `pybliographer` basiert auf den Gnome-Bibliotheken und trägt den Namen `pybliographic`. Beim Programmaufruf kann

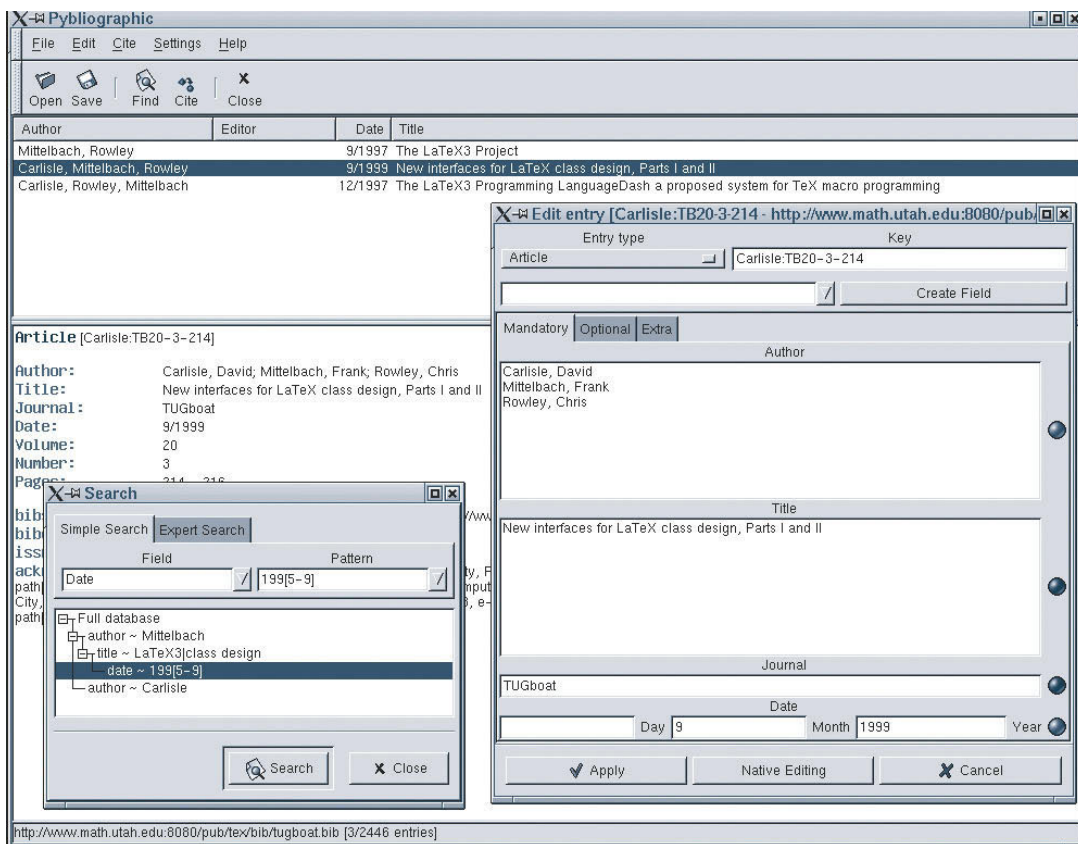


Abbildung 13.3: Die pybliographic-Benutzeroberfläche

man die Datenbank angeben, mit der man arbeiten möchte. Normalerweise handelt es sich um eine lokale Datei, es kann jedoch auch eine Remote-Datenbank sein, deren URL angegeben wird. Der Befehl

```
pybliographic http://www.math.utah.edu:8080/pub/tex/bib/tugboat.bib
```

beispielsweise öffnet einen Arbeitsbereich wie den in Abbildung 13.3 dargestellten. Er wird jedoch nicht genauso aussehen, da sich die graphische Benutzeroberfläche sehr flexibel anpassen lässt. Bei der vom Autor benutzten Version wurde z.B. im Hauptbildschirm zwischen der „author“-Spalte und dem Erscheinungsjahr eine Spalte „editor“ hinzugefügt. Über das Dialogfeld „Preferences“ (Settings → Preferences → Gnome) lassen sich andere Felder einblenden. Auf UNIX-Systemen werden die Einstellungen in der Datei `.pybrc.conf` gespeichert. Diese Datei lässt sich zwar nicht vom Anwender bearbeiten, man kann sie jedoch gegebenenfalls entfernen, um die ursprünglich vorgegebene Konfiguration wiederherzustellen.

Abbildung 13.3 zeigt noch einige weitere interessante Funktionen. Am Fuß des Hauptfensters sieht man, dass die geladene Datenbank 2446 Einträge

enthält (`tugboat.bib`), von denen zurzeit 3 angezeigt werden. Das liegt daran, dass die Datenbank nach Einträgen durchsucht wurde, die zum Muster des regulären Ausdrucks `Mittelbach` im Feld `author` passen (30 gefundene Einträge); diese wiederum wurden nach Einträgen durchsucht, deren `title`-Feld den Text `LaTeX3` oder `class design` enthält (5 gefundene Einträge); und innerhalb dieses Ergebnisses wurde die Suche dann auf Veröffentlichungen aus den Jahren 1995 bis 1999 beschränkt. Das Dialogfenster „Search“ zeigt die zurzeit definierten, verfügbaren hierarchischen Ansichten. Durch Anklicken der einzelnen Ansichten kann man zwischen ihnen hin- und herwechseln. Mit einem Rechtsklick werden nicht mehr benötigte Ansichten gelöscht. Die Suchfelder lassen sich einstellen, wobei die Anfangseinstellungen zunächst nur wenige Felder anbieten.

Hierarchische Suche

Um einen vorhandenen Eintrag zu bearbeiten, führt man im Hauptfenster einen Doppelklick darauf aus. Stattdessen kann man auch das Edit-Menü der Werkzeugleiste verwenden oder mit einem Rechtsklick das Kontextmenü aufrufen. Die letzten beiden Möglichkeiten stehen auch zum Löschen oder Hinzufügen von Einträgen zur Verfügung. Das Dialogfeld „Edit“ zeigt den Eintrag in einem Bearbeitungsformat, bei dem die Registerkarte „Mandatory“ geöffnet ist, welche die für den aktuellen Publikationstyp obligatorischen Felder anzeigt. Außerdem gibt es noch die optionalen Felder auf der Registerkarte „Optional“ und möglicherweise weitere Felder auf der Registerkarte „Extra“. Diese Einteilung wird nach den aktuellen Einstellungen vorgenommen und lässt sich leicht an die eigenen Erfordernisse anpassen. Das Programm `pybibliographic` kann zwar problemlos Datenbanken mit beliebigen Feldnamen laden, diese erscheinen jedoch alle auf der Registerkarte „Extra“. Das kann störend sein, wenn man mit einem erweiterten $\text{\LaTeX}$ -Stil arbeitet, wie etwa mit `jurabib`, der zusätzliche Felder als entweder erforderlich oder optional betrachtet. In solchen Fällen lohnt es sich, die vorgegebenen Einstellungen anzupassen (Settings → Entries, Fields).

Rechts der Felder sieht man runde Knöpfe, die grün oder rot sein können. Mit roten Knöpfen signalisiert `pybibliographic`, dass das jeweilige Feld Daten enthält, die das Programm nicht fehlerfrei parsen konnte, und dass beim Ändern des Textes wahrscheinlich Daten verloren gehen werden. Im Feld `title` beispielsweise konnte der Befehl `\LaTeX{}` nicht einwandfrei interpretiert werden, so dass das Programm stattdessen `LaTeX` anzeigt. Das Feld `journal` ist markiert, da die Datenbank eigentlich folgenden Text enthält:

*Warnung bei
problematischen
Inhalten*

```
Journal      = j-tugboat,
```

Dieser Verweis auf eine Kurzform ginge verloren, sobald man das betreffende Feld bearbeitet. Um solche Einträge zu bearbeiten, muss man auf „Native Editing“ umschalten, wie in Abbildung 13.4 auf der nächsten Seite zu sehen. Dies geschieht mittels der „Native Editing“-Schaltfläche im Dialogfenster „Edit entry“. Das Fenster verändert seine Form dann zu dem mittleren Fenster in Abbildung 13.4 und bietet ein $\text{\LaTeX}$ -Standardeingabeformat, in dem man nach Belieben editieren kann. Allerdings muss man hier selbst darauf achten, die $\text{\LaTeX}$ -Syntax einzuhalten. Wie im rechten Fenster der Abbildung zu sehen, kann man den „Native Editing“-Modus auch als Standardmodus einstellen.

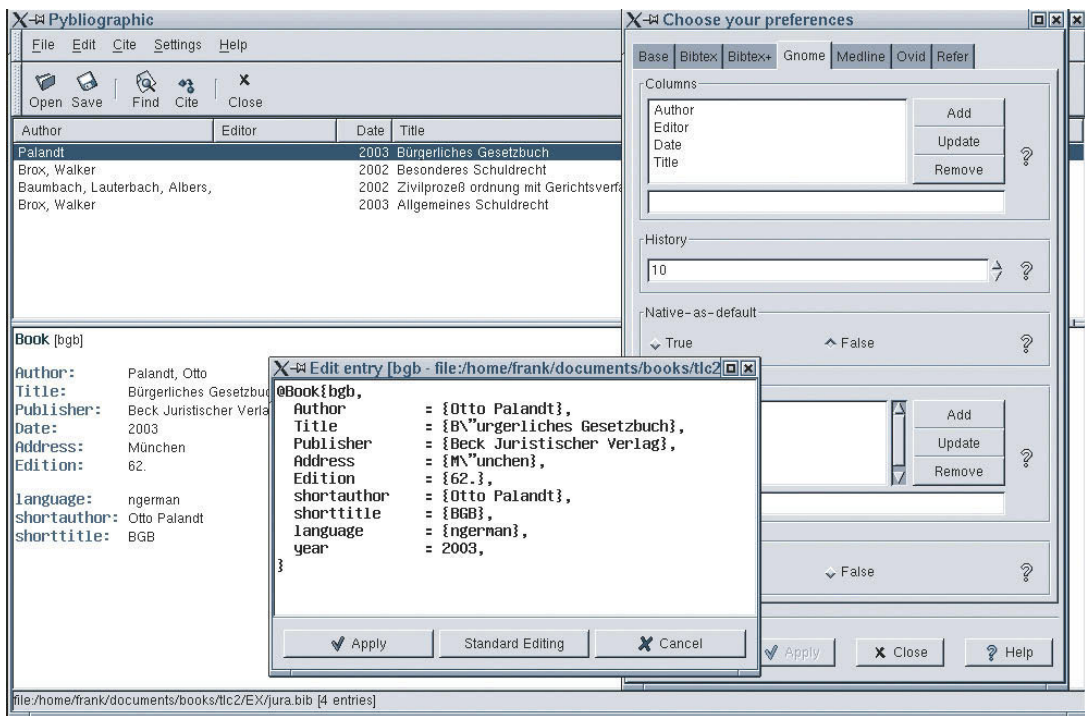


Abbildung 13.4: Native Editing in pybliographic

Vorgegebene Regeln zur Großschreibung

Beim Laden einer Datenbank normalisiert pybliographic die Großschreibung in einigen Feldern (z.B. in title). Da diese Überarbeitung im Rahmen der Aufbereitung für eine bestimmte Zeitschrift besser von BibTEX vorgenommen wird, sollte man in Erwägung ziehen, diese Funktion auszuschalten (Settings → Preferences → Bibtex+ → Capitalize). Bei nicht englischen Sprachen muss man sie sogar ausschalten, damit Substantive nicht plötzlich fälschlicherweise klein geschrieben werden.

Die Distribution enthält auch einige Kommandozeilenskripts. In der Dokumentation wird erläutert, wie man diese um weitere ergänzt. Mit pybconvert z.B. kann man Dateien in andere Formate konvertieren. Das Skript

```
pybconvert bibtex..refer tex.bib
```

konvertiert die BibTEX-Datenbank tex.bib in das Refer-Format und erzeugt dabei Ausgaben, wie die folgende:

```
%T A handbook for scholars
%P xi + 348
%I Oxford University Press
%F vLeunen:92
%D 92
%C Walton Street, Oxford OX2 6DP, UK
%A van Leunen, Mary-Claire
```

Je nach Inhalt der einzelnen Felder kann es dabei zu Warnungen wie der folgenden kommen: „warning: unable to convert ‘\textsl’“ (Warnung: Kann ‘\textsl’ nicht konvertieren). Das Programm kann solche Befehle nämlich nicht in ein T_EX-fremdes Format wie Refer konvertieren. In diesem Falle sollte man die Ergebnisse soweit erforderlich per Hand korrigieren.

Das Skript `pycompact` gleicht dem bereits besprochenen `perl`-Skript `aux2bib` und der Option `-x` des `bibtool`-Programms. Im Unterschied zu `bibtool` schließt es jedoch keine Einträge mit ein, auf die Querverweise erfolgen. Daher ist `bibtool`, sofern verfügbar, die bessere Wahl.

Ein recht interessantes Skript ist `pybcheck`, das eine Liste von BibT_EX-Datenbanken oder einen Verzeichnisnamen als Argument erwartet. Es überprüft dann alle Datenbanken auf Syntaxfehler, doppelte Schlüssel und andere Probleme. Führt man z.B. `pybcheck EX` aus, so erhält man

```
file 'EX/jura.bib' is ok [4 entries]
file 'EX/tex.bib' is ok [12 entries]
```

Das Skript überprüft die Datenbanken jeweils nur einzeln, so dass es nicht bemerkt, wenn verschiedene Dateien gleiche Schlüssel enthalten.

Emacs-Benutzer können den Befehl direkt über `M-x compile` gefolgt von `pybcheck datei(en)` aus einem Compile-Buffer heraus aufrufen. Vom Ergebnisfenster aus kann man dann durch Klicken mit der mittleren Maustaste sofort zu einem der entdeckten Fehler springen.

13.4.6 JBibtexManager – Datenbankverwaltung in Java

Das Programm `JBibtexManager` von Nizar Batada ist ein in Java geschriebener BibT_EX-Datenbankmanager (siehe Abbildung 13.5 auf der nächsten Seite). Aufgrund der gewählten Programmiersprache funktioniert er auf allen Plattformen, auf denen Java ab der Version 1.4 verfügbar ist (z.B. Windows, UNIX-Derivate, Mac).

Das Programm erlaubt das Suchen nach Autor, Herausgeber, Titel und Schlüsselwörtern; das Sortieren nach Publikationstyp, Verweisschlüssel, Autor, Erscheinungsjahr, Titel, Zeitschrift, Herausgeber und Schlüsselwörtern; und es verfügt natürlich über die Standardfunktionen zum Bearbeiten der Einträge, einschließlich Hinzufügen, Löschen, Kopieren und Einfügen, auch zwischen Bibliographien. Es entdeckt automatisch doppelte Verweisschlüssel, wenn Datenbanken zusammengeführt werden. Außerdem bietet es die Möglichkeit, eine Bibliographie nach doppelten Einträgen zu durchsuchen (also Einträgen, die sich, wenn überhaupt, nur im Verweisschlüssel voneinander unterscheiden).

Wie `pybliographic` kann auch dieses Programm Daten aus verschiedenen bibliographischen Formaten importieren: BibT_EX, INSPEC, ISI (Web of Knowledge), Medline (XML), Ovid und Scifinder. Es verfügt über Exportfilter für HTML und einfachen Text. Bei Formaten, die keinen Verweisschlüssel beinhalten, erzeugt es automatisch passende Schlüssel, sofern die Autoren Daten so strukturiert sind, dass das Programm sie interpretieren kann.

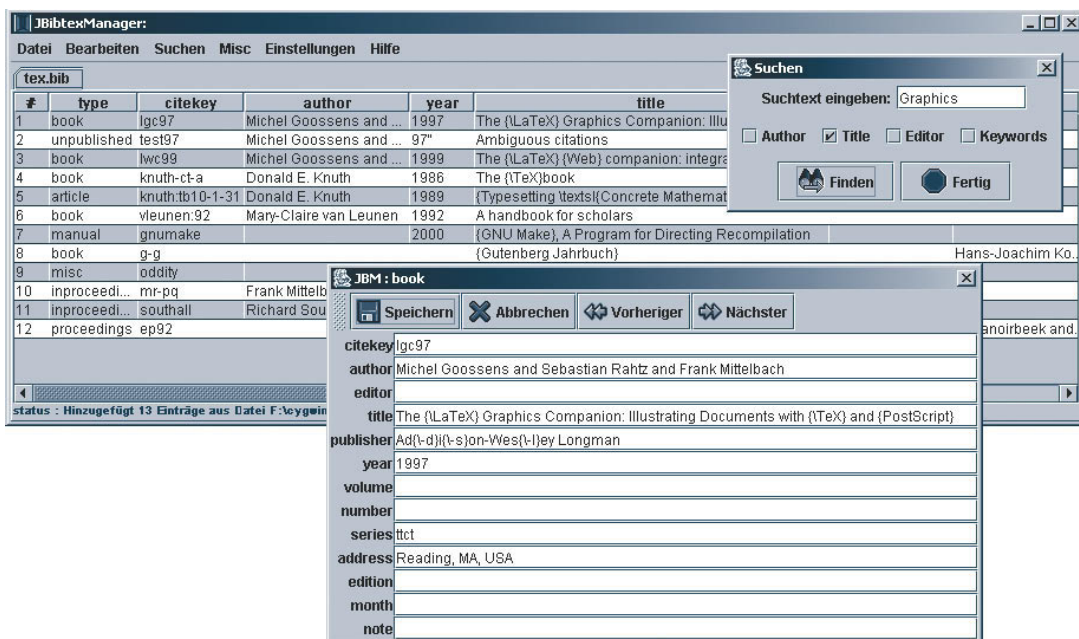


Abbildung 13.5: Die JBibTexManager-Benutzeroberfläche (deutsche Version)

Obwohl JBibTexManager hauptsächlich für die Verarbeitung von BibTeX-Datenbanken gedacht ist, können beim ersten Import solcher Dateien einige Schwierigkeiten auftreten, da nicht alle Syntaxvarianten des BibTeX-Datenbankformates unterstützt werden. Es sollte vor allem in jeder Zeile nur ein Feld stehen. Beim Parsen des GNUMake-Eintrags in der Beispieldatenbank `tex.bib` würden also Fehler auftreten. Außerdem werden Einträge nur erkannt, wenn ihr Publikationstyp (der mit einem @-Zeichen anfängt) in der ersten Spalte beginnt. Andernfalls wird der Eintrag fälschlicherweise als Kommentar interpretiert und nicht übernommen.¹

Diese Probleme treten natürlich nur beim ersten Laden einer extern erstellten Datenbank auf. Hat das System die Daten erst einmal akzeptiert, dann werden sie so gespeichert, dass sie auch immer wieder problemlos geladen werden können. Ein Weg, diese Anfangsprobleme zu umgehen, besteht darin, die externe Datenbank zunächst mit einem Werkzeug wie bibtool oder bibclean vorzubereiten, da die Einträge nach der Überprüfung und der Formatierung durch einen Prettyprinter in einem akzeptablen Format vorliegen.

Unbekannte Felder in einem Datenbankeintrag sind nicht sichtbar und lassen sich nur im „raw BibTeX“-Modus der neuesten Programmversion bearbeiten. Man kann jedoch anpassen, welche Felder das Programm für jeden einzelnen Publikationstyp erkennt, so dass man es auch für erweiterte BibTeX-Stile benutzen kann, wie sie etwa von den Paketen jurabib- oder natbib verwendet werden.

¹Die meisten dieser Einschränkungen sind in der neuen Version von JBibTexManager behoben.

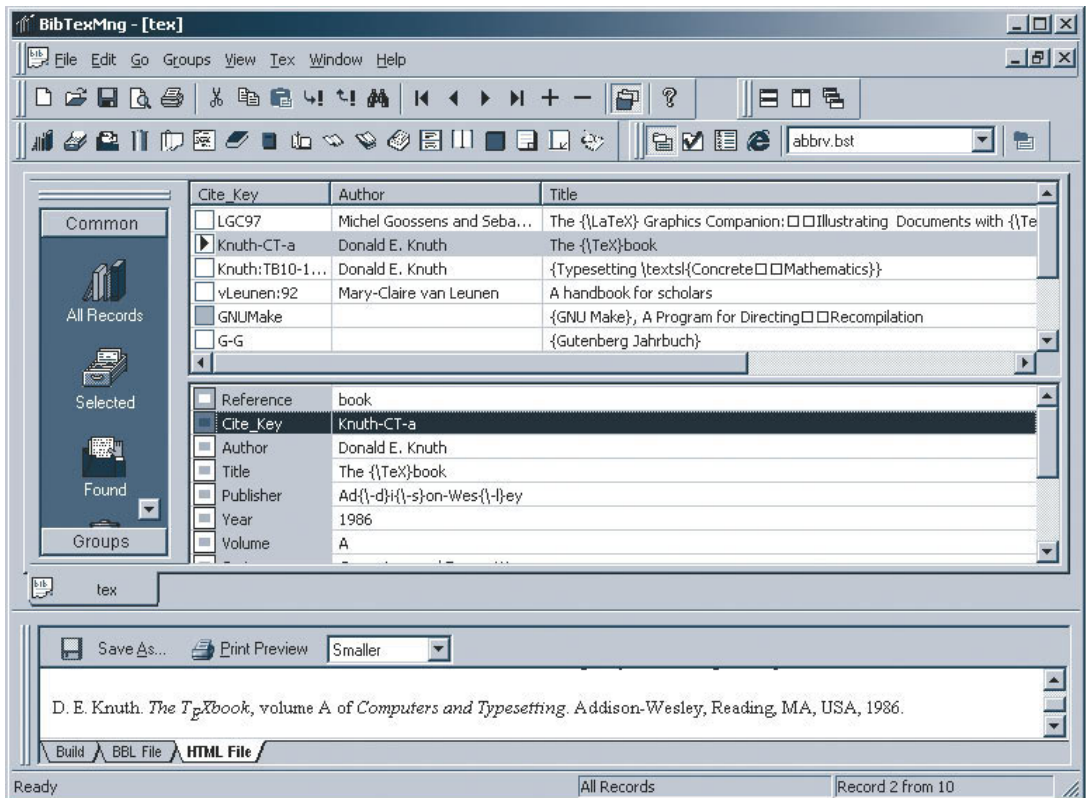


Abbildung 13.6: Die BibTeXMng-Benutzeroberfläche

Das Programm ist nicht auf dem CTAN erhältlich. Seine momentane Internetadresse ist <http://jabref.sourceforge.net/>, wo es 2004 mit einem ähnlichen Projekt namens BibKeeper unter dem neuen Namen JabRef zusammengeführt wurde.

13.4.7 BibTeXMng – Ein Datenbankmanager für Windows

Bei dem Programm BibTeXMng von Petr und Nikolay Vabishchevich handelt es sich um einen Bib_TE_X-Datenbankmanager für Windows (siehe Abbildung 13.6). Er unterstützt alle typischen Verwaltungsaufgaben: Bearbeiten, Suchen, Sortieren, Verschieben und Kopieren von Einträgen aus einer Datei in eine andere.

Im Gegensatz zu pybliographic oder JBibtexManager verarbeitet das Programm BibTeXMng ausschließlich Bib_TE_X-Datenbanken; es verfügt über keinerlei Import- oder Exportfunktionen für andere bibliographische Formate. Als einzige „fremde“ Formate werden .bbl- und .htm-Dateien unterstützt (d.h. man verarbeitet eine Auswahl von Einträgen aus dem Programm heraus mit Bib_TE_X oder Bib_TE_X8 oder erzeugt eine HTML-Datei aus dieser Auswahl von Einträgen).

In seiner aktuellen Version kennt das Programm leider nur die Standardpublikationstypen von BibTeX (siehe Tabelle 13.1 auf Seite 791), die BibTeX-Standardfelder (Tabelle 13.2) sowie folgende Felder:

abstract, affiliation, contents, copyright, isbn, issn, keywords, language, lccn, location, mrnumber, price, size und url

Alle anderen Felder werden beim ersten Laden einer BibTeX-Datenbank stillschweigend verworfen; das gilt auch für Publikationstypen, die nicht zum Standard gehören. Damit ist das Programm ungeeignet, wenn man BibTeX-Stile wie jurabib verwenden möchte, die zusätzliche Felder oder Typen einführen, da diese im Programm nicht dargestellt werden können. Es eignet sich jedoch für die meisten verfügbaren Stile, einschließlich der für natbib entwickelten (wie z.B. Stile, die mit custom-bib erzeugt wurden).

*Für jurabib et al. nicht
geeignet*

Man sollte auch beachten, dass das BibTexMng-Programm keine @string-Deklarationen unterstützt. Sind diese in einer extern erzeugten BibTeX-Datenbank vorhanden, so müssen sie erst entfernt werden, bevor man die Datenbank mit BibTexMng verwenden kann. Andernfalls treten beim Parsen der Einträge Fehler auf. Zur Unterstützung bietet das Programm eine Bereinigungsfunktion für externe Datenbanken an (unter File → Cleaning of BibTeX database). Diese Funktion ersetzt alle Kurzformen durch ihre Definitionen und entfernt gegebenenfalls vorhandene unbekannte Felder.

13.5 Formatieren von Bibliographien mit BibTeX-Stilen

Nachdem nun bekannt ist, wie man BibTeX-Datenbankeinträge mithilfe vielfältiger Werkzeuge generieren und bearbeiten kann, ist es Zeit, den eigentlichen Zweck des BibTeX-Programms näher zu beleuchten. Dieser besteht im Erzeugen eines Literaturverzeichnisses, das (je nach Inhalt des betreffenden Dokumentes) eine bestimmte Auswahl von Einträgen enthält, die bestimmten Konventionen entsprechend formatiert sind.

Zunächst wird anhand von Beispielergebnissen, die mittels Standard- und anderer Stile erzeugt wurden, die Verwendung existierender Stile besprochen. Danach wird gezeigt, wie sich mithilfe des custom-bib-Paketes auf einfache Weise angepasste Stile für nahezu jeden Zweck erstellen lassen.

13.5.1 Eine Sammlung von BibTeX-Stildateien

Verschiedene Organisationen oder Einzelpersonen haben BibTeX-Stildateien entwickelt, die den Layoutrichtlinien bestimmter Zeitschriften oder Verlagshäuser entsprechen. Nelson Beebe hat eine große Anzahl von BibTeX-Stildateien zusammengestellt. Zu jedem Stil gibt es eine Beispieldatei, die den Effekt des jeweiligen Stils veranschaulicht.¹ Einige dieser Stile – beispielsweise `authordate(i)`, `jmb` und `named` – müssen zusammen mit ihren zuge-

¹Anhang C gibt Aufschluss darüber, wie man diese Dateien aus einem der TeX-Archive erhält und ob sie nicht vielleicht schon auf dem eigenen System vorliegen.

hörigen L^AT_EX-Paketen verwendet werden, damit sie die gewünschten Effekte erzielen. Diese sind in Tabelle 13.4 genannt.

Man kann einen Bibliographiestil auch selbst anpassen, indem man die in der Tabelle aufgeführten leicht verändert. In Abschnitt 13.6.3 wird beschrieben, wie man dabei vorgeht. Oder man erzeugt mithilfe des Programms custom-bib eine eigene Stildatei, entsprechend der Erklärungen in Abschnitt 13.5.2 auf Seite 828.

Tabelle 13.4: Eine Auswahl von BibTeX-Stildateien

<i>Stilname</i>	<i>Beschreibung</i>
abbrv.bst	BibTeX-Standardstil
abbrvnat.bst	natbib-Variante des abbrv-Stils
abstract.bst	Abgewandelter alpha-Stil mit abstract-Schlüsselwort
acm.bst	BibTeX-Stil der Association for Computing Machinery
agsm.bst	BibTeX-Stil für Publikationen der australischen Regierung
alpha.bst	BibTeX-Standardstil
amsalpha.bst	alpha-ähnlicher BibTeX-Stil für $\mathcal{A}\mathcal{M}\mathcal{S}$ -T _E X
amsplain.bst	plain-ähnlicher BibTeX-Stil für $\mathcal{A}\mathcal{M}\mathcal{S}$ -T _E X (numerische Label)
annotate.bst	Abgewandelter BibTeX-Stil alpha mit annotate-Schlüsselwort
annotation.bst	Abgewandelter BibTeX-Stil plain mit annotate-Schlüsselwort
apa.bst	BibTeX-Stil der American Psychology Association
apalike.bst	Variante des BibTeX-Stils apa
apalike	L ^A T _E X-Paket für den Stil apalike.bst
apalike2.bst	Variante des BibTeX-Stils apalike
astron.bst	<i>Astronomy</i> -BibTeX-Stil
authordate <i>i</i> .bst	<i>i</i> =[1,4]; Reihe von BibTeX-Stilen, die Autor-Jahr-Verweise erzeugen
authordate1-4	L ^A T _E X-Paket für den BibTeX-Stil authordate <i>i</i> .bst
bbs.bst	BibTeX-Stil der <i>Behavioral and Brain Sciences</i>
cbe.bst	BibTeX-Stil des Council of Biology Editors (auch für Fachzeitschriften, wie <i>American Naturalist</i> und <i>Evolution</i>)
cell.bst	Leicht modifizierter BibTeX-Stil jmb
harvard	L ^A T _E X-Paket für die Harvard-Stile (z.B. agsm)
humanbio.bst	BibTeX-Stil für <i>Human Biology</i>
humannat.bst	Fachzeitschriften <i>Human Nature</i> und <i>American Anthropologist</i>
ieeetr.bst	BibTeX-Stil für die <i>Transactions of the Institute of Electrical and Electronic Engineers</i>
Fortsetzung auf nächster Seite	

Fortsetzung von letzter Seite	
Stilname	Beschreibung
is-abbrev.bst	B _B T _E X-Stil abbrev mit zusätzlichem ISSN- und ISBN-Schlüsselwort
is-alpha.bst	B _B T _E X-Stil alpha mit zusätzlichem ISSN- und ISBN-Schlüsselwort
is-plain.bst	B _B T _E X-Stil plain mit zusätzlichem ISSN- und ISBN-Schlüsselwort
is-unsrt.bst	B _B T _E X-Stil unsrt mit zusätzlichem ISSN- und ISBN-Schlüsselwort
jmb.bst	B _B T _E X-Stil des <i>Journal of Molecular Biology</i>
jmb	L ^A T _E X-Paket für den Stil jmb.bst
jox.bst	Stil zum jurabib-Paket (Oxfordstil)
jtb.bst	B _B T _E X-Stil des <i>Journal of Theoretical Biology</i>
jurabib.bst	Stil zum jurabib-Paket
jureco.bst	Stil zum jurabib-Paket (kompakt)
jurunsrt.bst	Stil zum jurabib-Paket (unsortiert)
kluwer.bst	Kluwer Academic Publishers B _B T _E X-Stil
named.bst	B _B T _E X-Stil mit [Autor(en)-Jahr-]Zitierschema
named	L ^A T _E X-Paket für den Stil named.bst
namunsrt.bst	Mit Namen versehene Variante des B _B T _E X-Stils unsrt
nar.bst	B _B T _E X-Stil des <i>Nucleic Acid Research</i>
nar	L ^A T _E X-Paket für den Stil nar.bst
nature.bst	B _B T _E X-Stil zu <i>Nature</i>
nature	L ^A T _E X-Paket für den Stil nature.bst
newapa.bst	Abwandlung des Stils apalike.bst
newapa	L ^A T _E X-Paket für den Stil newapa.bst
phaip.bst	B _B T _E X-Stil der Fachzeitschriften des <i>American Institute of Physics</i>
phapalik.bst	B _B T _E X-Stil der American Psychology Association
phcpc.bst	B _B T _E X-Stil der <i>Computer Physics Communications</i>
phiaea.bst	B _B T _E X-Stil für die Konferenzen der International Atomic Energy Agency
phjcp.bst	B _B T _E X-Stil des <i>Journal of Computational Physics</i>
phnf.bst	<i>Nuclear Fusion</i> -B _B T _E X-Stil
phnflet.bst	B _B T _E X-Stil der <i>Nuclear Fusion Letters</i>
phpf.bst	B _B T _E X-Stil zu <i>Physics of Fluids</i>
phppcf.bst	Physik-Version des B _B T _E X-Stils apalike
phreport.bst	B _B T _E X-Stil für interne Fachberichte zur Physik
phrmp.bst	B _B T _E X-Stil der <i>Reviews of Modern Physics</i>
plain.bst	B _B T _E X-Standardstil
plainnat.bst	natbib-Variante des Stils plain
plainyr.bst	B _B T _E X-Stil plain mit primärer Sortierung nach Erscheinungsjahr
Fortsetzung auf nächster Seite	

Fortsetzung von letzter Seite	
Stilname	Beschreibung
siam.bst	BibTeX-Stil der Society of Industrial and Applied Mathematics
unsrt.bst	BibTeX-Standardstil
unsrtnat.bst	natbib-Variante des Stils unsrt

Theoretisch kann man das Erscheinungsbild eines Literaturverzeichnisses einfach durch Verwendung eines anderen BibTeX-Stils verändern. In der Praxis gibt es jedoch ein paar Einschränkungen: Die Schnittstelle für BibTeX-Stile wurde von einigen Entwicklern so erweitert, dass ihre Stile in L^AT_EX zusätzliche Unterstützung benötigen. In Kapitel 12 wurden bereits einige Beispiele dazu gezeigt. So benötigen zum Beispiel alle Stile zum Autor-Jahr-Schema ein spezielles L^AT_EX-Paket, wie etwa natbib oder harvard, und die BibTeX-Stile zum jurabib-Paket funktionieren auch nur, wenn dieses Paket geladen ist.

Alles in allem funktioniert dieses Verfahren jedoch sehr gut. Das wird in diesem Abschnitt anhand verschiedener Beispiele belegt, indem ein gleich bleibendes Beispieldokument mithilfe unterschiedlicher BibTeX-Stile (und gegebenenfalls der erforderlichen Zusatzpakete) formatiert wird. Zu diesem Zweck werden fünf Quellen aus der mittlerweile gut bekannten Datenbank aus Abbildung 12.2 von Seite 714 zitiert: ein Artikel und ein Buch von Donald Knuth, die zeigen, wie verschiedene Veröffentlichungen des gleichen Autors gehandhabt werden; das Handbuch der Free Software Foundation, als Eintrag ohne Autor; der unveröffentlichte Eintrag mit vielen Autoren und dem besonderen BibTeX-Text „and others“; sowie eine Veröffentlichung aus einem Konferenzbericht, für den BibTeX zusätzliche Daten aus einem weiteren Eintrag heranziehen muss.

Im ersten Beispiel wird durch die folgende Eingabe der BibTeX-Standardstil plain verwendet:

```
\bibliographystyle{plain}
\nocite{Knuth:TB10-1,GNUMake,MR-PQ,Knuth-CT-a,test97}
\bibliography{tex}
```

Für das endgültige Dokument muss L^AT_EX die Beispieldatei einmal verarbeiten, damit die Quellenverweise in die .aux-Datei geschrieben werden. Danach verarbeitet BibTeX die erzeugte .aux-Datei, wobei die relevanten Einträge aus der BibTeX-Datenbank tex.bib ausgelesen werden. Der eigentliche Bibliographiestil, nach dessen Vorgaben die Datenbankeinträge zur späteren Verarbeitung durch L^AT_EX in die .bbl-Datei ausgegeben werden, ist im L^AT_EX-Dokument über den Befehl \bibliographystyle festgelegt. Abschließend wird L^AT_EX noch zweimal ausgeführt: einmal um die .bbl-Datei zu laden und ein zweites Mal um alle Verweise aufzulösen.¹ Dieser Ablauf wurde in Abschnitt 12.1.3 auf Seite 711 genauer beschrieben. Dort ist der Datenfluss zudem auch graphisch dargestellt (Abbildung 12.1).

Der Stil plain arbeitet mit numerischen Labels (in eckigen Klammern) und die Einträge sind alphabetisch nach Autor, Erscheinungsjahr und Titel

¹Für dieses Beispiel ist sogar nur ein Durchlauf erforderlich. Es gibt nämlich keine Querverweise, da überall der Befehl \nocite verwendet wurde.

sortiert, wobei Titel in Kleinbuchstaben konvertiert werden (inkorrekt im zweiten Eintrag). Das GNU-Handbuch wurde anhand der Organisation einsortiert. Damit erhält man die folgende Ausgabe:

- [1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.
- [2] Michel Goossens, Ben User, Joe Doe, et al. Mehrdeutige quellenverweise. Eingereicht bei IBM Journal of Research and Development, 1997.
- [3] Donald E. Knuth. *The T_EXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
- [4] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.
- [5] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Vanoirbeek and Coray [6], pages 261–273.
- [6] Christine Vanoirbeek and Giovanni Coray, editors. *EP92—Proceedings of Electronic Publishing*, '92, Cambridge, 1992. Cambridge University Press.

Bsp. 13-5-1

Wird plain durch abbrv ersetzt, so sieht das Ergebnis ganz ähnlich aus. Die Einträge sind jedoch kompakter, da Vornamen, Monate und vordefinierte Zeitschriftennamen (Tabelle 13.3 auf Seite 800) abgekürzt werden. Beispielsweise ergibt die Kurzform `ibmjrd` in der zweiten Quelle nun „IBM J. Res. Dev.“ anstelle von „IBM Journal of Research and Development“.

- [1] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.
- [2] M. Goossens, B. User, J. Doe, et al. Mehrdeutige quellenverweise. Eingereicht bei IBM J. Res. Dev., 1997.
- [3] D. E. Knuth. *The T_EXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
- [4] D. E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, Apr. 1989.
- [5] F. Mittelbach and C. Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Vanoirbeek and Coray [6], pages 261–273.
- [6] C. Vanoirbeek and G. Coray, editors. *EP92—Proceedings of Electronic Publishing*, '92, Cambridge, 1992. Cambridge University Press.

Bsp. 13-5-2

Der BibT_EX-Standardstil `unsrt` erzeugt die gleichen Ergebnisse wie `plain`, nur dass die Einträge nicht sortiert sondern in der Reihenfolge erscheinen, in der sie das erste Mal zitiert werden. Unter den Standardstilen gibt es keine Kombination aus `unsrt` und `abbrv`, nötigenfalls wäre es jedoch ein Leichtes,

die Unterschiede zwischen `plain` und `abbrv` in `unsrt` zu integrieren, so dass man einen neuen Stil erhält.

- [1] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.
- [2] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.
- [3] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Vanoirbeek and Coray [6], pages 261–273.
- [4] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
- [5] Michel Goossens, Ben User, Joe Doe, et al. Mehrdeutige quellenverweise. Eingereicht bei IBM Journal of Research and Development, 1997.
- [6] Christine Vanoirbeek and Giovanni Coray, editors. *EP92—Proceedings of Electronic Publishing*, '92, Cambridge, 1992. Cambridge University Press.

Bsp.
13-5-3

Der Standardstil `alpha` ist dem Stil `plain` ebenfalls sehr ähnlich, die Labels der Einträge werden hier jedoch aus dem Autornamen und dem Erscheinungsjahr gebildet. Das etwas seltsame Label des GNU-Handbuches rührt daher, dass der Eintrag ein `key`-Feld enthält, dessen erste drei Buchstaben als Teil des Labels herangezogen wurden. Beachtenswert ist auch das interessante Label für die Quelle mit mehr als drei Autoren. Die Publikationen werden hier nach dem Label sortiert, so dass das GNU-Handbuch erst an vierter Stelle steht.

- [GUD⁺97] Michel Goossens, Ben User, Joe Doe, et al. Mehrdeutige quellenverweise. Eingereicht bei IBM Journal of Research and Development, 1997.
- [Knu86] Donald E. Knuth. *The TeXbook*, volume A of *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.
- [Knu89] Donald E. Knuth. Typesetting Concrete Mathematics. *TUGboat*, 10(1):31–36, April 1989.
- [mak00] Free Software Foundation, Boston, Massachusetts. *GNU Make, A Program for Directing Recompilation*, 2000.
- [MR92] Frank Mittelbach and Chris Rowley. The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? In Vanoirbeek and Coray [VC92], pages 261–273.
- [VC92] Christine Vanoirbeek and Giovanni Coray, editors. *EP92—Proceedings of Electronic Publishing*, '92, Cambridge, 1992. Cambridge University Press.

Bsp.
13-5-4

Viele BibTeX-Stile erzeugen sparsamere oder umfangreichere Varianten der Standardstil-Layouts. Der `phaip`-Stil für die Zeitschriften des American

Institute of Physics beispielsweise setzt ein unsortiertes Layout (also in der Reihenfolge der Zitate) ein, lässt dabei jedoch die Artikelüberschriften aus, verwendet abgekürzte Autorennamen und kennzeichnet Herausgeber von Konferenzberichten durch eine andere Struktur. Man beachte, dass der Eintrag mit mehr als drei Autoren hier komprimiert wurde und somit nur noch den ersten Autor zeigt.

- [1] D. E. Knuth, *TUGboat* **10**, 31 (1989).
- [2] Free Software Foundation, Boston, Massachusetts, *GNU Make, A Program for Directing Recompilation*, 2000.
- [3] F. Mittelbach and C. Rowley, The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography?, In Vanoirbeek and Coray [6], pages 261–273.
- [4] D. E. Knuth, *The T_EXbook*, volume A of *Computers and Typesetting*, Addison-Wesley, Reading, MA, USA, 1986.
- [5] M. Goossens et al., Mehrdeutige quellenverweise, Eingereicht bei IBM J. Res. Dev., 1997.
- [6] C. Vanoirbeek and G. Coray, editors, *EP92—Proceedings of Electronic Publishing*, '92, Cambridge, 1992, Cambridge University Press.

Bsp.
13-5-5

Bei Stilen, die ein Autor-Jahr-Schema verwenden, verändert sich das Layout normalerweise viel drastischer. Zum Beispiel werden die Label üblicherweise unterdrückt, da ja nach Autoren nachgeschlagen wird. Der *chicago*-Stil etwa zeigt den oder die Autorennamen in abgekürzter Form (mit nachgestelltem Vornamen), gefolgt vom Erscheinungsjahr, das in Klammern steht. Außerdem ist hier noch eine andere Art zu sehen, wie man Herausgeber von Konferenzberichten darstellen kann, und im Englischen erhält man anstelle des Wortes „pages“ die Abkürzung „pp.“ Für dieses Beispiel wurde zur Unterstützung des Autor-Jahr-Schemas das *natbib*-Paket geladen.

- Free Software Foundation (2000). *GNU Make, A Program for Directing Recompilation*. Boston, Massachusetts: Free Software Foundation.
- Goossens, M., B. User, J. Doe, et al. (1997). Mehrdeutige quellenverweise. Eingereicht bei IBM Journal of Research and Development.
- Knuth, D. E. (1986). *The T_EXbook*, Volume A of *Computers and Typesetting*. Reading, MA, USA: Addison-Wesley.
- Knuth, D. E. (1989, April). Typesetting Concrete Mathematics. *TUGboat* 10(1), 31–36.
- Mittelbach, F. and C. Rowley (1992). The pursuit of quality: How can automated typesetting achieve the highest standards of craft typography? See Vanoirbeek and Coray (1992), pp. 261–273.
- Vanoirbeek, C. and G. Coray (Eds.) (1992). *EP92—Proceedings of Electronic Publishing*, '92, Cambridge. Cambridge University Press.

Bsp.
13-5-6

Als letztes Beispiel wird noch eine weitere Art von Layout angezeigt, die sich mithilfe des `jurabib`-Paketes erzielen lässt. Da hier mehr Anpassungen vorgenommen werden müssen, wird die Eingabe noch einmal angezeigt. Der Trick, mit dem hier die Überschrift unterdrückt wird, ist für echte Dokumente *nicht* geeignet, da der Leerraum um die Überschrift erhalten bliebe!

```
\usepackage[bibformat=ibidem]{jurabib}
\bibliographystyle{jurabib}
\jbuseidemhrule           % Linie erzeugen bei gleichem Autor
\renewcommand\refname{}   % Überschrift für Beispiel unterdrücken
\nocite{Knuth:TB10-1,GNUmake,MR-PQ,Knuth-CT-a,test97,LGC97}
\bibliography{tex}
```

In diesem Layout wird der Name des Autors durch eine Linie ersetzt, wenn er zuvor bereits aufgeführt wurde. Bei mehreren Autoren muss dazu die gesamte Liste übereinstimmen (vgl. die ersten beiden Einträge). Hier werden auch zum ersten Mal ISBN- und ISSN-Nummern angezeigt, sofern sie in den Einträgen vorhanden sind. Bei näherer Betrachtung lassen sich viele kleinere und größere Unterschiede entdecken. So behält dieses Layout als erstes die Schreibweise der Titel von Artikeln und Konferenzbeiträgen aus der Datenbank bei, anstatt sie in Kleinschreibung umzuwandeln. (Auf Seite 840 wird erklärt, wie man BibTeX-Stile verändern kann, damit sie sich so verhalten.)

Da das `jurabib`-Paket in erster Linie für Rechtstexte gedacht war, liegt hier einer der Stile vor, der keine vorgegebenen Texte für die Fachzeitschriften aus Tabelle 13.3 auf Seite 800 enthält. Entsprechend ist der zweite Eintrag unvollständig. In diesem Falle gibt BibTeX eine Warnung über den fehlenden Text aus. Man kann dann eine entsprechende Definition in der Datenbankdatei nachpflegen, oder je nach Vorliebe auch in einer separaten Datenbankdatei, die nur geladen wird, wenn man sie benötigt.

Goossens, Michel/Rahtz, Sebastian/Mittelbach, Frank: The L^AT_EX Graphics Companion: Illustrating Documents with T_EX and PostScript. Reading, MA, USA: Addison-Wesley Longman, 1997, Tools and Techniques for Computer Typesetting, xxi + 554, ISBN 0-201-85469-4

Goossens, Michel et al.: Mehrdeutige Quellenverweise. 1997, Eingereicht bei

Knuth, Donald E.: The T_EXbook. Volume A, Computers and Typesetting. Reading, MA, USA: Addison-Wesley, 1986, ix + 483, ISBN 0-201-13447-0

—— Typesetting Concrete Mathematics. TUGboat, 10 April 1989, Nr. 1, 31–36, ISSN 0896-3207

Free Software Foundation: GNU Make, A Program for Directing Recompilation. 2000

Mittelbach, Frank/Rowley, Chris: The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography? In **Vanoirbeek/Coray:** EP92, 261–273

Vanoirbeek, Christine/Coray, Giovanni, editors: EP92—Proceedings of Electronic Publishing, '92. Cambridge: Cambridge University Press, 1992

13.5.2 custom-bib – BibTeX-Stile leicht gemacht

Bisher wurde besprochen, wie man das Layout eines Literaturverzeichnisses mithilfe unterschiedlicher Bibliographiestile beeinflussen kann. Wird für eine Zeitschrift oder einen Verlag, für den man schreibt, ein bestimmter BibTeX-Stil empfohlen, so hat man alles, was man benötigt. Wahrscheinlicher ist jedoch, dass man eine ausführliche Richtlinie erhält, wie Literaturhinweise aussehen müssen, ohne Hinweis auf irgendeinen bestimmten BibTeX-Stil – vermutlich kennt der Verlag nicht einmal das Programm.

Eine Möglichkeit besteht natürlich darin, nach einem fertigen Stil zu suchen, der genau passt oder nur leicht angepasst werden muss (siehe Abschnitt 13.6.3). Angesichts der zahlreichen Varianten, die normalerweise für jedes setzerische Detail existieren, sind die Chancen, dass man einen geeigneten Stil findet, jedoch recht gering. Man betrachte nur einmal die folgenden neun gängigen Anforderungen an die Darstellung der Autorennamen:

<i>Anforderung</i>	<i>Beispiel</i>
Ganzer Name, Nachname zuletzt	Donald Ervin Knuth/Michael Frederick Plass
Ganzer Name, Nachname zuerst	Knuth, Donald Ervin/Plass, Michael Frederick
Initialen und Nachname	D. E. Knuth/M. F. Plass
Nachname mit Initialen	Knuth, D. E./Plass, M. F.
Nachname mit Initialen ohne Punkte	Knuth D E/Plass M F
Nachname mit verketteten Initialen	Knuth DE/Plass MF
Nachname mit Initialen ohne Leerraum	Knuth D.E./Plass M.F.
Nur erster Autor umgestellt mit Initialen	Knuth, D. E./M. F. Plass
Nur erster Autor umgestellt mit ganzen Namen	Knuth, Donald Ervin/Michael Frederick Plass

Tabelle 13.5: Anforderungen an die Namensdarstellung

Nimmt man noch Vorgaben für das Trennzeichen hinzu (z.B. Komma, Semikolon, Schrägstrich), für die Schrift der Autorennamen (also Serifenschrift, fett, Kapitälchen, kursiv usw.) und vielleicht noch eine unterschiedliche Schrift für Nach- und Vornamen, so erhält man schon allein für die Angabe von Autoren in einem Literaturverzeichnis über 500 verschiedene Stile. Diese Explosion von Kombinationen lässt sich natürlich nicht durch vorgefertigte Stile abdecken.

Angesichts dieser Problematik begann Patrick Daly, der Entwickler des natbib-Paketes, bereits 1993 mit der Arbeit an einem System, das anhand von Fragen, wie den zuvor gestellten (insgesamt über 70!), einen passenden BibTeX-Stil in Form einer den Antworten entsprechenden .bst-Datei erzeugen kann.

Das System arbeitet in zwei Schritten: Zunächst erfolgt die Sammelphase, in der interaktiv Fragen gestellt werden, und dann die Generierungsphase, in der anhand der erteilten Antworten ein BibTeX-Stil zusammengestellt wird. Beide Phasen basieren ausschließlich auf L^AT_EX, so dass sie ohne Zusatzprogramme auf jeder Plattform ausgeführt werden können.

Zum Sammeln der Antworten startet man unter L^AT_EX das Programm makebst.tex und beantwortet die gestellten Fragen. Die meisten Fragen erscheinen mit einer Auswahlliste verschiedener Antworten. Die voreingestellte

Antwort ist mit einem * gekennzeichnet und wird einfach durch Drücken der Eingabetaste ($\langle return \rangle$) ausgewählt. Die anderen Antwortmöglichkeiten werden durch Eingabe des Buchstabens, der in Klammern vor der jeweiligen Option steht, ausgewählt. Gibt man einen nicht angebotenen Buchstaben ein, so wird die voreingestellte Antwort übernommen.

Initialisieren des Systems

Es folgen die ersten Fragen, die eine Sonderrolle spielen, da mit ihrer Hilfe das System initialisiert wird. Es wird jeweils die empfohlene Antwort angezeigt.

Do you want a description of the usage? (NO)
(Möchten Sie eine Bedienungsanleitung? (Nein))

Gibt man y ein, so erhält man eine Beschreibung des (zuvor bereits erläuterten) Verfahrens; ansonsten hat die Frage keine Auswirkung.

Enter the name of the MASTER file (default=merlin.mbs)
(Geben Sie den Namen der MASTER-Datei ein (voreingestellt=merlin.mbs))

Die richtige Antwort ist hier die Eingabetaste ($\langle return \rangle$). Die voreingestellte Datei merlin.mbs ist zurzeit die einzig verfügbare Masterdatei für die Erstellung der Stile. Dies kann sich natürlich noch ändern.

Name of the final OUTPUT .bst file? (default extension=bst)
(Name der fertigen AUSGABE-.bst-Datei? (voreingestellte Erweiterung=bst))

Gefragt ist der Name der neuen $\text{\LaTeX}$ -Stildatei ohne Erweiterung, wie z.B. ttct (für die Schriftenreihe Tools and Techniques for Computer Typesetting). Als Ergebnis der ersten Phase erhält man dann eine Datei namens ttct.dbj, aus der in der zweiten Phase die $\text{\LaTeX}$ -Stildatei ttct.bst erzeugt wird.

Give a comment line to include in the style file.
 Something like for which journals it is applicable.
(Geben Sie eine Kommentarzeile für die Stildatei ein. Zum Beispiel für welche Zeitschriften sie gedacht ist.)

Hier kann man einen beliebigen Text eingeben. Man beachte jedoch, dass die Eingabetaste den Kommentar beendet. Die Zeile wird in die Ergebnisdateien übertragen und zeigt, für welchen Zweck der jeweilige $\text{\LaTeX}$ -Stil gedacht ist.

Do you want verbose comments? (NO)
(Möchten Sie ausführliche Kommentare? (Nein))

Durch die Antwort y wird der Kontext weiterer Fragen in folgender Form angezeigt:

```
<<STYLE OF CITATIONS:
...
>>STYLE OF CITATIONS:
(«STIL DER QUELLENVERWEISE: ... »STIL DER QUELLENVERWEISE:)
```


catalan	Katalanisch	italian	Italienisch
dansk	Dänisch	norsk	Norwegisch
dutch	Niederländisch	polski	Polnisch
esperant	Esperanto	portuges	Portugiesisch
finnish	Finnisch	slovene	Slowenisch
french	Französisch	spanish	Spanisch
german	Deutsch		

Tabelle 13.6: In custom-bib unterstützte Sprachen (Sommer 2003)

Hier muss jeder selbst entscheiden, ob diese Darstellung eine Hilfe ist. Die Voreinstellung zeigt diese Zusatzinformationen nicht an.

Name of language definition file (default=merlin.mbs)

(Name der Sprachdefinitionsdatei (voreingestellt=merlin.mbs))

Sprachunterstützung

Um einen BibTeX-Stil für eine nicht englische Sprache zu erzeugen, kann man hier den Namen der Sprache eingeben (also z.B. `german` für Deutsch). In Tabelle 13.6 sind die zurzeit unterstützten Sprachen aufgeführt. Andernfalls drückt man einfach die Eingabetaste.

Include file(s) for extra journal names? (NO)

(Datei(en) für besondere Namen von Fachzeitschriften einbinden? (Nein))

Mit `y` kann man vordefinierte Zeitschriftennamen für bestimmte Fachgebiete in den BibTeX-Stil aufnehmen. In diesem Fall wird man als Nächstes aufgefordert anzugeben, in welchen Dateien die vordefinierten Namen enthalten sind (wobei wiederum geeignete Voreinstellungen angeboten werden).

Damit ist die erste Gruppe von Fragen zur Initialisierung des Systems abgeschlossen. Als Nächstes folgen zahlreiche Fragen zum Layout und zu funktionellen Einzelheiten. Diese lassen sich in drei Kategorien einteilen:

Zitierschema Die hier getroffene Wahl wirkt sich auf die weiteren Fragen aus. Zum Autor-Jahr-Schema werden z.B. andere Fragen gestellt als zum numerischen Schema.

Erweiterungen Diese Fragen klären, ob zusätzliche BibTeX-Felder unterstützt werden sollen, also ob z.B. ein `url`-Feld integriert wird.

Typographische Einzelheiten Hier wählt man aus, wie die einzelnen Teile der Literatureinträge formatiert werden sollen. Einige der Wahlmöglichkeiten sind wiederum vom Zitierschema abhängig.

Es ist zwar möglich, seine Auswahl in der zweiten Bearbeitungsphase noch einmal zu ändern (oder ganz von vorne zu beginnen). Am besten sollte man sich jedoch bereits zu Beginn des interaktiven Programms darüber im Klaren sein, welches Zitierschema und welche Erweiterungen man wünscht. Die typographischen Einzelheiten lassen sich dagegen auch im Nachhinein wesentlich leichter ändern, sollte dies nötig sein. Daher werden die zentralen Fragen hier etwas ausführlicher beleuchtet.

Auswählen des Zitierschemas

Das Zitierschema wird durch Beantwortung der folgenden Frage festgelegt:

STYLE OF CITATIONS:

- (*) Numerical as in standard $\text{\LaTeX}$
- (a) Author-year with some non-standard interface
- (b) Alpha style, Jon90 or JWB90 for single or multiple authors
- (o) Alpha style, Jon90 even for multiple authors
- (f) Alpha style, Jones90 (full name of first author)
- (c) Cite key (special for listing contents of bib file)

(STIL DER QUELLENVERWEISE: () Numerisch wie in Standard- $\text{\LaTeX}$ (a) Autor-Jahr mit nicht standardgemäßer Schnittstelle (b) Alpha-Stil, Jon90 bzw. JWB90 bei einzelnen bzw. mehreren Autoren (o) Alpha-Stil, Jon90 selbst bei mehreren Autoren (f) Alpha-Stil, Jones90 (ganzer Name des ersten Autors) (c) Zitierschlüssel (speziell um die Inhalte der bib-Datei aufzulisten))*

Die Auswahl „numerical“ ist voreingestellt. Um einen Stil zum Autor-Jahr-Schema zu erzeugen, wählt man a (und ignoriert den Hinweis auf die „nicht standardgemäße Schnittstelle“). Verweise im Alpha-Stil erhält man mit b, o oder f, je nach bevorzugter Labelform. Die Auswahl c ist nur für Stile interessant, mit deren Hilfe $\text{\LaTeX}$ -Datenbanken dargestellt werden sollen; sie eignet sich also nicht zum Erzeugen von Literaturverzeichnissen.

Im Falle der voreingestellten Antwort (also numerisches Zitierschema) lautet die nächste Frage:

HTML OUTPUT (if non author-year citations)

- (*) Normal $\text{\LaTeX}$ output
- (h) Hypertext output, in HTML code, in paragraphs
- (n) Hypertext list with sequence numbers
- (k) Hypertext with keys for viewing databases

(HTML AUSGABE (sofern keine Autor-Jahr-Verweise) () Normale $\text{\LaTeX}$ -Ausgabe (h) Hypertext-Ausgabe, im HTML-Code, in Absätzen (n) Hypertext-Liste mit laufender Nummer (k) Hypertext mit Schlüsseln um Datenbanken zu sichten)*

Hier ist die Voreinstellung zu wählen. Alle anderen Möglichkeiten erzeugen $\text{\LaTeX}$ -Stile, die HTML-Daten ausgeben (welche zunächst noch weiter bearbeitet werden müssen, bevor sie im Browser betrachtet werden können). Diese Funktionen befinden sich noch in der Erprobung.

Bei der Wahl eines Autor-Jahr-Zitierschemas (also a) muss man als Nächstes entscheiden, auf welcher unterstützenden $\text{\LaTeX}$ -Schnittstelle dieses aufsetzen soll:

AUTHOR--YEAR SUPPORT SYSTEM (if author-year citations)

- (*) Natbib for use with natbib v5.3 or later
- (o) Older Natbib without full authors citations
- (l) Apalike for use with apalike.sty
- (h) Harvard system with harvard.sty
- (a) Astronomy system with astron.sty
- (c) Chicago system with chicago.sty

- (n) Named system with `named.sty`
- (d) Author-date system with `authordate1-4.sty`

(AUTOR-JAHR-UNTERSTÜTZUNG (bei Autor-Jahr-Verweisen) () Natbib mit natbib ab Version 5.3 (o) ältere Natbib-Versionen ohne ausführliche Autorenverweise (l) Apa-ähnlich mit `apalike.sty` (h) Harvard-Schema mit `harvard.sty` (a) Astronomy-Schema mit `astron.sty` (c) Chicago-Schema mit `chicago.sty` (n) System mit Namen mit `named.sty` (d) Autor-Jahr-Schema mit `authordate1-4.sty`)*

Die voreingestellte Auswahl, `natbib`, ist normalerweise die Beste. Sie bietet alle in den Abschnitten 12.3.2 und 12.4.1 beschriebenen Möglichkeiten. Man sollte *nicht* die Option `o` wählen. Verfügt man bereits über Dokumente, die etwa Verweisbefehle aus dem `harvard`-Paket verwenden (siehe Beispiel 12-3-4 auf Seite 725), dann wäre die Option `h` am besten geeignet. Aus dem gleichen Grund können unter bestimmten Umständen die anderen Möglichkeiten die richtige Wahl sein. Um möglichst portierbare Dokumente zu erhalten, ist jedoch immer `natbib` zu bevorzugen. Man sollte auch beachten, dass einige der anderen in der Auswahl genannten Pakete in gängigen $\text{\LaTeX}$ -Distributionen nicht mehr enthalten sind.

Festlegen der unterstützten Erweiterungen

Abgesehen von den Publikationstypen (Tabelle 13.1 auf Seite 791) und Feldern (Tabelle 13.2) des $\text{\BibTeX}$ -Standards kann man `makebst.tex` auch anweisen, zusätzliche Felder als optionale Datenbankfelder zu unterstützen. Diese werden dann ebenfalls verwendet, sofern sie in der Datenbank vorhanden sind. Einige dieser Erweiterungen sind in den Voreinstellungen deaktiviert, obwohl es fast immer sinnvoll wäre, sie in die $\text{\BibTeX}$ -Stildatei zu integrieren.

LANGUAGE FIELD

- (*) No language field
- (l) Add language field to switch hyphenation patterns temporarily

(SPRACHENFELD () Kein Sprachenfeld (l) Sprachenfeld language hinzufügen, um die Silbentrennung zeitweise umzuschalten))*

Die Antwort `l` ist eine große Hilfe bei der korrekten Darstellung von Titeln in anderen Sprachen. Beispiel 12-5-6 auf Seite 744 zeigt die möglichen Probleme und erläutert, wie sie sich mit einem vorhandenen `Language`-Feld lösen lassen (siehe Beispiel 12-5-36 auf Seite 760). Entsprechend empfiehlt es sich hier, von den Voreinstellungen abzuweichen.

ANNOTATIONS:

- (*) No annotations will be recognized
- (a) Annotations in `annotate` field or in `.tex` file of citekey name

(KOMMENTARE: () Kommentare werden nicht erkannt (a) Kommentare in `annotate`-Feldern oder einer `.tex`-Datei mit dem Namen des Zitierschlüssels (`citekey`))*

Wählt man a, so unterstützt die zu erzeugende .bst-Datei ein `annotate`-Feld sowie Kommentare, die in Dateien mit dem Namen des `<citekey>`s und der Erweiterung .tex gespeichert sind. Anders als im jurabib-Paket, das ebenfalls über eine solche Funktion verfügt, lässt sich diese Möglichkeit jedoch nicht durch eine Paketoption ein- oder ausschalten. Da man diese Funktion sicherlich je nach Dokument aktivieren oder deaktivieren möchte, kann es besser sein, zwei separate BibTeX-Stile zu erzeugen, die sich lediglich in dieser Hinsicht voneinander unterscheiden.

Das nicht zum Standard gehörige Feld `eid` (elektronische Kennung) wird automatisch von allen erzeugten Stilen unterstützt. Die Felder `doi`, `isbn` und `issn` sind in den Voreinstellungen ausgewählt, lassen sich jedoch abwählen. Insbesondere zur Unterstützung des REVTeX-Paketes der American Physical Society lassen sich noch eine Reihe weiterer Felder hinzufügen.

Und schließlich kann man noch einstellen, ob URLs unterstützt werden, indem man eine andere als die folgende voreingestellte Option wählt.

```
URL ADDRESS: (without REVTeX fields)
(*) No URL for electronic (Internet) documents
(u) Include URL as regular item block
(n) URL as note
(l) URL on new line after rest of reference
```

(URL-ADRESSE: (ohne REVTeX-Felder) () Keine URL für elektronische (Internet-) Dokumente (u) URL als regulären Datenblock einschließen (n) URL als Anmerkung (l) URL in neuer Zeile nach dem übrigen Eintrag)*

Die Autoren dieses Buches empfehlen URLs zu unterstützen, da Verweise auf elektronische Dokumente immer selbstverständlicher werden. In der Bibliographie wird die URL durch die Tags `\urlprefix\url{feldwert}` gekennzeichnet, wobei für beide Befehle voreingestellte Definitionen existieren. Durch Laden des `url`-Paketes lässt sich ein besserer Zeilenumbruch erzielen.

Eine der letzten Fragen bietet die folgende Auswahl:

```
COMPATIBILITY WITH PLAIN TEX:
(*) Use LaTeX commands which may not work with Plain TeX
(t) Use only Plain TeX commands for fonts and testing
```

(Kompatibilität zu PLAIN-TEX: () Auch Plain-TeX fremde LaTeX-Befehle verwenden (t) Für Schriften und zum Testen nur Plain-TeX-Befehle verwenden)*

Hier sollte man die Voreinstellung unbedingt beibehalten! L^AT_EX 2_ε gibt es nun seit gut zehn Jahren und NFSS sollte mittlerweile den Weg in jedes Wohnzimmer gefunden haben. Davon abgesehen gehören die Plain-TeX-Befehle (wie `\rm`, `\bf` usw.) offiziell schon gar nicht mehr zu L^AT_EX. Einige Dokumentenklassen definieren sie noch (aus Gründen der Kompatibilität zu L^AT_EX 2.09) – andere jedoch nicht. Wählt man also die veraltete Syntax, dann funktioniert der BibTeX-Stil möglicherweise unter bestimmten Umständen nicht einwandfrei.¹

¹Achtung: In älteren Versionen lautete die Frage „NEW FONT SELECTION SCHEME“ und die veralteten Befehle waren voreingestellt. Hier ist also Vorsicht geboten.

Man beachte, dass Fragen zu den Erweiterungen mit solchen zu typographischen Einzelheiten gemischt auftreten und nicht notwendigerweise in der hier vorgestellten Reihenfolge erscheinen.

Festlegen der typographischen Einzelheiten

Die verbleibenden Fragen (derer es noch viele gibt) betreffen typographische Einzelheiten, wie etwa die Formatierung von Autorennamen, Darstellung von Zeitschriftendaten und viele weitere Themen. Als Beispiel sei hier der Fragenblock gezeigt, der sich mit der Formatierung von Artikelüberschriften befasst:

TITLE OF ARTICLE:

- (*) Title plain with no special font
- (i) Title italic (\em)
- (q) Title and punctuation in single quotes ('Title,' ..)
- (d) Title and punctuation in double quotes ("Title,'" ..)
- (g) Title and punctuation in guillemets (<<Title,>> ..)
- (x) Title in single quotes ('Title', ..)
- (y) Title in double quotes ("Title'", ..)
- (z) Title in guillemets (<<Title>>, ..)

(ARTIKELÜBERSCHRIFT: () Schlichte Überschrift ohne besondere Schrift (i) Überschrift kursiv (q) Überschrift und Satzzeichen in Anführungsstrichen ('Titel', ...) (d) Überschrift und Satzzeichen in doppelten Anführungsstrichen ("Titel", ...) (g) Überschrift und Satzzeichen in Guillemets («Titel», ...) (x) Überschrift in Anführungsstrichen ('Titel', ...) (y) Überschrift in doppelten Anführungsstrichen ("Titel'", ...) (z) Überschrift in Guillemets («Titel», ...))*

Eine falsche Wahl an dieser Stelle ist kein Grund zur Verzweiflung. Dieser Fehler lässt sich wie im Folgenden beschrieben in der zweiten Verarbeitungsphase beheben.

Erzeugen des BibTeX-Stils aus den gegebenen Antworten

Hat man `makebst.tex` unter $\text{\LaTeX}$ ausgeführt und all diese Fragen beantwortet, so erhält man als Ergebnis eine neue Datei mit der Erweiterung `.dbj`. Diese enthält alle gewählten Optionen in einer besonderen Form, die von `DOCSTRIP` verarbeitet werden kann, mit dessen Hilfe dann der fertige BibTeX-Stil erzeugt wird. (Das Programm `DOCSTRIP` ist in Abschnitt 14.2 beschrieben.) Technisch ausgedrückt enthält eine Masterdatei für BibTeX-Stile (voreingestellt ist `merlin.mbs`) verschiedene Code-Abschnitte, die nun in Abhängigkeit von den `DOCSTRIP`-Optionen ausgewählt werden. Durch die zuvor besprochene interaktive Auswahl der Menü-Einträge, werden einige dieser Code-Abschnitte aktiviert und ermöglichen so die erforderliche Anpassung.

Hat man bei der Frage nach der neuen `.bst`-Datei `ttct` eingegeben, so verfügt man nun über eine Datei namens `ttct.dbj`. Um daraus den fertigen BibTeX-Stil zu erzeugen, muss man damit jetzt nur noch den folgenden Befehl eingeben:

```
latex ttct.dbj
```

Der Inhalt der in der ersten Phase erzeugten .dbj-Dateien ist gut dokumentiert und so dargestellt, das er sich auf recht einfache Weise noch weiter anpassen lässt. Angenommen man hat bei der Frage nach den Artikelüberschriften auf der vorherigen Seite die Antwort y ausgewählt (also Anführungszeichen um die Überschrift), man benötigt aber eigentlich die Option d (Anführungszeichen um Überschrift und Satzzeichen). Dann muss man nur die .dbj-Datei mit einem Texteditor öffnen und nach dem Abschnitt suchen, der die Artikelüberschriften einstellt:

```
%-----
%TITLE OF ARTICLE:
%   %: (def) Title plain
% tit-it,%: Title italic
% tit-qq,qt-s,%: Title and punctuation in single quotes
% tit-qq,%: Title and punctuation in double quotes
% tit-qq,qt-g,%: Title and punctuation in guillemets
% tit-qq,qt-s,qx,%: Title in single quotes
%   tit-qq,qx,%: Title in double quotes
% tit-qq,qt-g,qx,%: Title in guillemets
%-----
```

Um das Verhalten zu ändern, muss man lediglich das Kommentarzeichen vor der gewünschten Zeile entfernen und vor der zurzeit gewählten Option ein Kommentarzeichen einfügen:

```
%-----
%TITLE OF ARTICLE:
%   %: (def) Title plain
% tit-it,%: Title italic
% tit-qq,qt-s,%: Title and punctuation in single quotes
%   tit-qq,%: Title and punctuation in double quotes
% tit-qq,qt-g,%: Title and punctuation in guillemets
% tit-qq,qt-s,qx,%: Title in single quotes
% tit-qq,qx,%: Title in double quotes
% tit-qq,qt-g,qx,%: Title in guillemets
%-----
```

Dann verarbeitet man die Datei noch einmal mit $\text{\LaTeX}$, um den aktualisierten BibTeX-Stil zu erhalten.

13.6 Die BibTeX-Programmiersprache

Dieser Abschnitt bietet eine komprimierte Einführung in die Programmiersprache der BibTeX-Stildateien. Die Informationen sollten ausreichen, um kleine Änderungen an bestehenden Stildateien vorzunehmen. Weitergehende Informationen sind in dem von Oren Patashnik geschriebenen Artikel „Designing BibTeX Styles“ [138] nachzulesen.

Die BibTeX-Stile verwenden eine Postfix-Stapelverarbeitungssprache (wie PostScript), um BibTeX mitzuteilen, wie die Einträge für das Literaturverzeichnis zu formatieren sind. Die Sprache verfügt über zehn Befehle zur Verarbeitung ihrer Objekte: Konstanten, Variablen, Funktionen, den Stapel und die Eintragsliste. Diese Befehle sind in Tabelle 13.7 auf Seite 838 beschrieben.

BibTeX kennt zwei Arten von Funktionen: systemimmanente Funktionen, die von BibTeX selbst bereitgestellt werden (siehe Tabelle 13.8 auf Seite 839); und Benutzerfunktionen, die entweder mithilfe des MACRO- oder des FUNCTION-Befehls definiert werden.

Zwischen zwei Anführungszeichen, die konstante Zeichenfolgen (so genannte Stringkonstanten) begrenzen, kann man alle druckbaren ASCII-Zeichen verwenden. Obwohl BibTeX ansonsten Groß- und Kleinschreibung ignoriert, beachtet es die Schreibweise innerhalb eines Strings. Auch Leerzeichen in Stringkonstanten werden berücksichtigt und die Konstanten dürfen nicht umbrochen werden.

Variablen- und Funktionsnamen dürfen nicht mit einer Ziffer beginnen und keines der zehn unzulässigen Zeichen von Seite 798 enthalten. BibTeX ignoriert die Groß- und Kleinschreibung in Variablen-, Funktions- und Makronamen.

Konstanten und Variablen können als Integer (ganze Zahlen) oder String (Zeichenfolge) vorliegen; die booleschen Werte wahr und falsch werden durch die Integerwerte 1 bzw. 0 dargestellt.

Es gibt drei Arten von Variablen:

Globale Variablen Bei den globalen Variablen handelt es sich entweder um Integer- oder Stringwerte, die durch einen INTEGERS- bzw. STRINGS-Befehl deklariert werden.

Eintragsvariablen Hier handelt es sich um Integer- oder Stringvariablen, die durch den ENTRY-Befehl deklariert werden. Diese Variablen speichern einen Wert je Quelleneintrag, der aus den BibTeX-Datenbanken eingelesen wird.

Felder Diese Variablen sind nur lesbare Stringwerte, in denen Informationen aus der Datenbankdatei gespeichert werden. Ihre Werte werden durch den Befehl READ gesetzt. Genau wie bei den Eintragsvariablen existiert auch hier ein Wert je Eintrag.

13.6.1 BibTeX-Befehle und systemimmanente Funktionen

Die zehn BibTeX-Befehle sind in Tabelle 13.7 auf Seite 838 kurz beschrieben. Die Namen erscheinen zwar in Großbuchstaben, BibTeX ignoriert aber die Groß- oder Kleinschreibung.

Es empfiehlt sich (ist aber nicht zwingend), zwischen Befehlen mindestens eine Leerzeile, innerhalb eines Befehls aber keine Leerzeile zu lassen. Durch diese Konvention kann BibTeX syntaktisch falsche Einträge leichter erkennen und überspringen.

Tabelle 13.8 auf Seite 839 gibt einen kurzen Überblick über die 37 systemimmanenten Funktionen in BibTeX (nähere Einzelheiten siehe [138]). Jede systemimmanente Funktion, in deren Name ein Buchstabe vorkommt, endet mit einem \$-Zeichen.

13.6.2 Der Dokumentationsstil `btxbst.doc`

Oren Patashniks Standard-BibTEX-Stildateien `abbrv`, `alpha`, `plain` und `unsrt` basieren auf einer generischen Datei namens `btxbst.doc`, die sehr gut dokumentiert ist, und die einen detaillierten Einblick in die Arbeitsweise der BibTEX-Stildateien gibt.

Bei den Standardstilen gibt es zwei grundlegende Arten der Formatierung von Labels: *alphabetisch*, wie [Lam84], und *numerisch*, wie [34]. Literaturhinweise lassen sich auf drei Arten anordnen:

Sortierte, alphabetische Labels Alphabetisch sortiert, zunächst nach Zitierlabel, dann nach Autor(en) (oder dem entsprechenden Ersatzfeld), dann nach Erscheinungsjahr und Titel.

Sortierte, numerische Labels Alphabetisch sortiert, zunächst nach Autor(en) (oder dem entsprechenden Ersatzfeld), dann nach Erscheinungsjahr und Titel.

Unsortiert In der Reihenfolge angeordnet, in der die Quellen im Text zitiert werden.

Der grundlegende Ablauf in einer Stildatei wird durch die folgenden Kommandozeilen gesteuert, die sich am Ende der Datei `btxbst.doc` befinden:

```
EXECUTE {begin.bib}           % Präambel und \begin{thebibliography} erzeugen
EXECUTE {init.state.consts}    % Initialisieren der Statuskonstanten
ITERATE {call.type$}          % Durchlaufen der Einträge und Erstellen der Ausgabe
EXECUTE {end.bib}             % Schreiben des \end{thebibliography}-Befehls
```

Diese Befehle werden in den Tabellen 13.7 und 13.8 erläutert.

Der Programmcode einer Stildatei beginnt mit der Deklaration der verfügbaren Felder durch die `ENTRY`-Deklaration und mit den Stringvariablen, die für die Erstellung des gedruckten Labels verwendet werden sollen.

Danach folgen einige Funktionen für die Formatierung der Grundbestandteile eines Eintrags. Es gibt Funktionen für jedes der Basisfelder. Die Funktion `format.names` unterteilt Namen in ihre „First von Last, Junior“-Teile, trennt sie durch Kommas und setzt ein „and“ vor den letzten Namen (bzw. beendet die Aufzählung mit „et al.“, wenn bei mehreren Autoren als Letztes „others“ genannt ist). Die Funktion `format.authors` verarbeitet Autorendaten und `format.editors` bearbeitet Herausgeberdaten (sie hängt den passenden Titel an: „, editor“ oder „, editors“).

Der nächste Teil der Datei enthält alle die Funktionen, welche die verschiedenen Publikationstypen definieren, die in einer `.bib`-Datei akzeptiert werden, also Funktionen wie `article` und `book`. Diese Funktionen erzeugen die eigentliche Ausgabe, die für den jeweiligen Eintrag in die `.bbl`-Datei übernommen wird. Sie müssen dem `READ`-Befehl vorangehen. Wer Stile entwickelt, sollte außerdem die Funktion `default.type` für unbekannte Publikationstypen vorsehen.

Jede Eintragsfunktion beginnt mit dem Aufruf von `output.bibitem`. Dadurch werden der `\bibitem`-Befehl und seine Argumente in die `.bbl`-Datei geschrieben.

ENTRY <i>{feld-liste} {integer-variablen-liste} {string-variablen-liste}</i> Deklariert die Feld- und Eintragsvariablen; $\text{\LaTeX}$ deklariert automatisch ein zusätzliches Feld <code>crossref</code> für Querverweise, sowie eine zusätzliche String-Eintragsvariable <code>sort.key</code>, die vom <code>SORT</code>-Befehl verwendet wird. Je Stildatei darf nur ein <code>ENTRY</code>-Befehl vorhanden sein. Beispielsweise enthalten die Stile <code>alpha</code> und <code>plain</code> jeweils folgende Angaben: <pre>ENTRY { address author booktitle ... } {} { label extra.label sort.label } ENTRY { address author booktitle ... } {} { label }</pre>
EXECUTE <i>{funktionsname}</i> Führt eine einzelne Funktion aus. <pre>EXECUTE {begin.bib}</pre>
FUNCTION <i>{funktionsname} {definition}</i> Definiert eine neue Funktion; diese lässt sich nicht außerhalb der Stildatei verändern. <pre>FUNCTION {end.bib} { newline\$ "\end{thebibliography}" write\$ newline\$ }</pre>
MACRO <i>{makroname} {definition}</i> Definiert ein String-Makro; dieses lässt sich außerhalb der Stildatei verändern. <pre>MACRO {feb} {"February"}</pre>
INTEGERS <i>{globale-integer-variablen-liste}</i> Deklariert globale Integervariablen. <pre>INTEGERS { longest.label.width last.extra.num }</pre>
STRINGS <i>{globale-string-variablen-liste}</i> Deklariert globale Stringvariablen. <pre>STRINGS { longest.label last.sort.label next.extra }</pre>
ITERATE <i>{funktionsname}</i> Führt eine einzelne Funktion einmal je Listeneintrag in der aktuellen Reihenfolge der Liste aus. <pre>ITERATE {longest.label.pass}</pre>
REVERSE <i>{funktionsname}</i> Führt eine einzelne Funktion einmal je Listeneintrag in der umgekehrten Reihenfolge der Liste aus. <pre>REVERSE {reverse.pass}</pre>
READ Extrahiert für jeden Listeneintrag die Feldwerte aus der Datenbankdatei; je Stildatei darf nur ein <code>READ</code>-Befehl vorhanden sein. Die <code>ENTRY</code>- und <code>MACRO</code>-Befehle müssen dem <code>READ</code>-Befehl vorausgehen.
SORT Sortiert die Liste der Einträge anhand der Werte der String-Eintragsvariablen <code>sort.key</code>.

Tabelle 13.7: Befehle der $\text{\LaTeX}$ -Programmiersprache

$T_1 \ T_2 \ >$	(T)	1 (wenn $T_1 > T_2$) oder 0 (ansonsten)
$T_1 \ T_2 \ <$	(T)	1 (wenn $T_1 < T_2$) oder 0 (ansonsten)
$T_1 \ T_2 \ =$	(T)	1 (wenn $T_1 = T_2$) oder 0 (ansonsten)
$S_1 \ S_2 \ =$	(T)	1 (wenn $S_1 = S_2$) oder 0 (ansonsten)
$T_1 \ T_2 \ +$	($T_1 + T_2$)	Zwei Integer addieren
$T_1 \ T_2 \ -$	($T_1 - T_2$)	Zwei Integer voneinander subtrahieren
$S_1 \ S_2 \ *$	($S_1 S_2$)	Zwei Strings verketteten
$\mathcal{L} \ \mathcal{V} \ :=$		$\mathcal{V}$ den Wert von $\mathcal{L}$ zuweisen
$S \ \text{add.period}\$$	(S)	Punkt an einen String anhängen, es sei denn dieser endet auf '.', '?' oder '!'
$\text{call.type}\$$		Funktion ausführen, deren Name einem Publikationstyp entspricht (z.B. book)
$S \ \text{"t"} \ \text{change.case}\$$	(S)	S in Kleinbuchstaben umwandeln, außer am Anfang
$S \ \text{"l"} \ \text{change.case}\$$	(S)	S immer in Kleinbuchstaben umwandeln
$S \ \text{"u"} \ \text{change.case}\$$	(S)	S immer in Großbuchstaben umwandeln
$S \ \text{chr.to.int}\$$	(T)	Einzelnes Stringzeichen in ASCII übersetzen
$\text{cite}\$ \ (\text{zitier-string})$		$\backslash\text{cite}$ -Befehlsargument zum Stapel hinzufügen (push)
$\mathcal{L} \ \text{duplicate}\$$	($\mathcal{L} \ \mathcal{L}$)	Eintrag duplizieren
$\mathcal{L} \ \text{empty}\$$	(T)	1 (wenn $\mathcal{L}$ fehlendes Feld oder leerer String ist) oder 0 (ansonsten)
$S_1 \ T \ S_2 \ \text{format.name}\$$	(S)	T Namen S_1 entsprechend der Namenskonventionen S_2 formatieren
$T \ \mathcal{F}_1 \ \mathcal{F}_2 \ \text{if}\$$		$\mathcal{F}_1$ ausführen, wenn $T > 0$, sonst $\mathcal{F}_2$ ausführen
$T \ \text{int.to.chr}\$$	(S)	Integer in Zeichen mittels ASCII-Zeichentabelle übersetzen
$T \ \text{int.to.str}\$$	(S)	String-Entsprechung des Integers zum Stapel hinzufügen
$\mathcal{L} \ \text{missing}\$$	(T)	1 (wenn $\mathcal{L}$ fehlendes Feld ist) oder 0 (ansonsten)
$\text{newline}\$$		Neue Zeile in der .bb1-Datei beginnen
$S \ \text{num.names}\$$	(T)	Anzahl der Namen in S
$\mathcal{L} \ \text{pop}\$$		Oberstes Stapелеlement entfernen
$\text{preamble}\$$	(S)	Verkettung aller @preamble-Strings, die aus Datenbankdateien ausgelesen wurden, dem Stapel hinzufügen
$S \ \text{purify}\$$	(S)	Nicht alphanumerische Zeichen entfernen
$\text{quote}\$$	(S)	Buchstabenstring in Anführungszeichen dem Stapel hinzufügen
$\text{skip}\$$		Nichts tun
$\text{stack}\$$		Gesamten Stapel entfernen und ausgeben
$S \ T_1 \ T_2 \ \text{substring}\$$	(S)	Teilstring von S , der bei T_1 beginnt und eine Länge von T_2 hat
$\mathcal{L}_1 \ \mathcal{L}_2 \ \text{swap}\$$	($\mathcal{L}_2 \ \mathcal{L}_1$)	Literale tauschen
$S \ \text{text.length}\$$	(T)	Anzahl der „Text“-Zeichen
$S \ T \ \text{text.prefix}\$$	(S)	Erste T Zeichen von S
$\mathcal{L} \ \text{top}\$$		Oberes Element des Stapels entfernen und ausgeben
$\text{type}\$$	(S)	Publikationstyp des aktuellen Eintrags zum Stapel hinzufügen (z.B. book oder "" falls unbekannt)
$S \ \text{warning}\$$		Oberes (String-) Literal des Stapels entfernen und mit einer Warnung ausgeben
$\mathcal{F}_1 \ \mathcal{F}_2 \ \text{while}\$$		$\mathcal{F}_2$ ausführen, solange für den Resultwert T der Funktion $\mathcal{F}_1$ gilt $T > 0$
$S \ \text{width}\$$	(T)	Breite von S (in TeX-Einheiten) dem Stapel hinzufügen
$S \ \text{write}\$$		S in Ausgabepuffer schreiben

Tabelle 13.8: Systemimmanente Funktionen der BibTeX-Programmiersprache

Vor den systemimmanenten Funktionen steht jeweils die Variable, die sie vom Stapel entfernen. Sofern sie ein Ergebnis auf dem Stapel hinterlassen, ist dieses in Klammern angezeigt. Ein „Literal“ $\mathcal{L}$ ist ein Element des Stapels. Es kann ein Integer T , ein String S , eine Variable $\mathcal{V}$, eine Funktion $\mathcal{F}$ oder ein besonderer Wert sein, der ein fehlendes Feld anzeigt. Wenn das entfernte Literal den falschen Datentyp hat, meldet BibTeX einen Fehler und fügt dem Stapel den Integer 0 oder einen „null“-String hinzu, je nachdem, welchen Datentyp die Funktion ausgibt.

Danach werden die verschiedenen Felder formatiert und von der Funktion `output` oder `output.check` ausgegeben, die auch die Ausgabe von Trennzeichen (Kommas, Punkten, `\newblocks`) nach Bedarf steuert. Schließlich wird `fin.entry` aufgerufen, um den Satzendeppunkt hinzuzufügen und damit den Eintrag zu beenden.

Der nächste Abschnitt der `btxbst.doc`-Datei enthält Definitionen für die Monatsnamen und für bestimmte gängige Fachzeitschriften. Je nach Stil können vollständige oder abgekürzte Namen verwendet werden. Auf diese Definitionen folgt der `READ`-Befehl, welcher die Einträge aus der `.bib`-Datei ausliest.

Danach werden die Labels für die bibliographischen Einträge zusammengesetzt. Welche Felder für den ersten Teil eines Labels verwendet werden, hängt dabei vom jeweiligen Publikationstyp ab.

Anschließend werden die Labels für die Sortierung vorbereitet. Bei der Sortierung wird der Sortierschlüssel für jeden Eintrag von der `presort`-Funktion berechnet. Bei alphabetischen Labels müssen eventuell zusätzliche Buchstaben (a, b usw.) angehängt werden, um eine einheitliche Sortierreihenfolge zu erzielen. Dazu sind zwei weitere Sortierdurchläufe erforderlich. Bei numerischen Labels kann man entweder die sortierte oder die ursprüngliche Reihenfolge verwenden. In beiden Fällen muss man für die `thebibliography`-Umgebung das längste Label im Auge behalten. Schließlich wird die `.bbl`-Datei geschrieben, indem die Einträge durchlaufen werden und für jeden einzelnen die Funktion `call.type$` ausgeführt wird.

13.6.3 Kleine Änderungen an Stildateien

Häufig ist es notwendig, kleinere Änderungen an einer bestehenden Stildatei vorzunehmen, um den Anforderungen eines bestimmten Verlages zu genügen. Zunächst wird gezeigt, wie man eine (manchmal unerwünschte) Standardfunktion von `BBTEX`-Stilen ausschalten kann: das Umwandeln von Titeln in Kleinbuchstaben. Meistens möchte man die Titel wohl in der Form beibehalten, wie sie eingegeben wurden. Zu diesem Zweck lässt sich eine Variante des Stils `unsrt` erstellen. Im Beispiel erhält diese Variante den Namen `myunsrt`, da sie sich vom ursprünglichen Stil unterscheidet. Für andere Stile kann ein ähnliches Verfahren verwendet werden.

Ein Blick in die Tabelle 13.8 auf der vorherigen Seite lässt schon annehmen, dass die Funktion `change.case$` für die Änderung der Groß- oder Kleinschreibung verantwortlich ist. Mithilfe eines Editors findet man bei der Suche nach dem genannten String schnell heraus, dass die Funktion `format.title` verändert werden muss. Diese Funktion sieht vor und nach der Anpassung folgendermaßen aus:

```
FUNCTION {format.title}
{ title empty$
  { "" }
  { title "t" change.case$ }
  if$
}
```

Vor der Anpassung

```
FUNCTION {format.title}
{ title empty$
  { "" }
  { title } % <== modifiziert
  if$
}
```

Nach der Anpassung

Anhand der Tabelle 13.8 auf Seite 839 kann man die Logik der Funktion und den Ersetzungsvorgang nachvollziehen.

Auch die Funktion `format.edition` muss auf ähnliche Weise abgeändert werden. Hier kann man auf die innere „if“-Anweisung verzichten, da die resultierenden Fälle identisch wären.

<pre> FUNCTION {format.edition} { edition empty\$ { "" } { output.state mid.sentence = { edition "l" change.case\$ " edition" * } { edition "t" change.case\$ " edition" * } if\$ } if\$ } </pre> <i>Vor der Anpassung</i>	<pre> FUNCTION {format.edition} { edition empty\$ { "" } { edition " edition" * } if\$ } </pre> <i>Nach der Anpassung</i>

In den Funktionen `format.chapter.pages`, `format.thesis.type` und `format.tr.number` müssen ähnliche Änderungen vorgenommen werden.

Hinzufügen neuer Felder

Manchmal möchte man vielleicht ein zusätzliches Feld verwenden. Als Beispiel wird hier ein `annotate`-Feld hinzugefügt. Dies kann auf zwei Weisen geschehen: auf die im Stil `annotate` oder auf die im Stil `annotation` verwendete Art. Zunächst erfolgt ein Blick auf die einfachere Lösung. Der auf dem `plain`-Stil basierende Stil `annotation` erweitert zunächst die `ENTRY`-Definitionsliste um das Feld `annotate`. Danach wird die Funktion `fin.entry` so verändert, dass sie das zusätzliche Feld verarbeitet. Wie in dem Beispiel für die Funktion `book` zu sehen, wird `fin.entry` am Ende jeder Funktion aufgerufen, die einen Publikationstyp definiert.

<pre> FUNCTION {fin.entry} { add.period\$ write\$ newline\$ } </pre> <i>Vor der Anpassung</i>	<pre> FUNCTION {fin.entry} { add.period\$ write\$ newline\$ "\begin{quotation}\noindent\textsc{Key:\ }" cite\$ * write\$ annotate missing\$ 'skip\$ { "\\textsc{Annotation:\ }" write\$ annotate write\$ } if\$ "\end{quotation}" write\$ newline\$ } </pre> <i>Nach der Anpassung</i>

Nach Ausgabe des Verweistextes innerhalb einer quotation-Umgebung wird hinter dem Text „Annotation:“, der eine neue Zeile beginnt, der eigentliche Kommentartext eingefügt. Fehlt das Feld, so erfolgt keine Textausgabe – der Test `annotate missing$` führt dann den `skip$`-Zweig des `if$`-Befehls aus.

Der andere Stil, `annotate`, der auf dem `alpha`-Stil basiert, verfolgt einen etwas komplizierteren Ansatz. Nachdem das Element `annotate` in die `ENTRY`-Definitionsliste aufgenommen wurde, wird die Funktion `format.annotate` erstellt, um dieses zusätzliche Feld zu formatieren. Die Funktion hat eine ähnliche logische Struktur wie der oben gezeigte Code.

```
FUNCTION {format.annotate}
{ annotate empty$
  { "" }
  { " \begin{quotation}\noindent " annotate *
    " \end{quotation}" * }
  if$
}
```

Die Formatierungsalgorithmen für die in Tabelle 13.1 auf Seite 791 vorgestellten Publikationstypen enthalten außerdem jeweils direkt hinter dem Aufruf von `fin.entry` die zusätzliche Zeile `format.annotate write$`.

Sprachunterstützung

Um einen `BWTeX`-Stil an andere Sprachen als die englische anzupassen, muss man zumindest die festen englischen Texte in den `BWTeX`-Stildateien übersetzen, wie etwa „Annotation“ im letzten Abschnitt.

Zunächst sollte man eine Stildatei bearbeiten und an den entsprechenden Stellen die neuen Begriffe einfügen. Da man nur mit einer Sprache arbeitet, kann man gleichzeitig auch die entsprechenden sprachspezifischen typographischen Konventionen einführen. Ein Beispiel für diesen Ansatz ist der Stil `nederlands` von Werenfried Spit. Er basiert auf dem Stil `harvard` und wurde nach den Empfehlungen von Van Dale (1982) an das Niederländische angepasst. In den folgenden Beispielen werden einige der dort angepassten Funktionen näher untersucht.

Im Niederländischen unterscheidet man nicht zwischen einem oder mehreren Herausgebern. Der niederländische Oberbegriff `redactie` ersetzt die beiden englischen Formen.

<pre>FUNCTION {format.editors} { editor empty\$ { "" } { editor format.names editor num.names\$ #1 > { " (eds)" * } { " (ed.)" * } if\$ } if\$ }</pre>	<pre>FUNCTION {format.editors} { editor empty\$ { "" } { editor format.names ", redactie" * } if\$ }</pre>
<i>Vor der Anpassung</i>	<i>Nach der Anpassung</i>

Die nächsten Beispiele verdeutlichen, dass man Einträge (in Form und Übersetzung) recht genau an eine bestimmte Sprache anpassen kann. In diesem Falle geht es um das Format des Herausgeberfeldes (edition). In diesem Beispiel werden bis zur dritten Auflage für die niederländische Sprache spezifische Begriffe verwendet.

Ab der vierten Auflage wird die generische Zeichenfolge `ie` benutzt, wobei `i` die Nummer der Auflage wiedergibt. Das Beispiel zeigt auch die Verschachtelung der `ife`-Anweisungen und die Verwendung des Befehls `change.casee`, mit dem die Groß-/Kleinschreibung geändert wird.

```
FUNCTION {format.edition}
{
  edition empty$
  { "" }
  { output.state mid.sentence =
    { edition "1" change.case$
      " edition" * }
    { edition "t" change.case$
      " edition" * }

    if$
  }
  if$
}
```

Vor der Anpassung

```
FUNCTION {format.edition}
{
  edition empty$
  { "" }
  { edition "1" =
    { "Eerste" }
    { edition "2" =
      { "Tweede" }
      { edition "3" =
        { "Derde" }
        { edition "\textsuperscript{e}" * }
        if$
      }
      if$
    }
    if$
    output.state mid.sentence =
      { "1" change.case$ " druk" * }
      { "t" change.case$ " druk" * }
      if$
    }
    if$
  }
}
```

Nach der Anpassung

Natürlich sollte man auch die Texte für die Monatsnamen ändern. Zudem kann man noch einige weitere sprachspezifische Strings definieren.

```
MACRO {jan} {"januari"}   MACRO {feb} {"februari"}
MACRO {mar} {"maart"}     MACRO {apr} {"april"}
...
MACRO {sep} {"september"} MACRO {oct} {"oktober"}
MACRO {nov} {"november"}  MACRO {dec} {"december"}
```

Des Weiteren muss die Sortierroutine für die Namen, also die Funktion `sort.format.names`, die sprachspezifischen Regeln kennen, nach denen die Namen in der passenden Reihenfolge ausgegeben werden.

Dabei ist auch zu berücksichtigen, dass die meisten Sprachen Artikel oder andere kurze Wörter enthalten, die bei der Sortierung von Titeln ignoriert werden sollten.

```
FUNCTION {sort.format.title}
{ 't :=
  "A " #2
  "An " #3
  "The " #4 t chop.word
  chop.word
  chop.word
  sortify
  #1 global.max$ substring$
}
```

Vor der Anpassung

```
FUNCTION {sort.format.title}
{ 't :=
  "De " #3
  "Een " #4 t chop.word
  chop.word
  sortify
  #1 global.max$ substring$
}
```

Nach der Anpassung

Hier kürzt die Funktion `chop.word` den obersten String auf dem Stapel um das angegebene Wort – in diesem Falle den bestimmten Artikel (De) und den unbestimmten (Een).

Dokumentieren eigener L^AT_EX-Pakete

14.1 doc – Dokumentieren von Programmcode	845
14.2 docstrip.tex – Lauffähigen Code erzeugen	857
14.3 ltxdoc – Eine einfache Dokumentationsklasse	867
14.4 Werkzeuge für die Versionsverwaltung	869

In diesem Kapitel wird das doc-System beschrieben, ein Verfahren zur Dokumentation von L^AT_EX-Makros und -Umgebungen. Ein Großteil des verfügbaren L^AT_EX-Codes ist mithilfe seiner Konventionen und unterstützenden Werkzeuge dokumentiert. Dabei ist das Grundprinzip, den L^AT_EX-Programmcode zusammen mit Kommentaren und genereller Dokumentation in einer Datei zu speichern. Aus dieser werden dann die Dokumentation oder die ausführbaren Paketdateien mithilfe eines Standardverfahrens extrahiert.

Dieses Kapitel geht ausführlich auf die Struktur ein, die derartige Dateien haben sollten. Es erklärt, wie man zusammen mit dem DOCSTRIP-Programm Prozeduren zur automatischen Installation erstellt, mit denen man L^AT_EX-Pakete verteilt und die zugehörige Dokumentation generiert. Außerdem zeigt dieses Kapitel, wie man von Dritten erzeugten Programmcode leichter versteht, diesen einfach installiert und die Dokumentation hierfür erstellt (nicht zwingenderweise in dieser Reihenfolge).

Zum Schluss dieses Kapitels wird kurz darauf eingegangen, wie Versionsverwaltung funktioniert und wie RCS/CVS-Daten mit L^AT_EX extrahiert werden können. Maßnahmen zur Versionsverwaltung können sich bei größeren Dokumentationsprojekten als nützlich erweisen.

14.1 doc – Dokumentieren von Programmcode

Das Konzept des „literate programming“ (die Idee, Software integriert mit ihrer Dokumentation zu entwerfen) wurde von Donald Knuth geprägt, als er für die Entwicklung des T_EX-Programms das WEB-System konzipierte, das Pascal-ähnlichen Metaquellcode und Dokumentation kombiniert. Dank der

integrierten Dokumentation war es relativ einfach, T_EX und seine Begleitprogramme praktisch auf jedes Computersystem in der Welt zu portieren.

Nach und nach begannen auch die Autoren von L^AT_EX-Paketen, die Bedeutung der Dokumentierung ihres L^AT_EX-Programmcodes zu erkennen. Viele verteilen ihre L^AT_EX-Makros heute, indem sie das `doc`-Paket (von Frank Mittelbach) und das zugehörige `DOCSTRIP`-Programm (das ursprünglich von Frank Mittelbach stammt und später durch Beiträge von Johannes Braams, Denys Duchier, Marcin Woliński und Mark Wooding erweitert wurde) verwenden. An dieser Stelle sollte erwähnt werden, dass es eine experimentelle Neuimplementierung mit zusätzlichen Funktionen und einer besseren und direkteren Benutzerschnittstelle gibt, die von Lars Hellström geschrieben wurde. Sie wird derzeit unter dem Namen `xdoc2` verteilt, was darauf hinweist, dass es sich um eine eingefrorene (und damit verwendbare) Momentaufnahme der Entwicklungsarbeit handelt; die Endversion wird `xdoc` heißen.

Beide Systeme ermöglichen es, den L^AT_EX-Programmcodes und die Dokumentation in ein und derselben T_EX-Quelldatei abzuspeichern. Der offensichtliche Vorteil dieses Verfahrens besteht darin, dass komplexe T_EX-Anweisungen mithilfe der Kommentare in der Datei leichter zu verstehen sind. Zusätzlich sind Aktualisierungen direkter durchzuführen, weil nur eine einzige Quelldatei geändert werden muss.

Das `doc`-Paket bietet eine Reihe von Befehlen und errichtet einige Konventionen, die speziell vorbereitete Quelldateien zulassen, die sowohl Code als auch dessen Dokumentation enthalten.

Um eine Dokumentation zu erstellen, benötigt man einen Treiber (gegebenenfalls in einer externen Datei, s.u.), der das `doc`-Paket lädt und dann die Quelldatei interpretiert. Um eine lauffähige Version des Codes zu erzeugen, muss zunächst das Quellpaket mit `DOCSTRIP` verarbeitet werden (siehe Abschnitt 14.2). Dieser Schritt wird gewöhnlich implizit durchgeführt, indem eine `.ins`-Datei bereitgestellt wird, die mit L^AT_EX bearbeitet wird.

In seiner einfachsten Form ist der Treiber für die Dokumentation eine externe Datei. Heute ist der Treiber jedoch üblicherweise Teil der Quelldatei, so dass es zum Erzeugen der Dokumentation ausreicht, die Quelldatei mit L^AT_EX zu bearbeiten. Die Möglichkeiten werden ausführlich in Abschnitt 14.1.4 besprochen.

Die wichtigsten Befehle und Konzepte werden in den nächsten Abschnitten erörtert. Tabelle 14.1 auf Seite 853 gibt einen Überblick über alle `doc`-Anwenderbefehle. Weitere Einzelheiten über diese Befehle sind in der dokumentierten Quelldatei `doc.dtx` des `doc`-Paketes zu finden, die auch als (wenn auch etwas veraltetes) gutes Beispiel für das `doc`-System dienen kann. Leser können zusätzlich auf das Tutorial „How to Package Your L^AT_EX Package“ von Scott Pakin zurückgreifen, das verschiedene Aspekte des `doc`-Paketes und des `DOCSTRIP`-Programms beschreibt. Dieses Tutorial ist auf CTAN unter <http://www.ctan.org/tex-archive/info/dtxtut> verfügbar.

14.1.1 Allgemeine Konventionen für Quelldateien

Eine L^AT_EX-Datei, die mit dem `doc`-System verwendet werden soll, besteht typischerweise aus mehreren *Dokumentationsteilen* kombiniert mit *Definitionsteilen*. Innerhalb der Dokumentationsteile beginnt jede Zeile in der ersten Spalte

mit einem Prozentzeichen (%). Die Zeile kann beliebige $\text{\TeX}$ - oder $\text{\LaTeX}$ -Befehle enthalten, wobei das %-Zeichen hier jedoch nicht als Kommentarzeichen verwendet werden kann. Anwenderkommentare können stattdessen durch ein vorangestelltes `^^A` eingegeben werden. Längere Textpassagen können ebenfalls in Kommentare umgewandelt werden, indem sie mit `%\iffalse ... %\fi` begrenzt werden. Alle anderen Abschnitte der Datei zählen zum Definitionsteil. Er enthält den zugehörigen Code, der im Dokumentationsteil beschrieben wird.

Je nachdem, wie die Programmcode-Teile strukturiert sind, kann man solch eine Datei direkt mit $\text{\LaTeX}$ verwenden; dies wird heutzutage jedoch nur selten gemacht. Stattdessen wird `DOCSTRIP` typischerweise verwendet, um die ausführbaren Dateien zu erzeugen. Bei ersterem Ansatz durchläuft $\text{\LaTeX}$ die Dokumentationsteile sehr schnell. Dabei fügt es die Makrodefinitionen zusammen, auch wenn diese über mehrere Definitionsteile verteilt sind.

Wenn eine gedruckte Dokumentation der Makros erstellt werden soll, müssen die Definitionsteile wortwörtlich, d.h. unverändert, gesetzt werden. Dazu müssen diese Teile in `macrocode`-Umgebungen eingebettet werden.

```
%\begin{macrocode}
  (Code Zeilen)
%\end{macrocode}
```

Zu beachten ist dabei, dass zwischen dem %-Zeichen und `\end{macrocode}` *genau* vier Leerzeichen eingegeben werden müssen. Der Grund hierfür ist, dass $\text{\LaTeX}$ bei der Bearbeitung der `macrocode`-Umgebung nicht nach dem `\end`-Befehl mit dem Argument `macrocode`, sondern nach dieser speziellen Zeichenkette sucht. Innerhalb eines Definitionsteils sind alle $\text{\TeX}$ -Befehle erlaubt. Auch das Prozentzeichen kann verwendet werden, um unerwünschte Leerzeichen am Zeilenende zu unterdrücken.

Anstelle der `macrocode`-Umgebung kann man auch `macrocode*` verwenden. Diese Umgebung erzielt das gleiche Resultat, mit der Ausnahme, dass Leerzeichen in der gedruckten Dokumentation als `_`-Zeichen erscheinen.

14.1.2 Beschreibung neuer Makros und Umgebungen

Die meisten Pakete enthalten Befehle und Umgebungen, die von Anwendern in ihren Dokumenten verwendet werden können. Das `doc`-Paket bietet eine Reihe von Konstrukten an, um ein kurzes Handbuch bereitzustellen, das die Funktionen dieser Pakete beschreibt.

```
\DescribeMacro{\befehl}    \DescribeEnv{umgebung}
```

Der Befehl `\DescribeMacro` hat ein Argument, das im Rand ausgedruckt wird und zusätzlich einen speziellen Indexeintrag erzeugt, z.B.

```
% \DescribeMacro{\DocInput} \DescribeMacro{\IndexInput}
% Und schließlich der \meta{input commands}-Teil ...
```

Ein ähnliches Makro namens `\DescribeEnv` kann verwendet werden, um auf die Beschreibung einer $\text{\LaTeX}$ -Umgebung hinzuweisen.

```
\begin{macro}{\befehl} \begin{environment}{umgebung}
```

Um die Definition eines neuen Makros zu dokumentieren, wird die Umgebung `macro` verwendet. Diese Umgebung erhält den Namen des neuen Makros als Argument. Der Inhalt des Argumentes wird im Rand ausgegeben und in den Index aufgenommen. Tatsächlich werden für die Verwendung und die Definition eines Makros unterschiedliche Indexformatierungen benutzt, um das Referenzieren zu erleichtern. Hier ein Beispiel direkt aus den englischen Quelldateien des `doc`-Paketes:

```
% \begin{macro}{\MacroTopsep}
%   Here is the default value for the \verb+\MacroTopsep+
%   parameter used above.
%   \begin{macrocode}
\newlength\MacroTopsep
\setlength\MacroTopsep{7pt plus 2pt minus 2pt}
%   \end{macrocode}
% \end{macro}
```

Eine weitere Umgebung mit dem wenig phantasievollen Namen `environment` dokumentiert den Programmcode von Umgebungen. Sie funktioniert wie die `macro`-Umgebung, erwartet aber als Argument den Namen einer Umgebung.

```
\MakeShortVerb{\c} \MakeShortVerb*{\c} \DeleteShortVerb{\c}
```

Wenn viele Textstellen unverändert wiedergegeben werden sollen (z.B. Befehlsnamen), ist es sehr umständlich, stets `\verb+...+` einzugeben. Aus diesem Grund bietet das `doc`-Paket eine Abkürzungsmöglichkeit, mit der man ein Zeichen `c`, das ansonsten im Dokument nur sehr selten auftaucht, als Begrenzungssymbol für die wortwörtliche Ausgabe verwenden kann. Häufig wird das `"`-Zeichen gewählt, aber wenn dieses Zeichen bereits für einen anderen Zweck verwendet wird, z.B. um Umlaute zu erzeugen, kann man auch den senkrechten Strich „|“ benutzen. Nachdem der Befehl `\MakeShortVerb{\c}` im Dokument eingeführt worden ist, wird die Eingabe `c text c` äquivalent zu `\verb c text c`.

Die Variante `\MakeShortVerb*` bewirkt dasselbe, verwendet jedoch `\verb*`. Wenn `c` später in seiner ursprünglichen Bedeutung verwendet werden soll, genügt die Eingabe `\DeleteShortVerb{\c}`. Man kann zwischen beiden Formen (Kurzform und Originalbedeutung) beliebig oft hin und her wechseln.¹ Zu beachten ist, dass solche Kurzformen für `\verb` genauso wie `\verb` selbst nicht im Argument eines anderen Befehls vorkommen dürfen. Die Zeichen können jedoch uneingeschränkt in `verbatim`- und `macrocode`-Umgebungen verwendet werden.

Die dokumentierte Paketdatei kann in zwei Teile unterteilt werden: Der erste enthält die allgemeine Beschreibung und der zweite gibt eine ausführliche Beschreibung der Makroimplementierung wieder. Beim Erzeugen der Dokumentation kann der Anwender den letzteren Teil unterdrücken, wenn an

¹Diese Funktion gibt es auch als eigenständiges Paket `shortvrb`; dieses wurde in Abschnitt Abschnitt 3.4 besprochen. Siehe Beispiel 3-4-2 auf Seite 160.

der Trennstelle zwischen diesen beiden Teilen der Befehl `\StopEventually` eingefügt wurde.

```
\StopEventually{schlussteil} \Finale
```

Das Makro `\StopEventually` hat ein Argument, in welches die gesamte Information geschrieben wird, die erscheinen soll, wenn der Anwender beschließt, das Setzen des Dokumentes an dieser Stelle zu beenden (z.B. eine Bibliographie, die in der Regel am Ende des Dokumentes platziert wird). Wenn die Treiberdatei eine `\OnlyDescription`-Deklaration enthält, bearbeitet L^AT_EX das Argument von `\StopEventually` und hört dann auf, die Datei zu lesen.¹ Andernfalls speichert das Makro `\StopEventually` sein Argument in einem Makro namens `\Finale` ab. Dieses kann im späteren Verlauf verwendet werden, um die gespeicherte Information zu verarbeiten (üblicherweise ganz am Schluss, daher der Name). Durch dieses Verfahren ist es nicht erforderlich, Änderungen an zwei Stellen vorzunehmen.²

Zur Dokumentation von Änderungen kann man den Befehl `\changes` in den Beschreibungsteil des geänderten Codes einfügen.

```
\changes{version}{datum}{text}
```

Die Information im Befehl `\changes` kann dazu verwendet werden, eine Hilfsdatei zu erstellen (hierzu wird L^AT_EXs `\glossary`-Befehl verwendet), die nach entsprechender Formatierung ausgedruckt werden kann. Damit die Änderungsdaten aufgezeichnet werden, muss der Befehl `\RecordChanges` in der Treiberdatei angegeben werden. Um die Änderungsdaten zu lesen und auszu-drucken, wird der Befehl `\PrintChanges` an geeigneter Stelle (in der Regel nach dem Befehl `\PrintIndex`) in die Treiberdatei eingefügt.

Eine sortierte Datei mit den Änderungen erhält man, indem man die Glossar-Rohdatei mit `MakeIndex` bearbeitet und dabei eine passende Stildatei verwendet (z.B. `gglo.ist`, die mit der doc-Distribution mitgeliefert wird; für weitere Informationen über die Bearbeitung von Glossaren durch `MakeIndex` siehe Abschnitt 11.1.6 auf Seite 673).

14.1.3 Indexieren aller verwendeten Makros

Innerhalb der Umgebungen `macrocode` und `macrocode*` werden für jeden verwendeten Befehlsnamen Indexeinträge erstellt. Mit dem so generierten Indexregister lässt sich leicht herausfinden, wo ein bestimmtes Makro eingesetzt wird. Da T_EX jedoch erheblich langsamer arbeitet, wenn es solch eine große Anzahl an Indexeinträgen erstellen muss, kann man diese Funktion mithilfe des Befehls `\DisableCrossrefs` in der Treiberdatei deaktivieren. Mit dem Befehl `\EnableCrossrefs` wird sie wieder aktiviert.

¹Der etwas seltsame Befehlsname beruht auf einem Missverständnis des Paketautors: er meinte „eventuell“ und verwendete daher den englischen Begriff „eventually“, der jedoch eine andere Bedeutung hat (nämlich „schließlich“). Als er seinen Fehler bemerkte, wurde das Makro schon einige Jahre verwendet.

²Voreingestellt ist, das ganze Dokument zu setzen. Dies kann auch explizit mit dem Makro `\AlsoImplementation` festgelegt werden.

Möglichkeiten zur Feinsteuerung bietet der Befehl `\DoNotIndex`. Er hat ein Argument, das eine durch Kommas getrennte Liste von Befehlen enthält, die *nicht* in das Stichwortverzeichnis aufgenommen werden. Es kann mehr als ein `\DoNotIndex`-Befehl vorhanden sein, in diesem Fall werden ihre Inhalte kombiniert. Häufig wird dieses Makro verwendet, um Standardbefehle aus dem Index auszuschließen, die diesen nur aufblähen würden.

Das Anlegen (oder Nicht-Anlegen) des Indexes wird durch eine der nachfolgenden Deklarationen gesteuert, die in der Präambel der Treiberdatei angegeben werden (wird keine der Deklarationen spezifiziert, dann wird kein Stichwortverzeichnis erstellt). Bei Angabe des Befehls `\PageIndex` referenzieren alle Indexeinträge auf die zugehörige Seitennummer; bei Verwendung von `\CodelineIndex` erzeugen mit `\DescribeMacro` und `\DescribeEnv` erstellte Indexeinträge Seitennummern, während jene, die von den `macro`- und `macrocode`-Umgebungen erstellt wurden, auf die Zeilen referenzieren, in welchen der Befehl steht. Dazu werden alle Codezeilen automatisch durchnummeriert.

Wenn die Indexeinträge erstellt werden, müssen sie durch ein externes Programm wie *MakeIndex* (siehe Kapitel 11) sortiert werden. Da das `doc`-Paket spezielle Konventionen für Indexeinträge verwendet, muss *MakeIndex* mit dem Schalter `-s` ausgeführt werden (siehe Abschnitt 11.2.4 auf Seite 681), um eine passende Stildatei anzugeben – z.B. `gind.ist`, die mit dem `doc`-Paket mitgeliefert wird.

Um den sortierten Index zu lesen und auszudrucken, muss der Befehl `\PrintIndex` gegen Ende der Treiberdatei eingefügt werden, typischerweise hinter einer eventuell vorhandenen Bibliographie.

14.1.4 Der Dokumentationstreiber

Um für einen Satz von Makros mit dem `doc`-System eine Dokumentation zu erstellen, muss eine Treiberdatei angelegt werden, welche die folgenden Merkmale besitzt:

```
\documentclass[optionen]{dokumentenklasse}
\usepackage{doc}
 $\langle präambel \rangle$ 
\begin{document}
   $\langle eingabebefehle \rangle$ 
\end{document}
```

Die $\langle dokumentenklasse \rangle$ kann jede zugelassene Klasse sein, z.B. `article` oder `ltxdoc` (beschrieben in Abschnitt 14.3); in letzterem Fall wird das `doc`-Paket bereits von der Klasse geladen. In der $\langle präambel \rangle$ sollte man Deklarationen einfügen, die das Verhalten des `doc`-Systems steuern, wie etwa `\DisableCrossrefs`, `\OnlyDescription` und `\CodelineIndex`.

```
\DocInput{dateiname}   \IndexInput{dateiname}
```

Der Teil $\langle eingabebefehle \rangle$ sollte schließlich einen oder mehrere `\DocInput`- und/oder `\IndexInput`-Befehle enthalten. Der `\DocInput`-Befehl wird für Dateien verwendet, die für das `doc`-System ausgelegt sind, während der

`\IndexInput`-Befehl für Makrodateien verwendet werden kann, die nicht den Konventionen des `doc`-Systems entsprechen. Dieser Befehl akzeptiert als Argument einen Dateinamen und erstellt einen unformatierten Ausdruck der Datei, wobei er alle Befehle in den Index schreibt. Diese Funktion kann sehr praktisch sein, wenn man etwas über Makros erfahren möchte, die nicht ausreichend dokumentiert sind.

Es ist auch möglich, die Befehle `\PrintIndex` und `\PrintChanges` (wenn die Änderungen mit `\RecordChanges` aufgezeichnet werden) einzusetzen. Einige Leute binden sie direkt in die Quelldatei ein, aber es ist empfehlenswerter, sie in den Treiber zu einzufügen. Dadurch wird es möglich, mehrere Pakete in einem Dokument zu kombinieren und einen Gesamtindex zu erstellen.

Wie in der Einleitung erwähnt, ist der Treiber heutzutage meistens keine separate Datei, sondern wird direkt in die Quelldatei eingebunden. Wie dies funktioniert, wird im nächsten Abschnitt erklärt.

14.1.5 Bedingter Code in der Quelldatei

Die bisher besprochenen Funktionen können verwendet werden, um eine $\text{\LaTeX}$ -Quelldatei im „literate programming“-Stil zu erzeugen. Diese kann direkt verwendet werden, indem sie als Paket geladen wird (wobei $\text{\TeX}$ die Kommentare umgeht), oder sie kann, wie im vorangegangenen Abschnitt beschrieben, mithilfe einer Treiberdatei ausgedruckt werden. Hierfür muss aber die Struktur einer solchen Datei linear sein; mit anderen Worten: $\text{\TeX}$ lädt den *gesamten* Code genau in der Reihenfolge, wie er in der Datei aufgeführt ist.

Nach einigen Erfahrungen mit dem `doc`-System stellte es sich bald heraus, dass es eine wertvolle Erweiterung wäre, wenn man einsatzfähige Dateien erzeugen könnte, indem man sie aus mehreren Quelldateien zusammensetzt oder sie z.B. aus Teilen von einer oder mehreren Quelldateien extrahiert. Aus diesem Grund wurde das `doc`-System in zwei Richtungen ausgebaut:

- Es wurde eine Syntax entwickelt, um Teile des Codes zu markieren, so dass man auf die Bestandteile separat referenzieren konnte.
- Das `DOCSTRIP`-Programm (siehe Abschnitt 14.2), das ursprünglich nur verwendet wurde, um die Kommentare aus `doc`-Dateien zu entfernen, wurde erweitert, um eine Skriptsprache bereitzustellen, in der man angeben kann, wie eine sofort einsatzfähige Datei aus den markierten Code-Teilen von einer oder mehreren Quelldateien generiert werden kann.

Natürlich kann eine Quelldatei, die solch einen bedingten Code enthält, nicht mehr direkt verwendet werden. Es bedarf des vorherigen Einsatzes des `DOCSTRIP`-Programms, um sie in eine einsatzfähige Datei zu verwandeln. Die erweiterten Möglichkeiten, die dieser Ansatz bietet, machen jedoch den Aufwand, der durch den zusätzlichen Produktionsschritt während der Installation entsteht, bei weitem wett, so dass heute fast alle Anwendungen von `doc` diese Funktion nutzen.

Programmfragmente, die nur unter bestimmten Bedingungen hinzugefügt werden sollen, werden in der Quelldatei mit „Tags“ gekennzeichnet. Das einfachste Format besteht aus einem `<*name>`- und `</name>`-Paar, das einen Teil des Codes umrahmt. Dies ermöglicht dem Anwender, bestimmte Teile ein-

bzw. auszuschließen, indem er in einem DOCSTRIP-Skript auf ihren *namen* verweist. Diese Tags müssen – mit einem vorangestellten %-Zeichen – immer am Anfang einer Zeile stehen, z.B.

```
%<*style>
    einige Programmzeilen
%</style>
```

Es ist möglich, mehr als einen Tag mit einem Abschnitt zu verknüpfen, indem man mehrere *namen* mit den booleschen Operatoren | für das logische Oder, & für das logische Und und ! für die Negation kombiniert (wobei die Operatoren von links nach rechts abgearbeitet werden). Zum Beispiel bedeutet

```
%<*Aname|Bname&!Cname>
    einige Programmzeilen
%</Aname|Bname&!Cname>
```

dass dieser Block eingebunden werden soll, wenn entweder Aname angefordert wurde oder zwar Bname, aber nicht Cname.

Es gibt zwei weitere Formen von Direktiven, mit denen einzelne Programmzeilen eingebunden oder ausgeschlossen werden können. Eine Zeile, die mit %<+*name*> beginnt, wird eingebunden (ohne das Tag), wenn *name* angefordert wird. Eine Zeile, die mit %<-*name*> beginnt, wird eingebunden, wenn *name* während der DOCSTRIP-Bearbeitung *nicht* angefordert wird.

Die obigen Befehle können ineinander verschachtelt werden. Dabei werden die inneren Tags nur dann ausgewertet, wenn die äußeren Tags wahr sind (d.h. wenn der gesamte Block eingebunden werden soll).

```
%<*Aname>
    Programmzeile 1
%<+Bname> Programmzeile 2
%<-Bname> Programmzeile 3
    Programmzeile 4
%</Aname>
```

Hier wird nichts eingebunden, wenn Aname nicht angefordert wird. Wenn er angefordert wird, werden die Programmzeilen 1, 2 und 4 ausgegeben, wenn zusätzlich Bname angefordert wird; ansonsten werden die Programmzeilen 1, 3 und 4 ausgegeben.

Bleibt nur noch die Frage, wie man mit bedingtem Code die Treiberdatei in die Hauptquelldatei einfügen kann. Hierfür muss der Code für den Treiber als erster Codeblock eingebunden werden und davor und dahinter Tags eingefügt werden (z.B. `driver`). Wenn der Anwender dann die Quelldatei mit L^AT_EX bearbeitet, ist der Treibercode der erste Programmcode, dem keine %-Zeichen vorangestellt sind, und er wird ausgeführt. Da er auf `\end{document}` endet, führt L^AT_EX in der Datei keinen weiteren Programmcode aus. Die Dokumentation wird dann unter der Annahme gesetzt, dass der Treiber die gesamte Datei mit `\DocInput` lädt. Um die eigentlichen Paketdateien zu generieren, muss man ein DOCSTRIP-Skript verwenden (siehe Abschnitt 14.2 auf Seite 857), das den Treibercode ignoriert, indem es keinen Code aus einem Block anfordert, der mit dem Tag `driver` gekennzeichnet ist.

Tabelle 14.1: Überblick über Befehle des doc-Paketes

Präambel und Eingabebefehle	
<code>\AlsoImplementation</code>	Formatiert die gesamte Datei, einschließlich Code-Teil, die entsprechend der doc-Konventionen markiert ist (voreingestellt).
<code>\CharacterTable{zeichentabelle}</code>	Benutzerschnittstelle zur Zeichenüberprüfung.
<code>\CheckModules</code>	Spezielle Formatierung für die Modulmarker von DOCSTRIP (voreingestellt).
<code>\Checksum{prüfsumme}</code>	Benutzerschnittstelle für eine einfache Prüfsumme (Anzahl der Backslash-Zeichen im Programmcode).
<code>\CodelineIndex</code>	Indexiert die Befehle anhand der Zeilennummern.
<code>\CodelineNumbered</code>	Nummeriert die Zeilen ohne Indexierung der Befehle.
<code>\DisableCrossrefs</code>	Keine Erstellung von Indexeinträgen für Befehle im Programmcode.
<code>\DocInput{datei}</code>	Einlesen von <i>datei</i> gemäß der doc-Konventionen.
<code>\DontCheckModules</code>	Keine spezielle Formatierung der Modulmarker von DOCSTRIP.
<code>\EnableCrossrefs</code>	Erstellt Indexeinträge für Befehle im Programmcode.
<code>\IndexInput{datei}</code>	Einlesen der <i>datei</i> , wortwörtliche Ausgabe und Erstellung einer Querverweisliste für Befehle.
<code>\OnlyDescription</code>	Keine Formatierung des Codes; Abbruch bei <code>\StopEventually</code> .
<code>\PageIndex</code>	Indexiert Befehle anhand der Seitennummern.
<code>\PrintChanges</code>	Druckt die Änderungsliste hier aus.
<code>\PrintIndex</code>	Druckt den Index hier aus.
<code>\RecordChanges</code>	Erstellt eine Änderungsliste.
Dokumentenstrukturbefehle	
<code>\bslash</code>	Druckt ein Backslash-Zeichen (\) aus. Nur nützlich in Typewriter Fonts!
<code>\DeleteShortVerb{\zeichen}</code>	Macht für <i>zeichen</i> die vorangegangene Definition von <code>\MakeShortVerb</code> oder <code>\MakeShortVerb*</code> rückgängig.
<code>\DescribeEnv{umg}</code>	Kennzeichnet die Stelle im Text, an der die Umgebung <i>umg</i> beschrieben ist.
<code>\DescribeMacro{befehl}</code>	Kennzeichnet die Stelle im Text, an der das Makro <i>befehl</i> beschrieben ist.

Überblick über Befehle des doc-Paketes (Forts.)

`\begin{environment}{umg}`

Umgebung, in der Code und Beschreibung der Umgebung *umg* eingebettet sind.

`\Finale`

Befehl, der am Ende des Dokumentes ausgeführt wird (siehe auch `\StopEventually`).

`\begin{macro}{\befehl}`

Umgebung, in der Code und Beschreibung des Makros `\befehl` eingebettet sind.

`\begin{macrocode}`

Umgebung, in welcher der T_EX-Code eingebettet ist.

`\begin{macrocode*}`

Wie die *macrocode*-Umgebung, wobei Leerzeichen als `\` ausgegeben werden.

`\MakeShortVerb{\zeichen}`

Definiert *zeichen* als Kurzform für `\verb`.

`\MakeShortVerb*{\zeichen}`

Definiert *zeichen* als Kurzform für `\verb*`.

`\meta{arg}`

Druckt das Argument als Metawort, d.h. erzeugt folgendes Resultat: *(arg)*.

`\SpecialEscapechar{\zeichen}`

Definiert neues Escape-Zeichen *zeichen*, das anstelle von `\` verwendet werden soll.

`\StopEventually{befehle}`

Das Argument *befehle* gibt an, welche Befehle am Ende des Dokumentes ausgeführt werden (sie werden in `\Finale` gespeichert).

`\begin{verbatim}`

Leicht geänderte Version von L^AT_EXs Standard-*verbatim*-Umgebung, in der Prozentzeichen in der ersten Spalte ignoriert werden.

`\begin{verbatim*}`

Wie die *verbatim*-Umgebung, wobei Leerzeichen als `\` ausgegeben werden.

Indexbefehle

`\*`

Symbol, das in Indexeinträgen zum Verweis auf übergeordnete Einträge verwendet wird (Voreinstellung: `~`).

`\actualchar`

Zeichen zur Trennung des „Sortierschlüssels“ und des tatsächlichen Eintrags in einem Indexeintrag (Voreinstellung: `=`).

`\DoNotIndex{befehl1, ..., befehln}`

Namen der Befehle, die nicht im Index erscheinen sollen.

`\encapchar`

Zeichen zur Trennung des tatsächlichen Indexeintrags und des Befehls, durch den die Seitennummer in einem Indexeintrag formatiert wird (Voreinstellung: `!`).

`\IndexMin`

Längenparameter (Voreinstellung: 80pt), der den minimalen Leerraum angibt, der auf einer Seite vorhanden sein muss, damit der Index auf der aktuellen Seite beginnen kann.

*Überblick über Befehle des doc-Paketes (Forts.)***\IndexParms**

Makro, das die Formatierung der Indexspalten steuert.

\IndexPrologue{*text*}

Überschreibt den Standardtext, der normalerweise oberhalb des Index steht.

\levelchar

Zeichen zur Trennung verschiedener Indexebenen in einem Indexeintrag (Voreinstellung: >).

\main{*zahl*}

Definiert den Stil zur Formatierung der Seitennummern oder Zeilennummern der Indexeinträge für Haupteinträge (Voreinstellung: unterstrichene Ziffern).

\quotechar

Zeichen, das die besondere Bedeutung des folgenden Zeichens in einem Indexeintrag unterdrückt (Voreinstellung: !).

\SortIndex{*schlüssel*}{*eintrag*}

Erstellt einen Indexeintrag für *eintrag*, der unter *schlüssel* einsortiert ist.

\SpecialEnvIndex{*eintrag*}

Erstellt einen Indexeintrag für die Umgebung *eintrag*.

\SpecialIndex{\befehl}

Erstellt einen Indexeintrag für *\befehl* (wobei das Argument wortwörtlich im Index erscheint).

\SpecialMainEnvIndex{*umg*}

Erstellt einen Haupteintrag für eine Umgebung mit der Seitenkapselungsfunktion *\main*.

\SpecialMainIndex{\befehl}

Erstellt einen Haupteintrag für ein Makro mit der Seitenkapselungsfunktion *\main*.

\SpecialUsageIndex{\befehl}

Erstellt einen Indexeintrag für ein Makro mit der Seitenkapselungsfunktion *\usage*.

\usage{*zahl*}

Definiert den Stil, mit dem die Seitennummern der Indexeinträge auf Befehlsbeschreibungen formatiert werden (Voreinstellung: kursive Ziffern).

\verbatimchar

Zeichen zur Begrenzung von *\verb*-Konstruktionen innerhalb eines Indexeintrags (Voreinstellung: +).

Änderungs- und Versionsinformation**\changes{*version*}{*datum*}{*grund*}**

Zeichnet Änderungsinformationen in der Änderungsliste auf.

\docdate

Enthält häufig das Datum der letzten Aktualisierung der Dokumentation.

\filedate

Enthält häufig das Datum der letzten Aktualisierung des Programmcodes.

\filename

Enthält häufig den Namen der Quelldatei.

\fileversion

Enthält häufig die Versionsnummer der Quelldatei.

*Überblick über Befehle des doc-Paketes (Forts.)***\GlossaryMin**

Längenparameter (Voreinstellung: 80pt), der den minimalen Leerraum angibt, der auf einer Seite vorhanden sein muss, damit die Änderungsliste auf der aktuellen Seite beginnen kann.

\GlossaryParms

Makro, das die Formatierung der Spalten in der Änderungsliste steuert.

\GlossaryPrologue{*text*}

Überschreibt den Standardtext, der normalerweise oberhalb der Änderungsliste steht.

Layout- und Formatierungsparameter**\@idxitem**

Makro, das angibt, wie Indexeinträge gesetzt werden (standardmäßig werden sie in einem Absatz mit um 30pt eingerückten Folgezeilen gesetzt).

\AltMacroFont

Zeichensatz, der für das Setzen des Codes von DOCSTRIP-Modulen verwendet wird (Voreinstellung: `\small\ttfamily\slshape`).

\DocstyleParms

Makro, das die Formatierung des T_EX-Codes steuert.

\generalname

Überschrift für die Änderungseinträge auf oberster Ebene.

\MacrocodeTopsep

Vertikaler Abstand vor und nach jeder macrocode-Umgebung.

\MacroFont

Zeichensatz, der für das Setzen des Hauptteils des Codes verwendet wird (Voreinstellung: `\small\ttfamily`).

\MacroIndent

Breite des Einzugs für alle Programmcode-Zeilen.

\MacroTopsep

Vertikaler Abstand vor und nach jeder macro-Umgebung.

\MakePercentComment

Aktiviert „%“ als T_EXs Anfangszeichen für Kommentare.

\MakePercentIgnore

Deaktiviert „%“ als T_EXs Anfangszeichen für Kommentare.

\MakePrivateLetters

Makro zur Definition der Symbole, die als zusätzliche „Buchstaben“ in Befehlsnamen erlaubt sind (Voreinstellung: @).

\Module

Makro mit einem Argument, das die Formatierung von DOCSTRIP-Modulmarkern definiert.

\PrintDescribeEnv

Makro mit einem Argument, das die Formatierung von `\DescribeEnv` definiert.

\PrintDescribeMacro

Makro mit einem Argument, das die Formatierung von `\DescribeMacro` definiert.

\PrintEnvName

Wie `\PrintDescribeEnv`, jedoch für das Argument der `environment`-Umgebung.

*Überblick über Befehle des doc-Paketes (Forts.)***\PrintMacroName**

Wie `\PrintDescribeMacro`, jedoch zum Setzen des Argumentes der macro-Umgebung.

\ps@titlepage

Makro, das den Seitenstil für die Titelseite von Artikeln definiert, die in einer Zeitschrift zusammengefasst sind (Voreinstellung: `\ps@plain`).

StandardModuleDepth

Zähler, der die höchste Ebene der DOCSTRIP-Direktiven angibt, die noch mit `\MacroFont` formatiert werden. Tiefer verschachtelte Anweisungen werden mit `\AltMacroFont` definiert (voreingestellter Wert 0 bzw. 1 in `ltxdoc`).

\theCodelineNo

Steuert das Setzen der Zeilennummern (standardmäßig arabische Zahlen in der Schriftgröße für Hoch- und Tiefstellungen).

14.2 docstrip.tex – Lauffähigen Code erzeugen

Als doc in den späten 80er Jahren geschrieben wurde, war das Ziel, eine „literate programming“-Umgebung [86] für $\LaTeX$ bereitzustellen, in der $\LaTeX$ -Code und Dokumentation in einer Quelldatei vermischt sind. Wie sich bald herausstellte, wurde das Laden einer Datei durch $\TeX$ deutlich langsamer, wenn $\TeX$ auch den gesamten Dokumentationsteil analysieren (und anschließend ignorieren) musste.¹ Um dieses Problem zu vermeiden, suchte Frank Mittelbach nach Wegen, um alle Kommentare automatisch aus den Dateien zu entfernen, die für das doc-System geschrieben wurden.

Entwickelt man externe Programme für solch einen Zweck, so besteht das Problem, dass es nicht sicher ist, ob sie für das Betriebssystem des Anwenders verfügbar sind. Selbst wenn sie es sind, sind sie möglicherweise nicht installiert. Auf einem System, das $\LaTeX$ ausführen kann, ist jedoch ein Programm immer verfügbar: das $\TeX$ -Programm selbst. Um die größtmögliche Portierbarkeit zu erreichen, wurde das DOCSTRIP-Programm in einer systemnahen $\TeX$ -Sprache geschrieben. Seit den Anfangstagen hat das Programm viele Revisionen durchlaufen, und sein Verwendungszweck hat sich geändert: von einem einfachen, bereinigenden Hilfsprogramm zu einem vollständig anpassbaren Installationstool – das sogar Dateien an die richtigen Verzeichnisse auf dem Zielsystem verteilen kann. Johannes Braams, Denys Duchier, Marcin Woliński, Mark Wooding, David Carlisle und andere trugen zu dieser Metamorphose bei; Einzelheiten zu den Entwicklungsschritten sind in der dokumentierten Quelldatei nachzulesen (die natürlich „literate programming“ verwendet). Heute wird das DOCSTRIP-Programm vornehmlich für folgende Zwecke eingesetzt:

- Entfernung der Dokumentation (d.h. der Zeilen, die mit einem %-Zeichen in der ersten Spalte beginnen) aus der „literate programming“-Quelldatei.
- Erzeugung von einsatzfähigen Programmcode-Dateien durch Verwendung von Code aus einer oder mehrerer Dateien und Einbindung von Teilen daraus nach festgelegten Optionen.

¹Zu dieser Zeit konnte die Erstellung einer einzigen Seite durch $\TeX$ leicht eine halbe Minute oder länger dauern.

- Wenn gewünscht, automatische Installation der erzeugten Dateien in den richtigen Verzeichnissen auf dem Zielrechner, wodurch das Laden von Updates und Zusätzen zu einer L^AT_EX-Installation erheblich vereinfacht wird.

Insbesondere die letzte Möglichkeit ist nicht sehr bekannt, verdient es aber durchaus, weiter verbreitet zu werden, weil sie sich mit relativ geringerem Aufwand umsetzen lässt.

14.2.1 Aufrufen des docstrip-Programms

Von Anfang an konnte DOCSTRIP interaktiv ausgeführt werden, indem `docstrip.tex` mit L^AT_EX bearbeitet wurde:

```
latex docstrip.tex
```

L^AT_EX stellt daraufhin ein paar Fragen, etwa welche Datei wie verarbeitet werden soll. Nachdem der Anwender diese beantwortet hat, tritt DOCSTRIP in Aktion und entfernt die Kommentare aus der Quelldatei.

Dieses Verfahren diente ursprünglich nur dazu, Kommentare zu entfernen. Bei den heutigen Quelldateien, die bedingten Code enthalten und teilweise mit anderen kombiniert werden, um die endgültige „ausführbare“ Datei zu bilden, reicht dies jedoch nicht mehr aus. Stattdessen stellen die Paketentwickler heute eine Installationsdatei bereit (die per Konvention die Erweiterung `.ins` hat): sie wird verwendet, um hinter den Kulissen DOCSTRIP aufzurufen. Hierfür gibt der Anwender lediglich Folgendes an:

```
latex name.ins
```

Durch diesen Befehl werden alle „ausführbaren“ Dateien aus den Quelldateien generiert und gegebenenfalls an den richtigen Stellen installiert. Alle Standard-distributionen von L^AT_EX (z.B. `base`, `graphics` und `tools`) werden in dieser Form verteilt, genauso wie die meisten von anderen Entwicklern beigetragenen Pakete, die in diesem Buch beschrieben werden.

Der nächste Abschnitt geht darauf ein, wie man eigene Installations-skripts für DOCSTRIP erstellen kann. Anschließend zeigt Abschnitt 14.2.3, wie DOCSTRIP so eingerichtet werden kann, dass die generierten Dateien automatisch an den richtigen Stellen installiert werden.

14.2.2 docstrip-Skriptbefehle

Ein DOCSTRIP-Installationsskript hat folgende allgemeine Form:

```
\input docstrip
<andere DOCSTRIP-Befehle>
\endbatchfile
```

Zunächst wird der DOCSTRIP-Code mithilfe des T_EX-Befehls `\input` (ohne geschweifte Klammern um den Dateinamen) geladen. Dies macht es möglich,

solch ein Skript mit anderen T_EX-Formaten als L^AT_EX zu bearbeiten. Danach folgen die DOCSTRIP-Befehle, welche die neuen Dateien tatsächlich generieren, mit dem Anwender kommunizieren und andere notwendige Aufgaben durchführen. Am Ende des Skripts wird `\endbatchfile` aufgerufen. Ohne diese Anweisung würde DOCSTRIP die Eingabeaufforderung `*` anzeigen, während es auf weitere Eingaben des Anwenders wartet.

Erstellen neuer Dateien

Ein DOCSTRIP-Skript wird hauptsächlich erstellt, um zu beschreiben, welche Dateien aus welchen Quellen generiert werden sollen und welche optionalen (getaggten) Programmteile eingebunden werden sollen. Dies geschieht mithilfe von `\generate`-Deklarationen.

```
\generate{\file{ergebnisdatei_1}{\from{quelledatei_1}{tag-liste_1}
                                     \from{quelledatei_2}{tag-liste_2}
                                     \from{quelledatei_3}{tag-liste_3}
                                     ...
                                     \from{quelledatei_n}{tag-liste_n}}
...
\file{ergebnisdatei_n}{...}
}
```

Im Argument von `\generate` werden die *ergebnisdateien* angegeben, die mithilfe von `\file`-Deklarationen erstellt werden sollen. Das zweite Argument von `\file` enthält einen oder mehrere `\from`-Befehle, welche die *quelledateien* auflisten, die zur Generierung der *ergebnisdatei* verwendet werden sollen. Das zweite Argument jeder `\from`-Deklaration gibt die *tag-liste* an, die für eine bestimmte *quelledatei* verwendet werden soll. Damit werden nur die Teile des Programmcodes, die mit den entsprechenden Tags gekennzeichnet sind, sowie alle ungetaggten Quellcode Teile aus dieser Datei eingebunden (siehe Abschnitt 14.1.5 auf Seite 851).

Die *quelledateien* werden in der angegebenen Reihenfolge verwendet: zunächst wird der Code aus der *quelledatei₁* eingebunden (gemäß der Tag-Spezifikation), dann der Code aus der *quelledatei₂* usw. Die *tag-listen* der einzelnen `\from`-Befehle bestehen aus durch Kommas getrennte Tags, welche den einzubindenden Programmcode kennzeichnen.

Mit der oben gezeigten Syntax für `\generate` ist es möglich, aus einer oder mehreren *quelledateien* eine *ergebnisdatei* zu erzeugen, wenn man nur eine einzelne `\file`-Deklaration verwendet. Dadurch, dass `\generate` so häufig wie notwendig verwendet werden kann, ist dieses Verfahren flexibel genug, um jede Art von Distribution zu erzeugen. In dieser Form ist es jedoch nicht sehr effektiv. Angenommen, es gäbe eine große Quelldatei, aus der viele kleine Dateien erzeugt werden sollen – z.B. der Quellcode des doc-Paketes (`doc.dtx`), aus dem die Dateien `doc.sty`, `shortvrb.sty`, `gind.ist` und `gglo.ist` generiert werden. Diese Datei ist nahezu 5000 Zeilen lang, so dass DOCSTRIP mit vier `\generate`-Deklarationen 20000 Zeilen verarbeiten müsste. Um dieses Verfahren zu beschleunigen, ermöglicht `\generate` die Angabe mehrerer `\file`-Befehle in seinem Argument. Diese Dateien werden parallel verarbeitet,

so dass die *quelldateien* nur einmal geöffnet werden und der Quellcode für die *ergebnisdateien* parallel generiert wird.

```
\generate{\file{doc.sty}{\from{doc.dtx}{package}}
          \file{shortvrb.sty}{\from{doc.dtx}{shortvrb}}
          \usepostamble\istpost
          \file{gind.ist}{\from{doc.dtx}{gind}}
          \file{gglo.ist}{\from{doc.dtx}{gglo}}}
```

Wie zu erkennen ist, sind einige andere Befehle (z.B. `\usepostamble`) im Argument des `\generate`-Befehls zugelassen. Im obigen Beispiel hat dies den Effekt, dass der Standardnachspann durch einen anderen ersetzt wird (da der Standardnachspann am Ende der generierten Datei ein `\endinput` hinzufügt, was in einer Stildatei für *MakeIndex* nicht wünschenswert ist).

Beschränkungen bei
paralleler
Extrahierung

Bei diesem Ansatz gibt es einige Beschränkungen. Zum Beispiel gibt DOCSTRIP eine Fehlermeldung aus, wenn die Reihenfolge der Quelldateien in einem `\file`-Befehl mit der Reihenfolge in einem anderen kollidiert (die genauen Regeln werden in der DOCSTRIP-Dokumentation [121] besprochen). Wenn dies geschieht, ist die einfachste Lösung, zwei separate `\generate`-Deklarationen zu verwenden.

Kommunizieren mit dem Anwender

Die Skriptsprache von DOCSTRIP bietet begrenzt Möglichkeiten zur Kommunikation mit dem Anwender. Zu bedenken ist jedoch, dass interaktive Fragen, die manchmal durchaus sinnvoll sein können, den Installationsvorgang sehr zähflüssig machen können. Daher sollten diese Werkzeuge nur mit Vorsicht verwendet werden.

`\Msg{meldung}      \Ask{befehl}{frage}`

Der `\Msg`-Befehl kann verwendet werden, um eine *meldung* am Bildschirm auszugeben; damit bietet er eine ähnliche Funktionalität wie L^AT_EXs `\typeout`-Befehl. `\Ask` ist ähnlich wie L^AT_EXs `\typein`-Befehl, mit dem Unterschied, dass bei Beantwortung einer *frage* kein angehängtes Leerzeichen erzeugt wird, wenn die Return-Taste gedrückt wird. Auf diese Weise können einfache Fragen gestellt werden (über eine Art Low-Level-Programmierung). Ein Beispiel:

```
\Ask\answer{Should we continue? (y/n)}
\ifx\answer\y                    % \y enthält y für Vergleiche
% Code für die Antwort y        % Auch vordefiniert: \n und \yes
\else
% andernfalls
\fi
```

`\ifToplevel{code}`

Eventuell soll eine bestimmte Information nur dann ausgegeben werden oder ein bestimmter Code nur dann ausgeführt werden, wenn ein DOCSTRIP-Skript allein ausgeführt wird und nicht als Teil einer größeren Installation aufgerufen wird (siehe unten). Solch eine Information oder solch ein Code kann

im Argument eines `\ifToplevel`-Befehls angegeben werden. So gibt beispielsweise jedes einzelne Installationsskript der $\text{\LaTeX}$ -Basisdistribution an, was mit den generierten Dateien passieren soll. Bei Verwendung des Master-Installationsskripts `unpack.ins` werden die Meldungen in den Unterskripts jedoch unterdrückt, um zu vermeiden, dass dieselbe Information ständig wiederholt wird.

`\askforoverwritetrue    \askforoverwritefalse`

Bevor DOCSTRIP seine Ausgabe in eine Datei schreibt, prüft es, ob dieser Vorgang eine bestehende Version dieser Datei überschreibe. Wenn ja, wird der Anwender standardmäßig gefragt, ob die Datei überschrieben werden soll. Diese Nachfrage kann im DOCSTRIP-Skript explizit deaktiviert werden (oder umgekehrt, wenn sie deaktiviert war), und zwar mit dem Befehl `\askforoverwritefalse` bzw. `\askforoverwritetrue`.

`\askonceonly`

In einem Distributionsskript die Einstellung `\askforoverwritefalse` zu verwenden, ist meist nicht angemessen, da es effektiv bedeutet, dass die Dateien anderer Personen unter allen Umständen überschrieben werden dürfen – komme was wolle. Auf der anderen Seite wäre es bei größeren Installationen, wie etwa der Basisdistribution von $\text{\LaTeX}$, nicht sehr hilfreich, wenn bei Hunderten von Dateien einzeln nachgefragt würde. Aus diesem Grund bietet DOCSTRIP die Deklaration `\askonceonly`. Mit dieser erhält der Anwender bei der ersten Nachfrage die Möglichkeit, künftige Nachfragen zu unterdrücken. DOCSTRIP geht einfach davon aus, dass diese mit „yes“ beantwortet würden. Dies betrifft jedoch *alle* zukünftigen Fragen (egal, ob sie manuell von `\Ask` gestellt werden oder durch das Überschreiben einer Datei zustande kommen).

`\showprogress    \keepsilent`

Zum Zeitvertreib und weil in der ursprünglichen Implementierung alles so langsam war, kann man DOCSTRIP veranlassen, den Verlauf anzuzeigen, wenn Kommentare entfernt werden und neue Dateien zusammengesetzt werden. Die meisten Skripte laufen heute jedoch im stillen Modus ab.

Master-Installationsskripts

Bei großen Distributionen, wie etwa der $\text{\LaTeX}$ -Basisdistribution, ist es bequem, DOCSTRIP-Skripte zur Bearbeitung einzelner Teile bereitzustellen. `format.ins` erzeugt beispielsweise die zentrale Formatdatei `latex.ltx` und ihre Konfigurationsdateien, wie `fonttext.cfg`, während `classes.ins` die Standardklassen, wie die Dateien `article.cls` und `report.cls`, erzeugt.

Trotzdem soll der Anwender nicht gezwungen werden, ein Dutzend oder mehr Installationsskripts auszuführen (im Fall der $\text{\LaTeX}$ -Basisdistribution sind es 30). Daher bietet DOCSTRIP den Befehl `\batchinput`. Mit diesem können Installationsskripts in einem Master-Installationsskript eingebunden werden. Für diesen Zweck sollte nicht `\input` verwendet werden, denn dieser Befehl ist exklusiv für das einmalige Laden des DOCSTRIP-Codes reserviert (wie schon erläutert wurde), ansonsten wird er ignoriert. Ein gutes Beispiel

für solch ein Master-Installationsskript ist die Datei `unpack.ins` aus der Basisdistribution von L^AT_EX – abgesehen von der Tatsache, dass sie zu Beginn speziellen „handgemachten“ Programmcode enthält, der dafür sorgt, dass sie mit `initex` bearbeitet werden kann.

Einrichten von Vor- und Nachspann

Wie bereits erwähnt, schreibt DOCSTRIP nicht nur die ausgewählten Programmzeilen in die Ausgabedateien, sondern stellt ihnen auch eine *Präambel* voran und schließt jede Datei mit einem *Nachspann* ab. Für beide Operationen gibt es Standardtexte, aber in der Regel definiert ein DOCSTRIP-Skript explizit, was an diesen Stellen verwendet werden soll, z.B. ein Urheberrechtsverweis oder die Rechtsverzichtserklärung (siehe auch [108]).

<code>\preamble</code>	<code>\postamble</code>
<code><textzeilen></code>	<code><textzeilen></code>
<code>\endpreamble</code>	<code>\endpostamble</code>

Die Information, die am Anfang der DOCSTRIP-Ausgabedatei hinzugefügt werden soll, sollte zwischen den Befehlen `\preamble` und `\endpreamble` stehen. Die Zeilen, die am Ende hinzugefügt werden sollen, sollten zwischen den Befehlen `\postamble` und `\endpostamble` stehen. Alles, was DOCSTRIP für Präambel und den Nachspann findet, wird in die Ausgabedatei geschrieben, wobei zwei %-Zeichen vorangestellt werden (oder, genauer gesagt, die aktuelle Definition des Befehls `\MetaPrefix`). Im Allgemeinen sollte nur reiner Text verwendet werden. Möglicherweise vorkommende Befehlsnamen, die wortwörtlich ausgegeben werden sollen, gibt man am besten in der Form `\string\foo` ein. Neben der Anwenderpräambel fügt DOCSTRIP auch Informationen über die aktuelle Datei (d.h. ihr Name und die Quelldateien, aus denen sie generiert wurde) hinzu. Diese Information wird immer hinzugefügt, es sei denn, `\nopreamble` wird verwendet (siehe unten), oder man weicht von dem Standardverfahren zur Erstellung der Präambel ab (erläutert in der Dokumentation des DOCSTRIP-Paketes [121]).

Es besteht auch die Möglichkeit, eine Anzahl von „benannten“ Präambeln zu definieren und später beim Erstellen von Dateien auf diese zu verweisen. In der Tat ist dies die übliche Vorgehensweise, um Präambeln in größeren Projekten zu erzeugen.

<code>\declarepreamble\befehl<text>\endpreamble</code>	<code>\usepreamble\befehl</code>
<code>\declarepostamble\befehl<text>\endpostamble</code>	<code>\usepostamble\befehl</code>

Die Deklaration `\declarepreamble` funktioniert wie `\preamble`, mit der Ausnahme, dass sie den Präambeltext zur späteren Verwendung im `\befehl` speichert. Diese Präambel wird aktiviert, indem `\usepreamble` in einem DOCSTRIP-Skript aufgerufen wird. Für den Nachspann gibt es die Deklarationen `\declarepostamble` und `\usepostamble`. Beispiele finden sich in allen DOCSTRIP-Installationsskripten in den Distributionen der L^AT_EX-Standardkomponenten.

<code>\nopreamble</code>	<code>\nopostamble</code>

Die Erstellung einer Präambel oder eines Nachspans kann mit den Deklarationen `\nopreamble` bzw. `\nopostamble` unterdrückt werden.

14.2.3 Installationsunterstützung und Konfiguration

Vor ein paar Jahren entschied sich die T_EX-Anwendergemeinschaft für eine Standardverzeichnisstruktur für T_EX-Installationen (T_EX Directory Structure, TDS), die für alle Plattformen ausgelegt ist, für welche T_EX und L^AT_EX verfügbar sind [165]. Seither wurde dieser Standard bis zu dem Stand weiterentwickelt, der heute von den meisten T_EX-Distributionen verwendet wird.

Um die Integration neuer Pakete in eine TDS-konforme Installation zu erleichtern oder um Paket-Upgrades zu installieren, wurde das DOCSTRIP-Programm erweitert, so dass es unter bestimmten Umständen angewiesen werden kann, die generierten Dateien automatisch an den richtigen Stellen in dieser Struktur zu installieren. Damit dies funktioniert, müssen die DOCSTRIP-Skripte bestimmte Anweisungen enthalten. Zusätzlich muss der Anwender das DOCSTRIP-Programm konfigurieren, indem er eine `docstrip.cfg`-Datei bereitstellt, die für die Installation auf dem aktuellen System geeignet ist.

<code>\usedir{relativer-verzeichnispfad}</code>

Entwickler eines DOCSTRIP-Skripts haben noch eine kleine Zusatzaufgabe: Für jede generierte Datei muss ihre Position im TDS-Verzeichnisbaum bekannt sein, bzw. definiert werden. Diese Information wird im DOCSTRIP-Skript direkt vor dem Aufruf des `\generate`-Befehls oder im Argument von `\generate` vor der nächsten `\file`-Deklaration mit `\usedir` in Form eines Verzeichnispfades angegeben, und zwar relativ zum TDS-Stammverzeichnis. Zum Beispiel gibt die Datei `format.ins` in der Standarddistribution von L^AT_EX an:

```
\usedir{tex/latex/base}
\generate{\file{latex.ltx}{\from{tldirchk.dtx}{initex,2ekernel,dircheck}
                        \from{ltplain.dtx}{2ekernel}
                        ...}
          \file{tracefmt.sty}{\from{ltfsstrc.dtx}{package,trace}}
          \file{flafter.sty}{\from{ltoutput.dtx}{flafter}}
          ...}
```

Dadurch wird die L^AT_EX-Formatdatei (und andere) im korrekten Verzeichnis platziert. In komplexeren Paketen müssen Dateien möglicherweise, je nach ihrem Typ, auf verschiedene Verzeichnisse verteilt werden. So enthält beispielsweise das Installationsskript für das `jurabib`-Paket folgende Anweisungen:

```
\generate{
  \usedir{tex/latex/jurabib}
  \file{jurabib.sty}{\from{jurabib.dtx}{package}}
  \file{dejbbib.ldf}{\from{jurabib.dtx}{german}}
  ...
  \usedir{bibtex/bst/jurabib}
  \file{jurabib.bst}{\from{jurabib.dtx}{jurabst}}
}
```

```

...
\usedir{doc/latex/jurabib}
\file{jbstest.tex}{\from{jurabib.dtx}{test}}
...
}

```

Damit werden die Dateien, die von L^AT_EX benötigt werden, in `tex/latex/jurabib` generiert, die B^BT_EX-Stile in `bibtex/bst/jurabib`, Testdokumente in `doc/latex/jurabib` usw.

Die `\usedir`-Deklaration an sich hat noch keine Auswirkung: DOCSTRIP erzeugt Dateien immer noch ausschließlich im aktuellen Verzeichnis. Um DOCSTRIP zu ermöglichen, solche `\usedir`-Deklationen zu nutzen, muss man ihm eine Konfigurationsdatei (`docstrip.cfg`) zur Verfügung stellen, die eine Deklaration für das Stammverzeichnis des Zielsystems enthält sowie eine Reihe von Übersetzungen in lokale Verzeichnisnamen für die Pfade, die im Argument von `\usedir` angegeben werden.

```

\BaseDirectory{verzeichnis}
\DeclareDir{usedir-pfad}{lokale-übersetzung}

```

Die Deklaration `\BaseDirectory` gibt den absoluten Pfad zum Stammverzeichnis der T_EX-Installation an; weitere Pfade werden dann relativ zu diesem Verzeichnis angegeben. Zusätzlich muss für jeden verwendeten *relativen-verzeichnispfad* im Argument von `\usedir` eine Übersetzung in lokale Verzeichnisnamen angegeben werden. Um DOCSTRIP beispielsweise die von der emT_EX-Distribution verwendete Verzeichnisstruktur beizubringen, ist es möglicherweise nötig, eine Reihe von Deklationen anzugeben, etwa:

```

\BaseDirectory{c:/emtex}
\DeclareDir{tex/latex/base}{texinputs/latex}
\DeclareDir{tex/latex/jurabib}{texinputs/latex}

```

Sobald DOCSTRIP das `\BaseDirectory` kennt, versucht es, alle `\usedir`-Deklationen in seinen Skripten zu interpretieren. Wenn es eine Deklaration findet, für die es keine Übersetzung in lokale Verzeichnisnamen kennt (durch `\DeclareDir`), gibt es eine Meldung aus und erstellt die Datei stattdessen im aktuellen Verzeichnis. Man sollte dann eine passende Deklaration in der `.cfg`-Datei hinzufügen.

Manchmal ist es notwendig, einige Dateien außerhalb des Basisverzeichnisses abzulegen, z.B. wenn sich das B^BT_EX-Programm auf einer anderen Festplatte befindet. In diesem Fall sollte man die Sternform von `\DeclareDir` verwenden, die als zweites Argument einen absoluten Pfadnamen erwartet. Ein Beispiel:

```

\DeclareDir*{bibtex/bst/jurabib}{d:/bibtex/bst}

```

*Installations-
verzeichnisse müssen
vorhanden sein*

Da T_EX keine neuen Verzeichnisse generieren kann, ist es unbedingt erforderlich, dass alle lokalen Verzeichnisse, die mit `\DeclareDir` angegeben werden, tatsächlich existieren. Wenn bei Ausführung eines DOCSTRIP-Skripts

ein Verzeichnis nicht vorhanden ist, wird eine T_EX-Fehlermeldung ausgegeben, die besagt, dass in eine bestimmte Datei nicht geschrieben werden kann und eine andere angegeben werden muss.

Auf einem vollständig TDS-konformen System funktioniert die Übersetzung in lokale Verzeichnisnamen problemlos. Zum Beispiel weisen die folgenden Deklarationen DOCSTRIP an, die Dateien nicht im Hauptverzeichnisbaum, sondern in einem lokalen TDS-Verzeichnisbaum (hier `texmf-local`) zu installieren:

```
\BaseDirectory{/usr/local/lib/texmf-local}
\DeclareDir{tex/latex/base}{tex/latex/base}
\DeclareDir{tex/latex/jurabib}{tex/latex/jurabib}
\DeclareDir{bibtex/bst/jurabib}{bibtex/bst/jurabib}
```

In diesem Fall sollte man sicherstellen, dass der lokale Baum zuerst durchsucht wird.

\UseTDS

Um die Konfiguration zu vereinfachen, die zur Beschreibung einer TDS-konformen Installation notwendig ist, bietet DOCSTRIP die Deklaration `\UseTDS`. Diese weist das Programm an, den `\usedir`-Befehl wortwörtlich zu verwenden, wenn keine explizite `\DeclareDir`-Deklaration angegeben ist. Daher reicht es bei den meisten Installationen aus, in der `.cfg`-Datei eine `\UseTDS`- und eine `\BaseDirectory`-Deklaration anzugeben.

Standardmäßig erstellt DOCSTRIP Dateien nur im aktuellen Arbeitsverzeichnis. Auch wenn eine Konfigurationsdatei angegeben ist, die eine `\BaseDirectory`-Deklaration enthält, schreibt es nur in Verzeichnisse, die explizit mit `\DeclareDir` angegeben werden, oder, wenn `\UseTDS` verwendet wird, in die entsprechenden TDS-Verzeichnisse unterhalb des eigenen Basisverzeichnisses. An anderen Stellen überschreibt es Dateien nicht, auch wenn es (heute, zu Zeiten von *Viren* und anderen hässlichen Kreaturen) klar sein sollte, dass T_EX an sich dazu in der Lage ist und daher ein Sicherheitsrisiko darstellen könnte. Tatsächlich lassen es einige Installationen (z.B. jene auf der T_EX Live CD) nicht zu, dass T_EX in Dateien mit absoluten Pfadnamen oder Dateien schreibt, die mit einem Punkt beginnen, es sei denn, dass dies explizit erlaubt wurde. Auf dem System des Autors muss man beispielsweise Folgendes angeben

*Überlegungen zur
Sicherheit*

```
openout_any=r latex jurabib.ins
```

um die DOCSTRIP-Funktionen für die automatische Installation nutzen zu können.

\maxfiles{zahl} \maxoutfiles{zahl}

Es gibt zwei weitere Deklarationen, die man einer DOCSTRIP-Konfigurationsdatei hinzufügen kann. Auf einigen Betriebssystemen gibt es eine Beschränkung, was die Anzahl der Dateien angeht, die von einem Programm geöffnet werden können. Wenn dies der Fall ist, kann man mit einer `\maxfiles`-

Deklaration die Gesamtanzahl an geöffneten Dateien und mit `\maxoutfiles` die Gesamtanzahl an gleichzeitig geöffneten Ausgabedateien begrenzen (für letztere hat T_EX selbst eine Begrenzung von 16). Diese Deklarationen sollten nur verwendet werden, wenn es nötig ist.

14.2.4 Verwendung mit anderen Programmiersprachen

Mit einigen Einschränkungen ist es möglich, die DOCSTRIP-Funktionen zu verwenden, um Dateien, die nicht für eine T_EX-Installation gedacht sind, zu erstellen und zu verteilen. Man sollte jedoch nicht vergessen, dass DOCSTRIP beim Lesen der Quelldatei zeilenweise arbeitet. Eine Aktion wie das Entpacken einer Binärdatei kann daher nur zu unbrauchbaren Dateien führen.

Ändern der
Kommentarzeichen

Des weiteren ist es wahrscheinlich, dass Präambeln und Nachspann mit den Syntaxanforderungen der Sprache kollidieren, für welche die Datei gedacht ist. Die Erstellung eines Shellskripts mit mehreren Zeilen, die mit `%` beginnen, ist eher nicht empfehlenswert. Dieses Problem kann umgangen werden, indem man `\MetaPrefix` ändert (der standardmäßig `\DoubleperCent` erzeugt). Bei einem Shellskript, in dem wahrscheinlich das `#`-Zeichen als Kommentarzeichen verwendet wird, kann diese Veränderung etwas heikel sein, weil T_EX `#` als Sonderzeichen betrachtet. Eine Möglichkeit, ein einzelnes Hash-Zeichen als ein `\MetaPrefix` zu erzeugen, zeigt folgendes Beispiel:

```
\renewcommand\MetaPrefix{\string##}
```

Zur Standardeinstellung kehrt man mit der folgenden Definition zurück:

```
\renewcommand\MetaPrefix{\DoubleperCent}
```

Unverändert kopieren

Ein weiteres potentiell Problem, auf das man achten muss, sind Zeilen, die mit einem einzelnen Prozentzeichen beginnen; diese entfernt DOCSTRIP standardmäßig. Manchmal möchte man diese Zeilen aber erhalten. Dies lässt sich erreichen, indem man den entsprechenden Block in zwei spezielle Zeilen einschließt, und zwar folgendermaßen:

```
%<<tag-name
    (unverändert zu kopierende Programmzeilen)
%tag-name
```

Man kann einen beliebigen *tag-namen* wählen. Wichtig ist, dass dieser „Verbatim“-Block endet, wenn DOCSTRIP auf eine einzelne Zeile stößt, die ein Prozentzeichen gefolgt vom *tag-namen* enthält. Außerdem ist es wichtig, dass der *tag-name* nicht für einen bedingten Ausschluss oder eine bedingte Einbindung verwendet wird, sondern ausschließlich für die Angabe des Blocks, der unverändert kopiert werden soll. Wenn solch ein Block nur unter bestimmten Umständen geschrieben werden soll, was über das zweite Argument von `\from` gesteuert wird, muss er zusätzlich in ein Paar bedingter Tags eingeschlossen werden (siehe Abschnitt 14.1.5).

14.3 ltxdoc – Eine einfache Dokumentationsklasse

Die ltxdoc-Klasse wurde konzipiert, um die wesentlichen Quelldateien von L^AT_EX zu dokumentieren, die verwendet werden, um das L^AT_EX-Format und alle Pakete zu erstellen, die als Teil der Standarddistribution verteilt werden. Diese Klasse basiert auf der article-Klasse, erweitert diese aber um ein paar Befehle, die für die Dokumentierung des L^AT_EX-Programmcodes hilfreich sind. Außerdem enthält es ein paar Layouteinstellungen, die speziell auf die typischen Anforderungen einer Quelldatei gemäß der doc-Konventionen zugeschnitten sind (zum Beispiel durch eine Zeilenbreite entsprechend 72 Zeichen in Typewriter Fonts und einem breiteren linken Rand, in dem lange Makronamen Platz haben).

Eine spezielle Eigenschaft dieser Klasse ist, dass sie verwendet werden kann, um aus einer größeren Anzahl von Quelldateien im doc-Stil ein einzelnes Dokument zu erstellen. Dies hat den Vorteil, dass man einen kompletten Index der Makroverwendungen über alle Quelldateien generieren kann. Die Treiberdatei `source2e.tex` erzeugt beispielsweise den dokumentierten Quellcode der 40 Dateien, aus denen der L^AT_EX-Kernel besteht. Sie erstellt ein Dokument mit fast 600 Seiten, einschließlich einem Index und einem Änderungsverlauf (der bis in die frühen 90er Jahre zurückgeht).

14.3.1 Von ltxdoc bereitgestellte Erweiterungen

Als Erweiterungen bietet die Klasse eine kleine Auswahl an Befehlen, die L^AT_EX-Befehle und ihre Argumente beschreiben. Diese Befehle sollten tatsächlich im doc-Paket enthalten sein, aber durch einige unglückliche Umstände wurden sie nie zu diesem hinzugefügt.

```
\cmd{\name}   \cs{name}
\margin{arg}  \oarg{arg}  \parg{arg}
```

Der Befehl `\cmd` druckt den Befehl *name* in Typewriter Fonts aus; `\cmd{\foo}` ergibt beispielsweise `\foo`. Im Gegensatz zu der Anweisung `\verb+\foo+` (die ansonsten ähnlich funktioniert) kann er überall verwendet werden – selbst in den Argumenten anderer Befehle. Der Befehl `\cs` bietet dieselbe Funktionalität für all jene, die eine Syntax ohne das Backslash-Zeichen bevorzugen. Tatsächlich ist er sogar leistungsfähiger, weil er auch Befehle setzen kann, die als `\outer` deklariert wurden – ein Low-Level T_EX-Konzept, das normalerweise nicht in L^AT_EX verwendet wird. Des weiteren bewirkt ltxdoc, dass „|“ als Kurzform von `\verb` verwendet wird, so dass man in der Dokumentation einfach `|\foo|` schreiben kann. Wenn dies aus irgendwelchen Gründen nicht erwünscht ist, muss man den Befehl in der Quelldatei (nach `\begin{document}`) mithilfe von `\DeleteShortVerb{|\}` deaktivieren.

Die Befehle `\margin`, `\oarg` und `\parg` erzeugen die L^AT_EX-Syntax für obbligatorische, optionale bzw. Argumente der picture-Umgebung. Die Eingabe

```
\cs{makebox}\parg{x,y}\oarg{pos}\margin{text}
```

erzeugt das (wahrscheinlich wenig bekannte) Syntaxdiagramm für L^AT_EXs `\makebox` in der picture-Umgebung: `\makebox(\langle x,y\rangle)[\langle pos\rangle]{\langle text\rangle}`.

```
\DocInclude{datei}
```

Der Befehl `\DocInclude` ähnelt `\include`, außer dass er `\DocInput` für die *datei* (mit der impliziten Erweiterung `.dtx` oder `.fdd`) verwendet, anstatt `\input` für eine *datei* (mit der impliziten Erweiterung `.tex`) zu verwenden. Dieser Befehl wird in `source2e.tex` verwendet, um alle `.dtx`-Dateien „einzubinden“, die den L^AT_EX-Kernel bilden.

14.3.2 Konfiguration der ltxdoc-Ausgabe

Um Dokumente mit der `ltxdoc`-Klasse anzupassen, kann man eine Konfigurationsdatei erstellen (`ltxdoc.cfg`). Diese Konfigurationsdatei wird immer dann gelesen, wenn die `ltxdoc`-Klasse verwendet wird. Damit kann sie verwendet werden, um das Setzen aller Quelldateien an eigene Anforderungen anzupassen, ohne dass man eine Unmenge an kleinen Treiberdateien bearbeiten muss (was die manuelle Alternative wäre).

Wenn `ltxdoc.cfg` in einem Verzeichnis installiert ist, das standardmäßig von L^AT_EX durchsucht wird, wird sie mit der `ltxdoc`-Klasse auf alle Dokumentationsdateien angewendet. Wenn sie in dem aktuellen Verzeichnis eingefügt wurde, wird sie nur auf Dokumente angewendet, die in diesem Verzeichnis verarbeitet werden.

Die einfachste Form der Anpassung ist, eine oder mehrere Optionen an die `article`-Klasse zu übergeben, auf der `ltxdoc` basiert. Wenn man beispielsweise die gesamte Dokumentation für die Ausgabe auf DIN A4-Papier formatieren möchte, sollte die Zeile

```
\PassOptionsToClass{a4paper}{article}
```

in der Datei `ltxdoc.cfg` hinzugefügt werden. Diese muss dann in einem Verzeichnis platziert werden, das von L^AT_EX durchsucht wird.

Wie in Abschnitt 14.1.2 erörtert, trennt der Befehl `\StopEventually` die Quelldateien in einen Benutzerdokumentations- und einen Implementierungsteil. Um ausschließlich das Benutzerhandbuch zu erstellen, verfügt das `doc`-Paket über den Befehl `\OnlyDescription`, der den Implementierungsteil unterdrückt. Dieser Befehl kann auch in der Konfigurationsdatei verwendet werden. Da aber das `doc`-Paket nach dem Lesen der Konfigurationsdatei geladen wird, muss die Ausführung von `\OnlyDescription` verschoben werden. Am einfachsten gelingt dies mit `\AtBeginDocument`:

```
\AtBeginDocument{\OnlyDescription}
```

Beispielsweise generiert die dokumentierte Quelldatei des `fixltx2e`-Paketes, die Datei `fixltx2e.dtx`, 30 Seiten dokumentierten Programmcodes, wenn

```
latex fixltx2e.dtx
```

ohne Konfigurationsdatei ausgeführt wird. Die meisten Anwender interessieren sich jedoch nicht wirklich dafür, *wie* bestimmte Makros aus dem L^AT_EX-Kernel in diesem Paket korrigiert werden, sondern möchten nur wissen, welche Probleme sich lösen lassen, wenn man es lädt. Mit der obigen

Konfigurationszeile wird die Ausgabe auf ein zehnsseitiges Benutzerhandbuch reduziert, in dem nur die Probleme aufgeführt sind, die gelöst wurden.

Wenn der Treiber `source2e.tex` für die Kerneldokumentation verarbeitet wird, werden standardmäßig Index und Änderungsverlauf erstellt, wobei jedoch für einzelne Dateien kein Index erzeugt wird. Wer zusätzlich an einer ausführlichen Quellcodeliste interessiert ist, möchte möglicherweise auch einen Index erstellen. Auch hier können die vom `doc`-Paket bereitgestellten Indexbefehle verwendet werden, wobei ihre Ausführung wiederum verschoben werden muss. Eine dementsprechend erweiterte Konfigurationsdatei könnte in etwa folgendermaßen aussehen:

```
\AtBeginDocument{\AlsoImplementation % erzwingen, alles zu verarbeiten
                  \CodelineIndex      % Index per Codezeile auswählen
                  \EnableCrossrefs }  % dies aktivieren
\AtEndDocument{\PrintIndex}
```

Wenn ein Änderungsverlauf erstellt werden soll, müssen ähnliche Zeilen hinzugefügt werden. Zu bedenken ist, dass das `doc`-Paket `.idx`- und `.glo`-Dateien mit einer speziellen Syntax erzeugt, die entsprechende Stildateien erfordert. Diese werden mit *MakeIndex* bearbeitet (siehe Abschnitt 14.1.3 auf Seite 849).

14.4 Werkzeuge für die Versionsverwaltung

Wenn man ein Programm entwickelt oder ein großes Dokument, wie ein Benutzerhandbuch oder ein Buch (wie dieses hier) verfasst, spielt die Versionskontrolle eine wichtige Rolle. Darunter versteht man die Aufgabe, ein Softwaresystem, das aus vielen Versionen und Konfigurationen besteht, gut zu organisieren. Das *Revision Control System* ist ein Softwaretool, das bei dieser Aufgabe helfen kann. RCS verwaltet die Versionen von Textdokumenten – insbesondere Quellcode von Programmen, Dokumentation und Testdaten. Es automatisiert das Speichern, Abrufen, Protokollieren und Kennzeichnen von Versionen. Darüber hinaus bietet es Auswahlmechanismen, mit denen Konfigurationen zusammengestellt werden können. Außerdem ist es in der Lage, Informationen zur Versionsverwaltung im Textdokument hinzuzufügen, und zwar in sogenannten *RCS-Feldern*.

Das *Concurrent Versions System* (CVS; siehe <http://www.cvshome.org>), das ursprünglich als Frontend für RCS entwickelt wurde, erweitert die Versionsverwaltung von einer reinen Ansammlung von Dateien in einem einzigen Verzeichnis zu einer hierarchischen Zusammenstellung von Verzeichnissen, die versionskontrollierte Dateien enthalten. Diese Verzeichnisse und Dateien können kombiniert werden, um eine Softwareversion zu erstellen. CVS stellt die notwendigen Funktionen bereit, um diese Softwareversionen zu verwalten und die gleichzeitige Bearbeitung von Quelldateien durch mehrere Softwareentwickler zu steuern.

RCS und CVS bieten eine Schnittstelle zur Schlüsselwort-Ersetzung, in der Felder, die eine bestimmte Struktur aufweisen, mit Verwaltungsinformationen aktualisiert werden, sobald eine Datei in das System eing检echt wird. Die wichtigsten Schlüsselwörter sind `$Author$` (Benutzerkonto der

Person, die eincheckt), `$Date$` (Datum und Uhrzeit des Eincheckens in UTC), `$Id$` (Kombinationsfeld aus Dateiname, Version, Datum, Uhrzeit, Autor, Status und, optional, gesperrt durch), `$RCSfile$` (Archivdatei ohne Pfadname), `$Revision$` (Versionsnummer, die der Version zugewiesen ist) und `$Source$` (vollständiger Pfadname der Archivdatei). Anfangs werden in der Quelldatei einfach ein oder mehrere dieser Schlüsselwörter hinzugefügt (z.B. `$Id$`). Beim ersten Einchecken werden diese durch die Struktur `$\langle\textit{schlüsselwort}\rangle:\langle\textit{wert}\rangle$` ersetzt, wie das nächste Beispiel zeigt. Bei weiteren Eincheck-Vorgängen wird der `\langle\textit{wert}\rangle`, wenn erforderlich, aktualisiert.

Wenn man L^AT_EX-Dokumente einer Versionsverwaltung unterzieht, wird man häufig auf die Daten der RCS-Felder im Dokument zugreifen wollen – vielleicht um das Datum des letzten Eincheckens und die Versionsnummer in die lebenden Kolumnentitel einzufügen. Da die Syntax Dollarzeichen verwendet (die in L^AT_EX Formeln kennzeichnen), kann man die Schlüsselwörter nicht direkt im Text verwenden. Es gibt Pakete, die L^AT_EX-Tags bereitstellen, mit denen auf diese Daten auf eine Weise zugegriffen werden kann, die für den Satz geeignet ist.

14.4.1 rcs – Zugriff auf einzelne Schlüsselwörter

Das rcs-Paket, das von Joachim Schrod geschrieben wurde, ermöglicht es, aus beliebigen Schlüsselwortfeldern RCS-Daten zu extrahieren und die Daten zur späteren Verwendung in Befehlsnamen zu speichern.

<code>\RCS \$\textit{schlüsselwort}\$</code>	<code>\RCS \$\textit{schlüsselwort}:\langle\textit{wert}\rangle\$</code>
<code>\RCSdef \$\textit{schlüsselwort}\$</code>	<code>\RCSdef \$\textit{schlüsselwort}:\langle\textit{wert}\rangle\$</code>

Der `\RCS`-Befehl analysiert eine Zeichenfolge, die durch Dollarzeichen begrenzt ist, auf ein *schlüsselwort* und den zugehörigen *wert* hin. Er kann die beiden oben gezeigten Varianten erkennen. Aus dem *schlüsselwort* konstruiert er einen Befehlsnamen `\RCSschlüsselwort`, der später benutzt werden kann, um den *wert* abzurufen. Das *schlüsselwort* kann eine beliebige Zeichenfolge sein, die nur aus Buchstaben besteht, die in einem Befehlsnamen verwendet werden können. Auf diese Weise ist man nicht auf die oben genannten RCS-Schlüsselwortnamen beschränkt (das RCS/CVS-System aktualisiert jedoch nur diese Schlüsselwörter automatisch). Der `\RCSdef`-Befehl funktioniert wie `\RCS`, gibt aber zusätzlich das Schlüsselwort und den Wert auf dem Bildschirm aus.

Im nächsten Beispiel werden vier typische Schlüssel ausgelesen und ihre Werte später im Text gesetzt. Da alle Beispiele in diesem Buch aus dem Quellcode des Buches automatisch generiert werden (siehe Seite 171), entsprechen die Werte, die hinter den Schlüsselwörtern zu sehen sind, jenen, die in der Datei für dieses Kapitel verwendet werden.

```
\selectlanguage{ngerman} \usepackage{rcs}
\RCS $Date: 2007/01/08 22:34:19 $ \RCS $Author: frank $
\RCS $RCSfile: ch14.tex,v $ \RCS $Revision: 1.44 $

Die Datei \RCSRCSfile{} hat die Versionsnummer
\RCSRevision. Das letzte Einchecken erfolgte durch
\RCSAuthor{} am \RCSDate{} um \RCSTime\, \textsc{utc}.
```

Die Datei `ch14.tex,v` hat die Versionsnummer 1.44. Das letzte Einchecken erfolgte durch frank am 8. Januar 2007 um 22:34:19 UTC.

Bsp.
14-4-1

Bei genauerer Betrachtung dieses Beispiels kann man sehen, dass `\RCSDate` den Wert von `$Date$` (welcher aus einem numerischen Datenformat und der Zeitangabe besteht) nicht reproduziert, sondern stattdessen eine Datumsangabe erzeugt, die der von `\today` erzeugten verdächtig ähnlich sieht. Was genau passiert, ist Folgendes: Der Wert wird intern geparkt, und das Datum des Auscheckens im von `\today` verwendeten Format wird in `\RCSDate` gespeichert. Auf diese Weise können sprachspezifische Pakete (z.B. aus dem babel-System) ihre eigenen Methoden zur Darstellung des Datums bereitstellen.

Damit dies funktioniert, muss allerdings die Bedeutung von `\today` seine endgültige Bedeutung besitzen *bevor* das Paket `rcs` geladen wird; im Beispiel wurde deshalb ein `\selectlanguage` Befehl am Anfang eingefügt, um die Sprache in babel einzustellen.

Für Schlüsselwörter, deren Werte weiter bearbeitet werden, wird der ursprüngliche Wert automatisch im Befehl `\RCSRawschlüsselwort` (zum Beispiel `\RCSRawDate`) bereitgestellt. Es besteht die Möglichkeit, für andere Schlüsselwörter eigene Bearbeitungsroutrinen bereitzustellen; wie dies geht, wird in der Paketdokumentation (`rcs-user.tex`) beschrieben.

Zur bequemeren Handhabung bietet das Paket ein paar zusätzliche Befehle. Eine syntaktische Analyse von `$Date$` kann z.B. mit dem Befehl `\RCSdate` (klein geschriebenes „d“) anstelle des oben verwendeten Befehls `\RCS` durchgeführt werden. Dies ist äquivalent zum Aufruf von

```
\RCS $Date: 2007/01/08 22:34:19 $      \date{\RCSDate}
```

Das letzte Datum des Eincheckens wird damit automatisch als Datum im Dokumententitel verwendet.¹ Natürlich ist der Befehl `\RCSDate` immer noch für andere Verwendungen verfügbar.

Eine weitere Alternative zu `\RCS` besteht darin, den Befehl `\RCSID` zum Parsen eines Schlüsselwortes zu verwenden. Er richtet nicht nur den entsprechenden `\RCSschlüsselwort`-Befehl ein, der den Wert erhält, sondern setzt das Schlüsselwort und den Wert unverändert in den Kolumnenfuß. Dieser Befehl kann höchstens einmal verwendet werden (weil bei jedem Aufruf die Fußzeile überschrieben wird), er wird am besten mit dem Schlüsselwort `$Id$` oder `$Header$` kombiniert.

Da das `rcs`-Paket L^AT_EXs Schnittstelle für das Kolumnentitel-Layout mehr oder weniger umgeht, funktioniert der Befehl nicht, wenn man in der Quelldatei `\pagestyle`-Befehle verwendet, welche den lebenden Kolumnentitel aktualisieren. In diesem Fall sollte `\RCS` verwendet werden und die relevante Information mithilfe der Verfahren und Pakete, die in Abschnitt 4.4 beschrieben sind, manuell dem Kolumnentitel hinzugefügt werden.

Das Paket enthält auch Code, um die Änderungsverläufe von RCS zu setzen, die mit dem Schlüsselwort `$Log$` generiert werden können. Dies wird für die meisten Leser jedoch nicht von Nutzen sein, weil es eine spezielle RCS-Version erfordert und nicht mit CVS funktioniert. Bei Interesse sollte der Leser die Paketdokumentation konsultieren.

¹Häufig sieht man in Dokumenten `\date{\today}`; dies ist aber meistens schlecht, weil damit nicht das Datum der letzten Änderung, sondern das Datum des letzten Formatierungslaufes erzeugt wird.

14.4.2 rcsinfo – Parsen des Schlüsselwortes \$Id\$

Im Gegensatz zum rcs-Paket, das jede Zeichenfolge bearbeitet, die der RCS/ CVS-Schlüsselwort-Syntax entspricht, konzentriert sich das rcsinfo-Paket von Jürgen Vollmer auf ein einziges Schlüsselwort, nämlich \$Id\$.

```
\rcsInfo $Id$      \rcsInfo $Id:␣wert␣$
```

Wenn ein *wert* vorhanden ist, parst der `\rcsInfo`-Befehl diesen und speichert alle darin enthaltenen Informationen zur späteren Verwendung in einer Reihe von Befehlen ab. Andernfalls fügt er in diese Befehle Standardwerte ein – im Falle des Datumswertes das aktuelle Datum des L^AT_EX-Laufs und für alle anderen fehlenden Werte Texte wie `--owner--`.

Das folgende Beispiel zeigt alle Befehle, die vom Paket angeboten werden, zusammen mit den entsprechenden Ausgaben. Wie man sehen kann, hängt `\rcsInfoLongDate` von der aktuellen Sprache ab. Hier wird das italienische Format ausgegeben.

	<code>\usepackage[italian]{babel} \usepackage{rcsinfo}</code>	
	<code>\rcsInfo \$Id: ch14.tex,v 1.44 2007/01/08 22:34:19 frank Exp \$</code>	
ch14.tex 1.44	<code>\rcsInfoFile \quad \rcsInfoRevision</code>	<code>\par</code>
2007/01/08 22:34:19	<code>\rcsInfoDate \quad \rcsInfoTime</code>	<code>\par</code>
8 gennaio 2007	<code>\rcsInfoLongDate</code>	<code>\par</code>
2007 1 8	<code>\rcsInfoYear \quad \rcsInfoMonth \quad \rcsInfoDay</code>	<code>\par</code>
frank Exp --not-locked--	<code>\rcsInfoOwner \quad \rcsInfoStatus \quad \rcsInfoLocker</code>	

Bsp.
14-4-2

Das Paket enthält ein paar Optionen, mit denen sein Verhalten geändert werden kann:

today/notoday Standardmäßig ändert `\rcsinfo` L^AT_EXs interne Datumsinformation (Zeitpunkt des aktuellen L^AT_EX-Laufs) in die erhaltene „check-in“ Information um. Benutzt man den Befehl `\today` im Dokument, generiert dieser somit eine Datumsangabe basierend auf dieser Information. Wenn `notoday` verwendet wird, erzeugt `\today`, wie auch in anderen Dokumenten, eine Datumsangabe, die das Datum des L^AT_EX-Laufs anzeigt.

fancyhdr/nofancy Wenn `fancyhdr` angegeben wird, gibt das rcsinfo-Paket eine Anzahl von `fancyhdr`-Deklarationen aus, die in der Fußzeile einen lebenden Kolumnentitel einrichten. Da man trotzdem noch eine eigene Definition für den lebenden Kolumnentitel angeben und alles mit `\pagestyle{fancy}` aktivieren muss, ist es vielleicht sogar besser, den gesamten Prozess manuell zu steuern und alles selbst einzurichten.

long/short Diese Option funktioniert nur, wenn die Option `fancyhdr` verwendet wird. Sie entscheidet, ob eine lange (voreingestellt) oder eine kurze Datumsangabe in der Fußzeile verwendet wird.

Für jene, die ihre L^AT_EX-Dokumente mithilfe des `latex2html`-Programms nach HTML umwandeln wollen [59, Kapitel 3], bietet `rcsinfo` direkte Unterstützung in Form einer perl-Datei, `rcsinfo.perl`; diese Datei muss in der `latex2html`-Installation im richtigen Verzeichnis eingefügt werden. Nähere Informationen findet man im rcsinfo-Handbuch.

A n h a n g A

Präambeln, Pakete und Klassen in L^AT_EX – Ein Überblick

A.1 Verknüpfen von Markup und Formatierung	873
A.2 Seiten-Markup – Boxen und Linien	893
A.3 Erweiterte Steuerfunktionen	904
A.4 Aufbau von Paket- und Klassendateien	910

Dieser Anhang gibt einen Überblick darüber, welche Konzepte der Programmgestaltung in L^AT_EX zugrunde liegen. Hier wird erläutert, wie man neue Befehle und Umgebungen definiert, darunter auch solche mit einem optionalen Argument. Des Weiteren wird besprochen, wie L^AT_EX mit Zählern umgeht und diese darstellt; außerdem erfolgt eine Einführung in horizontale und vertikale Längenparameter und deren Handhabung. Der zweite Abschnitt befasst sich mit dem wichtigen Thema der (L^A)T_EX-Boxen und ihrem Gebrauch. Ein gutes Verständnis dieser Thematik ist wichtig, um die in diesem Buch präsentierten Informationen wirklich schätzen und einsetzen zu können. Der dritte Abschnitt ist den beiden Paketdateien `calc` und `ifthen` gewidmet, die Berechnungen und das Erstellen von Steuerfunktionen mit L^AT_EX erleichtern. Sie wurden in vielen Codebeispielen in diesem Buch verwendet. Abschließend folgt eine detaillierte Beschreibung der L^AT_EX 2_ε-Schnittstelle, mit deren Hilfe man eigene Paket- und Klassendateien mit Optionen definieren kann.

A.1 Verknüpfen von Markup und Formatierung

Dieser Abschnitt erläutert die L^AT_EX-Syntax zum Definieren von Befehlen und Umgebungen. Es ist wichtig, dass man ausschließlich die im Folgenden beschriebenen L^AT_EX-Konstrukte verwendet und nicht die Low-Level T_EX-Befehle.

Zum einen lassen sich so die Vorteile der $\text{\LaTeX}$ -Konsistenzprüfung nutzen, zum anderen sind so definierte Befehle (wahrscheinlich) unverändert in zukünftigen $\text{\LaTeX}$ -Versionen verwendbar.

A.1.1 Befehls- und Umgebungsnamen

Befehle

In der aktuellen $\text{\LaTeX}$ -Version kann man Akzentbuchstaben und andere Nicht-ASCII-Zeichen direkt in die Quelldatei eingeben, so dass man annehmen möchte, dass sie auch in Befehls- und Umgebungsnamen zulässig sein sollten (z.B. `\größer`). Dies ist jedoch nicht der Fall: Mehrbuchstabige $\text{\LaTeX}$ -Befehlsnamen müssen aus den einfachen ASCII-Buchstaben bestehen, also `a...z` und `A...Z`¹. Das heißt, `\vspace*` ist eigentlich an sich kein Befehl, sondern besteht aus dem Befehl `\vspace` gefolgt von dem Modifikator `*`. Rein technisch gesehen könnte man auch `\vspace_*` schreiben (da das Leerzeichen ignoriert wird) oder sogar den `*` in die nächste Zeile des Dokumentes setzen.²

Umgebungen

Umgebungsnamen hingegen werden anders gehandhabt. In ihrem Falle ist der `*` sehr wohl Bestandteil des Namens und ein vorangehendes Leerzeichen wird nicht ignoriert. Schreibt man also `\begin{figure_*}`, dann wird das Leerzeichen als Bestandteil des Namens angesehen und der Beginn der `figure*`-Umgebung nicht als solcher erkannt. Der Unterschied rührt von gewissen Einzelheiten bei der Implementierung und scheint darauf hinzudeuten, dass Umgebungsnamen noch einige zusätzliche ASCII-Zeichen enthalten dürfen, wie z.B.:

```
\newenvironment{foo.bar:baz_mit_leerzeichen}{}{}
```

Das lässt sich jedoch nicht verallgemeinern, da solche Umgebungsnamen, je nachdem welche zusätzlichen Pakete geladen sind, nicht mehr erkannt werden oder seltsame Fehler erzeugen. Daher sollte man sich besser nicht auf diese (Fehl-)Funktion einlassen, sondern sich an die offiziell unterstützten Namenskonventionen halten, in denen nur Klein- und Großbuchstaben sowie das Sternsymbol erlaubt sind.

Zitier- und Labelschlüssel

Genau genommen gelten die gleichen Regeln auch für `\cite` und `\label`-Schlüssel. Es hat sich jedoch allgemein durchgesetzt, Doppelpunkte in Schlüsseln zu verwenden (z.B. `sec:cmds`), so dass die meisten Pakete zumindest den Einsatz von Doppelpunkten in solchen Schlüsseln unterstützen. Zeichen außerhalb des ASCII-Bereiches und $\text{\LaTeX}$ -Syntaxzeichen (wie `_` oder `#`) dürfen niemals in Namen vorkommen, egal ob es sich um Schlüssel, Zähler, Umgebungen oder Befehlsnamen mit mehreren Zeichen handelt.

Bei Befehlsnamen mit nur einem Zeichen sieht das schon wieder anders aus: Hier darf jedes (einzelne) Zeichen verwendet werden. So ist beispielsweise `\$` ein gültiger $\text{\LaTeX}$ -Befehl, während `\foo$bar` als Befehl `\foo` interpretiert würde, hinter dem eine mathematische Formel beginnt (angezeigt durch `$`) gefolgt von den (mathematischen) Zeichen `b`, `a` und `r`. Jeder weitere Text würde dann ebenfalls im Formelmodus gesetzt.

¹Genau genommen ist das nicht ganz richtig, da man $\text{\TeX}$ so konfigurieren kann, dass es auch andere Konstellationen unterstützt. Es gibt jedoch gute Gründe, warum das in Standard- $\text{\LaTeX}$ nicht geschieht. Einige dieser Gründe wurden in Abschnitt 7.11 im Zusammenhang mit dem $\text{\LaTeX}$ -Kodierungsmodell besprochen.

²Es gilt als schlechter Stil, diese Form in einem Dokument einzusetzen, man kann aber leider niemanden daran hindern.

L^AT_EX-Befehle (also diejenigen Konstrukte, die mit einem Backslash beginnen) lassen sich in drei Grundkategorien einteilen: Dokumentenbefehle, Paket- und Klassenbefehle und interne „Kernel“-Befehle.

Dokumentenbefehle, wie etwa `\section`, `\emph` und `\sum`, haben normalerweise (möglichst) kurze, kleingeschriebene Namen.

Dokumentenbefehle

Klassen- und Paketbefehle haben üblicherweise längere Namen, bestehend aus Klein- und Großbuchstaben, wie etwa `\InputIfFileExists` und `\RequirePackage`. Einige von ihnen können auch im Quelltext noch genutzt werden, aber viele funktionieren nicht mehr, sobald `\begin{document}` verarbeitet wurde.

Klassen- und Paketbefehle

Die meisten internen L^AT_EX-Befehle, wie z.B. `\@tempcnta`, `\@ifnextchar` und `\z@` enthalten ein @ im Befehlsnamen. Dadurch ist sichergestellt, dass Anwender diese Namen nicht versehentlich für eigene Befehle verwenden. Das bedeutet jedoch auch, dass sie in Dokumenten, selbst in der Präambel, nur unter besonderen Vorsichtsmaßnahmen verwendet werden können.

Interne L^AT_EX-Befehle

Ein paar Beispiele in diesem Buch machen deutlich, dass man trotzdem hin und wieder „internen Programmcode“ in die Präambel aufnehmen muss. Dazu benötigt man die Befehle `\makeatletter` und `\makeatother` – werden diese vergessen, so kann das einige seltsame Fehler nach sich ziehen. Auf Seite 884 ist zu sehen, wie man sie verwendet. Man beachte, dass Paket- und Klassendateien diese Befehle auf *keinen Fall* enthalten dürfen: Der Befehl `\makeatletter` wird dort nicht benötigt, da er vor dem Lesen solcher Dateien ohnehin aktiviert wird; und `\makeatother` würde dieses Verhalten vorzeitig stoppen und damit ein absolutes Chaos anrichten.

 *Vorsicht bei internen Befehlen!*

Leider lassen sich die Befehlskategorien aus historischen Gründen oftmals nicht ganz klar unterscheiden. So ist z.B. `\hbox` ein interner Befehl, der vorzugsweise nur im L^AT_EX-Kern eingesetzt werden sollte, während es sich bei `\m@ne` um die Konstante -1 handelt, die genauso gut `\MinusOne` hätte heißen können.

Trotzdem ist diese Faustregel nützlich: Wenn ein Befehlsname ein @-Zeichen enthält, dann gehört er nicht zu den Anwenderbefehlen von L^AT_EX und damit kann sich sein Verhalten in zukünftigen Versionen ändern! Solche Befehle sollten immer nur mit Bedacht verwendet werden. Befehle mit Groß- und Kleinbuchstaben und die im *L^AT_EX Manual* [106] beschriebenen werden hingegen auch in zukünftigen Versionen von L^AT_EX 2_ε garantiert weiterhin unterstützt.

A.1.2 Definieren neuer Befehle

Oftmals lohnt es sich, neue Befehle zu definieren, z.B. zur Darstellung häufig verwendeter Zeichenfolgen oder wiederholt auftretender Befehlskombinationen. Ein neuer Befehl wird mithilfe des `\newcommand`-Befehls definiert, der über ein optionales Argument verfügt. Dieses gibt die Anzahl der Argumente des neuen Befehls an.

```
\newcommand{befehl}[narg]{befehlsdefinition}
```

Die Anzahl der Argumente liegt im Bereich $0 \leq narg \leq 9$. Verfügt der neue Befehl nicht über Argumente, so kann man die `[0]` weglassen. Innerhalb des

Teils *befehlsdefinition* werden die Argumente als #1 bis #narg aufgeführt.

PostScript und seine Variante Encapsulated PostScript werden häufig verwendet, um Graphiken in L^AT_EX-Dokumente einzubinden ...

```
\newcommand{\PS}{Post\~Script}
\newcommand{\EPS}{Encapsulated \PS}
\PS{} und seine Variante \EPS{} werden häufig
verwendet, um Graphiken in \LaTeX{-Dokumente
einzubinden \ldots
```

Bsp.
A-1-1

Argumentklammern
weglassen

Das Argument *befehl* darf immer nur ein einzelnes „Token“ enthalten (den Namen des zu definierenden Befehls), so dass man die geschweiften Klammern hier weglassen kann. Zwar ist die Nutzung dieser Syntaxeigenschaft von T_EX sonst nicht zu empfehlen, bei \newcommand und ähnlichen Deklarationen ist diese Schreibweise jedoch durchaus üblich. Auch in diesem Buch wurde häufig die kürzere Syntax verwendet:

```
\newcommand\PS {Post\~Script}
\newcommand\EPS{Encapsulated \PS}
```

Man muss jedoch beachten, dass diese Schreibweise nur bei Argumenten möglich ist, die für T_EX auch einzelne Tokens sind (also Namen, die mit einem Backslash beginnen). Bei Umgebungs- oder Zählernamen wird das Verfahren fehlschlagen. So ist etwa die Syntax des Beispiels

```
\setcounter mycount {5}
\newenvironment myenv{...}{...}
```

in L^AT_EX ungültig.

Soll ein Befehl sowohl in Formeln als auch in Texten verwendet werden, dann muss man bei seiner Definition besondere Sorgfalt walten lassen. Man könnte natürlich beispielsweise \mbox benutzen, aber das wäre mit einer Reihe von Nachteilen verbunden.

Die Reihe $x_1, \dots, x_n$ oder $x_1, \dots, x_n + G_{x_1, \dots, x_n}$

```
\newcommand\xvec{\mbox{\$x_1,\ldots,x_n\$}}
Die Reihe \xvec\ oder $\xvec+G_{\xvec}$
```

Bsp.
A-1-2

Eine bessere Lösung bietet der L^AT_EX 2_ε-Befehl \ensuremath. Wie der Name schon sagt, sorgt \ensuremath dafür, dass sein Argument immer im Formelmodus gesetzt wird. Falls erforderlich umgibt er es dazu mit \$-Zeichen. Die zuvor gezeigte Definition sollte also durch die folgende ersetzt werden:

Die Reihe $x_1, \dots, x_n$ oder $x_1, \dots, x_n + G_{x_1, \dots, x_n}$

```
\newcommand\xvec{\ensuremath{x_1,\ldots,x_n}}
Die Reihe \xvec\ oder $\xvec+G_{\xvec}$
```

Bsp.
A-1-3

Diese Schreibweise bietet den zusätzlichen Vorteil, dass die Symbole auch in Hoch- und Tiefstellungen stets in der richtigen Größe erscheinen, was bei der Definition mit \mbox nicht der Fall ist.

Bereits vorhandene Befehle müssen mit dem Befehl \renewcommand umdefiniert werden, dessen Syntax ansonsten der von \newcommand entspricht. Durch Umdefinieren kann man auch die Anzahl der Argumente eines Befehls

ändern. Man könnte also den Befehl `\xvec` aus dem letzten Beispiel so umdefinieren, dass er über ein Argument verfügt:

Bsp.
A-1-4

Die Reihe $x_1, \dots, x_n$ oder

$x_1, \dots, x_n + G_{x_1, \dots, x_n}$

Die Reihe $x_1, \dots, x_n$ oder

$x_1, \dots, x_k + G_{x_1, \dots, x_k}$

`\raggedright`

`\newcommand\xvec{\ensuremath{x_1,\ldots,x_n}}`

Die Reihe `\xvec\` oder `$(\xvec+G_{\xvec})$ \par`

`\renewcommand\xvec[1]{\ensuremath{x_1,\ldots,x_{\#1}}}`

Die Reihe `\xvec{n}` oder `$(\xvec{k}+G_{\xvec{k}})$`

Beim Umdefinieren eines Befehls (oder einer Umgebung, wie im Folgenden noch beschrieben) muss man natürlich vorsichtig sein. Die entsprechenden Befehle könnten schon in einer Klasse oder einem Paket vorkommen, die man geladen hat: Man versuche nur einmal den Befehl `\uppercase` in einem Dokument umzudefinieren, das mithilfe der Klasse `book` gesetzt wird.

Befehle mit einem optionalen Argument

In $\text{\LaTeX}$ kann man Befehle auch so definieren, dass ihr erstes Argument optional ist, und zwar mit folgender Syntax:

```
\newcommand{befehl}[narg][voreinst]{befehlsdefinition}
```

Eine solche Befehlsdefinition kann z.B. folgendermaßen aussehen:

```
\newcommand\LB[1][3]{\linebreak[#1]}
```

Die Voreinstellung für das optionale Argument befindet sich im zweiten Paar eckiger Klammern – in diesem Falle das Zeichen „3“. Innerhalb der Befehlsdefinition hat das optionale Argument die Nummer #1, während etwaige obligatorische Argumente mit den Nummern #2 bis #narg adressiert werden. Gibt man also `\LB` ein, so ist das eine Kurzform für `\linebreak[3]`, während `\LB[2]` den tatsächlich angegebenen Wert verwendet. Man erhält also den gleichen Effekt, als würde man `\linebreak[2]` eingeben.

Im nächsten Beispiel wird der Befehl `\lvec` definiert, den man (aufgrund von `\ensuremath`) innerhalb und außerhalb von Formeln verwenden kann. Unter der Annahme, dass die obere Grenze normalerweise n ist, muss diese nur optional angegeben werden, während die Vektorvariable explizit angegeben werden muss.

Für die Reihe $x_1 + \dots + x_n$ gilt

$$x_1 + \dots + x_n = \sum_{k=1}^n G_{y_1 + \dots + y_k}$$

Bsp.
A-1-5

`\newcommand\lvec[2][n]`

`{\ensuremath{\#2_1+\cdots+\#2_{\#1}}}`

Für die Reihe `\lvec{x}` gilt

`[\ \lvec{x} = \sum_{k=1}^n G_{\lvec{k}{y}} \ ]`

Im Allgemeinen empfiehlt es sich, den Fall, der am häufigsten auftritt, durch die Befehlsform ohne Argument abzudecken und die seltener verwendeten Fälle durch die Form mit optionalem Argument.

Einschränkungen für Argumente

Wie bereits erklärt, können benutzerdefinierte Befehle über ein optionales Argument und insgesamt über bis zu neun Argumente verfügen. Wurde der Befehl mittels `\newcommand` definiert, kann bei seiner Verwendung jedes dieser Argumente einen beliebigen Text aufnehmen. Dabei gelten jedoch ein paar Einschränkungen:

- Alle geöffneten geschweiften Klammern müssen auch wieder geschlossen werden, da $\text{\LaTeX}$ sonst nicht feststellen kann, wo das Argument endet.
- Der Befehl `\verb`, die Umgebung `verbatim` und verwandte Befehle oder Umgebungen werden in Argumenten nicht unterstützt.
- In einem optionalen Argument müssen schließende eckige Klammern „]“ immer in geschweiften Klammern verborgen sein (so ist `\item[{a}]` beispielsweise zulässig). Ohne diese geschweiften Klammern würde die Klammer `]` fälschlicherweise als Ende des optionalen Argumentes interpretiert.

*Argumentinhalte
gezielt einschränken*

Der zulässige Inhalt für ein Argument lässt sich gezielt weiter einschränken, indem man die `\newcommand*`-Variante der Deklaration verwendet.

```
\newcommand*{befehl}[narg][voreinst]{befehlsdefinition}
```

Die Sternform funktioniert wie `\newcommand`, definiert jedoch einen *befehl*, der aus Sicht von $\text{\TeX}$ nicht `\long` ist. Das heißt, der neue Befehl akzeptiert keine Leerzeilen oder `\par`-Befehle in seinen Argumenten. Diese Einschränkung kann bei Befehlen nützlich sein, deren Argumente nicht für ganze Textabsätze gedacht sind.

*Verhältnis zu
 $\text{\TeX}$ -Basisbefehlen*

Befehle, die mithilfe des $\text{\TeX}$ -Basisbefehls `\def` definiert wurden, akzeptieren ohnehin keine `\par`-Befehle in ihrem Argument. In dieser Hinsicht entsprechen sie also mit `\newcommand*` definierten Befehlen. Die Low-Level Entsprechung zu `\newcommand` ist die $\text{\TeX}$ -Befehlsfolge `\long\def`.

Verschachteln neuer Befehle

Manchmal muss man Befehlsdefinitionen verschachteln, vor allem, wenn Befehlskombinationen als Teil einer neuen Umgebung definiert werden. Sofern der innere Befehl (bzw. die Umgebung) über Argumente verfügt, lassen sich diese nicht ohne Weiteres adressieren. Hier kann man natürlich nicht `#1`, `#2` usw. verwenden, da sich diese bereits auf die Argumente des äußeren Befehls bzw. der äußeren Umgebung beziehen. In $\text{\TeX}$ wird dieses Problem durch eine Verdoppelung der Doppelkreuze gelöst: `##1` bezieht sich also auf das erste Argument der inneren Definition. Bei drei verschachtelten Definitionen benötigt man entsprechend die Schreibweise `###1`.

Um dieses abstrakte Konzept etwas anschaulicher zu machen, wird im Folgenden der Befehl `\DEF1vec` definiert, welcher den Befehl `\1vec` aus Beispiel A-1-5 auf der vorherigen Seite immer wieder (um)definiert. Als erstes Argument erhält `\DEF1vec` den Vektornamen, der nun fest in die geänderte

Definition von `\lvec` integriert wird. Als zweites Argument dient der obere Index, der hier die Voreinstellung für das optionale Argument von `\lvec` sein wird. Da der Vektorname also nun Teil der Definition ist, verfügt `\lvec` nur über ein optionales Argument.

```
\newcommand\lvec{}
\newcommand\DEFlvec[2]{%
  \renewcommand\lvec[1][#2]%
    {\ensuremath{#1_1+\cdots
      + #1_{##1}}}%
}
\DEFlvec{x}{n} % Initialisierung
Zunächst:  $\lvec \neq \lvec[k]$  \par
\DEFlvec{y}{i} Nun:  $\lvec \neq \lvec[k]$ 
```

Bsp.
A-1-6

Zunächst: $x_1 + \cdots + x_n \neq x_1 + \cdots + x_k$
Nun: $y_1 + \cdots + y_i \neq y_1 + \cdots + y_k$

Es lohnt sich, das im letzten Beispiel angewandte Verfahren einmal genauer zu betrachten. Dazu versucht man am besten, sich die tatsächlich vorgenommenen Definitionen vorzustellen, z.B. wenn die „Initialisierung“ durchgeführt wird. Man beachte auch, dass eine Definition von `\lvec` auf oberster Ebene erforderlich ist: Ihre genaue Form ist irrelevant, aber ohne sie könnte der Befehl nicht innerhalb des `\DEFlvec`-Befehls „undefiniert“ werden.

Besondere Paket- und Klassendeklarationen

Neben `\newcommand` und `\renewcommand`, die ursprünglich als Benutzerbefehle gedacht waren (z.B. für die Dokumentenpräambel), bietet L^AT_EX noch einige andere Möglichkeiten, Befehle für Klassen- und Paketdateien zu definieren bzw. umzudefinieren.

```
\providecommand*{befehl}[narg][voreinst]{befehlsdefinition}
```

Diese Deklaration funktioniert genau wie `\newcommand` und `\newcommand*`, bis auf die Tatsache, dass sie einfach ignoriert wird, wenn der zu definierende Befehl bereits existiert. Eine solche Eigenschaft ist hilfreich bei Quellcode, der in mehreren Dokumenten verwendet wird, wie etwa bibliographische Einträge. So kann man beispielsweise in der `@preamble` von Bib_TE_X anstelle von `\newcommand` den Befehl `\providecommand` verwenden, um Logos oder andere Befehle zu definieren, die in den Bib_TE_X-Einträgen verwendet werden. Damit lassen sich Fehlermeldungen vermeiden, falls diese Befehle bereits im Dokument definiert sind.

```
\DeclareRobustCommand*{befehl}[narg][voreinst]{befehlsdefinition}
```

Dieser Befehl verfügt über die gleichen Argumente wie `\newcommand` und `\newcommand*`, deklariert jedoch einen „robusten“ Befehl, auch wenn Teile des Codes innerhalb der `befehlsdefinition` zerbrechlich sind. Mithilfe dieses Befehls kann man neue robuste Befehle definieren oder existierende Befehle zu robusten umdefinieren. In der Protokolldatei wird vermerkt, dass der genannte `befehl` umdefiniert wurde und es erfolgt keine Fehlermeldung.

```
\CheckCommand*{befehl}[narg][voreinst]{befehlsdefinition}
```

Auch dieser Befehl verfügt über die gleichen Argumente wie `\newcommand` und `\newcommand*`. Anstatt den angegebenen *befehl* zu definieren überprüft er jedoch, ob dessen aktuelle Definition mit derjenigen im Argument *befehlsdefinition* übereinstimmt. Weichen die Definitionen voneinander ab, oder akzeptiert eine Version `\par`-Befehle in ihrem Argument und die andere nicht, so wird eine Fehlermeldung ausgegeben. Dieser Befehl eignet sich zum Überprüfen des Systemstatus, bevor ein Paket damit beginnt Befehlsdefinitionen zu verändern. So kann man insbesondere kontrollieren, ob ein Befehl bereits von einem anderen Paket undefiniert wurde.

A.1.3 Definieren neuer Umgebungen

Umgebungen lassen sich mit den Deklarationen `\newenvironment` und `\renewenvironment` definieren bzw. undefinieren. Für jede der beiden Deklarationen muss angegeben werden, was geschehen soll, wenn eine Umgebung beginnt und wenn sie wieder endet. Bei einer Umgebung namens „myenv“ werden diese Vorgänge innerhalb des Dokumentes durch die Befehle `\begin{myenv}` und `\end{myenv}` eingeleitet.

```
\newenvironment{name}[narg]{begdef}{enddef}
\renewenvironment{name}[narg]{begdef}{enddef}
```

Wie bei der `\newcommand`-Deklaration muss die Anzahl der Argumente im Bereich $0 \leq narg \leq 9$ liegen. Auch hier kann man die `[0]` weglassen, wenn kein Argument benötigt wird. Innerhalb der Befehlsdefinition, *begdef*, werden die Argumente als `#1` bis `#narg` adressiert. Soweit Argumente vorhanden sind, werden diese beim *Eintreten* in die Umgebung mittels des Befehls `\begin{myenv}` folgendermaßen angegeben:

```
\begin{myenv}{arg_1}...{arg_k}
```

Beim *Verlassen* der Umgebung über den Befehl `\end{myenv}` können keine Argumente angegeben werden. Im Definitionsteil *enddef*, in dem die Aktionen definiert werden, die beim Verlassen der Umgebung *myenv* eintreten sollen, sind außerdem die Argumente, die beim Eintritt in die Umgebung mit dem Befehl `\begin{myenv}` initialisiert wurden, nicht länger gültig. Hier fällt also dem Entwickler die Aufgabe zu, die Daten, die am Ende einer Umgebung benötigt werden, selbst zu speichern (vgl. die in Beispiel A-1-8 auf Seite 882 definierte Umgebung *Citation*).

Technisch gesehen definiert eine `\newenvironment`-Deklaration für die Umgebung *myenv* einen Befehl `\myenv`, der während der Verarbeitung von `\begin{myenv}` aufgerufen wird, und einen Befehl `\endmyenv`, der (neben anderen Vorgängen) durch `\end{myenv}` ausgeführt wird. Manchmal werden in Paketen und Klassen diese Befehle anstelle der Umgebungstags verwendet, um verwandte Umgebungen oder Befehle zu definieren. Ein Beispiel, bei dem dieses Verfahren vorteilhaft ist, befindet sich auf Seite 480. In den meisten

Keine Argumente
im Ende-Tag 

anderen Fällen ist diese Vorgehensweise jedoch nicht zu empfehlen, es sei denn, man versteht genau, wie der $\text{\LaTeX}$ -Kern implementiert ist.

Im nächsten Beispiel wird eine Umgebung des Typs „Abstract“ definiert, die häufig für eine kurze Zusammenfassung eines Artikels oder Buches verwendet wird. Sie beginnt mit einem fett und zentriert gesetzten Titel, dem der Text der Zusammenfassung innerhalb einer `quote`-Umgebung folgt. Der abschließende `\par`-Befehl sorgt dafür, dass nachfolgender Text immer einen neuen Absatz beginnt.

Zusammenfassung

Diese Zusammenfassung erklärt, wie man anstehende Aufgaben löst.

Dies ist nachfolgender Text. Der Zusammenfassung folgender Text. Weiterer Text.

Bsp.
A-1-7

```
\newenvironment{Abstract}
{\begin{center}\normalfont\bfseries
 Zusammenfassung\end{center}\begin{quote}}%
{\end{quote}\par}
\begin{Abstract}
Diese Zusammenfassung erklärt, wie man
anstehende Aufgaben löst.
\end{Abstract}
Dies ist nachfolgender Text. Der Zusammenfassung
folgender Text. Weiterer Text.
```

Das zweite Beispiel ist etwas komplexer. Es zeigt, wie man eine `Citation`-Umgebung für Zitate berühmter Personen definieren kann.

Der folgende $\text{\LaTeX}$ -Code definiert den Zähler `Citctr` zum Nummerieren der Zitate, sowie eine Box `\Citname` zum Speichern des Namens der zitierten Person. Mithilfe der Box lässt sich der Name am Ende des Zitates dort wiedergeben, wo sich der Befehl `\end{Citation}` befindet. (Zur Erinnerung: Der Wert, der im Argument des Befehls `\begin{Citation}` angegeben wurde, steht ja an dieser Stelle bereits nicht mehr zur Verfügung.) Beim Eintritt in die Umgebung wird der Wert des Argumentes, der kursiv gesetzt ist, in der Box `\Citname` gespeichert und der Zähler erhöht.

Danach beginnt eine `description`-Umgebung. Diese Umgebung umschließt einen einzigen `\item`-Befehl, welcher das Wort „Zitat“, gefolgt von dem Wert des Zählers, ausgibt. Beim Verlassen der `Citation`-Umgebung werden zwei horizontale dehnbare Leerräume eingefügt, zwischen denen ein Zeilenumbruch zwar erlaubt, aber erschwert ist. Dieser Leerraum muss unbedingt erhalten bleiben, auch wenn davor oder danach ein Zeilenumbruch erfolgt. Darum wird er mit dem Befehl `\hspace*` gebildet. Außerdem wird noch ein Leerraum von einem `\quad` (Geviert) eingefügt, der für einen passenden Abstand zwischen Zitat und Name sorgt, wenn beide in der gleichen Zeile erscheinen. Erfolgt zwischen ihnen ein Zeilenumbruch, so fällt der Abstand weg.

Schließlich wird dann der Inhalt der Box `\Citname` gesetzt, bevor die `description`-Umgebung wieder endet. Dadurch erscheinen der Name des Verfassers rechtsbündig und die letzte Zeile des Zitates linksbündig. Dies gilt auch, wie im nächsten Beispiel zu sehen, wenn sie in verschiedenen Zeilen stehen. Ohne diese Anpassungen würde der Text des Zitates immer im Blocksatz erscheinen, oftmals mit sehr viel Weißraum zwischen den einzelnen Wörtern.

Die in diesem Beispiel verwendeten Zähler- und Boxenbefehle werden in den Abschnitten A.1.4 und A.2 besprochen.

		<pre> \newcounter{Citctr} \newsavebox{\Citname} \newenvironment{Citation}[1] { \sbox\Citname{\emph{#1}}% \stepcounter{Citctr}\begin{description} \item[Zitat \arabic{Citctr}]} {\hspace*\fill}\nolinebreak[1]% \quad\hspace*\fill}% %% \finalhyphendemerits=0 %% siehe unten \usebox{\Citname}\end{description}} </pre>
Zitat 1	Der Mensch ist das Maß aller Dinge. <i>Protagoras</i>	
Dies ist gewöhnlicher Text zwischen zwei Citation-Umgebungen.		<pre> \begin{Citation}{Protagoras} Der Mensch ist das Maß aller Dinge. \end{Citation} Dies ist gewöhnlicher Text zwischen zwei Citation-Umgebungen. \begin{Citation}{Blaise Pascal} On mourra seul. \end{Citation} Weiterer sonstiger Text \ldots \begin{Citation}{William Pitt} Necessity is the plea for every infringement of human freedom. \end{Citation} </pre>
Zitat 2	On mourra seul. <i>Blaise Pascal</i>	
Weiterer sonstiger Text ...		
Zitat 3	Necessity is the plea for every infringement of human freedom. <i>William Pitt</i>	

Bsp.
A-1-8

Überraschenderweise wird der Name beim letzten Zitat in eine eigene Zeile gesetzt, obwohl auf jeden Fall noch genügend Raum neben dem Zitat vorhanden wäre. Das liegt daran, dass der T_EX-Absatzalgorithmus Lösungen bevorzugt, bei denen die vorletzte Zeile nicht auf einen Trennstrich endet, so dass er durch den Umbruch am `\nolinebreak`-Befehl einen dreizeiligen Absatz erzeugt.

Trennstriche in
der vorletzten Zeile
eines Absatzes

Es gibt zwei Möglichkeiten, dieses Verhalten zu korrigieren. Zum einen kann man den Umbruch an dieser Stelle durch das optionale Argument [3] statt des Wertes [1] erschweren; dieser Ansatz würde hier funktionieren, aber nicht unbedingt in anderen Fällen. Zum anderen kann man den T_EX-Absatzalgorithmus anweisen, den Trennstrich nicht zu beachten, indem man den ganzzahligen Low-Level T_EX-Parameter `\finalhyphendemerits` auf null setzt. Das erfordert eine etwas ungewöhnliche Syntax, wie im vorigen Beispiel zu sehen ist. (Dort ist die Korrektur jedoch auskommentiert, damit das Verhalten ohne ihren Einfluss deutlich wird.)

Wie schon bei `\newcommand`, so kann man auch das erste Argument einer Umgebung als optional festlegen:

```
\newenvironment{name}[narg][voreinst]{begdef}{enddef}
```

Der Vorgabewert `voreinst` für das optionale Argument steht im zweiten Paar eckiger Klammern. Innerhalb des `begdef`-Teils, der beim Eintritt in die Umgebung `name` ausgeführt wird, kann man mit #1 auf das optionale Argument zugreifen. Etwaig vorhandene obligatorische Argumente werden als #2 bis #narg adressiert. Wird die Umgebung `name` ohne optionalen Parameter verwendet, so enthält #1 den als Voreinstellung (`voreinst`) festgelegten Wert.

Als Beispiel wird hier die `altDescription`-Umgebung aus Beispiel 3-3-27 auf Seite 157 reimplementiert, dieses Mal jedoch mit einem optionalen anstelle des obligatorischen Argumentes, das die Breite des Einzugs angibt. Ein weiterer Unterschied gegenüber der ursprünglichen Definition besteht darin, dass die Listenlabel soweit möglich rechtsbündig gesetzt werden (indem der Befehl `\hfil` in dem Befehl `\makelabel` links positioniert wird). Wird die Umgebung ohne optionales Argument verwendet, so hat der Einzug eine Breite von `1em` (also einem `\quad` oder Geviert). Indem man den breitesten Eintrag als optionales Argument angibt, erreicht man, dass die Beschreibungsteile aller Einträge sauber untereinander gesetzt werden.

Im Beispiel wird zunächst das (voreingestellte) Verhalten der Umgebung `altDescription` gezeigt und danach, wie dieses mithilfe des optionalen Argumentes verändert werden kann.

		<pre>\usepackage{calc} \newenvironment{altDescription}[1][\quad]{ \begin{list}{}{ \renewcommand{\makelabel}[1]{\hfil\textsf{##1}}% \settowidth\labelwidth{\makelabel{#1}}% \setlength\leftmargin{\labelwidth+\labelsep}} \end{list}}</pre>
Erstens	Ein kurzer Begriff, dessen Text umbricht.	
Langer Begriff	Langer Begriff.	
Noch längerer Begriff	Sehr langer Begriff.	
Erstens	Ein kurzer Eintrag, dessen Text umbricht.	<pre>\begin{altDescription} \item[Erstens] Ein kurzer Begriff, dessen Text umbricht. \item[Langer Begriff] Langer Begriff. \item[Noch längerer Begriff] Sehr langer Begriff. \end{altDescription}</pre>
Langer Begriff	Langer Begriff.	<pre>\begin{altDescription}[Langer Begriff] \item[Erstens] Ein kurzer Eintrag, dessen Text umbricht. \item[Langer Begriff] Langer Begriff.</pre>
Noch längerer Begriff	Sehr langer Begriff.	<pre>\item[Noch längerer Begriff] Sehr langer Begriff. \end{altDescription}</pre>

Bsp.
A-1-9

A.1.4 Definieren und Ändern von Zählern

Jede Nummer, die von $\text{\LaTeX}$ intern generiert wird, ist mit einem *Zähler* (Register) verknüpft. Der Zähler heißt normalerweise genauso wie die Umgebung oder der Befehl, der die Zahl erzeugt, nur dass er nicht mit „\“ beginnt. Die folgende Liste enthält alle Zähler der Standarddokumentenklassen in $\text{\LaTeX}$:

<code>part</code>	<code>paragraph</code>	<code>figure</code>	<code>enumi</code>
<code>chapter</code>	<code>subparagraph</code>	<code>table</code>	<code>enumii</code>
<code>section</code>	<code>page</code>	<code>footnote</code>	<code>enumiii</code>
<code>subsection</code>	<code>equation</code>	<code>mpfootnote</code>	<code>enumiv</code>
<code>subsubsection</code>			

Eine Umgebung, die mit dem Befehl `\newtheorem` deklariert wird, kann ebenfalls einen Zähler mit demselben Namen besitzen, es sei denn, das optionale Argument schreibt eine Nummerierung zusammen mit einer anderen Umgebung vor.

Der Wert des Zählers ist eine einzelne ganze Zahl. Zähler können miteinander kombiniert werden, wie es z.B. bei der Nummerierung von Abschnittsüberschriften üblich ist. So gibt etwa die Zahlenfolge 7.4.5 in den Dokumentenklassen `book` oder `report` an, dass es sich um den fünften Unterabschnitt des vierten Abschnitts im siebten Kapitel handelt.

Im Folgenden werden alle $\LaTeX$ -Basisbefehle zum Definieren, Verändern und Darstellen von Zählern beschrieben. Diese Befehle sind noch viel leistungsfähiger, wenn man sie gemeinsam mit dem Paket `calc` benutzt, das in Abschnitt A.3.1 besprochen wird.

```
\newcounter{zähler-neu}[zähler-alt]
```

Dieser Befehl definiert global einen neuen Zähler `zähler-neu` und initialisiert ihn mit null. Existiert bereits ein Zähler mit diesem Namen, so wird eine Fehlermeldung ausgegeben. Gibt man im optionalen Argument `zähler-alt` den Namen eines anderen Zählers an, dann wird der neu definierte Zähler `zähler-neu` automatisch zurückgesetzt, sobald der Zähler `zähler-alt` mit einem der Befehle `\stepcounter` oder `\refstepcounter` hochgezählt wird. Außerdem wird der Befehl `\thezähler-neu` so definiert, dass er `\arabic{zähler-neu}` ausgibt.

```
\addtoreset{zähler-rücksetzen}{zähler}
\removefromreset{zähler-rücksetzen}{zähler}
```

Der Vorgang, der festlegt, dass ein Zähler zurückgesetzt wird, sobald ein anderer erhöht wird, ist auch als Kernelbefehl `\addtoreset` verfügbar¹. Leider steht die gegenteilige Deklaration nicht als Kernelbefehl zur Verfügung, sondern nur durch Laden des Paketes `remreset`. Mithilfe dieses kleinen Paketes lassen sich Zähler gegebenenfalls auch wieder voneinander lösen. So legt beispielsweise die Klasse `report` fest, dass der Fußnotenzähler `footnote` zurückgesetzt wird, sobald ein neues Kapitel beginnt. Sollen die Fußnoten jedoch im ganzen Dokument fortlaufend nummeriert werden, dann hilft der folgende Code in der Präambel:

```
\usepackage{remreset}
\makeatletter \@removefromreset{footnote}{chapter} \makeatother
```

Eine entsprechende Programmierung² in einem Paket oder einer Klasse funktioniert ebenso gut.

```
\setcounter{zähler}{wert} \addtocounter{zähler}{wert}
```

Mithilfe des Befehls `\setcounter` wird der Zähler `zähler` global auf den Wert `wert` gesetzt. Durch `\addtocounter` wird er global um den Betrag `wert` erhöht.

¹Vgl. hierzu die `\numberwithin`-Deklaration des `amsmath`-Paketes. Sie wird in Abschnitt 8.2.14 auf Seite 497 besprochen.

²Innerhalb eines Paketes benötigt man `\RequirePackage` und *muss* `\makeatletter` und `\makeatother` weglassen!

Achtung: 
`\@removefromreset`
 benötigt ein Paket!

<code>\stepcounter{zähler}</code>	<code>\refstepcounter{zähler}</code>

Beide Befehle erhöhen den Zähler *zähler* global und setzen alle untergeordneten Zähler zurück, also all jene, die als optionales Argument *zähler-alt* des Befehls `\newcounter` oder als erstes Argument von `\@addtoreset` deklariert wurden. Der Befehl `\refstepcounter` legt darüber hinaus fest, dass sich der aktuelle `\ref`-Wert aus dem Text ergibt, den der Befehl `\thezähler` erzeugt. Dabei ist zu beachten, dass das Hochsetzen eines Zählers zwar ein globaler Vorgang ist, der aktuelle `\ref`-Wert jedoch nur lokal gesetzt wird, also nur in der aktuellen Gruppe gültig ist.

Entsprechend zeigt das nächste Beispiel nicht das gewünschte Ergebnis, sondern übernimmt stattdessen die Abschnitsnummer. Die richtige Lösung wäre, `\refstepcounter` vor dem Befehl `\textbf` zu platzieren.

5 Ein Misserfolg

Übung 5.a: Ein Test.

Übung 5.b: Weiterer Test.

Verweis auf die Übungen: 5 und 5.

```
\newcounter{ex} \renewcommand\theex{\thesection.\alph{ex}}
\newenvironment{EX}{\begin{flushleft}%
  \textbf{\refstepcounter{ex}Übung~\theex:}}
  {\end{flushleft}}
\setcounter{section}{4} % zum Testen
\section{Ein Misserfolg}
\begin{EX} \label{A} Ein Test. \end{EX}
\begin{EX} \label{B} Weiterer Test. \end{EX}
Verweis auf die Übungen: \ref{A} und \ref{B}.
```

Bsp.
A-1-10

<code>\value{zähler}</code>	<code>\arabic{zähler}</code>	<code>\roman{zähler}</code>	<code>\Roman{zähler}</code>
<code>\alph{zähler}</code>	<code>\Alph{zähler}</code>	<code>\fnsymbol{zähler}</code>	

Der Befehl `\value` übergibt den aktuellen Wert eines Zählers in all den Fällen, in welchen $\text{\LaTeX}$ eine Zahl erwartet, wie etwa im *wert*-Argument der Befehle `\setcounter` oder `\addtocounter`, oder für einen Zahlenvergleich durch den `\ifthenelse`-Befehl des *ifthen*-Paketes. Er ist jedoch *nicht* geeignet, den Wert eines Zählers darzustellen! Zu diesem Zweck gibt es eine Reihe von Darstellungsbefehlen, die alle den Namen des Zählers als Argument erwarten.

Der Befehl `\arabic` stellt den Zählerwert als arabische Ziffer dar. Durch die Befehle `\roman` und `\Roman` werden römische Ziffern in Klein- bzw. Großbuchstaben erzeugt.

Die verbleibenden Befehle eignen sich nur für Zähler, deren Wert in einem bestimmten Bereich liegt. Der Befehl `\alph` stellt den Wert als Kleinbuchstaben dar: a, b, c, ... z. Entsprechend sollte er im Bereich von 1 bis 26 liegen, da ansonsten eine Fehlermeldung erfolgt. Der Befehl `\Alph` erzeugt die entsprechenden Großbuchstaben. Der Befehl `\fnsymbol` zeigt den Zählerwert als traditionelles Fußnotensymbol (z.B. *, †). Hier darf der Wert nicht größer als 9 sein, es sei denn, man verwendet ein Erweiterungspaket wie *footmisc*. Das nächste Beispiel zeigt die praktische Anwendung all dieser Befehle.

 **Zählerdarstellungen**
mit begrenztem
Wertebereich

8, viii, VIII, h, H, ††
Anno Domini MCMXCIV

```
\newcounter{exa}\setcounter{exa}{8}
\arabic{exa}, \roman{exa}, \Roman{exa}, \alph{exa},
\Alph{exa}, \fnsymbol{exa} \par
\setcounter{exa}{1994} Anno Domini \scshape{\roman{exa}}
```

Bsp.
A-1-11

`\the⟨zähler⟩`

Der Befehl `\the⟨zähler⟩` steht jeweils als Kurzform zur Verfügung, um die voreingestellte Darstellung eines *zählers* zu erzeugen (so z.B. `\thesection` für den Abschnittszähler *section*). Wie bereits zuvor erwähnt, wird dieser Befehl durch die `\newcounter`-Deklaration so initialisiert, dass er die Darstellung `\arabic{zähler}` erzeugt. In $\text{\LaTeX}$ umfasst eine solche visuelle Darstellung jedoch oftmals mehr als nur eine einzelne Zahl. So gibt man beispielsweise im Falle des Abschnittszählers den Wert des aktuellen Abschnitts sowie den des Unterabschnitts an, usw. Daher wird der Befehl `\the⟨zähler⟩` üblicherweise so (um)definiert, dass er eine komplexere Darstellung erzeugt. Dieses Verfahren gewinnt noch mehr an Bedeutung, wenn man bedenkt, dass der Befehl `\refstepcounter` nicht nur einen bestimmten Zähler erhöht und dessen untergeordnete Zähler zurücksetzt. Er legt zudem fest, dass das „aktuelle“ Label (das von dem Befehl `\label` übernommen wird) dem Ergebnis des `\the⟨zähler⟩`-Befehls für den soeben erhöhten Zähler entspricht.

Beispielsweise findet man in der Standardklasse *article* die folgenden Definitionen für Abschnittszähler:

```

\newcounter{part}
\newcounter{section}
\newcounter{subsection}[section]
\newcounter{subsubsection}[subsection]
\renewcommand\thepart{\Roman{part}}
\renewcommand\thesection{\arabic{section}}
\renewcommand\thesubsection{\thesection.\arabic{subsection}}
\renewcommand\thesubsubsection
    {\thesubsection.\arabic{subsubsection}}

```

Hier ist zu sehen, dass die untergeordneten Zähler zurückgesetzt werden, sobald die ihnen übergeordneten erhöht werden. Ebenso wird deutlich, dass die Darstellung der jeweiligen Zähler (der `\the . . .`-Befehl) aus dem aktuellen Zähler und den ihm übergeordneten gebildet wird. Man beachte, dass sich der *part*-Zähler auf keine der untergeordneten Ebenen auswirkt.

Als weiteres Beispiel dient ein Blick auf Tabelle 3.6 auf Seite 137, welche den Aufbau der Zähler für nummerierte Listen zeigt. Diese Zähler werden eigentlich in der Datei `latex.ltx` definiert, die den $\text{\LaTeX}$ -Kernel enthält. Nur die Befehle für die Darstellung, das Präfix und das Labelfeld werden in den Standardklassendateien definiert, und zwar folgendermaßen:

```

\renewcommand\thenumi {\arabic{enumi}}      \renewcommand\thenumii{\alph{enumii}}
\renewcommand\thenumiii{\roman{enumiii}}    \renewcommand\thenumiv{\Alph{enumiv}}

\renewcommand\p@enumii{\thenumi}            \renewcommand\p@enumiii{\thenumi(\thenumii)}
\renewcommand\p@enumiv{\p@enumiii\thenumiii}

\newcommand\labelenumi {\thenumi.}          \newcommand\labelenumii{(\thenumii)}
\newcommand\labelenumiii{\thenumiii.}      \newcommand\labelenumiv{\thenumiv.}

```

Zum Abschluss erfolgt noch ein Blick darauf, wie die Standardklassen mit dem *equation*-Zähler umgehen. Genau wie die Listenzähler, wird auch dieser

Zähler in der Datei `latex.ltx` deklariert. In der Klasse `article` werden alle Formeln in einer Sequenz durchnummeriert:

```
\renewcommand\theequation{\arabic{equation}}
```

In den Klassen `report` und `book` dagegen wird der Formelzähler durch den Befehl `\@addtoreset` mit jedem neuen Kapitel zurückgesetzt:

```
\@addtoreset{equation}{chapter}
\renewcommand\theequation{\thechapter.\arabic{equation}}
```

Auch die Darstellung ist in beiden Fällen unterschiedlich.¹

A.1.5 Definieren und Ändern von Längenparametern

In (L^A)T_EX gibt es zwei Arten von Längenparametern: „feste“ Längen (in *The T_EXbook* [87] unter dem Namen `<dimen>` geführt), die unveränderlich sind, und „elastische“ Längen (in *The T_EXbook* `<skip>` genannt), die über eine natürliche Länge verfügen und sich bis zu einem gewissen Grad dehnen oder stauchen lassen. Neue Längen werden in L^AT_EX immer als `<skip>` angelegt, so dass man wählen kann, ob man sie mit festen oder elastischen Längen initialisiert, letzteres durch Angabe von `plus` und `minus`-Beträgen. Andererseits sind alle Standardlängen in L^AT_EX feste Längen, es sei denn, sie sind ausdrücklich in Anhang C des L^AT_EX-Manuals als elastische Längen aufgeführt. Im Folgenden werden nun die L^AT_EX-Befehle zur Handhabung solcher Längenparameter besprochen.

```
\newlength{befehl}
```

Die Deklaration `\newlength` legt ein neues (elastisches) Längenregister an und weist ihm den Befehlsnamen *befehl* zu. Existiert bereits ein Befehl dieses Namens, so erhält man eine Fehlermeldung. Die neue Länge wird zunächst auf null gesetzt. Wie bei dem Befehl `\newcommand` werden auch hier die Klammern um *befehl* häufig weggelassen, da das Argument aus einem einzigen Befehlsnamen bestehen muss.

```
\setlength{befehl}{länge}    \addtolength{befehl}{länge}
```

Durch diesen Befehl wird der Längenparameter *befehl* auf den Wert *länge* eingestellt, bzw. im Falle des Befehls `\addtolength` wird der angegebene Betrag dem aktuellen Wert hinzugefügt. In den folgenden Beispielen werden die Inhalte der Längenvariablen mithilfe des T_EX-Befehls `\the` dargestellt. Dieser benötigt den Namen des Registerbefehls *ohne* Klammern!

Mylen = 28.45274pt

Mylen = 28.45274pt plus 4.0pt minus
2.0pt

```
\newlength\Mylen
```

```
\setlength \Mylen{10mm}    Mylen = \the\Mylen
\addtolength\Mylen{0pt plus 4pt minus 2pt}
\par Mylen = \the\Mylen
```

Bsp.
A-1-12

¹Die tatsächliche Definition ist noch etwas komplexer, da die Kapitelnummer mithilfe von Low-Level-Befehlen unterdrückt wird, wenn sie null beträgt.

sp	Skalierter Punkt ($65536\text{sp} = 1\text{pt}$), die kleinste T _E X-Einheit	
pt	Punkt = $\frac{1}{72.27}\text{in} = 0.351\text{mm}$	
bp	Big point = $\frac{1}{72}\text{in} = 0.353\text{mm}$, auch großer oder PostScript-Punkt	
dd	Didot-Punkt = $\frac{1}{72}$ eines französischen Zolls = 0.376mm	
mm	Millimeter = 2.845pt	
pc	Pica = $12\text{pt} = 4.218\text{mm}$	
cc	Cicero = $12\text{dd} = 4.531\text{mm}$	
cm	Zentimeter = $10\text{mm} = 2.371\text{pc}$	
in	Zoll = $25.4\text{mm} = 72.27\text{pt} = 6.022\text{pc}$	
ex	Höhe eines kleinen „x“ im aktuellen Font (ungefähr)	
em	Breite des großen „M“ im aktuellen Font (ungefähr)	
mu	Mathematische Einheit ($18\text{mu} = 1\text{em}$) zum Positionieren im Formelmodus	

Tabelle A.1: L^AT_EX-Längenmaße

Wie in Tabelle A.1 gezeigt, lassen sich Längen in verschiedenen Einheiten angeben. Man beachte den Unterschied zwischen dem typographischen Punkt (pt), der normalerweise in T_EX verwendet wird, und dem (großen) Punkt, der z.B. in PostScript zum Einsatz kommt. Um Platz für eine EPS-Graphik zu reservieren, muss man die Maße der Bounding Box entsprechend in bp angeben, um den passenden Leerraum zu erhalten.

```
\settoheight{befehl}{text}
\settodepth{befehl}{text}
```

Anstatt explizit einen bestimmten Wert für die Länge anzugeben, kann man mithilfe eines dieser drei Befehle einen gegebenen Text vermessen und das Ergebnis als Wert zuweisen. Durch `\settoheight` wird der Wert des Längenparameters *befehl* auf die natürliche Länge der gesetzten Version von *text* eingestellt. Dadurch eignet sich dieser Befehl insbesondere zum Definieren von Längen, die sich an Textinhalten oder Schriftgrößen orientieren. Die beiden anderen Befehle funktionieren ähnlich, nur messen sie anstatt der Breite die Höhe bzw. Tiefe des gesetzten Textes.

Breite = 50.29977pt	<code>\newlength\Mylen</code>	<code>\raggedright%</code> um das Beispiel zu verschönern
Höhe = 6.7799pt	<code>\settoheight\Mylen{Typographie}</code>	Breite = <code>\the\Mylen</code> \\
Tiefe = 2.16492pt	<code>\settodepth\Mylen{Typographie}</code>	Höhe = <code>\the\Mylen</code> \\
Neuberechnung mit größerer Schrift:	<code>\settoheight\Mylen{Typographie}</code>	Tiefe = <code>\the\Mylen</code> \par
Breite = 60.35974pt	<code>\settoheight\Mylen{\large Typographie}</code>	Neuberechnung mit größerer Schrift: \\
	<code>\settoheight\Mylen{\large Typographie}</code>	Breite = <code>\the\Mylen</code>

Bsp.
A-1-13

```
\fill \stretch{dez-zahl}
```

Diese beiden elastischen Längen sind für das Argument von `\vspace` und ähnlichen Befehlen gedacht. Die elastische Länge `\fill` ist auf die natürliche Länge null voreingestellt, kann sich jedoch auf jeden beliebigen positiven Wert dehnen. Dieser Wert darf nicht geändert werden! Er wird an zahlreichen Stellen im Kernel verwendet und eine Änderung würde seltsame Resultate mit sich bringen.

<code>\hspace{<i>län</i>}</code>	Horizontaler Abstand der Breite <i>län</i> , der eine feste oder elastische Länge sein kann
<code>\enspace</code>	Horizontaler Abstand von einem Halbgeviert
<code>\quad</code>	Horizontaler Abstand von einem Geviert (1em im aktuellen Font)
<code>\qquad</code>	Doppeltes <code>\quad</code>
<code>\hfill</code>	Horizontaler elastischer Abstand, der sich von 0 bis ∞ dehnen lässt
<code>\hrulefill</code>	Wie <code>\hfill</code> , erzeugt jedoch eine durchgezogene horizontale Linie
<code>\dotfill</code>	Wie <code>\hfill</code> , erzeugt jedoch eine gepunktete Linie

Tabelle A.2: Vordefinierte horizontale Abstände

Eine oftmals nützlichere elastische Länge steht mit dem Befehl `\stretch` zur Verfügung, wobei `\fill` dem Befehl `\stretch{1}` entspricht. Allgemeiner ausgedrückt lässt sich `\stretch{dez-zahl}` auf *dez-zahl* mal `\fill` dehnen. Mit seiner Hilfe kann man die Position von Texten horizontal oder vertikal justieren, z.B. durch Abstände, die in einem bestimmten Verhältnis zueinander stehen. In Beispiel A-1-15 wird die praktische Anwendung gezeigt.

Horizontale Abstände

Tabelle A.2 zeigt die in $\text{\LaTeX}$ bekannten Befehle für horizontale Abstände. Mit dem Befehl `\hspace` lässt sich ein flexibler horizontaler Abstand beliebiger Breite erzeugen. Der Befehl `\hspace*` ist mit ihm identisch, nur dass sein Abstand niemals entfernt wird, auch nicht an einem Zeilenende.

Wie das folgende Beispiel zeigt, ist ein Leerzeichen vor oder nach einem `\hspace`- oder `\hspace*`-Befehl durchaus von Bedeutung:

Bsp. A-1-14	Ein	0,5 Zoll breiter Abstand.	<code>\par Ein\hspace{0.5in}0,5-Zoll breiter Abstand.</code>
	Ein	0,5 Zoll breiter Abstand.	<code>\par Ein \hspace{0.5in}0,5-Zoll breiter Abstand.</code>
	Ein	0,5 Zoll breiter Abstand.	<code>\par Ein \hspace{0.5in} 0,5-Zoll breiter Abstand.</code>

Das nächste Beispiel zeigt, wie sich mithilfe von elastischen Längen die Textposition in einer Zeile nachjustieren lässt. Man beachte, dass der Befehl `\hfill` eigentlich eine Abkürzung für `\hspace{\fill}` darstellt. Um die Eingabe zu erleichtern, wurde zudem ein Befehl `\HS` mit einem optionalen Argument definiert, der sich ohne dieses Argument wie `\hfill` verhält. Indem man einen Wert für seine Dehnbarkeit festlegt, kann man ihn jedoch mehr oder weniger elastisch als `\hfill` gestalten (wobei er mit dem Wert 1 die gleiche Wirkung hat wie `\hfill`).

				<pre> \newcommand{\HS}[1][1.]{\hspace{\stretch{#1}}} \begin{center} links \hfill rechts links \HS[2]\fbox{\$\frac{2}{5}\$}\HS[5] rechts links \HS mittig \hfill rechts links \hrulefill\ mittig \hrulefill\ rechts links \dotfill\ rechts links \dotfill\ \HS[.5] \dotfill\ rechts links \dotfill\ \HS \dotfill\ rechts links \dotfill\ \HS[2.] \dotfill\ rechts \end{center} </pre>			
Bsp. A-1-15	links		rechts	links		rechts	
	links	$\frac{2}{5}$	rechts	links	<code>\HS[2]\fbox{\$\frac{2}{5}\$}\HS[5]</code>	rechts	
	links	mittig	rechts	links	<code>\HS mittig \hfill</code>	rechts	
	links	_____ mittig _____	rechts	links	<code>\hrulefill\ mittig \hrulefill\</code>	rechts	
	links	 rechts		links	<code>\dotfill\</code>	rechts	
	links	 rechts		links	<code>\dotfill\ \HS[.5] \dotfill\</code>	rechts	
	links	 rechts		links	<code>\dotfill\ \HS \dotfill\</code>	rechts	
	links	 rechts		links	<code>\dotfill\ \HS[2.] \dotfill\</code>	rechts	

Vertikale Abstände

Vertikale Abstände werden mit dem Befehl `\vspace` erzeugt, der ähnlich funktioniert wie `\hspace`. Der Befehl `\vspace*` generiert einen vertikalen Abstand, der immer erhalten bleibt, selbst wenn er sich an einem Seitenumbruch befindet. Der `\vspace`-Befehl wird an einer solchen Stelle ignoriert. Tabelle A.3 auf der gegenüberliegenden Seite zeigt in $\text{\LaTeX}$ bekannte Befehle für vertikale Abstände, die in allen Standardklassen üblich sind.

Das Verhalten des `\vspace`-Befehls ist für $\text{\LaTeX}$ -Benutzer oftmals verwirrend. Innerhalb eines Absatzes wird der vertikale Abstand nach der Zeile eingefügt, die den Befehl enthält. Zwischen zwei Absätzen verhält sich `\vspace` jedoch wie erwartet.

Die Auswirkung eines `\vspace`-

Befehls innerhalb eines Absatzes ist etwas gewöhnungsbedürftig. Mit einem negativen Wert kann er vielleicht dazu dienen, den Durchschuss zu verkleinern.

Die `\vspace{3mm}`-Auswirkung eines `\verb!\vspace!`-Befehls innerhalb eines Absatzes ist etwas gewöhnungsbedürftig. Mit einem negativen Wert kann er vielleicht dazu dienen, den Durchschuss zu verkleinern.

`\vspace{\baselineskip}`

Das Einstellen der Abstände zwischen Absätzen ist da schon nützlicher und erlaubt, den Weißraum vor und nach abgesetztem Material zu steuern.

Das Einstellen der Abstände zwischen Absätzen ist da schon nützlicher und erlaubt, den Weißraum vor und nach abgesetztem Material zu steuern.

Bsp.
A-1-16

Elastische Längen, wie sie auf Seite 888 vorgestellt wurden, lassen sich auch für vertikale Elemente verwenden. Der Befehl `\vfill` ist eigentlich eine Kurzform für eine leere Zeile, auf die ein `\vspace{\fill}`-Befehl folgt. Generell kann man mithilfe des `\stretch`-Befehls in Verbindung mit `\vspace` das Layout einer ganzen Seite steuern. Das kann beim Gestalten einer Titelseite hilfreich sein: Soll der Titel um ein Drittel der Seite nach unten verschoben erscheinen, so setzt man einfach den Befehl `\vspace*{\stretch{1}}` davor und `\vspace*{\stretch{2}}` dahinter.

Geoffrey Chaucer
The Canterbury Tales

LONDON 1400

```
\newcommand\HRule{\noindent\rule{\linewidth}{1.5pt}}
\begin{titlepage}
  \vspace*{\stretch{1}}
  \HRule
  \begin{flushright}
    \LARGE Geoffrey Chaucer \\\
    The Canterbury Tales
  \end{flushright}
  \HRule
  \vspace*{\stretch{2}}
  \begin{center}
    \textsc{London 1400}
  \end{center}
\end{titlepage}
```

Bsp.
A-1-17

<code>\vspace{<i>län</i>}</code>	Vertikaler Abstand der Höhe <i>län</i> , der eine feste oder elastische Länge sein kann
<code>\smallskip</code>	Vertikaler Abstand der Größe <code>\smallskipamount</code> (voreingestellt auf ungefähr ein Viertel <code>\baselineskip</code>)
<code>\medskip</code>	Vertikaler Abstand der Größe <code>\medskipamount</code> (voreingestellt auf ungefähr die Hälfte von <code>\baselineskip</code>)
<code>\bigskip</code>	Vertikaler Abstand der Größe <code>\bigskipamount</code> (voreingestellt auf ungefähr ein <code>\baselineskip</code>)
<code>\vfill</code>	Vertikale elastische Länge, die sich von 0 bis ∞ dehnen lässt

Tabelle A.3: Vordefinierte vertikale Abstände

`\addvspace{abstand}`

Während der $\text{\LaTeX}$ -Benutzerbefehl `\vspace` kategorisch einen vertikalen Abstand einfügt (der nur an Seitenumbrüchen unterdrückt wird, und im Falle der Sternform nicht einmal dort), gibt es einen weiteren Befehl für vertikale Abstände, der oftmals im Kernel und in Paketdateien verwendet wird. Der Befehl `\addvspace` verfügt über eine etwas andere Semantik und ist, obwohl sein Name es vermuten lässt, kein Benutzerbefehl.

 *Mit Vorsicht
zu genießen – wenn
überhaupt*

Im Gegensatz zu `\vspace` ist der Befehl `\addvspace` nur im vertikalen Modus zulässig, also zwischen Absätzen. Im horizontalen Modus gibt er die berühmte Fehlermeldung „Something’s wrong—perhaps a missing `\item`“ („Etwas läuft hier schief, vielleicht fehlt ein `\item`“) aus, den die meisten $\text{\LaTeX}$ -Anwender kennen und lieben. Meistens hat diese Fehlermeldung gar nichts mit einem fehlenden oder falsch platzierten `\item`-Befehl zu tun, sondern weist nur auf einen falsch platzierten `\addvspace`-Befehl hin. Hier wird jedoch ein Teil der Entwicklungsgeschichte dieses Befehls deutlich: Er wurde ursprünglich nur dazu entwickelt und eingesetzt, um Leerraum zwischen den Elementen einer `list`-Umgebung einzufügen.

Der andere wichtige Unterschied zu `\vspace` besteht darin, dass `\addvspace` einen Abstand einfügt, dessen Größe von einem unmittelbar vorangehenden Leerraum abhängig ist. Die genauen Regeln stammen noch aus $\text{\LaTeX}$ 2.09 und weisen einige seltsame Unregelmäßigkeiten auf, die sich heute niemand mehr wirklich erklären kann. Aus Gründen der Rückwärtskompatibilität wird der Befehl jedoch in seiner bisherigen Form beibehalten. Ist s der durch `\addvspace` einzufügende Abstand und ℓ die Größe des (gegebenenfalls) vorhergehenden vertikalen Leerraums, dann gilt:

Wenn	$s < \text{Opt} < \ell$	dann	gehe um s zurück
andernfalls wenn	$\ell = \text{Opt}$	dann	füge einen zusätzlichen Leerraum der Höhe s ein
		sonst	erzeuge einen Abstand der Höhe $\max(\ell, s)$ aus beiden

Ignoriert man einmal die Sonderfälle aus den ersten beiden Zeilen der Regeln, dann lässt sich das Konzept von `\addvspace` folgendermaßen beschreiben: Hat man zwei vertikal orientierte Gebilde, wie etwa eine Liste und eine Überschrift, und beide sollen darüber und darunter einen gewissen Freiraum erhalten, dann ist es vielleicht keine so gute Idee, beide Abstände vollständig einzufügen, wenn

die Objekte direkt aufeinander folgen. In einem solchen Fall ist es gemeinhin besser, den größeren der beiden Werte zu verwenden. Darum findet man bei Listen, Überschriften und anderen gesetzten Elementen eher `\addvspace` als `\vspace`.

Das hat einige recht überraschende Auswirkungen: Folgen zwei solche abgesetzten Elemente aufeinander, dann wird nur der größere der sie umgebenden Abstände verwendet. Versucht man jedoch den vorhandenen Leerraum ein wenig zu vergrößern, etwa durch Einfügen des Befehls `\vspace{4pt}` zwischen den beiden Objekten, dann erhält man plötzlich einen wesentlich größeren Leerraum. In einer Befehlsfolge wie

```
\addvspace{10pt} \vspace{4pt} \addvspace{8pt}
```

kann der zweite `\addvspace`-Befehl den ersten nicht mehr wahrnehmen und fügt daher seinen ganzen Abstand ein, so dass man einen Leerraum von insgesamt 22pt erhält. Ohne den `\vspace`-Befehl in der Mitte ergibt sich nur ein Abstand von 10pt Höhe. Der `\vspace`-Befehl wird nicht mit dem nachstehenden `\addvspace` zusammengefasst, da er in Wirklichkeit einen Abstand von 4pt erzeugt, auf den ein Abstand von 0pt Höhe folgt, so dass die zweite Regel greift.

Merkt man, dass der Abstand nun zu groß ist und reduziert seine Korrektur auf z.B. `\vspace{2pt}`, so ändert sich kaum etwas, da man immer noch einen Leerraum von 20pt Höhe erhält. Was noch überraschender ist: Versucht man den ursprünglichen Abstand durch Einfügen eines Befehls wie `\vspace{-3pt}` zu verkleinern, dann erhält man einen Leerraum von 15pt Höhe – das ist immer noch mehr als zuvor.

Um wirklich einen Leerraum von 7pt zu erhalten, müsste man den Abstand um 11pt reduzieren. Leider gibt es keinen anderen Weg, die Größe des erforderlichen Abstandes zu bestimmen, als damit zu experimentieren oder in den Definitionen des oberen und unteren Objektes nachzusehen, welche Werte für `\addvspace` an der jeweiligen Stelle verwendet werden.

Die gleichen Schwierigkeiten treten auch auf, wenn ein anderes unsichtbares Objekt zwischen zwei aufeinander folgenden `\addvspace`-Befehlen steht. So verbirgt z.B. auch ein Farbwechsel-, `\index`- oder ein `\label`-Befehl einen vorangehenden `\addvspace`-Befehl, so dass plötzlich nicht mehr der größere, sondern die Summe der beiden Abstände erscheint.

`\addpenalty{penalty}`

Obwohl es sich bei `\addpenalty` eigentlich nicht um einen Längenparameter handelt, wird der Befehl trotzdem hier beschrieben, da er zur Kombination mit `\addvspace` gedacht ist. Mithilfe eines Penalty-Wertes weist $\TeX$ Positionen, an denen ein Umbruch erfolgen darf, einen bestimmten Grad an „Badness“ zu. Diese Badness signalisiert, wie günstig oder ungünstig eine bestimmte Stelle für einen Umbruch ist, also wie schwer oder leicht ein Umbruch an dieser Position erfolgen kann. Ein hoher Penalty-Wert bedeutet, dass die jeweilige Stelle für einen Umbruch eher ungeeignet ist, während ein negativer Penalty-Wert $\TeX$ signalisiert, dass die jeweilige Stelle für einen Zeilen- oder Seitenumbruch relativ günstig ist. In Anhang B.3.2 auf Seite 982 wird genauer gezeigt, wie der Absatzalgorithmus von $\TeX$ diese Werte handhabt. Die Einzelheiten dieses Verfahrens sind in den Kapiteln 14 und 15 von [87] beschrieben.

Der Befehl `\addpenalty` benötigt als Argument einen für $\TeX$ verständlichen Penalty-Wert (sinnvolle Werte liegen zwischen -10000 und 10000). Der in Kapitel 2 besprochene Befehl `\@startsection` beispielsweise verwendet `\addpenalty` mit einem voreingestellten Wert von -300 um anzuzeigen, dass sich der Raum vor

seiner Überschrift besonders gut für einen Umbruch eignet. Kombiniert man nun `\addpenalty` und `\addvspace` miteinander, so hat das zwei Auswirkungen:

- $\text{\LaTeX}$ verwendet immer noch den größeren von zwei Abständen, auch wenn `\addpenalty` zwischen den entsprechenden `\addvspace`-Befehlen erscheint.
- $\text{\LaTeX}$ bewegt den möglichen Umbruch „visuell“ an den Anfang des Leerraums, auch wenn ein `\addvspace`-Befehl vor `\addpenalty` steht.

Die zweite Eigenschaft ist wichtig um überflüssigen Weißraum am Fuß einer Seite zu vermeiden. Auf Seite 982 ist genauer erklärt, wie dieser Effekt erzielt wird.

A.2 Seiten-Markup – Boxen und Linien

Das Konzept, eine Seite aus Boxen aufzubauen, stellt eines der Grundprinzipien von $\text{\TeX}$ dar, und $\text{\LaTeX}$ nutzt dieses Verfahren durch verschiedene geeignete Befehle. Eine *Box* ist ein rechteckiges Objekt mit einer Höhe, Tiefe und Breite. Ihr Inhalt kann beliebig komplex aufgebaut sein, z.B. aus weiteren Boxen, Zeichen, Leerräumen und so weiter. Einmal zusammengestellt betrachtet $\text{\LaTeX}$ sie als ein einzelnes, festes Objekt, das sich in etwa so verhält wie ein (potentiell überdimensionales) Zeichen. Eine Box lässt sich weder teilen, noch am Zeilen- oder Seitenende umbrechen. Boxen können jedoch nach oben, unten, links und rechts bewegt werden. $\text{\LaTeX}$ kennt drei Arten von Boxen:

LR (links–rechts) Die Inhalte einer solchen Box werden von links nach rechts gesetzt. Es sind keine Zeilenumbrüche möglich und Befehle wie `\` und `\newline` werden ignoriert oder erzeugen Fehlermeldungen.

Par (Absätze) Diese Art von Box kann mehrere Zeilen enthalten, die genau wie normaler Text im Absatzmodus gesetzt werden. Die Absätze werden untereinander angeordnet. Ihre Breite wird durch einen vom Benutzer angegebenen Wert festgelegt.

Rule Diese (dünne oder dicke) Linie dient oftmals dazu, verschiedene logische Elemente auf der gedruckten Seite gegeneinander abzugrenzen, wie etwa die einzelnen Zeilen und Spalten einer Tabelle oder Kolumnentitel vom Haupttext.

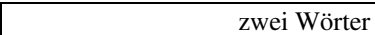
Alle $\text{\LaTeX}$ -Boxen beginnen im vertikalen Modus einen neuen Absatz, genau wie ein Buchstabe. $\text{\TeX}$ -Basisbefehle für Boxen (wie etwa `\hbox`) verhalten sich dagegen unterschiedlich, je nachdem, wo man sie verwendet. Es spricht einiges dagegen, die $\text{\TeX}$ -Basisbefehle direkt zu benutzen. Die Gründe werden in Abschnitt A.2.5 näher dargelegt. Bei Linien verhält es sich etwas anders, so dass die entsprechenden $\text{\TeX}$ -Basisbefehle für Linien hier besprochen werden.

A.2.1 LR-Boxen

<code>\mbox{<i>text</i>}</code>	<code>\fbox{<i>text</i>}</code>
<code>\makebox[<i>breite</i>][<i>pos</i>]{<i>text</i>}</code>	<code>\framebox[<i>breite</i>][<i>pos</i>]{<i>text</i>}</code>

Bei den Befehlen in der ersten Zeile wird der *text* in den geschweiften Klammern als Box ohne bzw. mit Rahmenlinie betrachtet. So ergibt z.B. `\fbox{zwei Wörter}` die Box zwei Wörter. Die beiden Befehle in der zweiten Zeile stellen eine verallgemeinerte Form der beiden aus der ersten Zeile

dar. Mit ihrer Hilfe kann der Anwender festlegen, wie breit die Box sein soll, und wo der Text darin positioniert wird.

zwei Wörter


```
\makebox[5cm]{zwei Wörter} \par
\framebox[5cm][r]{zwei Wörter}
```

Bsp.
A-2-1

Der Text lässt sich mit dem Positionsargument `[c]` (voreingestellt) zentrieren. Darüber hinaus kann man ihn auch linksbündig (`[l]`) oder rechtsbündig (`[r]`) ausrichten. Mithilfe des Argumentes `[s]` kann man den *text* zudem vom linken bis zum rechten Rand der Box dehnen, sofern er dehnbaren Leerraum enthält (z.B. in Form von `\hspace` oder den in Tabelle A.2 auf Seite 889 aufgeführten vordefinierten Abständen). Auch Wortzwischenräume lassen sich dehnen (und zu einem gewissen Grad stauchen), wie auf Seite 440 erläutert. Das Erscheinungsbild von gerahmten Boxen lässt sich mithilfe zweier Layoutparameter steuern:

`\fboxrule` Stärke der Rahmenlinien, die mit dem Befehl `\fbox` oder `\framebox` erzeugt werden; sie ist in allen Standardklassen auf 0.4pt voreingestellt.

`\fboxsep` Abstand zwischen dem Rand der von `\fbox` oder `\framebox` erzeugten Box und ihrem Inhalt; er ist in allen Standardklassen auf 3pt voreingestellt.

Alle Änderungen dieser Parameter folgen den normalen Regeln für Geltungsbereiche und wirken sich auf alle gerahmten Boxen im Geltungsbereich aus. So betrifft beispielsweise die Änderung von `\fboxsep` im nächsten Beispiel nur die zweite Box.

Normal

Breit & Fett

Fett

```
\fbox{Normal} \hfill
\setlength\fboxrule{2pt}%
{\setlength\fboxsep{2mm}\fbox{Breit & Fett}}
\hfill \fbox{Fett}
```

Bsp.
A-2-2

Für die Boxenbefehle, die über Argumente für die Maße der jeweiligen Boxen verfügen, kann man vier besondere Längenparameter verwenden: `\width`, `\height`, `\depth` und `\totalheight`. Sie bestimmen die natürliche Größe des Argumentes *text*, wobei sich `\totalheight` aus der Summe von `\height` und `\depth` ergibt.

Ein paar gute Tipps

Ein paar gute Tipps

Ein paar gute Tipps

```
\usepackage{calc}
```

```
\framebox{ Ein paar gute Tipps } \par
\framebox[\width + 9mm][s]{ Ein paar gute Tipps }
\par \framebox[1.5\width]{ Ein paar gute Tipps }
```

Bsp.
A-2-3

Boxen ohne horizontale Ausdehnung (Breite null) sind sehr praktisch, um Stellen auf einer Seite zu markieren (z.B. zum Positionieren von Fußnotenmarken in einem Titel) oder um Text in den Rand zu schreiben. Das Prinzip dieses Verfahrens ist im nächsten Beispiel zu sehen, in dem eine solche Box genutzt wird, um Text mit einer Markierung zu versehen, ohne seine zentrierte Position zu beeinflussen. Man beachte, dass der optionale Parameter `[l]` das

Material außen auf die rechte und [r] es auf die linke Seite setzt.

Bsp.
A-2-4

Ein Satz.¹²³
Weiterer Text in der Mitte.
³²¹Ein Satz.

```
\centering
Ein Satz.\makebox[0pt][l]{\textsuperscript{123}}\\
Weiterer Text in der Mitte. \\
\makebox[0cm][r]{\textsuperscript{321}}Ein Satz.
```

⇔ Wie am Rand dieser Zeile zu sehen, kann man Boxen ohne horizontale Ausdehnung dazu verwenden, Text in den Rand fließen zu lassen. Für diesen Effekt wurde dieser Absatz folgendermaßen eingeleitet:

```
\noindent\makebox[0cm][r]{\$\Longleftarrow$}%
Wie am Rand ...
```

Eine interessante Möglichkeit besteht auch im Hoch- oder Tiefstellen von Boxen. Dieser Effekt lässt sich mithilfe des sehr leistungsfähigen Befehls `\raisebox` erzielen, der über zwei obligatorische sowie zwei optionale Argumente verfügt:

```
\raisebox{versatz}[höhe][tiefe]{inhalt}
```

Um die mit dem angegebenen *inhalt* erstellte Box anzuheben oder zu senken, gibt man die Größe für den *versatz* als Länge an. Ein negativer Wert senkt die Box. Genau wie bei anderen Boxen kann man auch hier die Sonderbefehle `\height`, `\depth`, `\totalheight` oder sogar `\width` verwenden, um sich auf die natürlichen Maße des *inhalts* zu beziehen. Im nächsten Beispiel wird auf diese Weise das Wort „empor“ soweit angehoben, dass die Unterlänge des „p“ bündig mit der Grundlinie abschließt, und das Wort „hinab“ wird ganz unter die Grundlinie gesenkt.

Bsp.
A-2-5

x111x empor x222x x333x
 hinab

```
x111x \raisebox{\depth}{empor} x222x
\raisebox{-\height}{hinab} x333x
```

Normalerweise berücksichtigt $\text{\LaTeX}$ die zusätzliche Höhe und Tiefe bei der Berechnung des Zeilenzwischenraums, so dass eine angehobene oder gesenkte Box Zeilen auseinander schieben kann. Dieser Effekt lässt sich beeinflussen, indem man $\text{\LaTeX}$ angibt, welche *höhe* und *tiefe* für das Material tatsächlich auf der Seite eingerechnet werden soll. Das zweite Zeilenpaar macht deutlich, dass $\text{\LaTeX}$ nicht bemerkt, dass Text nach oben oder unten verschoben ist. Dementsprechend sind die Zeilen so gesetzt, als befände sich der gesamte Text auf der Grundlinie.

Bsp.
A-2-6

x111x x222x
hinab empor
x333x x444x

```
\begin{flushleft}
x111x \raisebox{-1ex}{hinab} x222x \\
x333x \raisebox{1ex}{empor} x444x \\[4mm]
x111x \raisebox{-1ex}[0cm][0cm]{hinab} x222x \\
x333x \raisebox{1ex}[0cm]{empor} x444x
\end{flushleft}
```

Eine sinnvollere Verwendung wird in Abschnitt 5.7 auf Seite 280 besprochen, in dem es um Tabellenspalten (in `tabular`) geht, die sich über mehrere Zeilen erstrecken.

A.2.2 Absatzboxen

Absatzboxen werden mithilfe des Befehls `\parbox` oder der `minipage`-Umgebung gesetzt. Dabei erscheint der *text* im Absatzmodus in einer Box der vorgegebenen *breite*. Die vertikale Ausrichtung der Box zur Grundlinie wird durch das einbuchstabige optionale Argument *pos* ([c], [t] oder [b]) bestimmt.

<code>\parbox[<i>pos</i>]{<i>breite</i>}{<i>text</i>}</code>	<code>\begin{minipage}[<i>pos</i>]{<i>breite</i>}</code> <i>text</i> <code>\end{minipage}</code>

Wie im nächsten Beispiel zu sehen, ist die zentrierte Position voreingestellt. Man beachte, dass L^AT_EX im Blocksatz (der voreingestellt ist) große Wortzwischenräume erzeugt, wenn die Boxen extrem schmal sind.

Der Inhalt der Absatzbox auf der linken Seite.	ZEILE	Die Absatzbox auf der rechten Seite; man beachte, dass der Text zerrupft wirkt, da L ^A T _E X ihn in schmalen Spalten nicht gleichmäßiger ausrichten kann.	<pre> \parbox{.3\linewidth}{Der Inhalt der Absatzbox auf der linken Seite.} \hfill ZEILE\hfill \parbox{.4\linewidth}{Die Absatzbox auf der rechten Seite; man beachte, dass der Text zerrupft wirkt, da \LaTeX{} ihn in schmalen Spalten nicht gleichmäßiger ausrichten kann.} </pre>

Bsp.
A-2-7

Die `minipage`-Umgebung ist sehr hilfreich für das gezielte Positionieren von Material auf einer Seite. Sie ist eine echte vollständige Kleinausgabe einer Seite und kann somit eigene Fußnoten, Absätze sowie `array`-, `tabular`-, `multicols`- und andere Umgebungen enthalten. Sie kann zwar keine Gleitobjekte oder `\marginpar`-Befehle aufnehmen, dafür jedoch selbst in `figure`- oder `table`-Umgebungen erscheinen, in denen sie häufig benutzt wird, um das Material innerhalb der Gleitobjektumgebung ansprechend zu gestalten. Das nächste Beispiel zeigt auf einfache Weise, wie `minipage`-Umgebungen in der Praxis aussehen. Die Grundlinie wird durch einen Halbgeviertstrich angezeigt, der mit dem Befehl `\HR` erzeugt wurde. Man beachte die *pos*-Argumente ([c], [t] bzw. [b]) in den drei `minipage`-Umgebungen.

	<pre> \newcommand\HR{\rule{.5em}{0.4pt}} \HR \begin{minipage}[b]{12mm} A A A A A A A A A A A A A \end{minipage}\HR \begin{minipage}[c]{12mm} B B B B B B B B B B B B B B B B \end{minipage}\HR \begin{minipage}[t]{12mm} C C C C C C C \end{minipage}\HR </pre>
<pre> A A A A A A A A A B B B B A A A B B B B A A A B B B B -A A A -B B B B- C C C C- B B B B C C C B B B B B B B </pre>	

Bsp.
A-2-8

Für kompliziertere Anordnungen muss man die verschiedenen `minipage`-Umgebungen gegebenenfalls verschachteln. Die weiteren Beispiele bieten sich

zum Vergleich an. Hier wird versucht, die beiden linken Blöcke oben bündig auszurichten und den Block, den diese beiden dann bilden, wiederum durch eine weitere minipage-Ebene an dem verbleibenden Block unten bündig auszurichten.

		<code>\newcommand\HR{\rule{.5em}{0.4pt}}</code>
	<code>C C C C</code>	<code>\HR\begin{minipage}[b]{30mm}</code>
<code>_A A A xx B B B B _C C C _</code>		<code>\begin{minipage}[t]{12mm}</code>
<code>A A A B B B B</code>		<code>A A A A A A A A A A A A A A</code>
<code>A A A B B B B</code>		<code>\end{minipage} xx \begin{minipage}[t]{12mm}</code>
<code>A A A B B B B</code>		<code>B B B B B B B B B B B B B B B B B B</code>
<code>A A A B B B B</code>		<code>\end{minipage}</code>
<code>A A A B B B B</code>		<code>\end{minipage}\HR</code>
<code>B B B</code>		<code>\begin{minipage}[b]{12mm} C C C C C C C \end{minipage}\HR</code>

Bsp.
A-2-9

Es entsteht jedoch nicht das erwartete Ergebnis. Stattdessen bilden die beiden nach oben ausgerichteten minipage-Umgebungen innerhalb der nach unten ausgerichteten einen Absatz, der aus einer Zeile besteht (die minipage-Umgebungen werden einfach als große Elemente in der xx-Zeile betrachtet). Darum stimmt die unterste Linie der rechten minipage-Umgebung immer noch mit derjenigen der xx-Zeile überein. Um das zu vermeiden, muss man, wie im nächsten Beispiel zu sehen, nach dem Absatz einen unsichtbaren Leer-raum einfügen.

		<code>\newcommand\HR{\rule{.5em}{0.4pt}}</code>
		<code>\HR\begin{minipage}[b]{30mm}</code>
		<code>\begin{minipage}[t]{12mm}</code>
<code>A A A xx B B B B</code>		<code>A A A A A A A A A A A A A A</code>
<code>A A A B B B B</code>		<code>\end{minipage} xx \begin{minipage}[t]{12mm}</code>
<code>A A A B B B B</code>		<code>B B B B B B B B B B B B B B B B B B</code>
<code>A A A B B B B</code>		<code>\end{minipage}</code>
<code>A A A B B B B</code>		<code>\par\vspace{0mm}</code>
<code>A A A B B B B C C C C</code>		<code>\end{minipage}\HR</code>
<code>_ B B B _C C C _</code>		<code>\begin{minipage}[b]{12mm} C C C C C C C \end{minipage}\HR</code>

Bsp.
A-2-10

Im folgenden Falle werden die beiden rechten Umgebungen innerhalb einer weiteren Umgebung oben bündig zueinander ausgerichtet. Diese wiederum wird unten bündig zur linken Umgebung gesetzt. Bei einem Vergleich mit dem vorigen Beispiel stellt man fest, dass sich die beiden Resultate stark voneinander unterscheiden, obwohl die Reihenfolge der Ausrichtungsparameter dieselbe ist. Die minipage-Umgebungen sind lediglich anders verschachtelt.

		<code>\newcommand\HR{\rule{.5em}{0.4pt}}</code>
		<code>\HR\begin{minipage}[b]{12mm}</code>
<code>B B B B xx C C C C</code>		<code>A A A A A A A A A A A A A A \end{minipage}\HR</code>
<code>A A A B B B B C C C</code>		<code>\begin{minipage}[b]{30mm} \begin{minipage}[t]{12mm}</code>
<code>A A A B B B B</code>		<code>B B B B B B B B B B B B B B B B B B</code>
<code>A A A B B B B</code>		<code>\end{minipage} xx</code>
<code>A A A B B B B</code>		<code>\begin{minipage}[t]{12mm} C C C C C C C \end{minipage}</code>
<code>A A A B B B B</code>		<code>\par\vspace{0mm}</code>
<code>_A A A _B B B</code>		<code>_ \end{minipage}\HR</code>

Bsp.
A-2-11

Auch hier musste wieder ein vertikaler Leerraum eingefügt werden, um eine korrekte Ausrichtung zu erzielen. Dies bringt jedoch nicht immer das gewünschte Ergebnis. Enthält die letzte Zeile der verschachtelten `minipage` wie im folgenden Beispiel etwa einen Buchstaben mit Unterlänge, so ist die Ausrichtung der Grundlinien nicht perfekt.

```

\newcommand\HR{\rule{.5em}{0.4pt}}
B B B xx C C C C \HR\begin{minipage}[b]{12mm}
B B B B C C C A A A A A A A A A A A A A A A \end{minipage}\HR
A A A B B B B \begin{minipage}[b]{30mm} \begin{minipage}[t]{12mm}
A A A B B B B B B B B B B B B B B B B B B B B B B B gg jj
A A A B B B B \end{minipage} xx
A A A B B B B \begin{minipage}[t]{12mm} C C C C C C C \end{minipage}
A A A B B B B \par\vspace{0mm}
_A A A _gg jj _ \end{minipage}\HR

```

Bsp.
A-2-12

Zur Korrektur muss ein (negativer) vertikaler Abstand hinzugefügt werden, der die Tiefe der Buchstaben kompensiert.

Dies lässt sich wohl am einfachsten (wenn auch am gefährlichsten) mithilfe des `TEX`-Basisbefehls `\prevdepth` erreichen. Dieses Längenregister lässt sich nur im vertikalen Modus verwenden (also nach dem Ende eines Absatzes) und enthält die Tiefe der vorhergehenden Zeile. Im nächsten Beispiel geht man mithilfe dieses Befehls um eben diesen Betrag zurück und simuliert so, dass sich der untere Rand der Box auf der Grundlinie der letzten Zeile befindet.

Diese Verwendung von `\prevdepth` ist jedoch mit Vorsicht zu genießen. Wie bereits erwähnt, erzeugt der Befehl außerhalb des vertikalen Modus eine Fehlermeldung. Außerdem überlädt `TEX` diesen Basisbefehl am Anfang einer vertikalen Box und hinter einer horizontalen Linie mit dem Wert -1000pt ¹. Verwendet man im Beispiel also `\vspace*` anstelle von `\vspace` so erfährt man eine unliebsame Überraschung, da `\vspace*` eine unsichtbare Linie setzt, die dafür sorgt, dass der Leerraum auch an einem Seitenumbruch erhalten bleibt. Der Wert des inneren `\prevdepth` betrüge also -1000pt so dass tatsächlich am Fuß der Box ein Abstand von 1000 Punkten eingefügt würde.

Überraschende
Effekte von
`\prevdepth`

```

\newcommand\HR{\rule{.5em}{0.4pt}}
B B B xx C C C C \HR\begin{minipage}[b]{12mm}
B B B B C C C A A A A A A A A A A A A A A A \end{minipage}\HR
A A A B B B B \begin{minipage}[b]{30mm} \begin{minipage}[t]{12mm}
A A A B B B B B B B B B B B B B B B B B B B B B B B gg jj
A A A B B B B \end{minipage} \par\vspace{-\prevdepth}
A A A B B B B \end{minipage} xx
A A A B B B B \begin{minipage}[t]{12mm} C C C C C C C \end{minipage}
A A A B B B B \par\vspace{0pt}
_A A A _gg jj _ \end{minipage}\HR

```

Bsp.
A-2-13

Manchmal ist es hilfreich, die vertikalen Abmessungen einer Absatzbox vorab festzulegen. Zu diesem Zweck bietet das moderne `LATEX` zusätzliche optionale Argumente für den Befehl `\parbox` und die `minipage`-Umgebung.

¹`TEX` errechnet anhand von `\prevdepth` den erforderlichen Durchschuss und ein Wert von -1000pt zeigt an, dass dieser Abstand unterdrückt werden soll.

```
\parbox[pos][höhe][innere-pos]{breite}{text}
\begin{minipage}[pos][höhe][innere-pos]{breite}text\end{minipage}
```

Das Argument *innere-pos* bestimmt, wo in der Box der *text* positioniert ist. Es kann den Wert t, c, b oder s annehmen. Ist kein Wert angegeben, so wird der Wert des Argumentes *pos* benutzt. Man kann sich die Argumente *höhe* und *innere-pos* als vertikale Entsprechungen von *breite* und *pos* des `\makebox`-Befehls vorstellen. Durch den Positionsparameter s wird der *text* vertikal auf die angegebene *höhe* gedehnt. Daher muss man hier gegebenenfalls vertikal dehnbare Abstände einbauen, z.B. mithilfe des `\vspace`-Befehls.

Genau wie bei den anderen Boxbefehlen kann man auch hier im optionalen Argument mit `\height`, `\totalheight` usw. auf die natürlichen Maße der Box Bezug nehmen.

```
\usepackage{calc}
```

Etwas Text ganz oben. Und ein paar Zeilen am unteren Rand der Box.	Hier ein paar Zeilen am oberen Rand der Box. Und nur eine Zeile hier unten.	<pre>xx \fbox{\parbox[b][\height+\baselineskip][s] {20mm}{Etwas Text ganz oben. \par\vfill Und ein paar Zeilen am unteren Rand der Box.}} \fbox{\parbox[b][\height+\baselineskip][s] {20mm}{Hier ein paar Zeilen am oberen Rand der Box. Und nur eine Zeile \par\vfill hier unten.}} xx</pre>

Bsp.
A-2-14

A.2.3 Linienboxen

Linienboxen in $\text{\LaTeX}$ werden mit dem Befehl `\rule` gezeichnet:

```
\rule[versatz]{breite}{höhe}
```

Durch den Befehl `\rule[4pt]{2cm}{1mm}` erhält man eine 2 cm lange Linie von 1 mm Stärke, die sich 4pt über der Grundlinie befindet: . Mithilfe des `\rule`-Befehls lassen sich auch Linienboxen ohne horizontale Ausdehnung (Breite null) erzeugen, also unsichtbare Linien, so genannte *Struts* (Streben). Diese Struts sind sehr nützlich, um die Höhe oder Breite einer gegebenen Box zu steuern, z.B. um die Höhe einer mit `\fbox` oder `\framebox` gerahmten Box zu vergrößern oder um den Abstand zwischen Tabellenzeilen lokal anzupassen. Man vergleiche folgende Beispiele:

x111x	etwas Text	x222x	mehr Text	x333x	<pre>x111x \fbox{etwas Text} x222x \fbox{\rule[-5mm]{0cm}{15mm}mehr Text} x333x</pre>

Bsp.
A-2-15

Wie bereits erwähnt, verhalten sich Boxen (auch Linien) in $\text{\LaTeX}$ wie Zeichen. Werden sie beispielsweise außerhalb eines Absatzes verwendet, so beginnen sie automatisch einen neuen Absatz. Bei Linien ist das nicht immer erwünscht. Um etwa eine Linie zwischen zwei Absätzen zu setzen, muss man

den Absatzeinzug mit dem Befehl `\noindent` unterdrücken. Ansonsten würde die Linie eingezogen und rechts aus dem Text herausragen.

... Text für unsere Seite, der immer und immer wieder genutzt wird.

Der nächste Absatz. Text für unsere Seite, der immer und immer wieder genutzt wird.

```
\newcommand\sample{ Text für unsere Seite,
  der immer und immer wieder genutzt wird.}

\ldots \sample \par
\noindent\rule{\linewidth}{0.4pt} \par
Der nächste Absatz. \sample
```

Bsp.
A-2-16

Aufgrund dieses Verhaltens befindet sich die Linie auf der Grundlinie eines einzeiligen Absatzes und liegt daher visuell viel dichter am nächsten Absatz. Mithilfe des optionalen Argumentes *versatz* kann man die Linie genau in der Mitte zwischen den beiden Zeilen platzieren. Der richtige Wert (in diesem Falle ungefähr 2.5pt) lässt sich jedoch nur durch Ausprobieren finden.

Eine mögliche Lösung besteht darin, den Zeilenzwischenraum mithilfe des TeX-Low-Level-Befehls `\nointerlineskip` zu unterdrücken und dann, wie im folgenden Beispiel, die erforderlichen Abstände explizit einzufügen. Dieses Mal wird der Befehl `\noindent` nicht verwendet, so dass die Linie um `\parindent` eingerückt erscheint. Dafür wird die Länge der Linie mithilfe des `calc`-Paketes so berechnet, dass auch auf der rechten Seite ein Freiraum von `\parindent` verbleibt.

... Text für unsere Seite, der immer und immer wieder genutzt wird.

Der nächste Absatz. Text für unsere Seite, der immer und immer wieder genutzt wird.

```
\usepackage{calc} % \sample wie zuvor

\ldots \sample \par
\nointerlineskip \vspace{5.8pt}
\rule{\linewidth-2\parindent}{0.4pt}\par
\nointerlineskip \vspace{5.8pt}
Der nächste Absatz. \sample
```

Bsp.
A-2-17

Die Summe aus den verwendeten vertikalen Abständen und der Linienstärke beträgt 12 Punkte (also den Wert von `\baselineskip`). Dadurch sind jedoch nicht die Grundlinien der beiden Absätze 12 Punkte voneinander entfernt: Vielmehr ist das der Abstand zwischen dem unteren Rand der letzten Zeile des ersten Absatzes (der durch das „g“ in „genutzt“ festgelegt wird) und dem oberen Rand der ersten Zeile des nächsten Absatzes (wie er durch das „A“ erzeugt wird). Wünscht man allerdings, dass die Grundlinien ein bestimmtes Raster einhalten, dann verwendet man besser eine Variante von Beispiel A-2-16 mit dem optionalen Argument *versatz*.

Klassen- oder Paketentwickler verwenden anstelle der Kombination von `\rule` und `\nointerlineskip` häufig die TeX-Basisbefehle für Linien. Ihr Vorteil liegt darin, dass sie automatisch den Durchschuss unterdrücken und dass man nicht alle Maße angeben muss. Allerdings bedienen sie sich einer ungewöhnlichen Syntax und man kann sie nicht einsetzen, wenn die Linie, wie im vorigen Beispiel, horizontal oder vertikal versetzt werden muss.

```
\hrule height höhe depth tiefe width breite \relax
\vrule height höhe depth tiefe width breite \relax
```

Der Basisbefehl `\hrule` darf nur zwischen Absätzen und `\vrule` nur innerhalb von Absätzen verwendet werden. Befinden sie sich am falschen Ort, so

	width	height	depth
<code>\hrule</code>	*	0.4pt	0.0pt
<code>\vrule</code>	0.4pt	*	*

Tabelle A.4: Voreinstellungen für Linienbasisbefehle in T_EX

beginnen oder beenden sie einen Absatz je nach Bedarf. Auf die Befehle können ein oder mehrere der Schlüsselwörter `height`, `depth` und `width` mit jeweils einem Längenwert folgen. Die Reihenfolge ist dabei beliebig und fehlende Schlüsselwörter erhalten die voreingestellten Werte aus Tabelle A.4. Ein Stern in dieser Tabelle bedeutet, dass sich die Linie bis zum Rand der äußeren Box erstreckt. Der Befehl `\relax` am Ende ist nicht unbedingt erforderlich. Er sorgt jedoch dafür, dass T_EX sicher weiß, dass die Linienangaben beendet sind und somit nicht fälschlicherweise Wörter im folgenden Text als Schlüsselwörter interpretiert.

Im nächsten Beispiel wird `\hrule` mit den Voreinstellungen verwendet, so dass man eine Linie der Stärke 0.4pt erhält, die sich über die gesamte Druckfahne erstreckt (da diese hier effektiv die äußere Box darstellt).

... Text für unsere Seite, der immer und immer wieder genutzt wird.

% \sample wie zuvor

```
\ldots \sample \par
\vspace{3pt}\hrule\relax\vspace{3pt}
Der nächste Absatz. \sample \par
```

Bsp.
A-2-18

Der nächste Absatz. Text für unsere Seite, der immer und immer wieder genutzt wird.

A.2.4 Arbeiten mit Boxregistern

Man kann Daten setzen und dann in einer benannten Box speichern, deren Inhalt sich immer wieder verwenden lässt.

<code>\newsavebox{befehl}</code>	Box deklarieren
<code>\sbox{befehl}{text}</code>	Box füllen
<code>\savebox{befehl}[breite][pos]{text}</code>	Box füllen
<code>\usebox{befehl}</code>	Inhalt verwenden

Der Befehl `\newsavebox` deklariert global einen Befehl `befehl` (z.B. `\mybox`), den man sich als benannte Zwischenablage vorstellen kann. In dieser Box kann man gesetztes Material zur (mehrfachen) Wiederverwendung speichern.

Die Befehle `\sbox` und `\savebox` gleichen `\mbox` und `\makebox`, nur dass sie die erstellte Box in einem (zuvor mit `\newsavebox` zugewiesenen) benannten Zwischenspeicher ablegen, anstatt sie direkt zu setzen. Mithilfe des Befehls `\usebox` lässt sich das gespeicherte Material dann wieder abrufen, ohne dass es dabei gelöscht wird. Der gleiche Zwischenspeicher (z.B. `\mybox`) lässt sich im Geltungsbereich der aktuellen Umgebung oder der in Klammern stehenden Gruppe immer wieder verwenden. Er enthält jeweils die Daten, die als Letztes darin gespeichert wurden.

Man muss darauf achten, den Befehlsnamen `\mybox` nicht direkt zu verwenden, da er lediglich eine Referenz auf die interne Registernummer der fraglichen Box enthält. Verwendet man `\mybox` allein, erzeugt dies kurioser-

 *Überraschende Zeichen in der Ausgabe*

weise das Zeichen, das sich im aktuellen Font an der Position befindet, auf welche die Nummer der Box verweist. Daher sollte man Boxen nur mithilfe der zuvor beschriebenen Befehle verarbeiten.

		<code>\newsavebox{\myboxa}\newsavebox{\myboxb}</code>	
		<code>\sbox{\myboxa}{in Box a}</code>	
		<code>\savebox{\myboxb}[2cm][1]{in Box b}</code>	
		<code>x1x \usebox{\myboxa} x2x \usebox{\myboxb} x3x</code>	
		<code>\savebox{\myboxb}[2cm][r]{in Box b}</code>	
<code>x1x in Box a</code>	<code>x2x in Box b</code>	<code>x3x</code>	<code>\par</code>
<code>x1x in Box a</code>	<code>x2x in Box b</code>	<code>x3x</code>	<code>x1x \usebox{\myboxa} x2x \usebox{\myboxb} x3x</code>

Bsp.
A-2-19

Zusätzlich zu den zuvor besprochenen Befehlen gibt es noch die Umgebung `lrbox` mit folgender Syntax:

```
\begin{lrbox}{befehl} text \end{lrbox}
```

Das Argument *befehl* sollte hier ein Boxregisterbefehl sein, der zuvor mit `\newsavebox` zugewiesen wurde. Die Umgebung `lrbox` speichert in dieser Box den *text* für die weitere Verwendung mit `\usebox`. Leerzeichen am Anfang und am Ende werden ignoriert. Somit handelt es sich bei `lrbox` vom Prinzip her um die Umgebungsvariante des Befehls `\sbox`. Man kann diese Umgebung gut nutzen, wenn man den Inhalt einer Umgebung in einer Box abspeichern möchte, um ihn später weiterzuverarbeiten. Der folgende Code beispielsweise definiert die Umgebung `fcolumn`, die sich wie eine `minipage`-Umgebung von der Breite einer Spalte verhält, dabei jedoch eingerahmt ist.

```
\usepackage{calc}
\newsavebox{\fcolbox} \newlength{\fcolwidth}
\newenvironment{fcolumn}[1][\linewidth]
{
  \setlength{\fcolwidth}{\#1-2\fboxsep-2\fboxrule}%
  \begin{lrbox}{\fcolbox}\begin{minipage}{\fcolwidth}
  \fcolumn{minipage}\end{lrbox}\noindent\fbbox{\fcolbox}}
\begin{fcolumn} In dieser Umgebung kann man Verbatim-Text
wie \verb=\fcolbox= verwenden. \end{fcolumn}
```

In dieser Umgebung kann man Verbatim-Text wie `\fcolbox` verwenden.

Bsp.
A-2-20

Die vorige Definition ist in mehrerer Hinsicht interessant. Die Umgebung wird mit einem optionalen Argument definiert, das die Breite der fertigen Box festlegt (voreingestellt auf `\linewidth`). In der nächsten Zeile wird (mithilfe des `calc`-Paketes) die interne Zeilenlänge berechnet, die an die `minipage`-Umgebung übergeben werden muss. Dabei ist der vom `\fbbox`-Befehl auf beiden Seiten hinzugefügte Leerraum wieder abzuziehen. Dann werden die Umgebungen `lrbox` und `minipage` aufgerufen, um den Inhalt der `fcolumn`-Umgebung in die Box `\fcolbox` zu schreiben. Am Ende der Umgebung werden diese Umgebungen wieder geschlossen. Danach wird die Box `\fcolbox` in einem `\fbbox`-Befehl gesetzt. Der vorangehende Befehl `\noindent` unterdrückt jeglichen Einzug für den Fall, dass die Umgebung am Anfang eines Absatzes steht oder selbst einen Absatz bildet.

Die in Abschnitt 10.1.1 auf Seite 613 beschriebene `boxedminipage`-Umgebung lässt sich auf ähnliche Weise implementieren. Der entscheidende Unterschied zum vorherigen Code besteht lediglich darin, dass `\noindent`

wegfällt und die Breite als obligatorisches und die Position als optionales Argument übergeben werden.

```
\usepackage{calc}
\newsavebox{\fcolbox} \newlength{\fcolwidth}
\newenvironment{boxedminipage}[2][c]
{
  \setlength{\fcolwidth}{#2-2\fboxsep-2\fboxrule}%
  \begin{lrbox}{\fcolbox}%
  \begin{minipage}[#1]{\fcolwidth}%
  \end{minipage}\end{lrbox}\fbox{\usebox{\fcolbox}}
}
links \begin{boxedminipage}[b]{3.2cm}
      In dieser Umgebung kann man Verbatim-Text
      wie \verb=\fcolbox= verwenden.
\end{boxedminipage}
rechts rechts
```

Bsp.
A-2-21

In dieser Umgebung
kann man Verbatim-
Text wie `\fcolbox`
verwenden.

links

rechts

Vergleicht man diese Definition mit dem eigentlichen Paketcode, der aus L^AT_EX 2.09 stammt, dann wird deutlich, dass die Programmiermöglichkeiten der aktuellen L^AT_EX-Version ihre Vorteile haben.

A.2.5 Boxbefehle und Farben

Auch wenn man in eigenen Dokumenten keine Farben verwenden möchte, kann man durch Berücksichtigen der Informationen in diesem Abschnitt zumindest sicherstellen, dass eigene Klassen oder Pakete mit dem `color`-Paket kompatibel sind. Dadurch ist denjenigen geholfen, welche gerade diese Klassen oder Pakete mit den Erweiterungen des `color`-Paketes kombinieren wollen.

Der einfachste Weg zu „sicheren Farben“ besteht darin, dass man anstelle von T_EX-Basisbefehlen immer die L^AT_EX-Boxbefehle verwendet: also eher `\sbox` als `\setbox`, `\mbox` statt `\hbox` und `\parbox` oder die `minipage`-Umgebung anstelle von `\vbox`. Die L^AT_EX-Befehle für Boxen verfügen über neue Optionen, durch die sie genauso leistungsfähig sind wie die Basisbefehle von T_EX.

Hier ein Beispiel dafür, was schief gehen kann, wenn man T_EX-Basisbefehle einsetzt: In der Anweisung `{\ttfamily text}` wird der Font direkt *vor* der Klammer `}` wiederhergestellt, während in dem ähnlich erscheinenden Konstrukt `{\color{green} text}` die Farbe direkt *nach* der letzten geschweiften Klammer wiederhergestellt wird. Normalerweise fällt dieser Unterschied nicht ins Gewicht. Angenommen, es liegt nun jedoch eine Boxzuweisung über einen T_EX-Basisbefehl vor, wie etwa:

```
\setbox0=\hbox{\color{green} irgendein Text}
```

Hier erfolgt die Wiederherstellung der Farbe nach der letzten geschweiften Klammer und wird damit *nicht* in der Box gespeichert. Wie negativ sich dies auswirkt, ist davon abhängig, wie Farben eingesetzt werden: Die Probleme können von einer falschen Farbgebung im übrigen Dokument bis zu Fehlern im dvi-Treiber reichen, mit dessen Hilfe das Dokument gedruckt werden soll.

Ein weiterer interessanter Befehl ist `\normalcolor`. Er verhält sich normalerweise einfach wie `\relax` (führt also nichts aus). Man kann ihn jedoch

wie den Befehl `\normalfont` verwenden, um Seitenbereiche wie Legenden oder Überschriften auf die „Hauptfarbe des Dokumentes“ einzustellen.

A.3 Erweiterte Steuerfunktionen

A.3.1 calc – Arithmetische Funktionen

Das Paket `calc` von Kresten Thorup und Frank Jensen enthält eine Reihe von Makros zur Erweiterung der arithmetischen Funktionen in $\text{\LaTeX}$. In $\text{\TeX}$ werden alltägliche Berechnungen mithilfe einfacher Low-Level-Operationen wie `\advance` und `\multiply` durchgeführt. Das `calc`-Paket definiert eine Arithmetik mit Infixnotation für $\text{\LaTeX}$. Dazu verändert es die $\text{\LaTeX}$ -Befehle `\setcounter`, `\addtocounter`, `\setlength` und `\addtolength` so, dass sie nicht nur einfache Zahlen und Längen, sondern Integer- und Längenausdrücke aufnehmen können.

Ein Integerausdruck kann ganze Zahlen, $\text{\TeX}$ -Integerregister, $\text{\LaTeX}$ -Zähler (z.B. `\value{zähler}`), Klammern und Binäroperatoren (`-`, `+`, `*`, `/`) enthalten. Um einen Zähler um fünf zu erhöhen, kann man schreiben:

```
\usepackage{calc} \newcounter{local}
\setcounter{local}{2} % Starteinstellung für das Beispiel
Der Wert beträgt zurzeit „2“.
Der Wert wurde nun zu „7“ verändert.
\setcounter{local}{\value{local}+5}
Der Wert wurde nun zu “\thelocal” verändert.
```

Bsp.
A-3-1

Als Beispiel bietet sich an, einen Befehl zu definieren, der die Uhrzeit ausgibt (das $\text{\TeX}$ -Register `\time` enthält die Anzahl der Minuten seit Mitternacht):

```
\usepackage{calc} \newcounter{hours}\newcounter{minutes}
\newcommand\printtime{\setcounter{hours}{\time/60}%
\setcounter{minutes}{\time-\value{hours}*60}%
\thehours h \theminutes min}
```

Es ist jetzt 21h 18min. Es ist jetzt `\printtime`.

Bsp.
A-3-2

In einem Längenausdruck müssen Teilausdrücke, die addiert oder subtrahiert werden, stets vom selben Typ sein, d.h. entweder Längen oder ganze Zahlen. Man darf also nicht „`2cm+4`“ schreiben, wohl aber einen Ausdruck wie „`2cm+4pt`“, da es sich bei beiden Teilen um Längen handelt. Dividieren und Multiplizieren lässt sich nur mit ganzen Zahlen, so dass „`2cm*4`“ ein zulässiger Teilausdruck ist, „`2cm*4pt`“ jedoch verboten. Außerdem muss der Längenteil als erstes stehen, „`4*2cm`“ ist also unzulässig.

Mithilfe der zuvor beschriebenen Befehle lässt sich die Breite einer Spalte in einem n -spaltigen Layout mit dem folgenden einzelnen Befehl berechnen (vorausgesetzt die Variable n ist als erstes Argument eines $\text{\LaTeX}$ -Makros gespeichert):

```
\setlength\linewidth{(\textwidth-\columnsep*(#1-1))/#1}
```

Die Einschränkung, dass man nur mit ganzen Zahlen multiplizieren oder durch solche teilen kann, wurde für Längenberechnungen aufgehoben. Hier sind auch reelle Zahlen erlaubt.

```
\real{dezimalkonstante} \ratio{längenausdruck}{längenausdruck}
```

Eine reelle Zahl lässt sich in zwei Formen darstellen: Der erste Befehl konvertiert die *dezimalkonstante* in eine für calc-Formeln geeignete Form. Die zweite Form gibt die reelle Zahl wieder, die man erhält, wenn man den Wert des ersten Ausdrucks durch den des zweiten teilt.

Angenommen, man will eine Graphik so skalieren, dass sie sich über die gesamte Seitenbreite (`\textwidth`) erstreckt. Sind dabei die Originalmaße der Abbildung durch die Längenvariablen `\Xsize` und `\Ysize` gegeben, so berechnet sich die Höhe der Abbildung nach der Skalierung folgendermaßen:

```
\setlength\newYsize{\Ysize*\ratio{\textwidth}{\Xsize}}
```

Das calc-Paket wird in vielen Beispielen dieses Buches verwendet. Wer darauf verzichten will, muss den jeweiligen Beispielcode in (L^AT_EX)-Basisbefehle umsetzen. So muss beispielsweise die Einstellung von `\fcolwidth` auf Seite 902 von

```
\setlength\fcolwidth{#1-2\fbboxsep-2\fbboxrule}%
```

in die folgenden Anweisungen übersetzt werden:

```
\setlength\fcolwidth{#1}
\addtolength\fcolwidth{-2\fbboxsep}
\addtolength\fcolwidth{-2\fbboxrule}
```

Abgesehen von der Tatsache, dass die Infixnotation des calc-Paketes sicherlich einfacher zu lesen ist (und sich viel einfacher verändern lässt), enthält das Paket zudem Befehle für die Division und Multiplikation, die mit L^AT_EX-Standardbefehlen nicht realisiert werden können. So benötigt man beispielsweise den folgenden Code, um die `\topmargin`-Berechnung von Seite 206 durchzuführen:

```
\setlength\topmargin{297mm}
\addtolength\topmargin{-\textheight}

% TeX-Berechnung für die es keine LaTeX-Form gibt:
\divide\topmargin by 3

\addtolength\topmargin{-1in}
\addtolength\topmargin{-\headheight}
\addtolength\topmargin{-\headsep}
```

A.3.2 ifthen – Verbesserte Steuerfunktionen

Manchmal möchte man vielleicht in Abhängigkeit von dem Wert eines logischen Ausdrucks verschiedene Inhalte setzen. Dies wird mit dem Standardpaket `ifthen` von Leslie Lamport möglich, das für die heutige L^AT_EX-Version von David Carlisle reimplementiert wurde. Das Paket definiert Befehle, durch die man mit L^AT_EX Steuerfunktionen erstellen kann.

```
\ifthenelse{test}{dann}{sonst}
```

Ist die Bedingung *test* wahr, so werden die Befehle im Argument *dann* ausgeführt. Ansonsten werden die Befehle in *sonst* ausgeführt.

Eine einfache Bedingung ist der Vergleich zweier ganzer Zahlen. Angenommen man will einen Zählerwert in einem englischen Text in englischer Schreibweise darstellen:

```
\usepackage{ifthen}
\newcommand\toEng[1]{\arabic{#1}\textsuperscript{%
\ifthenelse{\value{#1}=1}{st}{%
\ifthenelse{\value{#1}=2}{nd}{%
\ifthenelse{\value{#1}=3}{rd}{%
\ifthenelse{\value{#1}<20}{th}}}%
\typeout{Zu hoher Wert}}}}}
```

This is the 3rd section
in the 1st appendix.

This is the \toEng{section} section in the \toEng{chapter}
appendix.

Bsp.
A-3-3

Im nächsten Beispiel wird ein Befehl definiert, der die Uhrzeit in kurzer Schreibweise ausgibt. Es zeigt, wie komplexe Operationen (mithilfe des *calc*-Paketes) mit bedingten Steuerfunktionen kombiniert werden können.

```
\usepackage{ifthen,calc}
\newcounter{hours}\newcounter{minutes}
\newcommand\Printtime{\setcounter{hours}{\time/60}%
\setcounter{minutes}{\time-\value{hours}*60}%
\ifthenelse{\value{hours}<10}{0}{\thehours:%
\ifthenelse{\value{minutes}<10}{0}{\theminutes}}
```

Es ist jetzt „21:18“ Uhr.

Es ist jetzt "\Printtime" Uhr.

Bsp.
A-3-4

```
\equal{zeichenfolge1}{zeichenfolge2}
```

Der Befehl `\equal` ergibt das Ergebnis *true* (wahr), wenn *zeichenfolge₁* und *zeichenfolge₂* übereinstimmen, nachdem sie vollständig expandiert wurden. Bei Verwendung zerbrechlicher Befehle in einer der Zeichenfolgen ist besondere Vorsicht geboten: Sie müssen mit `\protect` geschützt werden.

```
\usepackage{ifthen,shortvrb} \MakeShortVerb\|
\newcommand\BB{\CC}\newcommand\CC{\DD}
\newcommand\DD{AA} \newcommand\EE{EE}
```

`\BB=\EE?` False.

`\BB=\CC?` True.

`\DD=\BB?` True.

```
| \BB |= | \EE | ? \ifthenelse{\equal{\BB}{\EE}}{True}{False}. \par
| \BB |= | \CC | ? \ifthenelse{\equal{\BB}{\CC}}{True}{False}. \par
| \DD |= | \BB | ? \ifthenelse{\equal{\DD}{\BB}}{True}{False}.
```

Bsp.
A-3-5

Eine praktische Anwendung für den vorigen Befehl wäre die Definition eines Befehls, der sein Argument ausgibt und außerdem in den Index einträgt. Der Hinweis auf die Stelle, an welcher der Begriff definiert wird, soll im Index

fett gedruckt werden, alle anderen in normaler Schrift. Für den selteneren Fall wird ein optionales Argument verwendet.

Bsp.
A-3-6

Der Begriff AAAA wird definiert. ...
Der Begriff AAAA wird besprochen.

```
\usepackage{ifthen}
\newcommand{\IX}[2][R]{\texttt{#2}%
\ifthenelse{\equal{#1}{D}}{%
\index{#2|textbf}}{\index{#2}}}
\ldots{} Der Begriff \IX{AAAA} wird definiert.
\ldots{} Der Begriff \IX{AAAA} wird besprochen.
```

Dadurch erhält man die gewünschte Darstellung in der .idx-Datei, in welcher Einträge der folgenden Art erzeugt werden:

```
\indexentry{AAAA|textbf}{907} \indexentry{AAAA}{907}
```

Ein komplexeres Beispiel, in dem man die vollständige Kontrolle darüber hat, welche Daten in den Index oder Text aufgenommen werden, bedient sich des erweiterten Indexbefehls `\IXE`, der im nächsten Beispiel definiert wird. Sein voreingestelltes optionales Argument „`!*!, !`“ enthält eine Zeichenfolge, die so (hoffentlich) in keinem Text vorkommt. Verwendet man den Befehl `\IXE` mit nur einem (normalen) Argument, so erscheinen im Index und im Text die gleichen Informationen. Mithilfe des optionalen Argumentes kann man im Index andere Informationen eingeben als sie im Text stehen. Im nächsten Beispiel sind alle möglichen Kombinationen aufgeführt. An den vertikalen Balken um die Befehle herum kann man erkennen, dass keine unerwünschten Leerzeichen erzeugt werden.

Bsp.
A-3-7

Text und Index identisch
|**beide**|. Text und Index unter-
schiedlich |**text**|. Nur im Index
||. Nur im Text |**nur-text**|. We-
der im Text noch im Index ||.

```
\usepackage{ifthen}
\newcommand\IXE[2][!*!, !]{%
\ifthenelse{\equal{#1}{!*!, !}}{%
\ifthenelse{\equal{#2}{}}{\textbf{#2}\index{#2}}{%
\ifthenelse{\equal{#1}{}}{\index{#1}}%
\ifthenelse{\equal{#2}{}}{\textbf{#2}}}}
Text und Index identisch |\IXE{beide}|.
Text und Index unterschiedlich |\IXE[index]{text}|.
Nur im Index |\IXE[nur-index]{}|.
Nur im Text |\IXE[] {nur-text}|.
Weder im Text noch im Index |\IXE[]{}|.
```

Die .idx-Datei enthält nur drei Einträge, da der Befehl mit leerem optionalem Argument „`[]`“ keinen Indexeintrag erzeugt.

```
\indexentry{beide}{907}
\indexentry{index}{907}
\indexentry{nur-index}{907}
```

<code>\boolean{name}</code> <code>\newboolean{name}</code> <code>\setboolean{name}{wert}</code>

L^AT_EX kennt einige Schalter, die den Wert `true` oder `false` (wahr oder falsch) annehmen können. Solche Schalter lassen sich mithilfe der Deklaration `\newboolean` selbst definieren, wobei das Argument *name* aus einer Folge

T _E X-Schalter (können nur abgefragt werden)	
<code>hmode</code>	true, wenn in horizontaler Richtung gesetzt wird (z.B. in einem Absatz oder einer LR Box).
<code>vmode</code>	true, wenn in vertikaler Richtung gesetzt wird (z.B., wenn T _E X sich zwischen zwei Absätzen befindet).
<code>mmode</code>	true, wenn T _E X eine Formel setzt.
L ^A T _E X-Schalter (die letzten beiden können gesetzt werden)	
<code>@twoside</code>	true, wenn L ^A T _E X für den zweiseitigen Ausdruck setzt.
<code>@twocolumn</code>	true, wenn L ^A T _E X im Standard-Zweispaltenmodus setzt (<code>false</code> innerhalb von <code>multicols</code> -Umgebungen).
<code>@firstcolumn</code>	true, wenn <code>@twocolumn</code> true ist und L ^A T _E X die erste Spalte setzt.
<code>@newlist</code>	true, wenn L ^A T _E X sich am Anfang einer Listenumgebung befindet (wird auf <code>false</code> gesetzt, sobald der Text <i>nach</i> dem ersten <code>\item</code> -Befehl erreicht ist).
<code>@inlabel</code>	true, nach einem <code>\item</code> -Befehl bis der darauf folgende Text erreicht ist.
<code>@noskipsec</code>	true, nach einer eingebetteten Überschrift, bis der darauf folgende Text erreicht ist.
<code>@afterindent</code>	Schalter, der von dem Befehl <code>\@afterheading</code> geprüft wird; dieser Befehl wird normalerweise in Überschriften verwendet und verhindert (bei <code>false</code>) den Einzug des nächsten Absatzes.
<code>@tempswa</code>	Temporärer Schalter, der von vielen L ^A T _E X-Befehlen zur internen Kommunikation verwendet wird.

Tabelle A.5: Interne `\boolean`-Schalter in L^AT_EX

von Buchstaben besteht.¹ Dieser Schalter wird zunächst auf `false` gesetzt. Er wird mit dem Befehl `\setboolean` geändert, dessen Argument *wert* entweder den Text `true` oder `false` enthält. Danach kann man den Wert mithilfe des Befehls `\boolean` im ersten Argument von `\ifthenelse` prüfen. Mit `\boolean` lassen sich alle derartigen internen L^AT_EX-Schalter testen (die gängigsten von ihnen sind in Tabelle A.5 aufgeführt). Als Beispiel könnte man überprüfen, ob ein Dokument ein Layout zum ein- oder zweiseitigen Druck verwendet.

Doppelseitiger Druck. `\usepackage{ifthen}`
`\ifthenelse{\boolean{@twoside}}{Doppel}{Ein}seitiger Druck.`

Bsp.
A-3-8

```
\lengthtest{test}
```

Der Befehl `\lengthtest` dient zum Vergleichen von Längen. In seinem Argument *test* kann er zwei Längen mithilfe eines der Operatoren `<`, `=` oder `>` miteinander vergleichen. Dabei können die Längen als explizite Werte wie `20cm` oder als mit `\newlength` definierte Namen angegeben werden.

Als Beispiel dient hier eine Abbildung mit den Maßen `\Xsize` und `\Ysize`. Diese Abbildung soll in einen rechteckigen Bereich mit den Abmessungen `\Xarea` und `\Yarea` eingepasst werden, ohne ihr Seitenverhältnis zu

¹Im L^AT_EX-Kernel und in den meisten Paketen werden solche Schalter mithilfe des T_EX-Basisbefehls `\newif` definiert.

verändern. Der folgende Code errechnet die neuen Seitenlängen der Abbildung (`\newX` und `\newY`). Der Trick liegt darin, zunächst die Seitenverhältnisse von Rechteck und Abbildung zu berechnen und dann anhand der Ergebnisse den Skalierungsfaktor zu bestimmen.

```
\newlength{\sizetmp} \newlength{\areatmp}
\setlength{\sizetmp}{1pt*\ratio{\Xsize}{\Ysize}}
\setlength{\areatmp}{1pt*\ratio{\Xarea}{\Yarea}}
\ifthenelse{\lengthtest{\sizetmp > \areatmp}}
  {\setlength{\newX}{\Xarea}%
   \setlength{\newY}{\newX*\ratio{\Ysize}{\Xsize}}}
  {\setlength{\newY}{\Yarea}%
   \setlength{\newX}{\newY*\ratio{\Xsize}{\Ysize}}}
```

`\isodd{zahl}`

Mithilfe des Befehls `\isodd` kann man überprüfen, ob es sich bei einer gegebenen *zahl* um eine gerade Zahl handelt. Erzeugt beispielsweise ein `\pageref`-Befehl eine gültige Zahl (was normalerweise der Fall ist), so kann man den Befehl folgendermaßen verwenden:

Bsp.
A-3-9

Dies ist eine
gerade Seite. Dies

6

ist eine spezielle
Seite.

7

```
\usepackage{ifthen} \newcounter{pl}
\newcommand\pcheck[2]{\stepcounter{pl}%
 \label{pl-\thep1}%
 \ifthenelse{\isodd{\pageref{pl-\thep1}}}%
  {#1}{#2}}
Dies ist eine \pcheck{ungerade}{gerade} Seite.
Dies ist eine \pcheck{spezielle}{dumme} Seite.
```

Der Befehl `\isodd` ist speziell auf die vorige Anwendung zugeschnitten, auch wenn es sein kann, dass das Ergebnis von `\pageref` im ersten $\text{\LaTeX}$ -Lauf noch nicht definiert ist. Man beachte, dass man anstelle von `\label` und `\pageref` nicht einfach `\thepage` verwenden kann, da die Seiten asynchron gebildet werden. Entsprechend kann es sein, dass der Code bereits ausgewertet wird, während die Seite noch nicht fertig ist. Später könnte die Ausgabe-routine von $\text{\LaTeX}$ diese Passage immer noch auf die nächste Seite verschieben, so dass das Ergebnis von `\thepage` nicht mehr gültig wäre.

`\whiledo{test}{ausführen}`

Der `\whiledo`-Befehl ist nützlich, um bestimmte sich wiederholende Befehlsfolgen auszuführen. Das folgende einfache Beispiel zeigt, wie der Befehl eingesetzt werden kann:

Bsp.
A-3-10

Ich soll im Unterricht nicht reden (1).
Ich soll im Unterricht nicht reden (2). Ich
soll im Unterricht nicht reden (3). Ich soll
im Unterricht nicht reden (4). Ich soll im
Unterricht nicht reden (5).

```
\usepackage{ifthen} \newcounter{howoften}
\setcounter{howoften}{1}
\whiledo{\value{howoften}<6}{Ich soll im
 Unterricht nicht reden (\thehowoften).
 \stepcounter{howoften}}
```


<code>\and</code>	<code>\or</code>	<code>\not</code>	<code>\(</code>	<code>\)</code>

Mehrfachbedingungen können durch die logischen Operatoren `\or`, `\and` und `\not` zu logischen Ausdrücken kombiniert werden, wobei die Befehle `\(` und `\)` als Klammern dienen. Hier ein einfaches Beispiel dazu:

```
\usepackage{ifthen}
\usepackage[ngerman]{babel}
\mdqon
\newcommand{\QU}[2]{%
  \ifthenelse{\(\equal{#1}{GER}\and \equal{#2}{ja}\)\or
    \(\equal{#1}{FRE}\and \equal{#2}{oui}\)}%
    {"OK"}{"nicht OK"}}
```

```
\mdqoff
```

```
Einverstanden \QU{GER}{ja} oder nicht \QU{GER}{nein}.
```

```
D'accord \QU{FRE}{oui} ou pas \QU{FRE}{non}?
```

Einverstanden „OK“ oder nicht „nicht OK“.

D'accord „OK“ ou pas „nicht OK“?

Bsp.
A-3-11

Um die deutschen Anführungszeichen zu erhalten, müssen "..." als babel-Kurzformen erkannt werden. In der Präambel sind derartige Kurzformen aber normalerweise deaktiviert, weshalb sie hier (für die Definition von `\QU`) mittels `\mdqon` explizit aktiviert wurden.

A.4 Aufbau von Paket- und Klassendateien

In diesem Abschnitt wird besprochen, welche Befehle Entwicklern von Paket- und Klassendateien zur Verfügung stehen. Auch wenn man keine eigenen Pakete entwickeln möchte, hilft dieser Abschnitt, den Aufbau und Inhalt von Klassen- und Paketdateien wie `book` oder `varioref` besser zu verstehen und so größeren Nutzen aus ihnen zu ziehen.

Generell sind Klassen- und Paketdateien gleich aufgebaut und bestehen aus folgenden Bereichen:

- ⟨Kenndaten⟩*
- ⟨Initialisierung⟩*
- ⟨Deklarieren von Optionen⟩*
- ⟨Ausführen von Optionen⟩*
- ⟨Laden von Paketen⟩*
- ⟨Hauptteil⟩*

Alle diese Bereiche sind optional. Die in den einzelnen Bereichen jeweils verfügbaren Befehle werden im Folgenden besprochen. Tabelle A.6 auf der gegenüberliegenden Seite bietet einen kurzen Überblick.

A.4.1 Die Kenndaten

In diesem Teil der Klassen- oder Paketdatei wird die Art der Datei definiert. Hier kann man zudem angeben, welche $\text{\LaTeX}$ -Distribution mindestens erforderlich ist, um die Datei korrekt zu verarbeiten.

<i>Kenndaten</i>	
<code>\NeedsTeXFormat{format}[datum]</code>	Benötigt das Format <i>format</i> (LaTeX2e) mit einem Versionsdatum, das nicht älter als <i>datum</i> ist.
<code>\ProvidesClass{name}[versionsdaten]</code> <code>\ProvidesPackage{name}[versionsdaten]</code>	Kennzeichnet die Klasse oder das Paket <i>name</i> und nennt ihre <i>versionsdaten</i> .
<code>\ProvidesFile{name}[versionsdaten]</code>	Kennzeichnet eine andere Datei <i>name</i> und nennt ihre <i>versionsdaten</i> .
<i>Deklarieren von Optionen</i>	
<code>\DeclareOption{option}{code}</code>	Deklariert den <i>code</i> , der für <i>option</i> ausgeführt werden soll.
<code>\PassOptionsToPackage{optionsliste}{paketname}</code>	Übergibt die <i>optionsliste</i> an das Paket <i>paketname</i> .
<code>\DeclareOption*{code}</code>	Deklariert den <i>code</i> , der für unbekannte Optionen ausgeführt werden soll.
<code>\CurrentOption</code>	Verweist auf die aktuelle Option zur Verwendung in <code>\DeclareOption*</code> .
<i>Ausführen von Optionen</i>	
<code>\ExecuteOptions{optionsliste}</code>	Führt Code für jede Option in der <i>optionsliste</i> aus.
<code>\ProcessOptions</code> <code>\ProcessOptions*</code>	Verarbeitet die angegebenen Optionen für die aktuelle Klasse oder das Paket; die Sternform hält die angegebene Reihenfolge ein.
<i>Laden von Paketen</i>	
<code>\RequirePackage[optionsliste]{paket}[datum]</code>	Lädt ein <i>paket</i> mit der angegebenen <i>optionsliste</i> und einem Versionsdatum, das nicht älter als <i>datum</i> ist.
<i>Sonderbefehle für Paket- und Klassendateien</i>	
<code>\AtEndOfPackage{code}</code> <code>\AtEndOfClass{code}</code>	Stellt die Ausführung von <i>code</i> bis zum Ende des aktuellen Paketes (oder der Klasse) zurück.
<code>\AtBeginDocument{code}</code> <code>\AtEndDocument{code}</code>	Führt <i>code</i> bei <code>\begin{document}</code> oder <code>\end{document}</code> aus.
<code>\IfFileExists{datei}{dann}{sonst}</code>	Führt den Code in <i>dann</i> aus, wenn die genannte <i>datei</i> existiert, ansonsten den Code in <i>sonst</i> .
<code>\InputIfFileExists{datei}{dann}{sonst}</code>	Wenn die angegebene <i>datei</i> existiert, wird <i>dann</i> ausgeführt und die Datei eingefügt; ansonsten wird der Code in <i>sonst</i> ausgeführt.
<i>Sonderbefehle für Klassendateien</i>	
<code>\LoadClass[optionsliste]{klasse}[datum]</code>	Wie <code>\RequirePackage</code> für Klassendateien; sieht jedoch keine globalen Optionen, die ihm nicht explizit übergeben wurden.
<code>\PassOptionsToClass{optionsliste}{klasse}</code>	Übergibt die <i>optionsliste</i> an die genannte <i>klasse</i> .
<code>\OptionNotUsed</code>	Zur Verwendung in <code>\DeclareOption*</code> soweit erforderlich.

Tabelle A.6: Befehle für Paket- und Klassendateien

`\ProvidesClass{name}[versionsdaten]`

Eine Klassendatei gibt sich mit dem Befehl `\ProvidesClass` zu erkennen. Das Argument *name* entspricht dem Namen, wie er im obligatorischen Argument des `\documentclass`-Befehls verwendet wird (also der Dateiname ohne Erweiterung). Das optionale Argument *versionsdaten* sollte, sofern vorhanden, mit einem Datum in der Form JJJJ/MM/TT beginnen, gefolgt von einem Leerzeichen und der Versionsnummer oder Kennzeichnung. Darauf kann noch weiterer Text folgen, der die Klasse näher beschreibt. Die Klasse `report` beispielsweise enthält in etwa folgenden Text:

```
\ProvidesClass{report}[2004/02/16 v1.4f
                        Standard LaTeX document class]
```

In einem Dokument kann man die *versionsdaten* nutzen, indem man dem `\documentclass`-Befehl das Datum folgendermaßen als zweites optionales Argument mitgibt:

```
\documentclass[twocolumn]{report}[2004/02/16]
```

Dadurch kann L^AT_EX die Klasse `report` darauf prüfen, ob sie am 16.2.2004 herausgegeben wurde bzw. noch neuer ist. Ist die Klassendatei älter, so wird eine Warnung ausgegeben. Verwendet man also eine neue Version einer Klassendatei und versendet sein Dokument an eine andere Stelle, dann erfahren die Anwender dort direkt, ob ihre L^AT_EX-Distribution veraltet ist.

`\ProvidesPackage{name}[versionsdaten]`

Dieser Befehl kennzeichnet eine Paketdatei. Er ist genauso aufgebaut wie der Befehl `\ProvidesClass`. Das in den *versionsdaten* genannte Datum kann auch hier als zweites optionales Argument für den `\usepackage`-Befehl verwendet werden, um sicherzugehen, dass eine aktuelle Version der Paketdatei geladen wird wie z.B.:

```
\usepackage[ngerman]{varioref}[2005/06/09]
```

`\ProvidesFile{dateiname}[versionsdaten]`

Dieser Befehl kennzeichnet alle anderen Arten von Dateien. Daher muss das Argument *dateiname* den vollständigen Namen inklusive der Erweiterung enthalten.

`\NeedsTeXFormat{format}[datum]`

Außer den zuvor genannten Befehlen enthält der Abschnitt *<Kenndaten>* normalerweise zudem eine `\NeedsTeXFormat`-Deklaration. Als Argument *format* muss hier L^AT_EX2_ε angegeben werden. Sofern das optionale Argument *datum* verwendet wird, sollte es das Erscheinungsdatum der mindestens benötigten L^AT_EX-Distribution im Format JJJJ/MM/TT enthalten, wie z.B.:

```
\NeedsTeXFormat{LaTeX2e}[2001/06/01]
```

Hier wird eine $\text{\LaTeX}$ 2 ϵ -Distribution benötigt, die am 1. Juni 2001 oder später veröffentlicht wurde. Durch diesen Befehl erhält jeder, der versucht den Programmcode mit einer älteren $\text{\LaTeX}$ -Version zu verwenden, eine Warnung, dass Fehler auftreten können. Eine neuere Version wird ohne Warnung akzeptiert, da davon ausgegangen wird, dass die Versionen aufwärtskompatibel sind.

Alle vier Deklarationen sind optional. Ihre Verwendung in veröffentlichten Klassen- und Paketdateien erleichtert jedoch die Pflege dieser Dateien.

A.4.2 Der Initialisierungsteil

Im Abschnitt *(Initialisierung)* kann man jeden gültigen $\text{\LaTeX}$ -Code verwenden. Dies schließt auch das Laden von Paketen mithilfe des `\RequirePackage`-Befehls ein (siehe Abschnitt A.4.5), falls der in ihnen enthaltene Programmcode in einer der Optionsdeklarationen benötigt wird. So kann man z.B. an dieser Stelle das `calc`-Paket laden, wenn man später damit arbeiten will. Normalerweise ist dieser Teil jedoch leer.

A.4.3 Deklarieren von Optionen

In diesem Teil werden alle dem Paket oder der Klasse bekannten Optionen mithilfe des `\DeclareOption`-Befehls deklariert. Hier dürfen keine Pakete geladen werden.

```
\DeclareOption{option}{code}
```

Das Argument *option* enthält den Namen der zu deklarierenden Option, und der Programmcode *code* wird beim Anfordern dieser Option ausgeführt. So ist zum Beispiel die Papierformat-Option `a4paper` normalerweise folgendermaßen definiert:

```
\DeclareOption{a4paper}{\setlength\paperheight{297mm}%
                        \setlength\paperwidth{210mm}}
```

Im Prinzip kann das *code*-Argument des Befehls `\DeclareOption` jede beliebige Anweisung enthalten – vom Setzen eines Flags bis hin zu komplexen Befehlsfolgen.

Eine wichtige Funktion, die in `\DeclareOption` verwendet werden sollte, ist `\PassOptionsToPackage`. Sie kann ein oder mehrere Optionen an ein anderes Paket übergeben, das erst später geladen wird.

```
\PassOptionsToPackage{optionsliste}{paketname}
```

Das Argument *optionsliste* enthält eine durch Kommas getrennte Liste von Optionen, die dem Paket *paketname* übergeben werden soll, wenn es im Abschnitt *(Laden von Paketen)* geladen wird¹. Angenommen, man möchte eine Klassendatei definieren, die zwei Pakete **A** und **B** verwendet, welche beide die

¹Der Paketentwickler muss selber darauf achten, diese Pakete dann auch tatsächlich zu laden. $\text{\LaTeX}$ überprüft nicht, ob die Pakete, die über `\PassOptionsToPackage` Optionen erhalten sollen, später auch wirklich geladen werden.

Option `infoshow` unterstützen. Mit folgender Deklaration könnte man diese Option auch in der Klassendatei unterstützen:

```
\DeclareOption{infoshow}{%
  \PassOptionsToPackage{infoshow}{A}%
  \PassOptionsToPackage{infoshow}{B}%
  {Code zum Unterstützen von infoshow in der Klasse}}
```

Wird eine Paketdatei mit einer ihr unbekannten Option geladen, dann gibt sie eine Warnung aus. Eine Klassendatei ignoriert die Option in einem solchen Fall stillschweigend. Sie geht davon aus, dass es sich um eine globale Option handelt, die an andere Pakete übergeben wird, welche mit `\usepackage` erst noch geladen werden. Dieses Verhalten ist jedoch nicht festgeschrieben und kann mithilfe der Deklaration `\DeclareOption*` geändert werden.

`\DeclareOption*{code}`

Befehl wirkt sich
nicht auf globale
Optionen aus!

Das Argument *code* gibt an, welche Aktion ausgeführt werden soll, wenn der `\usepackage`- oder der `\RequirePackage`-Befehl eine unbekannte Option übergibt. Innerhalb dieses Argumentes verweist `\CurrentOption` auf die fragliche Option. Um beispielsweise ein Paket zu schreiben, das den Funktionsumfang eines anderen Paketes erweitern soll, könnte man folgende Deklaration verwenden:

```
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{A}}
```

Dadurch werden alle vom aktuellen Paket nicht deklarierten Optionen an Paket A übergeben. Sofern keine `\DeclareOption*`-Deklaration vorhanden ist, wird, wie bereits beschrieben, die voreingestellte Handlung ausgeführt.

Kombiniert man `\DeclareOption*` und `\InputIfFileExists`, so kann die Handhabung von Optionen sogar mit Bedingungen verknüpft werden. Der folgende Code etwa sucht nach Dateien, deren Namen mit dem Optionsnamen gebildet wurden:

```
\DeclareOption*{\InputIfFileExists{g-\CurrentOption.xyz}{}%
  {\OptionNotUsed
   \PackageWarning{irgendeiname}%
    {Option \CurrentOption\space unbekannt}}}
```

Wird die Datei `g-option.xyz` gefunden, dann wird sie auch geladen; ansonsten wird die Option mit einer Warnung ignoriert.

A.4.4 Ausführen von Optionen

Nachdem alle Optionen deklariert sind, werden normalerweise zwei Arten von Aktionen durchgeführt. Man kann sofern gewünscht einige Voreinstellungen festlegen, wie etwa ein Standardpapierformat. Danach müssen alle übergebenen Optionen geprüft und der zugehörige Programmcode ausgeführt werden.

`\ExecuteOptions{optionsliste}`

Der Befehl `\ExecuteOptions` führt den Programmcode aller Optionen der Liste *optionsliste* aus, und zwar in der dort angegebenen Reihenfolge. Durch Ausführen des Programmcodes, der zuvor mit einem `\DeclareOption`-Befehl definiert wurde, kann man so auf einfache Weise Voreinstellungen festlegen. Die Standardklasse `book` beispielsweise führt folgende Anweisung aus:

```
\ExecuteOptions{letterpaper,twoside,10pt}
```

Man kann `\ExecuteOptions` auch verwenden, wenn man andere Optionen deklariert, wie etwa beim Definieren einer Option, die automatisch andere nach sich zieht. Der Befehl `\ExecuteOptions` darf nur vor dem Befehl `\ProcessOptions` verwendet werden, da `\ProcessOptions` als eine seiner letzten Aktionen den gesamten Speicher zurückfordert, den der Code für die deklarierten Optionen eingenommen hat.

`\ProcessOptions`

Beim Erreichen des `\ProcessOptions`-Befehls prüft dieser die Liste der Optionen für die gegebene Klasse oder das Paket und führt den entsprechenden Programmcode aus. Genauer gesagt werden im Falle eines Paketes, die globalen Optionen (wie im `\documentclass`-Befehl festgelegt) und die direkt angegebenen Optionen (das optionale Argument des `\usepackage`- oder `\RequirePackage`-Befehls) geprüft. Dabei wird für jede durch das Paket deklarierte Option der entsprechende Programmcode ausgeführt. Dies geschieht entsprechend der Reihenfolge der Optionen in der `\DeclareOption`-Deklaration des Paketes und nicht anhand derjenigen im `\usepackage`-Befehl. Nicht bekannte globale Optionen werden ignoriert. Bei allen anderen unbekannten Optionen wird der mit dem Befehl `\DeclareOption*` festgelegte Programmcode ausgeführt; fehlt dieser, so erfolgt eine Fehlermeldung.

Daher reagieren Pakete, die zum Deklarieren von Optionen lediglich eine `\DeclareOption*`-Deklaration verwenden, nicht auf globale Optionen des `\documentclass`-Befehls, sondern akzeptieren nur diejenigen, die explizit mit `\usepackage` oder `\RequirePackage` deklariert wurden.

Bei Klassendateien verhält sich `\ProcessOptions` genauso, jedoch ohne die Sonderbehandlung globaler Optionen.

Bei der Verwendung von `\ProcessOptions` gibt es ein potentielles Problem: Der Befehl sucht nach einem auf ihn folgenden Stern (selbst in der nachfolgenden Zeile), was fälschlicherweise dazu führen kann, dass er nach ihm stehende Befehle expandiert. Dieses Risiko lässt sich vermeiden, indem man den Befehl `\relax` an sein Ende setzt, um die Suche sofort zu beenden und die Ausführung der Optionen einzuleiten.

*Unerwünschtes
Ausführen von
Befehlen verhindern*

`\ProcessOptions*`

Bei manchen Paketen kann es besser sein, wenn sie ihre Optionen in der Reihenfolge verarbeiten, in der sie im `\usepackage`-Befehl auftreten und nicht in derjenigen der `\DeclareOption`-Befehle. So soll beispielsweise im

babel-Paket die zuletzt aufgeführte Sprachoption die Hauptsprache des Dokumentes festlegen. Ein solches Paket kann seine Optionen mithilfe von `\ProcessOptions*` anstatt `\ProcessOptions` in der angegebenen Reihenfolge ausführen.

A.4.5 Laden von Paketen

Sobald die Optionen abgearbeitet sind, ist es möglicherweise an der Zeit ein oder mehrere zusätzliche Pakete zu laden, wie etwa diejenigen, an die man mithilfe des Befehls `\PassOptionsToPackage` Optionen übergeben hat.

`\RequirePackage[optionsliste]{paket}[datum]`

Dieser Befehl ist die Paket- bzw. Klassenentsprechung für den Dokumentenbefehl `\usepackage`. Sofern das genannte *paket* noch nicht geladen wurde, geschieht dies nun, und zwar mit den in der *optionsliste* genannten Optionen, den globalen Optionen des Befehls `\documentclass` und allen über `\PassOptionsToPackage` an dieses Paket übergebenen Optionen.

L^AT_EX lädt jedes Paket nur einmal, da es oftmals riskant ist, den Programmcode eines Paketes mehrfach auszuführen. Benötigt man nun ein Paket mit einer bestimmten Reihe von Optionen und dieses Paket wurde zuvor bereits mit anderen, aber nicht allen aktuell benötigten Optionen geladen, so steht der Anwender der Paketdatei vor einem Problem. In diesem Fall gibt L^AT_EX eine Fehlermeldung aus, die den Anwender der Paketdatei über den Konflikt informiert und vorschlägt, das Paket über den `\usepackage`-Befehl mit allen erforderlichen Optionen zu laden.

Mit dem optionalen Argument *datum* kann man festlegen, dass eine Paketversion nicht älter sein darf, als ein bestimmtes Datum. Damit dieses Verfahren funktioniert, muss das benötigte Paket selbst über eine Datumsangabe in seiner `\ProvidesPackage`-Deklaration verfügen.

`\RequirePackageWithOptions{paket}[datum]`

Dieser Befehl verhält sich genau wie `\RequirePackage` bis auf die Tatsache, dass ihm genau die gleichen Optionen übergeben werden, wie dem aufrufenden Paket bzw. der Klasse. Dadurch lässt sich das Erzeugen von Paketvarianten vereinfachen, welche die gleichen Optionen benötigen wie das Original (vgl. auch die Erläuterungen zu `\LoadClassWithOptions` auf Seite 920).

A.4.6 Der Hauptteil

Dieser letzte Teil der Datei definiert die Eigenschaften und implementiert die Funktionen, die von der angegebenen Dokumentenklasse oder dem entsprechenden Paket zur Verfügung gestellt werden. Er kann jeden gültigen L^AT_EX-Befehl enthalten und definiert in der Regel neue Befehle und Strukturen. Es empfiehlt sich, L^AT_EX-Standardbefehle zu verwenden, wie etwa `\newlength`, `\newcommand`, `\CheckCommand` usw., und sich nicht auf T_EX-Basisbefehle zu verlassen, da diese nicht überprüfen, ob Konflikte mit anderen Paketen auftreten können.

A.4.7 Sonderbefehle für Paket- und Klassendateien

```
\AtEndOfPackage{code}   \AtEndOfClass{code}
```

Manchmal muss man die Ausführung von Programmcode bis an das Ende des aktuellen Paketes oder der aktuellen Klassendatei zurückstellen. Die hier aufgeführten Deklarationen speichern ihr Argument *code* und führen es erst aus, wenn das Ende des Paketes bzw. der Klasse erreicht ist. Enthält eine Datei mehrere Deklarationen dieser Art, dann wird der *code* gesammelt und schließlich in der Reihenfolge der Deklarationen ausgeführt.

```
\AtBeginDocument{code}   \AtEndDocument{code}
```

Andere wichtige Stellen, an denen man möglicherweise aufgeschobenen Programmcode ausführen möchte, sind der Anfang und das Ende des Dokumentes, oder genauer gesagt die Stellen, an denen `\begin{document}` und `\end{document}` verarbeitet werden. Mithilfe der obigen Befehle können Pakete Programmcode in diese Umgebung einfügen, ohne dabei in Konflikt mit anderen Paketen zu geraten, die dasselbe versuchen.

Man beachte jedoch, dass jeglicher Code im `\AtBeginDocument`-Befehl noch zur Präambel gehört. Dadurch ist dieser Bereich bestimmten Einschränkungen unterworfen; so kann dort insbesondere kein Material gesetzt werden.

```
\IfFileExists{datei}{dann}{sonst}  
\InputIfFileExists{datei}{dann}{sonst}
```

Versucht ein Paket oder eine Klasse eine nicht existierende Datei mit dem Befehl `\input` einzulesen, so gerät der Anwender in eine $\TeX$ -Dateifehlerschleife. Diese kann man nur durch Eingeben eines gültigen Dateinamens verlassen. Dieses Problem kann man vermeiden, wenn man stattdessen den Befehl `\IfFileExists` benutzt. Das Argument *datei* gibt den Namen der Datei an, deren Vorhandensein man überprüfen möchte. Findet $\LaTeX$ die Datei, so werden die Befehle des *dann*-Argumentes ausgeführt, andernfalls die des *sonst*-Argumentes. Der Befehl `\InputIfFileExists` prüft nicht nur, ob die angegebene Datei vorhanden ist, sondern fügt sie auch sofort nach dem Ausführen des *dann*-Codes ein. Danach wird der in *datei* aufgeführte Name in die Dateiliste aufgenommen, die mit dem Befehl `\listfiles` angezeigt wird.

```
\PackageWarning{paket}{warnung}  
\PackageWarningNoLine{paket}{warnung}  
\PackageInfo{paket}{info}
```

Entdeckt ein Paket ein Problem, so kann es den Anwender durch eine Warnung auf dem Bildschirm darauf hinweisen. Findet beispielsweise das Paket `multicol` in einer Box die Umgebung `multicols*` (die unausbalancierte Spalten erzeugt¹), dann benutzt es `\PackageWarning` um den Anwender auf das

¹Spalten in einer Box müssen unbedingt in ihrer Länge ausbalanciert werden. Andernfalls wird alles Material in die erste Spalte platziert, da Boxen (im Gegensatz zu einer Seite) in vertikaler Richtung beliebig groß werden können.

Problem (und die ergriffenen Korrekturmaßnahmen) hinzuweisen:

```
\PackageWarning{multicol}{multicols* inside a box does
not make sense.\MessageBreak Going to balance anyway}
```

Dieser Befehl erzeugt eine Warnung, die mithilfe des Befehls `\MessageBreak` explizit auf zwei Zeilen umbrochen wird:

```
Package multicol Warning: multicols* inside a box does not make sense.
(multicol)                Going to balance anyway on input line 6.
```

Die aktuelle Zeilennummer wird automatisch angefügt. Manchmal wäre es hilfreich, auch den Namen der aktuellen Datei anzugeben – diese Information steht auf der Makroebene aber leider nicht zur Verfügung.

Je nach Art des Problems kann es wichtig sein, dem Anwender die Zeile der Quelldatei zu nennen, in welcher das Problem aufgetreten ist. In anderen Fällen wiederum ist diese Information nicht von Bedeutung, etwa wenn das Problem beim Laden des Paketes auftritt. In solchen Fällen sollte man die Befehlsvariante `\PackageWarningNoLine` verwenden: Sie erzeugt die gleiche Ausgabe wie `\PackageWarning`, fügt jedoch keinen Hinweis auf die Zeilennummer hinzu.

Sofern der Warnhinweis nicht so wichtig ist und nur in der Protokolldatei erscheinen soll, kann man den Befehl `\PackageInfo` benutzen. Hat man beispielsweise das Paket `shortvrb` geladen und führt die Deklaration `\MakeShortVerb\=` aus, dann erscheint in der Protokolldatei:

```
Package shortvrb Info: Made = a short reference for \verb on input line 3.
```

Es gibt keinen `\PackageInfoNoLine`-Befehl. Will man die Zeilennummer in einem Hinweis unbedingt unterdrücken, so kann man dies mit `\@gobble` als letztem Befehl im zweiten Argument des `\PackageInfo`-Befehls bewirken.

`\PackageError{paket}{kurztext}{langtext}`

Ist das entdeckte Problem so schwerwiegend, dass der Anwender eingreifen muss, dann kann man anstelle der Warnung eine Fehlermeldung ausgeben. Sobald der Fehler auftritt, wird sofort das Argument *kurztext* ausgegeben und die Verarbeitung angehalten. Trifft etwa das `inputenc`-Paket auf ein ihm unbekanntes 8-Bit-Zeichen, dann erzeugt es die folgende Fehlermeldung:

```
! Package inputenc Error: Keyboard character used is undefined
(inputenc)                in inputencoding 'latin1'.
```

```
See the inputenc package documentation for explanation.
Type H <return> for immediate help.
...
```

```
1.5 abc^^G
?
```

Durch Drücken der Taste „h“ oder „H“ wird das Argument *langtext* ausgegeben, in diesem Falle:

```
You need to provide a definition with \DeclareInputText
or \DeclareInputMath before using this key.
```

Auch hier kann man Zeilenumbrüche in Fehlermeldungen und Hilfetexten mit `\MessageBreak` explizit festlegen.

```
\ClassWarning{klasse}{warnung}
\ClassWarningNoLine{klasse}{warnung}
\ClassInfo{klasse}{info}
\ClassError{klasse}{kurztext}{langtext}
```

Hinweise, Warnungen und Fehlermeldungen sind nicht nur für Pakete verfügbar – für Klassen gibt es ähnliche Befehle. Sie unterscheiden sich nur in den von ihnen erzeugten Texten: Die Klassenbefehle geben anstatt „Package“ an den entsprechenden Stellen den Text „Class“ aus.

A.4.8 Sonderbefehle Klassendateien

Manchmal ist es hilfreich, aus einer vorhandenen allgemeinen Dokumentenklasse eine angepasste Klassendatei zu erzeugen. Dazu stehen zwei Befehle zur Verfügung.

```
\LoadClass[optionsliste]{klasse}[datum]
```

Der `\LoadClass`-Befehl verhält sich wie `\RequirePackage` mit den folgenden drei Ausnahmen:

- Er kann nur in Klassendateien verwendet werden.
- Jede Klasse darf höchstens einen `\LoadClass`-Befehl enthalten.
- Die im Argument angegebene *klasse* sieht keine globalen Optionen, es sein denn, diese sind im Argument *optionsliste* aufgeführt oder werden ihr explizit mit dem Befehl `\PassOptionsToClass` übergeben.

```
\PassOptionsToClass{optionsliste}{klasse}
```

Mithilfe des Befehls `\PassOptionsToClass` kann man Optionen an eine solche allgemeine Klasse übergeben. Ein Beispiel für eine derartige Erweiterung einer Klassendatei ist in Abbildung A.1 auf der nächsten Seite dargestellt. Dort wird eine Klassendatei namens `myart` definiert, die zwei zusätzliche Optionen akzeptiert: zum einen `cropmarks`, um Beschnittlinien zum Beschneiden von Seiten zu erzeugen, und zum anderen `bind`, um die Druckseiten zugunsten eines größeren Bundstegs leicht nach außen zu verschieben. Außerdem wird sie um die Umgebung `Notes` erweitert.

```

% ----- Kenndaten -----
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{myart}[1994/01/01]
% ----- Initialisierung -----
\RequirePackage{ifthen}      \newboolean{cropmarks}
% ----- Deklarieren von Optionen -----
\DeclareOption{cropmarks}{\setboolean{cropmarks}{true}}
\DeclareOption{bind}        {\AtEndOfClass{\addtolength\oddsidemargin{.5in}%
\addtolength\evensidemargin{-.5in}}}
\DeclareOption*              {\PassOptionsToClass{\CurrentOption}{article}}
% ----- Ausführen von Optionen -----
\ProcessOptions \relax      % vgl. Hinweis auf S. 915!
% ----- Laden von Paketen -----
\LoadClass{article}         % der eigentliche Code
% ----- Hauptteil -----
\newenvironment{Notes}{...}{...} % die neue Umgebung
\ifthenelse{\boolean{cropmarks}} % Unterstützung für cropmarks
  {\renewcommand{\ps@plain}{...} ...}{

```

Abbildung A.1: Eine Klassendatei, die article erweitert

Die Option `cropmarks` wird als boolescher Schalter implementiert und sie definiert verschiedene `\pagestyle`-Befehle um, wenn der Schalter auf `true` steht. Die Option `bind` verändert die Werte von `\oddsidemargin` und `\evensidemargin`. Zu dem Zeitpunkt, an dem die `bind`-Option abgearbeitet wird, haben diese Längenregister jedoch noch nicht ihren endgültigen Wert, denn dieser wird erst später gesetzt, wenn `\LoadClass` die Klasse `article` lädt. Daher wird die Änderung mithilfe des Befehls `\AtEndOfClass` bis zum Ende der Klassendatei `myart` zurückgestellt.

`\OptionNotUsed`

Ergibt sich ein komplexeres *code*-Argument für `\DeclareOption*` in einer Klassendatei (z.B. weil man einige Optionen verarbeiten, andere jedoch zurückweisen möchte), so muss man $\LaTeX$ möglicherweise mithilfe des Befehls `\OptionNotUsed` explizit darüber informieren, dass eine Option nicht akzeptiert wurde. Andernfalls geht $\LaTeX$ davon aus, dass die Option verwendet wurde und erzeugt keine Warnung, wenn sie von keinem weiteren Paket übernommen wird.

`\LoadClassWithOptions{klasse}[datum]`

Dieser Befehl gleicht dem Befehl `\LoadClass`. Er ruft die angegebene *klasse* jedoch immer mit exakt der Optionsliste auf, die auch von der aktuellen Klasse verwendet wird, und nicht mit den Optionen, die explizit oder mithilfe von `\PassOptionsToClass` übergeben wurden. Er ist hauptsächlich dazu gedacht, eine Klasse auf einer anderen aufzubauen, wie z.B.:

`\LoadClassWithOptions{article}`

Dem sollte man die folgende, etwas andere Konstruktion gegenüberstellen:

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

So wie sie hier verwendet werden, haben beide Versionen mehr oder weniger den gleichen Effekt. Die Variante mit `\LoadClassWithOptions` ist jedoch etwas schneller (und leichter einzugeben). Deklariert die Klasse jedoch eigene Optionen, dann haben die beiden Konstruktionen unterschiedliche Auswirkungen. Man vergleiche beispielsweise

```
\DeclareOption{landscape}{...}
\ProcessOptions
\LoadClassWithOptions{article}
```

mit

```
\DeclareOption{landscape}{...}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{article}}
\ProcessOptions
\LoadClass{article}
```

Im ersten Beispiel wird die Klasse `article` nur mit `landscape` aufgerufen, wenn auch die aktuelle Klasse mit dieser Option aufgerufen wird. Im zweiten Beispiel wird die `landscape`-Option nie an die Klasse `article` übergeben, da das vorgegebene Verfahren nur Optionen übergibt, die *nicht* explizit deklariert wurden.

```
\@ifpackageloaded{paket}{dann}{sonst}
\@ifpackagelater{paket}{datum}{dann}{sonst}
\@ifpackagewith{paket}{optionen}{dann}{sonst}
```

In einigen Fällen kann es hilfreich sein, Informationen darüber zu erhalten, ob ein Paket bereits zuvor geladen wurde, und wenn ja, wie. Um dies zu erfahren, stehen Klassen- und Paketentwicklern drei Befehle zur Verfügung. Mit dem Befehl `\@ifpackageloaded` findet man heraus, ob das genannte *paket* bereits geladen wurde. Ist es geladen, so wird der Code im Argument *dann* ausgeführt, andernfalls derjenige im Argument *sonst*. Der Befehl `\@ifpackagelater` gibt Aufschluss darüber, ob das genannte *paket* mit einem neueren Versionsdatum als *datum* geladen wurde. Und mithilfe des Befehls `\@ifpackagewith` erfährt man, ob ein *paket* mindestens mit den in der durch Kommas unterteilten Liste *optionen* aufgeführten Optionen geladen wurde.

Das `fontenc`-Paket lässt sich nicht mit diesen Befehlen prüfen. Es täuscht nämlich vor, dass es niemals geladen wurde, damit man es immer wieder mit anderen Optionen erneut laden kann. Nähere Einzelheiten hierzu enthält die Datei `ltoutenc.dtx` der L^AT_EX-Distribution.

A.4.9 Eine minimalistische Klassendatei

Jede Klassendatei *muss* vier Dinge enthalten: eine Definition für den Befehl `\normalsize`, sowie Werte für `\textwidth` und `\textheight` und den Stil der Seitennummerierung. Eine kleinstmögliche Dokumentenklassendatei¹ sähe also folgendermaßen aus:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{minimal}[1995/10/30 Standard LaTeX minimal class]
\renewcommand\normalsize{\fontsize{10pt}{12pt}\selectfont}
\setlength\textwidth{6.5in}
\setlength\textheight{8in}
\pagenumbering{arabic}      % wird benötigt, obwohl diese Klasse
                           % gar keine Seitenzahlen anzeigt
```

Diese Klassendatei unterstützt allerdings keine Fußnoten, Marginalien, Gleitobjekte oder sonstige Funktionen. Die meisten Klassen enthalten dementsprechend natürlich mehr als dieses Minimum!

¹Diese Klasse ist als `minimal.cls` in der Standarddistribution enthalten.

Anhang B

Finden und Lösen von Problemen

B.1 Fehlermeldungen	924
B.2 Warnungen und informelle Meldungen	961
B.3 $\TeX$ und $\LaTeX$ -Tracingbefehle.	977

In einer idealen Welt lassen sich alle Dokumente, die man erzeugt, problemlos kompilieren und sie werden genau in der gewünschten guten Qualität ausgegeben. Wer sich so glücklich schätzt, braucht diesen Anhang niemals zu Rate ziehen. Wer hingegen doch auf die ein oder andere Schwierigkeit stößt, sollte hier Hilfe finden, um sein Problem auf einfache Weise zu lösen.

Dieser Anhang beginnt mit einer alphabetischen Liste aller Fehlermeldungen, also der Meldungen, bei denen $\LaTeX$ stoppt und um Rat fragt. „Alle“ bedeutet in diesem Falle alle $\LaTeX$ -Kernelfehler (sie beginnen mit `LaTeX Error:`), so gut wie alle $\TeX$ -Fehler (die also direkt vom darunter liegenden Satzsystem ausgegeben werden) sowie die Fehlermeldungen der folgenden Pakete: `amsmath`, `babel`, `docstrip`, `calc`, `color`, `graphics`, `graphicx`, `inputenc`, `fontenc` und `textcomp`. Fehlermeldungen anderer Pakete, die sich mit

! Package *<Paket>* Error: *<Fehlertext>*

identifizieren, gehören nicht dazu, sofern das *<Paket>* nicht in der oben aufgeführten Liste steht. Bei solchen Fehlermeldungen sollte man unter der Beschreibung des betreffenden Paketes in diesem Buch oder in seiner Originaldokumentation nachschlagen.

Doch auch wenn kein wirklicher Fehler vorliegt, der die Verarbeitung stoppt, können Warnungen und Hinweise auf dem Bildschirm oder in der Protokolldatei erscheinen. Diese werden in Abschnitt B.2 behandelt, in dem alle zentralen $\LaTeX$ - und alle wichtigen $\TeX$ -Meldungen aufgeführt sind, die gegebenenfalls besonderer Aufmerksamkeit bedürfen. Dort findet man auch eine Erklärung der möglichen Ursachen sowie Vorschläge zu ihrer Behebung.

Der letzte Abschnitt beschäftigt sich mit Werkzeugen zur Fehlerüberwachung für Situationen, in denen kein oder kein genügend aufschlussreicher Warnhinweis existiert. Dort wird zunächst erläutert, wie man Befehlsdefinitionen und Registerwerte anzeigen kann, und danach, wie man Probleme beim Seitenumbruch untersucht und löst. Darauf folgen Vorschläge, wie sich Schwierigkeiten beim Umbrechen von Absätzen erkennen und lösen lassen. Der Abschnitt schließt mit einer Beschreibung des Paketes `trace`, das hilfreich ist, um die Ausführung von Befehlen zu überwachen, sofern eigene Definitionen oder solche anderer Entwickler unerwartete Auswirkungen zeigen.

Ein Teil der Informationen in diesem Anhang bezieht sich eher auf Low-Level- $\text{\LaTeX}$, und wurde somit nach Wissen der Autoren bisher noch nie in einem „ $\text{\LaTeX}$ “-Buch beschrieben. Diese Informationen sind jedoch trotzdem wichtig. Es ist nicht wirklich hilfreich, Leser auf andere Bücher wie etwa *The $\text{\TeX}$ book* zu verweisen, da die meisten dort gegebenen Ratschläge zu Plain- $\text{\TeX}$ sich nicht auf $\text{\LaTeX}$ anwenden lassen oder subtile Fehler erzeugen, wenn man sie anwendet. Darum soll dieses Buch selbst so umfassend wie möglich über die $\text{\TeX}$ -Engine informieren, die $\text{\LaTeX}$ zugrunde liegt, soweit dies im $\text{\LaTeX}$ -Kontext als sinnvoll erscheint.

B.1 Fehlermeldungen

Wenn $\text{\LaTeX}$ mit einer Fehlermeldung stoppt, wird neben der Fehlerbeschreibung die Nummer der Zeile ausgegeben, bis zu der das Quelldokument verarbeitet wurde. Aus Gründen der Speicherverwaltung, die mit dem Aufbau von $\text{\TeX}$ selbst zusammenhängen, erfolgt jedoch keine Angabe darüber, zu welcher Quelldatei diese Zeilennummer gehört. Bei einfachen Dokumenten ist das kein Problem. Verarbeitet man jedoch Dokumente, die auf mehrere Dateien verteilt sind, dann muss man die Bildschirmausgabe oder die Protokolldatei aufmerksam studieren, um herauszufinden, welche Datei $\text{\LaTeX}$ gerade verarbeitete, als der Fehler auftrat.

Immer wenn $\text{\LaTeX}$ beginnt, eine neue Datei zu lesen, wird eine öffnende runde Klammer angezeigt, der direkt der Name der Datei folgt. Sobald $\text{\LaTeX}$ die Datei zu Ende gelesen hat, wird eine schließende runde Klammer ausgegeben. Außerdem zeigt $\text{\LaTeX}$ eine öffnende eckige Klammer gefolgt von der aktuellen Seitenzahl an, sobald es beginnt, eine Seite für die Ausgabe vorzubereiten. Trifft man also auf eine Meldung wie die folgende:

```
(./trial.tex [1] (./ch-1.tex [2] [3] (./table-1.tex [4] [5])) [6]
! Undefined control sequence.
<argument> Ein \textss
               {Test}
1.235 \section{Ein \textss{Test}}
                                   \label{sec:test}
?
```

so kann man darauf schließen, dass der Fehler im Argument irgendeines Befehls (`<argument>`) erfolgte und entdeckt wurde, als $\text{\LaTeX}$ das Material für Seite 7 zusammenstellte. Das Programm konnte noch den Großteil von Zeile 235

*Die Quellcodezeile für
einen Fehler finden*

der Datei `ch-1.tex` lesen. In diesem Beispiel ist der Fehler in der Quellcodezeile deutlich zu erkennen: Der Befehl `\textsf` im Argument des `\section`-Befehls wurde fälschlicherweise `\textss` geschrieben. In einigen Fällen gibt es jedoch nur einen unklaren oder gar keinen Zusammenhang zwischen Fehler und Quellcodezeile.

Definiert man beispielsweise `\renewcommand\thepart{\Alp{part}}`, so tritt der Schreibfehler erst dann zutage, wenn man den `\part`-Befehl verwendet, der intern diese Definition ausführt. In diesem Fall erhält man dann folgende Meldung:

```
! Undefined control sequence.
\thepart ->\Alp
      {part}
1.167 \part{Test}
```

Hier befindet sich der eigentliche Fehler nicht in Zeile 167, ja wahrscheinlich nicht einmal in der aktuellen Datei: Der Befehl `\part` ruft einfach nur die fehlerhafte Definition des Befehls `\thepart` auf.

Manchmal stößt $\text{\LaTeX}$ beim Erzeugen einer neuen Seite auf einen Fehler. Da es sich hierbei um einen asynchronen Vorgang handelt, gibt die in der Fehlermeldung genannte Quellcodezeile in diesem Falle keinerlei Aufschluss über die Fehlerursache. Wenn man sich hier fragt, was denn der Fehler mit der Quellcodezeile zu tun haben soll, tut man dies zu Recht: Sie haben tatsächlich nichts miteinander zu tun. Dazu folgendes Beispiel:

```
! Undefined control sequence.
\thepage ->\romen
      {page}
1.33 D
      ieser Beispieltext soll die Seite füllen.
```

Benötigt man zusätzliche Informationen über einen Fehler oder darüber, wie $\text{\LaTeX}$ weiter verfahren möchte, so kann man auf das `?` nach der Fehlermeldung ein `\h` eingeben. Bei einem $\text{\TeX}$ -Fehler wie dem zuvor gezeigten erhält man die Antwort:

```
? h
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., '\hobx'), type 'I' and the correct
spelling (e.g., 'I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.

(Der Befehl am Ende der oberen Zeile der Fehlermeldung wurde nie
\def'iniert. Falls ein Tippfehler vorliegt (z.B. „\hobx“), geben Sie 'I' und die
korrekte Schreibweise ein (z.B. „I\hbox“). Ansonsten fahren Sie einfach fort
und ich ignoriere den nicht definierten Code.)
```

Hier kann man schon sehen, welche Tücken ein Ratschlag birgt, der direkt von der $\text{\TeX}$ -Engine kommt: Man muss ihn womöglich erst einmal übersetzen, da er sich häufig auf Befehle bezieht, die nicht unbedingt etwas mit

L^AT_EX-Dokumenten zu tun haben. So müsste man beispielsweise das `\def` im L^AT_EX-Kontext als `\newcommand` oder `\renewcommand` lesen. Bei echten L^AT_EX-Fehlern ist dies nicht der Fall, obwohl auch hier manchmal nicht sonderlich hilfreiche Ratschläge erfolgen:

You're in trouble here. Try typing `<return>` to proceed.
If that doesn't work, type `X <return>` to quit.

(Hier gibt es ein Problem. Versuchen Sie mit `<return>` fortzufahren. Wenn das nicht funktioniert, geben Sie `X <return>` ein um abzubrechen.)

Vielen Dank, das hätte man auch selbst gewusst! Trotzdem lohnt es sich immer, die Informationen abzufragen, da viele Meldungen doch detailliertere Hinweise enthalten.

*Den Stack teilweise
expandierter Makros
anzeigen*

Um zusätzliche Informationen zu einem aufgetretenen Fehler zu erhalten, kann man auch den Zähler `errorcontextlines` auf einen höheren positiven Wert setzen. In diesem Falle listet L^AT_EX den Stack (Stapelspeicher) der aktuell ausgeführten Makros auf:

```

1  ! Undefined control sequence.
2  \thepage ->\romen
3      {page}
4  \@oddfont ->\reset@font \hfil \thepage
5      \hfil
6  \@outputpage ...lor \hb@xt@ \textwidth {\@thefoot
7      }\color@endbox }}\globa...
8
9  \@opcol ...lumn \@outputdblcol \else \@outputpage
10      \fi \global \@mparbotto...
11 <output> ...specialoutput \else \@makecol \@opcol
12      \@startcolumn \@whilesw...
13 <to be read again>
14      D
15 1.33 D
16   ieser Beispieltext soll die Seite füllen.
```

Diese Meldung ist von unten nach oben zu lesen: L^AT_EX hat das D gesehen (Zeile 15 und 16), vermerkt jedoch, dass diese Stelle noch einmal gelesen werden muss (`<to be read again>`, Zeile 13 und 14), da mittlerweile auf die Ausgaberoutine umgeschaltet wurde. Dort gelangte das Programm bis zum Ausführen des Befehls `\@opcol` (Zeile 11 und 12), welcher wiederum den Befehl `\@outputpage` aufrief (Zeile 9 und 10), der seinerseits den Befehl `\@thefoot` ausführte (Zeile 6 und 7). Zeile 4 ist etwas seltsam, da sie sich überraschenderweise auf `\@oddfont` anstatt auf `\@thefoot` bezieht. Letzterer Befehl expandiert jedoch zu `\@oddfont`, so dass er sofort vollständig verarbeitet wird und gar nicht erst auf dem Stapel teilweise expandierter Makros landet. Innerhalb von `\@oddfont` gelangte L^AT_EX bis zum Aufruf des Befehls `\thepage`, der seinerseits zu `\romen` erweitert wurde (Zeile 2 und 3). Und diesen Befehl bemängelt das Programm dann schließlich als nicht definiert (Zeile 1).

Glücklicherweise reicht es meistens, die Fehlermeldung und die Quellcodezeile anzuzeigen. Darum ist der Zähler `errorcontextlines` in L^AT_EX auf `-1` voreingestellt, was bedeutet, dass kein weiterer Kontext angezeigt wird.

Es kann auch zu Fehlern kommen, wenn L^AT_EX eine Hilfsdatei verarbeitet, die dazu dient, Informationen von einem L^AT_EX-Lauf zum nächsten zu übertragen (z.B. .aux- oder .toc-Dateien). Die Daten in einer solchen Datei können aufgrund eines früheren L^AT_EX-Laufs fehlerhaft sein. Selbst wenn der entsprechende Fehler in der Quelldatei bereits behoben ist, können immer noch Spuren davon in solchen externen Dateien verblieben sein. Manchmal muss man sie daher löschen, bevor man L^AT_EX erneut ausführt. Oftmals verschwinden die Probleme jedoch bereits nach einem weiteren Durchgang.

 *Hartnäckige Fehler*

Eine beliebte Quelle für derart hinterhältige Fehler in L^AT_EX sind die so genannten *zerbrechlichen* Befehle, sofern diese ungeschützt in *bewegten Argumenten* verwendet werden. Technisch gesehen ist ein bewegtes Argument ein Argument, das L^AT_EX intern expandiert, ohne es direkt zu setzen (z.B. mithilfe des internen L^AT_EX-Befehls `\protected@edef1`). Als Faustregel kann man sich merken, dass ein solches Argument vor dem Setzen an eine andere Stelle bewegt wird. Dies trifft z.B. auf folgende Fälle zu: Argumente von Gliederungsbefehlen wie `\section`, die ins Inhaltsverzeichnis verschoben werden; das Argument des `\caption`-Befehls, der in das Abbildungs- oder Tabellenverzeichnis gelangt; die Argumente von `\markboth` und `\markright`.

 *Fehler aufgrund zerbrechlicher Befehle*

Eine gute, wenn auch nicht wirklich hilfreiche, Definition eines zerbrechlichen Befehls ist die Folgende: Er erzeugt einen Fehler, wenn ihm in einem bewegten Argument kein `\protect`-Befehl vorangeht. Mittlerweile sind die meisten L^AT_EX-Befehle robust, so dass ein solcher Schutz selten notwendig ist. Erhält man jedoch eine seltsame Fehlermeldung von einem Befehl, der in einem bewegten Argument steht, dann sollte man einfach versuchen, ihm ein `\protect` voranzustellen. Zentrale L^AT_EX-Befehle mit optionalen Argumenten sind üblicherweise zerbrechlich. Der Befehl `\sqrt[3]{-1}` ist jedoch robust, genau wie *alle* benutzerdefinierten Befehle mit einem optionalen Argument. Andererseits ist `\[...\]` zerbrechlich, wird jedoch robust, sobald man das `amsmath`-Paket geladen hat. Mit anderen Worten: Es gibt keine genauen Regeln dafür, zu welcher Kategorie ein Befehl gehört. Benutzerdefinierte Befehle mit obligatorischen Argumenten sind immer dann zerbrechlich, wenn ihre Definition zerbrechliche Befehle enthält. So ist beispielsweise die folgende Definition zerbrechlich:

```
\newcommand\frail{\ifthenelse{\value{section}<10 \and
                               \value{subsection}=1}%
                               {\typeout{Ja}}{\typeout{Nein}}}
```

Das liegt daran, dass das Vergleichsargument von `\ifthenelse` zerbrechlich ist. Angenommen, man verwendet den Befehl `\frail` im @-Ausdruck eines `tabular`-Befehls (nur als Beispiel, es ergibt natürlich keinen Sinn):

```
\nonstopmode      \begin{tabular}{@{\frail}l} x \end{tabular}
```

¹Es gibt das Gerücht, dass der T_EX-Basisbefehl `\edef` eben diesem Zweck dient. Seine Verwendung in eigenen Befehlen ist jedoch nicht ratsam, es sei denn, man ist sicher, dass der Basisbefehl niemals mit willkürlichen Eingaben seitens des Dokumentes in Berührung kommt. Stattdessen sollte man `\protected@edef` benutzen. Dieser Befehl verhindert nämlich, dass zerbrechliche Befehle, denen ein `\protect` vorangeht, auseinander gebrochen werden!

In diesem Falle würde man die folgenden 134 Fehler erhalten, bevor $\text{\LaTeX}$ schließlich aufgibt (die linke Spalte zeigt, wie häufig der jeweilige Fehler auftritt):

```

1 ! Argument of \@array has an extra }.
2 ! Argument of \@firstoftwo has an extra }.
1 ! Extra }, or forgotten $.
4 ! Extra }, or forgotten \endgroup.
1 ! LaTeX Error: Illegal character in array arg.
1 ! LaTeX Error: Can be used only in preamble.
51 ! Misplaced \cr.
2 ! Missing # inserted in alignment preamble.
1 ! Missing = inserted for \ifnum.
49 ! Missing \cr inserted.
2 ! Missing control sequence inserted.
2 ! Missing number, treated as zero.
1 ! Missing { inserted.
2 ! Missing } inserted.
1 ! Paragraph ended before \renewcommand was complete.
2 ! Paragraph ended before \reserved@b was complete.
1 ! Paragraph ended before \reserved@c was complete.
2 ! Undefined control sequence.
1 ! Use of \@argtabularcr doesn't match its definition.
7 ! Use of \@array doesn't match its definition.

```

Bei diesem Beispiel gelangt $\text{\TeX}$ nämlich in eine Schleife, in der es versucht, einen $\backslash\text{cr}$ -Befehl einzufügen, diese Idee aber sofort wieder verwirft, nur um danach wieder von vorne anzufangen.

Aus diesem Beispiel kann man Folgendes lernen: Immer wenn man auf einen seltsamen $\text{\TeX}$ -Fehler trifft, für den es keine einfache Erklärung (wie etwa einen falsch geschriebenen Befehlsnamen) gibt, ist die Ursache möglicherweise ein zerbrechlicher Befehl, der in einem bewegten Argument auseinander gebrochen wurde. Also sollte man einfach versuchen, den Befehl dort, wo der Fehler auftritt, mit $\backslash\text{protect}$ zu schützen. Da sich dieser Sachverhalt im Prinzip hinter jedem $\text{\TeX}$ -Fehler verbergen kann, wird er im Folgenden nicht mehr extra aufgeführt – immerhin werden hier ja noch mehr als 60 $\text{\TeX}$ -Fehlermeldungen erklärt.

Wie in Abschnitt A.1.1 besprochen, gibt es einige Einschränkungen hinsichtlich der Zeichen, die in den Verweisschlüssel-Argumenten der Befehle $\backslash\text{label}$ und $\backslash\text{bibitem}$ verwendet werden dürfen. Kurz gesagt, verhalten sich solche Schlüssel gelegentlich wie bewegte Argumente. Je nach Kombination der verwendeten Zeichen und je nachdem, welche Pakete geladen wurden, können hier alle möglichen grässlichen $\text{\TeX}$ -Fehler auftreten. In einem solchen Fall gibt es auch *keinen* Schutz durch den $\backslash\text{protect}$ -Befehl. Stattdessen muss man einen einfacheren Schlüssel benutzen, der mit den Syntaxregeln für solche Schlüssel übereinstimmt.

Alphabetisches Verzeichnis der $\text{\TeX}$ - und $\text{\LaTeX}$ -Fehler

In der folgenden Fehlerliste sind alle $\text{\TeX}$ - und Paketfehlermeldungen mit einem eingerahmten Hinweis über ihren Ursprung am Ende der Fehlermeldung gekennzeichnet. Bei nicht gekennzeichneten Fehlermeldungen handelt es

Alle $\text{\TeX}$ -Fehler
können von
zerbrechlichen
Befehlen in bewegten
Argumenten stammen!

Von Querverweis-
schlüsseln erzeugte
Fehler

sich um L^AT_EX-Meldungen, bei denen das Präfix „LaTeX Error:“ weggelassen wurde.

- * Wenn L^AT_EX stoppt und nur einen Stern anzeigt, dann wurde das Ende der Quelldatei erreicht, ohne dass eine Anweisung zum Abschließen des Verarbeitungslaufs (also `\end{document}` oder `\stop`) erfolgte. L^AT_EX wartet nun auf weitere Eingaben über die Tastatur. Zwar handelt es sich hier nicht direkt um einen Fehler, aber meistens bedeutet dieses Verhalten, dass etwas ziemlich schief gegangen ist. Sofern vorher keine Fehler aufgetreten sind und das Dokument wirklich mit `\end{document}` endet, kann es sein, dass man lediglich vergessen hat, eine `verbatim`-Umgebung zu schließen. Dadurch wurde der Rest des Dokumentes im „verbatim“-Modus verarbeitet.

Um zum Ursprung des Problems zu gelangen, sollte man `\end{foo}` eingeben. Dadurch erhält man entweder die Fehlermeldung „Environment ... ended by foo“, die anzeigt, welche Umgebung nach Ansicht von L^AT_EX noch nicht geschlossen wurde, oder es erfolgt keinerlei Reaktion. In diesem Falle weiß man, dass man sich tatsächlich noch in irgendeiner Art von „verbatim“-Kontext bewegt. In letzterem Falle sollte man versuchen, L^AT_EX durch Drücken von Strg-C oder der entsprechenden Tastenkombination für die jeweilige Installation zu unterbrechen. Danach reagiert man mit „x“ auf die „Unterbrechung“, um die Verarbeitung zu beenden. Daraufhin erhält man normalerweise durch einen Blick auf die letzte noch gesetzte Seite des Dokumentes einen Hinweis, an welcher Stelle das Problem begann.

`'<character>' invalid at this point` ^{calc}

'<zeichen>' an dieser Stelle ungültig

Das calc-Paket wurde geladen und eine der Formeln in `\setcounter`, `\setlength`, `\addtocounter` oder `\addtolength` verwendet eine Syntax, die von calc nicht unterstützt wird. Abschnitt A.3.1 enthält nähere Einzelheiten hierzu.

`<command> allowed only in math mode` ^{amsmath}

<befehl> nur im Formelmodus zulässig

Dieser Befehl oder diese Umgebung darf nur im Formelmodus verwendet werden. Man sollte sorgfältig überprüfen, welche Teile des Dokumentes fehlen.

`<name> undefined`

<name> nicht definiert

Dieser Fehler tritt auf, wenn man den Befehl `\renewcommand` für einen `<name>`n verwendet, der L^AT_EX nicht bekannt ist. Entweder wurde der `<name>` falsch geschrieben oder hier wird eigentlich der Befehl `\newcommand` benötigt.

`\< in mid line`

\< in Zeilenmitte

Der in einer `tabbing`-Umgebung definierte Befehl `\<` wurde mitten in einer Zeile verwendet. Er ist aber nur am Zeilenanfang zulässig (zum Beispiel nach einem `\\`-Befehl).

A `<Box>` was supposed to be here T_EX

Hier sollte eine <Box> sein

Dieser Fehler tritt auf, wenn man einen Boxbefehl wie `\sbox` mit einem ungültigen ersten Argument (das also nicht mit `\newsavebox` deklariert wurde) verwendet. Normalerweise erhält man zunächst die Fehlermeldung „Missing number, treated as zero“ (Fehlende Nummer, wird als null behandelt), die anzeigt, dass T_EX das Boxregister null verwendet.

Accent `<command>` not provided by font family `<name>` textcomp

Akzent(befehl) wird von Fontfamilie <name> nicht unterstützt

Das `textcomp`-Paket implementiert die TS1-Kodierung, die leider nur von sehr wenigen Fontfamilien für L^AT_EX vollständig unterstützt wird. Es wird kein Akzentzeichen gesetzt. In Abschnitt 7.5.4 ist beschrieben, wie man einen Ersatz für das entsprechende Zeichen bereitstellt.

Argument of `<command>` has an extra `}` T_EX

Argument von <befehl> hat eine überzählige }

Anstelle eines obligatorischen Befehlsargumentes wurde eine schließende geschweifte Klammer verwendet (z.B. `\mbox}`). Zerbrechliche Befehle, die in einem bewegten Argument nicht mit `\protect` geschützt sind, werden häufig so auseinander gebrochen, dass sie diesen oder einen der im Folgenden noch aufgeführten „Extra ...“-Fehler erzeugen.

Bad `\line` or `\vector` argument

Ungültiges \line- oder \vector-Argument

L^AT_EX gibt diesen Fehler aus, wenn man eine negative Länge oder eine unzulässige Neigung für einen `\line`- oder `\vector`-Befehl angegeben hat. Für letzteren Fall wurden in Kapitel 10 Alternativen aufgezeigt.

Bad math environment delimiter

Ungültiges Begrenzungszeichen für Formelumgebung

Dieser Fehler wird ausgelöst, wenn in einer Formel die Befehle `\(` oder `\[` bzw. wenn im normalen Text `\)` oder `\]` vorkommen. Hier muss man überprüfen, ob die entsprechenden Klammern im Dokument paarweise vorhanden sind.

`\begin{<env>}` allowed only in paragraph mode amsmath

`\begin{<umg>}` nur im Absatzmodus zulässig

An vielen Stellen, wie etwa im LR-, Text- oder Formelmodus ist es nicht sinnvoll, eine abgesetzte Formel zu verwenden. Die entsprechende abgesetzte Umgebung `<umg>` wird ignoriert.

`\begin{<env>}` on input line `<line number>` ended by `\end{<other env>}`

`\begin{<umg>}` in Zeile `<Zeilennummer>` endet durch `\end{<andere umg>}`

Dieser Fehler tritt auf, wenn L^AT_EX feststellt, dass die Umgebung `<umg>` fälschlicherweise durch das Ende-Tag der Umgebung `<andere umg>` beendet wurde. Hier hat man sehr wahrscheinlich einfach vergessen, die Umgebung `<umg>` zu schließen.

Eine andere mögliche Ursache kann darin liegen, dass man versucht hat, eine Verbatim-ähnliche Umgebung oder eine `amsmath`-Umgebung für

abgesetzte Formeln in der Definition der eigenen Umgebung zu verwenden. Das ist oftmals nicht möglich. In Abschnitt 3.4.3 auf Seite 173 sind Lösungen für Verbatim-ähnliche Umgebungen beschrieben.

Trifft keine der beiden vorigen Möglichkeiten zu und man ist absolut sicher, alle Umgebungen korrekt verschachtelt zu haben, dann muss es irgendwo zwischen dem Anfang der Umgebung `\umg` und dem Auftreten des Fehlers einen Befehl geben, der einen `\endgroup`-Befehl ausführt, dem kein passender `\begingroup`-Befehl voranging. Dadurch wird $\text{\LaTeX}$ vorgetäuscht, dass die Umgebung `\umg` an dieser Stelle endet. Um den Ursprung dieses Problems ausfindig zu machen, kann man das Ende-Tag solange näher an das Anfangs-Tag heranführen, bis das Problem verschwindet.

`\begin{split}` won't work here $\text{\LaTeX}$

`\begin{split}` funktioniert hier nicht

Entweder befindet sich diese `split`-Umgebung nicht in einer Formel oder man muss hier vielleicht den Befehl `aligned` verwenden.

Can be used only in preamble

Darf nur in der Präambel verwendet werden

$\text{\LaTeX}$ ist auf einen Befehl oder eine Umgebung gestoßen, die man nur in einem Paket oder der Präambel (also vor `\begin{document}`) verwenden sollte. Dieser Fehler kann auch durch eine zweite `\begin{document}`-Anweisung verursacht werden.

Cannot be used in preamble

Darf nicht in der Präambel verwendet werden

Einige Befehle, wie etwa `\nocite` dürfen nur im Hauptteil des Dokumentes (also nach `\begin{document}`) verwendet werden. Die Deklaration ist an die entsprechende Stelle zu verschieben.

Cannot define Unicode char value < 00A0 $\text{\LaTeX}$

Kann kein Unicode-Zeichen mit Wert < 00A0 definieren

Werte unter "00A0 (Dezimalwert 160) sind entweder als Unicode-Werte für Textzeichen unzulässig oder dürfen in $\text{\LaTeX}$ nicht umdefiniert werden.

Cannot determine size of graphic in `\includegraphics` $\text{\LaTeX}$

Kann Größe der Graphik in `\includegraphics` nicht bestimmen

Für den Befehl `\includegraphics` wurde keine explizite Bildgröße angegeben und $\text{\LaTeX}$ konnte diese nicht direkt aus der Graphik`\includegraphics` bestimmen. Normalerweise wird diese Information automatisch ausgelesen, z.B. bei `.eps`-Dateien aus den „Bounding-Box“-Daten. Je nach verwendetem Graphiktreiber ist $\text{\LaTeX}$ jedoch nicht in der Lage, die Daten aus binären Bitmaps, wie etwa `.jpg`-, `.gif`- oder `.png`-Dateien zu extrahieren.

Cannot include graphics of type: `\type` $\text{\LaTeX}$

Kann folgende Graphikart nicht einbinden: `\type`

Diesen Fehler erhält man, wenn man im zweiten Argument von `\DeclareGraphicsRule` oder als Schlüsselwort `type` des Befehls

`\includegraphics` einen Graphiktyp angegeben hat, der vom geladenen Graphiktreiber nicht unterstützt wird.

`\caption outside float`

\caption außerhalb eines Gleitobjektes

Ein `\caption`-Befehl befindet sich außerhalb einer Gleitobjektumgebung wie `figure` oder `table`. Diese Fehlermeldung wird von einigen der Erweiterungspakete, die in Kapitel 6 beschrieben sind, deaktiviert.

Command `<name>` already defined

Befehl `<name>` bereits definiert

Man versucht einen Befehl, eine Umgebung, eine neue Savebox, eine Länge oder einen Zähler unter einem Namen zu deklarieren, der in $\text{\LaTeX}$ bereits belegt ist. Die Deklaration wird ignoriert, und man muss einen anderen Namen wählen. Dieser Fehler tritt ebenfalls auf, wenn man `\newcommand` mit einem `<name>`n verwendet, der mit `\end...` beginnt, selbst wenn der Befehl `\renewcommand` diesen `<name>`n als noch nicht benutzt bezeichnet. Er wird auch ausgegeben, wenn man eine Umgebung `<name>` definieren will und der Befehl `\end<name>` bereits existiert. So kann man z.B. keine Umgebung namens `graf` definieren, da es in $\text{\TeX}$ bereits einen Low-Level-Befehl namens `\endgraf` gibt.

Command `<name>` invalid in math mode

Befehl `<name>` im Formelmodus ungültig

Hier handelt es sich entweder um eine Warnung oder eine Fehlermeldung, die anzeigt, dass man im Formelmodus einen Befehl verwendet hat, der nur in normalem Text zulässig ist. Im Falle der Fehlermeldung erhält man über `h` weitere Informationen.

Command `<name>` not defined as a math alphabet

Befehls `<name>` nicht als Mathematikalphabet definiert

Dieser Fehler tritt auf, wenn man versucht, eine `\SetMathAlphabet`-Deklaration mit einem `<name>`n zu verwenden, der nicht zuvor mit `\DeclareMathAlphabet` oder `\DeclareSymbolFontAlphabet` als mathematischer Alphabetbefehl deklariert wurde.

Corrupted NFSS tables

NFSS-Tabellen beschädigt

$\text{\LaTeX}$ wollte eine Fontersetzung vornehmen und hat dabei festgestellt, dass seine internen Tabellen nicht konsistent sind. Dieser Fehler tritt auf, wenn eine Fontersetzung angestoßen wurde und die Ersetzungsregeln eine Schleife enthalten (also Zirkelbezüge in den sub-Deklarationen vorhanden sind) oder wenn die voreingestellten Ersetzungsargumente für die aktuelle Kodierung auf eine Schriftformgruppe verweisen, die es nicht gibt.

Counter too large

Zu hoher Zählerwert

Dieser Fehler wird erzeugt, wenn man den aktuellen Wert eines Zählers mit `\fnsymbol`, `\alph` oder `\Alph` darstellen will und dieser Wert

außerhalb des Bereiches liegt, welcher sich mit dem gewählten Befehl darstellen lässt (vgl. die Erläuterungen auf Seite 123).

Dimension too large TeX

Zu große Länge

TeX kann nur Größen verarbeiten, die absolut gesehen kleiner sind als 16383.99998pt (ca. 226 Zoll bzw. 574 Zentimeter). Selbst auf einer großen Seite sollte dies eigentlich genügen.

\displaybreak cannot be applied here amsmath

\displaybreak lässt sich hier nicht anwenden

Eine äußere Umgebung wie etwa `split`, `aligned` oder `gathered` hat einen Block gebildet, der nicht umbrochen werden kann.

Division by 0 graphics/graphics

Division durch 0

Diesen Fehler erhält man normalerweise, wenn man eine Graphik mit der Höhe null skaliert. Das kann ganz unabsichtlich geschehen, beispielsweise indem man im optionalen Argument von `\includegraphics` die Werte `angle=-90,height=3cm` angibt. Die Rotation dreht das Bild auf die Seite und setzt so die Höhe auf null – einen Wert, der sich schwerlich skalieren lässt. In einem solchen Fall kann man stattdessen `totalheight` verwenden.

Double subscript TeX

Doppelte Tiefstellungen

Zwei Tiefstellungen erfolgen nacheinander (z.B. x_{i_2}) und L^AT_EX weiß nicht, ob man damit x_{i_2} oder x_{i_2} meint. Mithilfe von Klammern kann man die Tiefstellungen eindeutig kennzeichnen: $x_{\{i_2\}}$.

Double superscript TeX

Doppelte Hochstellungen

L^AT_EX ist auf zwei aufeinander folgende Hochstellungen gestoßen – vgl. die vorige Erklärung.

Encoding file ‘<name>’ not found fontenc

Fontkodierungsdatei „<name>“ nicht gefunden

Ruft man die Kodierung `<name>` auf, so versucht L^AT_EX die Definitionen dieser Kodierung aus der Datei `<name>.def` zu laden, wobei `<name>` zunächst in Kleinbuchstaben umgesetzt wird. Wenn diese Fontkodierungsdatei nicht existiert oder L^AT_EX sie nicht findet, erfolgt diese Fehlermeldung.

Encoding scheme <name> unknown

Kodierschema <name> unbekannt

Das in einer Deklaration oder in `\fontencoding` angegebene Kodierschema `<name>` ist dem System nicht bekannt. Entweder hat man vergessen, es mithilfe von `\DeclareFontEncoding` zu deklarieren oder der Name ist falsch geschrieben.

Environment $\langle name \rangle$ undefined*Umgebung $\langle name \rangle$ nicht definiert*

Diesen Fehler erhält man, wenn man den Befehl `\renewenvironment` mit einem Umgebungsnamen verwendet, den L^AT_EX nicht kennt. Entweder wurde $\langle name \rangle$ falsch geschrieben oder hier wird eigentlich der Befehl `\newenvironment` benötigt.

Erroneous nesting of equation structures; amsmath trying to recover with ‘aligned’*Fehlerhaft verschachtelte Formeln; versuche mit „aligned“ weiterzuarbeiten*

Nur bestimmte abgesetzte `amsmath`-Gebilde lassen sich verschachteln. Eines davon ist die Umgebung `aligned`, die hier vom System als Ersatz für die fälschlicherweise verschachtelte Umgebung eingefügt wird. Da man wahrscheinlich etwas anderes beabsichtigt hat, sollte man die falsch verschachtelte Umgebung entsprechend ändern.

Extra & on this line amsmath *Überzähliges & in dieser Zeile*

Dieser Fehler tritt nur bei veralteten `amsmath`-Umgebungen auf, die in diesem Buch nicht beschrieben sind. Trifft man auf diesen Fehler, dann hat er meist eine verheerende Wirkung und man sollte die Umgebung, in der er aufgetreten ist, sorgfältig prüfen.

Extra alignment tab has been changed to $\backslash cr$ $\text{T}_{\text{E}}\text{X}$ *Überzähliger Ausrichtungstabulator wurde durch $\backslash cr$ ersetzt*

Verwendet man eine ausgerichtete Struktur, wie etwa `tabular` oder eine der abgesetzten Formelumgebungen (z.B. `eqnarray` oder `split` aus dem `amsmath`-Paket), dann wird jede Zeile durch `&`-Zeichen in eine festgelegte Anzahl von Spalten unterteilt. Der Fehler besagt, dass zu viele dieser Zeichen vorhanden sind, wahrscheinlich, weil man am Ende einer Zeile ein `\\` vergessen hat. Das `\cr` ist die T_EX-Bezeichnung für das Zeilenende. Es entspricht jedoch nicht in jeder Hinsicht dem `\\`-Befehl in L^AT_EX.

Extra $\backslash endgroup$ $\text{T}_{\text{E}}\text{X}$ *Überzähliges $\backslash endgroup$*

T_EX ist auf einen `\endgroup`-Befehl gestoßen, dem kein entsprechender `\begingroup`-Befehl voranging.

Extra $\backslash or$ $\text{T}_{\text{E}}\text{X}$ *Überzähliges $\backslash or$*

T_EX hat einen Basisbefehl `\or` außerhalb einer entsprechenden Low-Level-Bedingung `\ifcase` gefunden. Das überzählige `\or` kann von einem falschen Gebrauch des Befehls `\ifthenelse` herrühren.

Extra $\backslash right$ $\text{T}_{\text{E}}\text{X}$ *Überzähliges $\backslash right$*

Diesen Fehler gibt T_EX aus, wenn es in einer Formel den Befehl `\right` ohne ein passendes `\left` findet. Man beachte, dass `\left`-`\right`-Paare Teil derselben „Unterformel“ sein müssen. Sie dürfen beispielsweise

nicht in ausgerichteten Strukturen durch ein & getrennt werden oder in verschiedenen Klammergruppen erscheinen.

Extra }, or forgotten \$^{TeX}

Überzähliges }- oder fehlendes \$-Zeichen

Dieser Fehler wird ausgelöst, wenn Begrenzungszeichen für Formeln (wie etwa $\$. \$$, $\backslash[...\backslash]$) und Klammergruppen nicht korrekt verschachtelt sind. TeX geht davon aus, dass eine }-Klammer überflüssig ist, wie in $\$x\}$, und ignoriert diese entsprechend. In diesem Beispiel wäre das zwar die richtige Entscheidung, bei der Formel $\mbox{\(a)}$ jedoch nicht. Dort fehlt ein schließendes $\backslash$, so dass durch Löschen der }-Klammer zusätzliche Fehler entstehen würden.

Extra }, or forgotten \endgroup^{TeX}

Überzähliges }-Zeichen oder fehlende \endgroup

Die aktuelle Gruppe wurde mit dem Befehl $\backslashbegin{group}$ eröffnet (der z.B. von $\backslashbegin{...}$ verwendet wurde). TeX fand jedoch anstelle des entsprechenden $\backslashendgroup$ -Befehls eine schließende }-Klammer. Man erhält diesen Fehler, wenn man im Hauptteil einer Umgebung eine einzelne }-Klammer platziert.

File ‘<name>’ not found

Datei „<name>“ nicht gefunden

L^ATeX versucht, die Datei $\langle name \rangle$ zu laden, kann diese jedoch nicht finden. Entweder ist sie nicht vorhanden oder das zugrunde liegende TeX-Programm sucht an der falschen Stelle. Sollte die Datei vorhanden sein und L^ATeX behauptet trotzdem das Gegenteil, dann kann es sein, dass die lokale TeX-Installation für einen schnelleren Dateizugriff ein Hash-Verfahren verwendet. In diesem Falle muss man die Installation mithilfe eines besonderen Programms auf neu installierte Dateien aufmerksam machen (z.B. bei der TeX Live-Distribution von der beiliegenden CD-ROM mit dem Programm `mktexlsr`).

Der Fehler wird von Befehlen wie $\backslashinput$ und $\backslashusepackage$ ausgegeben, wenn diese die angeforderte Datei nicht finden können. Als Antwort auf die Fehlermeldung kann man eine alternative Datei vorschlagen. Wird der neue Name ohne Erweiterung angegeben, so verwendet L^ATeX eine gegebenenfalls vorhandene bisherige Erweiterung. Mit der Eingabetaste ($\langle Enter \rangle$) kann man das Laden der Datei übergehen; mit x oder X bricht man den Verarbeitungslauf ab. In manchen Fällen erhält man statt dieser Meldung eine ähnlich lautende Low-Level-Fehlermeldung von TeX: „I can't find file ‘<name>’“. Aus dieser kommt man allerdings nicht so einfach heraus (siehe Eintrag auf Seite 937).

Wenn eine mit $\backslashincludegraphics$ angeforderte Graphikdatei fehlt, dann erfährt man durch Eingabe von h, welche Erweiterungen bei der Suche nach der Datei verwendet wurden.

File ended while scanning <something>^{TeX}

Dateiende erreicht beim Lesen von <etwas>

Diese Fehlermeldung ist Teil eines „Runaway...“-Fehlers, siehe die Erläuterungen auf Seite 948.

Float(s) lost*Gleitobjekt(e) verloren*

Ein oder mehrere Gleitobjekte (zum Beispiel `figure` oder `table`) oder `\marginpar`-Befehle wurden nicht gesetzt. Die wahrscheinlichste Ursache dafür ist, dass man eine Gleitobjektumgebung oder eine Marginalie versehentlich in einer Box platziert hat: in eine andere Gleitobjektumgebung oder Marginalie (`\marginpar`), oder in eine `minipage`-Umgebung, eine Absatzbox (`\parbox`) oder eine Fußnote (`\footnote`). Es kann sein, dass $\text{\LaTeX}$ dieses Problem erst ziemlich spät entdeckt, etwa wenn das Dokument fertig gestellt wird. Dadurch wird es sehr schwierig, die problematische Stelle in der Quelldatei zu finden. Die beste Lösung besteht dann darin, das Dokument immer wieder zu halbieren (z.B. mithilfe des Basisbefehls `\endinput`), bis der fehlerhafte Teil so klein ist, dass man den Fehler ausmachen kann.

Besteht das Problem nicht in einer fehlerhaften Verschachtelung, dann hat man es möglicherweise mit einem ernststen Programmfehler im Gleitobjektalgorithmus zu tun. Dieser kann von zusätzlich geladenen Paketen stammen.

Font family $\langle encoding \rangle + \langle familie \rangle$ unknown*Unbekannte Fontfamilie $\langle kodierung \rangle + \langle familie \rangle$*

Man hat versucht, eine Schriftformgruppe mit `\DeclareFontShape` zu deklarieren, ohne zuvor die `Font\langle familie \rangle` mithilfe des Befehls `\DeclareFontFamily` als Teil dieser $\langle kodierung \rangle$ zu deklarieren.

Font ' $\langle name \rangle$ ' not found*Zeichensatz „ $\langle name \rangle$ “ nicht gefunden*

Die internen Zeichensatztabellen von $\text{\LaTeX}$ enthalten falsche Daten, so dass $\text{\LaTeX}$ den externen Zeichensatz $\langle name \rangle$ nicht finden konnte. Entweder wurde dieser Font noch gar nicht installiert, $\text{\TeX}$ kann die zugehörige `.tfm`-Datei nicht finden oder die auf ihn verweisende `\DeclareFontShape`-Deklaration enthält einen Schreibfehler.

Font $\langle internal-name \rangle = \langle external-name \rangle$ not loadable: $\langle reason \rangle$ $\text{\TeX}$ *Font $\langle interner-name \rangle = \langle externer-name \rangle$ kann nicht geladen werden: $\langle ursache \rangle$*

$\text{\TeX}$ konnte einen Font mit der $\text{\LaTeX}$ -Bezeichnung $\langle internal-name \rangle$ nicht laden. Die Bezeichnung entspricht der NFSS-Konvention und besteht aus `\langle kodierung \rangle / \langle familie \rangle / \langle serie \rangle / \langle form \rangle / \langle grad \rangle`.¹ Sie könnte beispielsweise `\T1/cmr/m/it/10` (Computer Modern normal kursiv 10 Punkte in T1-Kodierung) lauten. Das gibt Aufschluss darüber, welcher Font problematisch ist, auch wenn man nicht viel dagegen tun kann. Für die $\langle ursache \rangle$ gibt es zwei Möglichkeiten:

Bad metric (TFM) file $\text{\TeX}$ *Fehlerhafte Fontmetrikdatei (TFM)*

Die $\text{\TeX}$ -Fontmetrikdatei für den Zeichensatz (also $\langle externer-name \rangle$.`tfm`) ist beschädigt. Möglicherweise verfügt die verwendete

¹Das ist tatsächlich ein einzelner Befehlsname. Aufgrund der enthaltenen Schrägstriche kann man ihn jedoch nicht direkt im Dokument eingeben.

Installation über einige Hilfsprogramme zum genauen Überprüfen von `.tfm`-Dateien. Normalerweise benötigt man dazu jedoch fachliche Unterstützung.

Metric (TFM) file not found TeX

Fontmetrikdatei (TFM) nicht gefunden

Die $\text{\TeX}$ -Fontmetrikdatei für den Zeichensatz (also `\external-name.tfm`) wurde nicht gefunden. Die vorhandene Installation unterstützt zwar vielleicht eine bestimmte Fontfamilie durch ein Paket (z.B. `cmbright`), die entsprechenden Fonts sind jedoch nicht verfügbar oder nicht richtig installiert.

Font $\langle\text{internal-name}\rangle=\langle\text{external}\rangle$ not loaded: Not enough room left TeX

Font $\langle\text{interner-name}\rangle=\langle\text{externer}\rangle$ nicht geladen: Nicht mehr genügend Speicher

$\text{\TeX}$ kann nur eine bestimmte Anzahl an Zeichensätzen laden und für $\langle\text{interner-name}\rangle$ war kein Platz mehr. Mithilfe des in Abschnitt 7.5.6 beschriebenen Paketes `tracefont` lässt sich herausfinden, welche Fonts bisher geladen wurden. Ein möglicher Grund für übermäßig viele geladene Fonts ist die Verwendung ungewöhnlicher Schriftgrade, für die $\text{\LaTeX}$ die entsprechenden Mathematikfonts berechnen und laden muss. Nähere Einzelheiten hierzu sind in Abschnitt 7.10.7 beschrieben.

Font shape $\langle\text{font shape}\rangle$ not found

Schriftform $\langle\text{schriftform}\rangle$ nicht gefunden

Diese Fehlermeldung deutet darauf hin, dass etwas mit einer `\DeclareFontShape`-Deklaration nicht stimmt. Vielleicht enthält sie keine Angaben zu Schriftgraden. Hier sollte überprüft werden, wie die fragliche Schriftformgruppe eingerichtet ist.

I can't find file ' $\langle\text{name}\rangle$ ' TeX

Ich kann Datei „ $\langle\text{name}\rangle$ “ nicht finden

Dieser Low-Level-Fehler wird von $\text{\TeX}$ ausgegeben, wenn das Programm eine Datei, die geladen werden soll, nicht findet. An dieser Stelle kommt man nur weiter, wenn man $\text{\TeX}$ eine Datei nennt, die es auch finden kann, oder indem man den Verarbeitungslauf vollständig abbricht (sofern das Betriebssystem dies zulässt). Um dieses Problem zu lösen, verfügen viele Installationen über eine Datei namens `null.tex`, so dass man an dieser Stelle einfach `null` eingeben kann. $\text{\LaTeX}$ verwendet normalerweise die Fehlermeldung „File ' $\langle\text{name}\rangle$ ' not found“ („Datei ' $\langle\text{name}\rangle$ ' nicht gefunden“), die verschiedene Aktionen seitens des Anwenders unterstützt. Je nach Implementierung des Paketes kann man aber eben auch den hier beschriebenen Fehler erhalten.

I can't write on file ' $\langle\text{name}\rangle$ ' TeX

Ich kann nicht in die Datei „ $\langle\text{name}\rangle$ “ schreiben

$\text{\TeX}$ hat keine Berechtigung, in die Datei $\langle\text{name}\rangle$ zu schreiben. Sie ist entweder schreibgeschützt oder für das entsprechende Verzeichnis liegen keine Schreibrechte vor. Bei manchen $\text{\TeX}$ -Versionen (z.B. derjenigen auf

der T_EX Live CD) kann vor dem Fehler noch folgende Zeile stehen:

```
tex: Not writing to /texmf/tex/latex/base/latex.ltx (openout_any = p)
```

Diese Installationen verhalten sich gewissermaßen per Voreinstellung „paranoid“ (daher das „p“ in der Meldung), wenn sie in Dateien schreiben sollen. Sie überschreiben nur Dateien unterhalb des aktuellen Verzeichnispfades und *keine*, die mit einem absoluten Pfad angegeben wurden oder deren Dateiname mit einem Punkt beginnt. Dieses Verhalten lässt sich über die Einstellungen in der Datei `texmf.cnf` ändern.

Illegal character in array arg

Unzulässiges Zeichen in array arg

Diesen Fehler erhält man, wenn das Spaltenformat für eine `tabular`- oder `array`-Umgebung oder einen `\multicolumn`-Befehl Zeichen enthält, die in L^AT_EX nicht als Spaltenformatkürzel definiert sind. Wahrscheinlich wurde die erweiterte Syntax des in Kapitel 5 beschriebenen `array`-Paketes verwendet, wobei man vergessen hat, das Paket in der Präambel zu laden (z.B. wenn man eine Tabelle von einem Dokument in ein anderes kopiert).

Illegal parameter number in definition of `\command` T_EX

Ungültige Argumentanzahl in der Definition von `\befehl`

Dieser Fehler tritt auf, wenn (um-)definierte Befehle oder Umgebungen im Ersetzungstext `#<ziffer>` verwenden, wobei die Ziffer höher ist als die deklarierte Anzahl der Argumente. Dieser Fehler kann implizit durch das Verschachteln von Deklarationsbefehlen wie `\newcommand` entstehen, wenn man dabei vergisst, dass Bezüge auf die Argumente der inneren Befehle durch Verdoppeln der `#`-Zeichen erfolgen. Nähere Einzelheiten hierzu sind auf Seite 878 erläutert. Eine andere mögliche Ursache kann darin bestehen, dass man auf Umgebungsargumente im zweiten obligatorischen Argument von `\newenvironment` oder `\renewenvironment` verweist.

Illegal unit of measure (pt inserted) T_EX

Ungültige Maßeinheit (pt eingefügt)

Man erhält diesen Fehler, wenn man beim Festlegen eines Wertes für einen Längenparameter die Maßeinheit falsch schreibt oder vergisst (vgl. Abschnitt A.1.5).

Improper argument for math accent: amsmath

Extra braces must be added to prevent wrong output

Falsches Argument für Akzent in Formeln:

Zusätzliche Klammern erforderlich um fehlerhafte Ausgabe zu vermeiden

Die gesamte „mit Akzent versehene Unterformel“ muss in geschweiften Klammern stehen.

Improper discretionary list T_EX

Ungültige „discretionary“-Liste

Diesen Fehler erzeugt T_EX, wenn es auf einen `\discretionary`-Befehl trifft, dessen Argument nach seiner Expansion etwas anderes als Zeichen, Boxen oder Unterschneidungen enthält.

Improper \hyphenation *Ungültiger \hyphenation-Befehl*

Um mithilfe des \hyphenation-Befehls eine bestimmte Silbentrennung vorgeben zu können, darf das Argument des Befehls nur Buchstaben enthalten und das Zeichen – zur Kennzeichnung der Stellen, an denen getrennt werden kann. Das Problem besteht darin, dass beispielsweise Akzentbuchstaben in einigen Fontkodierungen als einzelne Glyphen (zulässig) vorliegen, in anderen Kodierungen jedoch komplizierte Konstruktionen erzeugen, welche den Basisbefehl \accent benötigen. Verwendet man z.B. die T1-Kodierung, dann verweist \"u auf eine einzelne Glyphe. Damit ist die folgende Anweisung gültig:

```
\usepackage[T1]{fontenc} \hyphenation{T\"ur-stop-per}
```

Mit der voreingestellten Kodierung OT1 würde die gleiche Anweisung für die Silbentrennung den genannten Fehler erzeugen. Auf Seite 467 sind die Zeichenunterschiede zwischen den wichtigsten Kodierungen erläutert.

Improper \prevdepth *Unzulässiger Gebrauch von \prevdepth*

Hier wurde \the\prevdepth oder \showthe\prevdepth außerhalb des vertikalen Modus verwendet, was nicht zulässig ist. Dieser Fehler tritt auch auf, wenn man versehentlich ein Gleitobjekt (z.B. figure oder table) in einer Umgebung für abgesetzte Formeln platziert hat.

Improper \spacefactor *Unzulässiger Gebrauch von \spacefactor*

Hier wurde der Befehl \the\spacefactor oder \showthe\spacefactor außerhalb des horizontalen Modus verwendet, was nicht zulässig ist.

\include cannot be nested*\include darf nicht verschachtelt werden*

L^AT_EX ist in einer mit \include eingebundenen Datei auf einen weiteren \include-Befehl gestoßen. Aufgrund von Einschränkungen bei der Implementierung darf das nicht geschehen. Man sollte entweder den inneren \include-Befehl durch \input ersetzen oder die Struktur des Dokumentes so verändern, dass alle \include-Anweisungen direkt im Hauptdokument stehen.

Incompatible list can't be unboxed *Inkompatible Liste kann nicht entpackt werden*

T_EX wurde aufgefordert beim Erstellen einer vertikalen Liste eine Box mit horizontalem Material zu entpacken oder umgekehrt. Entweder ist man hier auf einen schweren Programmierfehler in einem Paket gestoßen oder man hat einige Befehle in einer Weise verwendet, die ausdrücklich nicht unterstützt wird. So erzeugen beispielsweise die Befehle des soul-Paketes diesen Fehler, wenn sie ineinander verschachtelt werden.

Incomplete *<conditional>*; all text was ignored after line *<number>* TeX

Unvollständige *<bedingung>*; gesamter Text nach Zeile *<nummer>* wurde ignoriert

L^AT_EX hat das Ende der aktuellen Eingabedatei erreicht, und eine Low-Level-Bedingung von T_EX ist noch nicht beendet (es gab keinen passenden `\fi`-Befehl).

Infinite glue shrinkage found *<somewhere>* TeX

<irgendwo> wurde ein Abstand gefunden, der sich unendlich verkleinern lässt

Um Absätze in Zeilen oder die Druckfahne in Seiten umbrechen zu können, geht T_EX davon aus, dass es keine elastische Länge gibt, die beliebig schrumpfen kann. Andernfalls wäre es ja möglich, beliebig viel Material in eine einzige Zeile oder auf eine einzige Seite zu setzen. Daher sind die Befehle wie `\hspace{0pt minus 1fil}` in einem Absatz bzw. `\vspace{0pt minus 1fil}` zwischen zwei Absätzen nicht zulässig und rufen diesen Fehler hervor. Mit *<irgendwo>* wird auf die Stelle verwiesen, an der das fehlerhafte Material gefunden wurde.

Interruption TeX

Unterbrechung

Diese Fehlermeldung erhält man, wenn man einen L^AT_EX-Lauf (mit STRG-C oder dem entsprechenden Befehl der jeweiligen Installation) unterbrochen hat. Sie sollte daher nicht weiter überraschen. Mit x gefolgt von der Eingabetaste (*<Return>*) kann man den Lauf dann vorzeitig abbrechen. Alleiniges Drücken der Eingabetaste setzt den Lauf fort.

Invalid use of *<command>* amsmath

Unzulässige Verwendung von *<befehl>*

Ein amsmath-Befehl wurde an einer Stelle benutzt, an der er keinen Sinn ergibt. Man sollte nachschlagen, wie dieser Befehl richtig verwendet wird.

Keyboard character used is undefined in input encoding *<name>* inputenc

Das verwendete Tastaturzeichen ist in der Eingabekodierung *<name>* nicht definiert

Im Dokument wurde eine 8-Bit-Zahl gefunden, die in der Eingabekodierung *<name>* nicht mit einem LICR-Objekt verknüpft ist (siehe Abschnitt 7.5.2 und 7.11.3). Man sollte überprüfen, ob das Dokument wirklich in der angegebenen Kodierung gespeichert ist.

Language definition file *<language>.ldf* not found babel

Sprachdefinitionsdatei *<sprache>.ldf* nicht gefunden

Wenn L^AT_EX beim Verarbeiten der Optionsliste für das babel-Paket auf eine unbekannte Option *<sprache>* stößt, versucht das Programm eine Datei namens *<sprache>.ldf* zu laden. Diese Meldung wird angezeigt, wenn L^AT_EX die entsprechende Datei nicht findet. Hier kann ein einfacher Schreibfehler vorliegen oder die Datei ist vielleicht nicht unter dem L^AT_EX-Suchpfad gespeichert.

Limit controls must follow a math operator TeX

Befehle für Grenzen müssen hinter einem mathematischen Operator stehen

Die Befehle `\limits` bzw. `\nolimits` dürfen nur hinter einem mathematischen Operator wie z.B. `\sum` stehen. In Tabelle 8.4 sind die gängigen Operatornamen aufgeführt.

\LoadClass in package file

\LoadClass in Paketdatei

Der Befehl `\LoadClass` darf nur in Klassendateien vorkommen (siehe Abschnitt A.4).

Lonely \item—perhaps a missing list environment

Einsames \item – vielleicht fehlt eine Listenumgebung

Der `\item`-Befehl muss in einer Listenumgebung stehen. $\text{\LaTeX}$ nimmt hier jedoch an, dass der fragliche Befehl außerhalb einer Liste steht.¹

Math alphabet identifier <id> is undefined in math version <name>

Der mathematische Alphabetbefehl <id> ist im Formellayout <name> nicht definiert

Der mathematische Alphabetbefehl `<id>` wurde in einem Formellayout `<name>` verwendet, für das er nicht eingerichtet ist. In der Präambel des Dokumentes sollte diesem Alphabetbefehl mithilfe einer zusätzlichen `\SetMathAlphabet`-Deklaration eine geeignete Schriftformgruppe zugewiesen werden.

Math version <name> is not defined

Formellayout <name> ist nicht definiert

Ein mathematisches Alphabet oder ein Symbolfont wurden einem in $\text{\LaTeX}$ unbekannten Formellayout zugewiesen. Entweder wurde der Name falsch geschrieben oder man hat vergessen, dieses Layout zu deklarieren. Vielleicht muss noch eine Paketdatei hinzugefügt werden. Es kann auch sein, dass das System das mit einem `\mathversion`-Befehl ausgewählte Formellayout nicht kennt.

Misplaced alignment tab character & TeX

Falsch positionierter Ausrichtungstabulator &

$\text{\LaTeX}$ ist außerhalb von `tabular`, `align` oder einer der anderen ausgerichteten Umgebungen auf ein `&`-Zeichen gestoßen. Das Zeichen `&` sollte man mit `\&` setzen. Ein möglicher Grund kann darin bestehen, dass die `amsmath`-Umgebung `cases` oder `matrix` verwendet wurde, ohne dass man das Paket geladen hat.

Misplaced \cr or Misplaced \crrc TeX

Falsch platziertes \cr oder Falsch platziertes \crrc

Ein `\cr` ist in $\text{\TeX}$ der Low-Level-Befehl für das Zeilenende in einer ausgerichteten Struktur (`\crrc` ist eine Variante davon). Er entspricht in etwa dem $\text{\LaTeX}$ -Befehl `\\`. $\text{\TeX}$ nimmt an, dass sich der fragliche Befehl außerhalb einer solchen Struktur befindet.

¹Im Gegensatz zu der Fehlermeldung „... perhaps a missing \item“ ist die $\text{\LaTeX}$ -Diagnose in diesem Falle normalerweise zutreffend.

Misplaced `\noalign`^[TeX]*Falsch platzierter `\noalign`-Befehl*

Der TeX-Basisbefehl `\noalign` dient intern dazu, zwischen Zeilen von abgesetzten ausgerichteten Formeln „nicht ausgerichtetes“ Material einzufügen. Daher darf er nur unmittelbar hinter einem Zeilenendebefehl stehen. Beispielsweise erhält man diesen Fehler, wenn man `\hline` außerhalb der Umgebungen `array` oder `tabular` verwendet bzw. innerhalb, jedoch nicht direkt hinter einem `\\`.

Misplaced `\omit`^[TeX]*Falsch platzierter `\omit`-Befehl*

Der TeX-Basisbefehl `\omit` wird intern dazu verwendet, Spaltenformate in ausgerichteten abgesetzten Strukturen zu ändern (z.B. um Zellen innerhalb einer `tabular`-Umgebung mithilfe von `\multicolumn` über mehrere Zeilen zu spannen). Der Befehl `\omit`, und damit auch die ihn aufrufenden Befehle, sind nur direkt am Anfang einer Zelle in einer ausgerichteten Umgebung erlaubt (also nach `\\` oder `&`).

Missing `\begin{document}`*`\begin{document}` fehlt*

Dieser Fehler tritt auf, wenn man bereits in der Dokumentenpräambel versucht Material zu setzen.¹ Wahrscheinlich wurde ein Fehler in einer Deklaration von L^AT_EX fehlinterpretiert. Der Fehler wird auch hervorgerufen, wenn Text in einer Zeile hinter `\begin{filecontents}` folgt.

Missing control sequence inserted^[TeX]*Fehlender Befehlsname eingefügt*

Der Befehl `\newcommand` oder `\renewcommand` wurde ohne einen Befehlsnamen (der mit einem Backslash beginnt) im ersten Argument verwendet.

Missing `\cr` inserted^[TeX]*Fehlendes `\cr` eingefügt*

TeX nimmt an, dass es Zeit ist, eine Zeile in einer ausgerichteten Struktur zu beenden, und hat zu diesem Zweck seinen entsprechenden Low-Level-Befehl dort eingefügt. Bezogen auf ein L^AT_EX-Dokument ist diese Annahme für gewöhnlich falsch, so dass der Wiederherstellungsversuch von TeX normalerweise scheitert.

Missing delimiter (`.` inserted)^[TeX]*Fehlendes Begrenzungszeichen (`.` eingefügt)*

Auf `\left`, `\right` oder einen der `\big..`-Befehle folgte kein Begrenzungszeichen. Zur Korrektur wurde der leere Begrenzer „.“ eingefügt. Abschnitt 8.5.3 auf Seite 510 enthält nähere Einzelheiten hierzu.

Missing `\endcsname` inserted^[TeX]*Fehlender `\endcsname`-Befehl eingefügt*

Dieser Fehler kann auftreten, wenn man Befehle als Teil eines Zählers oder einer Umgebung verwendet, wie z.B. `\newenvironment{B1}{\"ode}`.

¹Das Setzen innerhalb einer `\sbox`- oder `\savebox`-Deklaration ist zwar zulässig, sollte aber im Allgemeinen besser erst nach `\begin{document}` erfolgen, da einige Pakete erst an diesem Punkt vollständig eingerichtet sind.

Missing number, treated as zero^{TeX}*Fehlende Zahl, als null behandelt*

Dieser Fehler tritt auf, wenn TeX eine Zahl oder Länge erwartet, aber etwas anderes vorfindet. Er wird beispielsweise erzeugt, wenn man `\value{page}` anstelle von `\thepage` verwendet, da TeX bei einem einzelnen `\value`-Befehl erwartet, dass ein Low-Level-Zähler zugewiesen wird. Ganz allgemein wird dieser Fehler dadurch ausgelöst, dass man ein Längenregister ohne eine passende Zuweisungsfunktion wie `\setlength` benutzt. Zudem erhält man die Meldung, wenn auf einen `\usebox`-Befehl kein mit `\newsavebox` definierter Zwischenspeicher für eine Box folgt, da solche Zwischenspeicher intern durch Zahlen dargestellt werden.

Missing p-arg in array arg*Fehlendes p-Argument in array-Argument*

Im Argument von `tabular`, `array` oder `\multicolumn` befindet sich ein Spaltenformatkürzel `p`, dem kein Ausdruck in geschweiften Klammern folgt. Der Ausdruck müsste die Breite angeben.

Missing @-exp in array arg*Fehlender @-Ausdruck in array-Argument*

Im Argument von `tabular`, `array` oder `\multicolumn` befindet sich ein Spaltenformatkürzel `@`, dem kein Ausdruck in geschweiften Klammern folgt. Dieser Ausdruck müsste das Material für den Spaltenzwischenraum enthalten.

Missing # inserted in alignment preamble^{TeX}*Fehlendes #-Zeichen in Ausrichtungspräambel eingefügt*

Eine Ausrichtungspräambel legt das Spaltenlayout für eine ausgerichtete Struktur fest. TeX kennzeichnet den Teil einer Spalte, der Eingaben erhalten soll, intern mit einem #-Zeichen. Dieser Fehler tritt in L^ATeX wahrscheinlich nur als Folgefehler auf.

Missing = inserted for \ifnum^{TeX}*Fehlendes =-Zeichen für \ifnum eingefügt*

TeX bemängelt, dass auf die Low-Level-Bedingung `\ifnum` nicht zwei Zahlen folgen, die durch ein `<`-, `=`- oder `>`-Zeichen getrennt sind. Dieser Fehler kann auftreten, wenn man in `\ifthenelse`-Anweisungen den Vergleichsoperator vergisst.

Missing = inserted for \ifdim^{TeX}*Fehlendes =-Zeichen für \ifdim eingefügt*

Der Low-Level-Bedingung `\ifdim` folgt kein Vergleich zweier Längen.

Missing \$ inserted^{TeX}*Fehlendes \$-Zeichen eingefügt*

Entweder ist TeX in normalem Text auf ein Element gestoßen, das nur im Formelmodus zulässig ist (wie z.B. `\sum`, `\alpha`, `^`) oder umgekehrt beim Verarbeiten einer Formel auf ein reines Textelement (z.B. `\par`). Darum wurde ein `$`-Zeichen eingefügt, um in den Formelmodus zu wechseln bzw. diesen zu verlassen. Hat man beispielsweise versucht, einen

Unterstrich nur durch `_` anstelle von `\_` zu erhalten, dann setzt $\LaTeX$ den Rest des Absatzes im Formelmodus, wobei sehr wahrscheinlich noch weitere Fehler auftreten.

Missing `\endgroup` inserted ^{$\TeX$}

Fehlender `\endgroup`-Befehl eingefügt

Dieser Fehler weist darauf hin, dass eine Gruppierung im Dokument fehlerhaft verschachtelt ist. Umgebungen verwenden intern `\begingroup` und `\endgroup` und aus irgendeinem Grund nimmt $\TeX$ an, dass eine solche Gruppe nicht ordnungsgemäß geschlossen wurde. Wenn sich nicht feststellen lässt, warum ein Gruppe fehlerhaft ist, kann man versuchen, es mithilfe der `\showgroups`- oder der `\tracinggroups`-Funktion von $\epsilon\TeX$ herauszufinden. Das Verfahren wird auf Seite 958 beschrieben.

Missing `\right.` inserted ^{$\TeX$}

Fehlender `\right.`-Befehl eingefügt

Eine Formel enthält den Befehl `\left` ohne den zugehörigen `\right`-Befehl. Man erinnere sich, dass die Begrenzerpaare `\left` und `\right` zur gleichen „Unterformel“ gehören müssen. Sie dürfen beispielsweise nicht durch ein `&` in einer ausgerichteten Struktur getrennt werden oder in verschiedenen Gruppierungsblöcken erscheinen.

Missing `{` inserted ^{$\TeX$}

Fehlende `{`-Klammer eingefügt

$\TeX$ nimmt an, dass eine öffnende geschweifte Klammer fehlt, und hat diese eingefügt. Dieser Fehler wird z.B. durch eine einzelne `}`-Klammer in einer `tabular`-Zelle ausgelöst.

Missing `}` inserted ^{$\TeX$}

Fehlende `}`-Klammer eingefügt

Irgendetwas in der Gruppierungsstruktur des Dokumentes stimmt nicht und $\TeX$ versucht das Problem durch Einfügen einer schließenden geschweiften Klammer zu lösen. Entweder schafft dieser Versuch Abhilfe oder er erzeugt noch weitere Fehler. Das Problem ist normalerweise mit einem Blick auf das gesetzte Material zu erkennen. Wenn sich nicht feststellen lässt, warum die Gruppenstruktur fehlerhaft ist, kann man versuchen, es mithilfe der `\showgroups`- oder der `\tracinggroups`-Funktion von $\epsilon\TeX$ herauszufinden. Das Verfahren wird auf Seite 958 beschrieben.

Multiple `\label's`: label $\langle label \rangle$ will be lost^{amsmath}

Mehrfacher `\label`-Befehl: Das Label $\langle label \rangle$ wird ignoriert

In `amsmath`-Umgebungen für abgesetzte Formeln darf nur ein `\label`-Befehl je Gleichungszeile vorkommen. Es ist normalerweise am Besten, wenn man alle bis auf den letzten entfernt, da nur dieser wirksam ist.

Multiple `\tag`^{amsmath}

Mehrfacher `\tag`-Befehl

In der `amsmath`-Umgebung für abgesetzte Formeln darf nur ein `\tag`-Befehl je Gleichung vorkommen. Alle bis auf den ersten werden ignoriert.

No counter '*<name>*' defined*Zähler *<name>* ist nicht definiert*

Der Zähler *<name>*, auf den entweder in `\setcounter`, `\addtocounter` oder im optionalen Argument des `\newcounter`- oder des `\newtheorem`-Befehls verwiesen wird, ist L^AT_EX unbekannt. Er muss erst noch mithilfe des Befehls `\newcounter` deklariert werden.

No Cyrillic encoding definition files were found babel*Es wurden keine Definitionsdateien für Kyrillisch gefunden*

Die Sprachdefinitionsdateien für die unterstützten „kyrillischen Sprachen“ prüfen, ob eine der bekannten Fontkodierungsdateien für das Kyrillische (z.B. T2A oder T2B) zu finden ist. Andernfalls wird diese Fehlermeldung angezeigt und man muss erst noch die L^AT_EX-Unterstützung für Kyrillisch installieren.

No declaration for shape ''***Keine Deklaration für die Schriftform '*<schriftform>*'*

Die in einem `\DeclareFontShape`-Befehl verwendete Größenfunktion `sub` oder `ssub` verweist auf eine Ersatzschriftform, die das Fontauswahlsystem von L^AT_EX nicht kennt.

No driver specified color/graphics/graphicsx*Treiber nicht angegeben*

Eines der Pakete `graphics`, `graphicx` oder `color` wurde geladen, ohne dass eine Option für das Ausgabegerät angegeben wurde. Bei den meisten Installationen wird diese Einstellung mithilfe der Konfigurationsdateien `graphics.cfg` und `color.cfg` vorgenommen.

No room for a new '*<register>*' T_EX*Kein Platz für ein neues '*<register>*'*

Die im Dokument geladenen Pakete beanspruchen mehr interne Register (`\count`, `\dimen`, ...) als T_EX zur Verfügung stellt. Hier kann man versuchen das fragliche Dokument mit eT_EX zu verarbeiten und zusätzlich das etex-Paket zu laden.

No `\title` given*`\title` fehlt*

Eine L^AT_EX-Klasse hat den Befehl `\maketitle` ausgeführt, ohne dass sie eine `\title`-Deklaration entdecken konnte. Bei diesem Befehl ist nur `\date` optional.

Not a letter T_EX*Kein Buchstabe*

Hier wurde mit `\hyphenation` eine besondere Silbentrennung vorgegeben, wobei das Argument des Befehls jedoch einige Zeichen enthielt, die T_EX nicht als Buchstaben ansieht. So erzeugt beispielsweise die Anweisung `\hyphenation{la-ryn-gol-o-gist's}` einen solchen Fehler, da es sich bei ' nach den Kategorien von T_EX nicht um einen „Buchstaben“ handelt.

Not in outer par mode

Nicht im äußeren Absatzmodus

Dieser Fehler wird ausgegeben, wenn in einem Befehl oder einer Umgebung, die eine Box erzeugt, ein `\marginpar`-Befehl oder eine Gleitobjektumgebung wie `table` oder `figure` vorkommt. So darf man einen `\marginpar`-Befehl nicht in einer Fußnote, einem Gleitobjekt, einer tabular-Umgebung oder an einer ähnlichen Stelle verwenden, da diese alle Boxen erzeugen. Das bemängelte Objekt muss in den Fließtextbereich verschoben werden.

Number too big^{TeX}

Zu große Zahl

In einem `\setcounter`- oder `\addtocounter`-Befehl wurde eine Zahl zugewiesen oder verwendet, die größer als die größte oder kleiner als die kleinste Zahl ist, die TeX noch verarbeiten kann (absolut 2147483647, hexadezimal 7FFFFFFF). Dieser Fehler kann auch auftreten, wenn man mit `\setlength` oder `\addtolength` ein Längenregister ändert.

OK^{TeX}

Okay

Hier wurde ein TeX-Überwachungsbefehl wie `\show` oder `\showthe` verwendet. Nach Anzeige der Daten stoppt L^AT_EX mit dieser Meldung, damit man Eingaben in der Befehlszeile vornehmen kann (wie z.B. `i\show..` um einige andere Werte anzusehen). Sie wird auch ausgegeben, wenn `\tracingonline` positiv ist und Befehle verwendet werden, die normalerweise nur in die Protokolldatei schreiben (siehe nächste Fehlermeldung).

OK (see the transcript file)^{TeX}

Okay (siehe Protokolldatei)

Hier wurde ein TeX-Überwachungsbefehl wie `\showbox` oder `\showlists` benutzt, ohne dass man L^AT_EX angewiesen hat, die Ergebnisse auf dem Bildschirm auszugeben.

Old form `<command>` should be `\begin{<envname>}`^{amsmath}

Alte Form von `<befehl>` sollte `\begin{<umgname>}` heißen

Man hat eine der Umgebungen `cases`, `matrix` oder `pmatrix` in der nicht an `amsmath` angepassten Befehlsform verwendet (wahrscheinlich mit ihrer alten internen Syntax). Hier muss die `amsmath`-Form der Umgebung mit der internen Standardsyntax benutzt werden.

Only one # is allowed per tab^{TeX}

Je Tabulator ist nur ein #-Zeichen zulässig

Dieser Fehler deutet auf eine beschädigte Spaltenvorlage einer Ausrichtungsumgebung hin. In L^AT_EX kann er eigentlich nur durch einen zerbrechlichen Befehl in einem bewegten Argument hervorgerufen werden.

Option clash for package `<name>`

Optionskonflikt bei Paket `<name>`

Das Paket `<name>` wurde zweimal mit unterschiedlichen Optionen aufgerufen. Gibt man bei dieser Fehlermeldung `h` ein, so zeigt L^AT_EX die

widersprüchlichen Optionen an. Da $\text{\LaTeX}$ jedes Paket nur einmal lädt¹, gibt man am besten alle benötigten Optionen direkt beim ersten Mal an. Ist dies nicht möglich, da das Paket bereits zuvor von einer Klasse oder einem anderen Paket geladen wird, dann kann man versuchen, die benötigten Optionen dem `\documentclass`-Befehl als globale Optionen zu übergeben. Zur Not kann man ein Paket auch mit dem Befehl `\RequirePackage` noch vor `\documentclass` laden. Abschnitt 2.1.1 enthält nähere Einzelheiten hierzu.

Page height already too large

Seitenhöhe bereits zu groß

Man hat versucht, eine Seite, deren vertikale Größe bereits über 8191.99998pt (ungefähr 113 Zoll bzw. 287 Zentimeter) liegt, mit `\enlargethispage` zu vergrößern. $\text{\LaTeX}$ hält das für gefährlich groß und wird die Seite nicht wie gewünscht verlängern.

Paragraph ended before `\command` was complete $\text{\TeX}$

Absatz endete, bevor `\befehl` vollständig war

Wie bereits in Abschnitt A.1.2 besprochen, dürfen mit `\newcommand*` oder `\renewcommand*` definierte Befehle keine `\par`-Befehle oder Leerzeichen enthalten. Andernfalls erhält man zusammen mit diesem Fehler die Meldung „Runaway argument“. Der genannte `\befehl` muss nicht mit dem im Dokument verwendeten identisch sein. Beispielsweise wird für `\emph{. . \par . .}` in der Fehlermeldung der Befehl `\text@command` aufgeführt, also der interne Befehl, den `\emph` aufruft.

(Please type a command or say ‘\end’) $\text{\TeX}$

(Bitte einen Befehl oder „\end“ eingeben)

Man hat auf die Meldung * mit Drücken der Eingabetaste (`\Return`) reagiert. Diese Meldung wurde auf Seite 929 erläutert. $\text{\TeX}$ s Vorschlag ein `\end` einzugeben ist falsch!

`\pushtabs` and `\poptabs` don’t match

`\pushtabs` und `\poptabs` passen nicht zusammen

In einer tabbing-Umgebung wurde ein `\poptabs`-Befehl verwendet, ohne dass zuvor ein `\pushtabs`-Befehl erfolgte.

`\RequirePackage` or `\LoadClass` in Options Section

`\RequirePackage`- oder `\LoadClass`-Befehl in Optionsbereich

In einer Paket- oder Klassendatei befindet sich zwischen den `\DeclareOption`-Befehlen und der Anweisung `\ProcessOptions` ein `\RequirePackage`- oder ein `\LoadClass`-Befehl. In diesem Bereich dürfen keine Pakete oder Klassen geladen werden, da diese die Informationen über die aktuellen Optionen zerstören würden (Abschnitt A.4 erläutert nähere Einzelheiten). Will man ein Paket in Abhängigkeit von einer bestimmten Option laden, dann sollte man ein Flag setzen, wenn diese Option ausgewählt wurde, und das entsprechende Paket erst nach Beendigung des `\ProcessOptions`-Befehls laden.

¹Hier bildet das Paket `fontenc` die einzige Ausnahme. Es kann so oft wie nötig mit unterschiedlichen Optionen geladen werden (siehe Abschnitt 7.5.3 auf Seite 372).

Rotation not supported

Drehung nicht unterstützt

Hier wurde mit `\rotatebox` oder einem ähnlichen Befehl die Drehung eines Objektes angefordert. Diese wird jedoch von dem ausgewählten Graphiktreiber nicht unterstützt. $\text{\LaTeX}$ reserviert zwar den passenden Leerraum, das Bild kann im formatierten Dokument jedoch in der falschen Lage erscheinen.

Runaway *<something>*

Ende von *<etwas>* fehlt

$\text{\TeX}$ nimmt an, dass es schon zu lange nach dem Ende von *<etwas>* sucht, wobei *<etwas>* hier entweder für *argument*, *definition*, *preamble* oder *text* stehen kann. Sofern es sich nicht um einen Fehler in der Low-Level-Programmierung eines Paketes handelt, ist die wahrscheinlichste Ursache ein Argument. Beispielsweise kann eine vergessene schließende Klammer für ein Argument dazu führen, dass $\text{\TeX}$ bis ans Ende des Dokumentes weitersucht oder bis sein Arbeitsspeicher voll ist, je nachdem, was zuerst eintritt. Auch unvollständige `\newcommand`-, `\newenvironment`- und ähnliche Definitionen behaupten, dass noch ein Argument aussteht.

Die Fehlermeldung `Runaway definition` kann nur von fehlerhaften Low-Level-Definitionen hervorgerufen werden, welche $\text{\TeX}$ -Basisbefehle wie `\def` enthalten.

Die Meldung `Runaway preamble` weist darauf hin, dass es Probleme in einer ausgerichteten Struktur gibt (welche in herkömmlichen $\text{\LaTeX}$ -Dokumenten eigentlich nicht vorkommen sollten). `Runaway text` verweist normalerweise auf die Zuweisung eines Tokenregisters und sollte gar nicht auftreten, außer durch einen schweren Programmierfehler bei der Implementierung eines Paketes.

Anders als bei normalen Fehlermeldungen erfolgt kein Hinweis darauf, in welcher Zeile der Fehler auftrat, da $\text{\TeX}$ oftmals schon das Ende der Datei erreicht hat. Stattdessen wird der Anfang des gesammelten Materials angezeigt. Angenommen bei einer Definition fehlt die abschließende Klammer:

```
\newcommand\foo{bar
\begin{document} Irgendein Text \end{document}
```

In diesem Fall erhält man:

```
Runaway argument?
{bar \begin {document} Irgendein Text \end {document}
! File ended while scanning use of \@argdef.
<inserted text>
      \par
<*> samplefile.tex

?
```

$\text{\TeX}$ hat in diesem Falle versucht, die Situation durch Einfügen eines `\par`-Befehls zu retten. Da dies aber erst am Ende des gesamten Do-

kumentes geschah, war es nicht mehr besonders hilfreich. Anstelle des Hinweises „File ended while...“ kann hier auch eine andere Meldung erscheinen, wie etwa „Paragraph ended before...“.

Scaling not supported graphics/graphics

Skalierung wird nicht unterstützt

Hier wurde mit `\resizebox` oder einem ähnlichen Befehl die Skalierung eines Objektes angefordert. Diese wird jedoch von dem ausgewählten Graphiktreiber nicht unterstützt. $\text{\LaTeX}$ reserviert zwar den passenden Leerraum, das Bild wird im gedruckten Dokument jedoch in der Originalgröße (nicht skaliert) erscheinen.

Something's wrong-perhaps a missing \item

Irgendetwas läuft hier schief, vielleicht fehlt ein \item

Dieser Fehler wird durch einen `\addvspace`-Befehl im horizontalen Modus hervorgerufen. Die anschließende Bemerkung „perhaps a missing \item“ trifft leider nur selten zu. Vergisst man zum Beispiel bei einem `\mbox`-Befehl die schließende Klammer (wie etwa `\mbox{... \section{...}}`), dann tritt dieser Fehler auf, da der `\section`-Befehl, der intern `\addvspace` ausführt, nun im horizontalen Modus aufgerufen wird.

Hier gilt es herauszufinden, welcher Befehl den fraglichen `\addvspace`-Befehl aufgerufen hat, und dort zu prüfen, ob der aufrufende Befehl korrekt verwendet wurde. Auf Seite 891 wurde der `\addvspace`-Befehl ausführlich besprochen.

Sorry, I can't find <format> ... TeX

Leider kann ich das <format> ... nicht finden

Erhält man diese Meldung, dann wurde $\text{\LaTeX}$ gar nicht aufgerufen, da $\text{\TeX}$ das `<format>` mit den $\text{\LaTeX}$ -Basisdefinitionen nicht finden konnte. Das Problem liegt an der $\text{\TeX}$ -Installation und man sollte die Installationsdokumentation zu Rate ziehen.

Suggested extra height (<value>) dangerously large

Vorgeschlagene Zusatzhöhe (<wert>) gefährlich groß

Verwendet man den Befehl `\enlargethispage` mit diesem `<wert>`, dann wird die fertige Seite nach dem Dafürhalten von $\text{\LaTeX}$ zu groß.

Symbol font <name> is not defined

Symbolfont <name> ist nicht definiert

Hier wurde versucht, den Symbolfont `<name>` zu benutzen (zum Beispiel in einer `\DeclareMathSymbol`-Deklaration), ohne dass er zuvor mit `\DeclareSymbolFont` deklariert wurde.

Symbol <command> not provided by font family <name> textcomp

Symbol<befehl> wird nicht von Schriftfamilie <name> unterstützt

Das `textcomp`-Paket implementiert die TS1-Kodierung, die leider nur von sehr wenigen Fontfamilien für $\text{\LaTeX}$ vollständig unterstützt wird. Das Paket wird das fragliche Symbol in einer mittels `\textcompstubstdefault` voreingestellten Schriftfamilie setzen. Dieser Fehler lässt sich in eine

Warnung umwandeln, indem man das Paket `textcomp` mit der Option `warn` lädt. Abschnitt 7.5.4 enthält nähere Einzelheiten hierzu.

Tab overflow

Tabulator-Überlauf

L^AT_EX unterstützt bis zu dreizehn Tabulatorpositionen (`\=`) in einer tabbing-Umgebung – hier wurde jedoch eine höhere Anzahl verwendet. Werden sie nicht alle gleichzeitig benötigt, so kann man versuchen, das Problem mithilfe des `\pushtabs`-Befehls zu lösen und/oder mit Vorlagenzeilen (`\kill` am Zeilenende), in denen Tabulatoren neu eingestellt werden.

\tag not allowed here amsmath

\tag hier nicht zulässig

Der `\tag`-Befehl darf nur in der obersten Ebene einer abgesetzten Formel verwendet werden. Am besten setzt man ihn ans Ende der logischen Gleichung, zu der er gehört.

TeX capacity exceeded, <explanation> TeX

TeX-Kapazität überschritten, <erklärung>

T_EX ging irgendeine Art von Speicher aus, so dass es die Verarbeitung einstellen musste. Dieser Fehler wird in Abschnitt B.1.1 auf Seite 956 genauer besprochen.

Text line contains an invalid character TeX

Textzeile enthält ein ungültiges Zeichen

Die Eingabedatei enthält ein seltsames, nicht druckbares Zeichen, das von T_EX zurückgewiesen wird. Das kann passieren, wenn man die Datei mit einem Textverarbeitungsprogramm erstellt und sie dann nicht als „Text“ gespeichert hat.

The attribute <attrib> is unknown for language <lang> babel

Das Attribut <attribut> ist für die Sprache <sprache> unbekannt

Hier sollte für eine <sprache> ein Attribut aktiviert werden, das in der Sprachdefinitionsdatei für diese Sprache nicht definiert ist. Die `babel`-Dokumentation sollte näheren Aufschluss zu der fraglichen Sprache geben können.

The character '<char>' is not a shorthand character in <language> babel

Das Zeichen „<zeichen>“ dient in dieser <sprache> nicht als Kurzzeichen

Diese Fehlermeldung wird angezeigt, wenn ein Anwender dem Befehl `\shorthandon` ein <zeichen> übergibt, das in der aktuellen <sprache> nicht als Kurzzeichen definiert ist. Die Anweisung wird ignoriert.

The font size command \normalsize is not defined...

Der Schriftgrößenbefehl \normalsize ist nicht definiert...

Eine Klassendatei muss über ein bestimmtes Minimum an Definitionen verfügen, wozu auch der Befehl `\normalsize` gehört. Nähere Erläuterungen hierzu befinden sich in Abschnitt A.4.9 auf Seite 922.

There's no line here to end

Hier gibt es keine zu beendende Zeile

Dieser Fehler wird ausgelöst, wenn sich ein `\newline`- oder ein `\\`-Befehl außerhalb eines Absatzes befindet, also hinter einem `\par`-Befehl oder einer Leerzeile. Wollte man hier einen zusätzlichen vertikalen Abstand einfügen, so sollte man dies mithilfe von `\vspace` oder einem der anderen auf Seite 890 beschriebenen Befehlen tun.

This may be a LaTeX bug

Hier könnte ein LaTeX-Fehler vorliegen

Soweit es dem Autor bekannt ist, hat diese Meldung bisher noch nie wirklich auf einen LaTeX-Fehler hingewiesen. Sie zeigt jedoch an, dass LaTeX durch vorhergehende Fehler gehörig durcheinander geraten ist und den Überblick über den zu verarbeitenden Datenstrom verloren hat. Am besten stoppt man den Lauf, korrigiert zunächst die schon aufgelaufenen Fehler und versucht es dann noch einmal.

This NFSS system isn't set up properly

Dieses NFSS-System ist nicht richtig eingerichtet

Dieser Fehler tritt auf, wenn LaTeX einen Fehler entdeckt, während es bei `\begin{document}` versucht, die Tabellen für die Fontersetzung zu verifizieren. Er weist darauf hin, dass entweder eine `\DeclareFontSubstitution`- oder die `\DeclareErrorFont`-Deklaration¹ fehlerhaft ist. Diese Deklarationen müssen auf gültige Schriftformen verweisen, die mit `\DeclareFontShape` deklariert wurden. Durch Eingabe von `h` erhält man weitere Informationen. Außerdem sollte man den Systembetreuer verständigen. Wer selbst Systembetreuer ist, der sollte in diesem Falle das Ende von Abschnitt 7.10.5 lesen.

Too deeply nested

Zu tief verschachtelt

Standard-LaTeX unterstützt bei Listen sechs Verschachtelungsebenen. Diese Ebenen können bis zu vier `itemize`- oder `enumerate`-Listen enthalten. Dieser Fehler deutet darauf hin, dass das betreffende Dokument eine dieser Grenzen überschritten hat. Wahrscheinlich wurde eine Listenumgebung nicht korrekt geschlossen. Sollte man wirklich weitere Ebenen benötigen, so muss man die Basisdefinitionen von `list`, `itemize` und/oder `enumerate` in ein eigenes Paket kopieren und dann dort die fest vorgegebenen Konstanten ändern.

Too many columns in eqnarray environment

Zu viele Spalten in der Umgebung eqnarray

Die Umgebung `eqnarray` unterstützt höchstens drei Spalten (also zwei `&`-Zeichen pro Zeile). Für qualitativ hochwertigen Mathematiksatz sollte

¹Die Deklaration `\DeclareErrorFont` wird während der Installation benutzt und verweist auf einen Zeichensatz (Schriftform und -grad), der verwendet werden soll, wenn gar nichts anders mehr geht. Sie ist auf die Schrift Computer Modern Roman 10pt voreingestellt, die in jeder TeX-Installation vorhanden sein sollte. Nähere Einzelheiten sind in [109] erläutert.

man das in Kapitel 8 beschriebene `amsmath`-Paket in Betracht ziehen, das auch komplexere abgesetzte Formeln zulässt.

Too many math alphabets used in version $\langle name \rangle$

Im Formellayout $\langle name \rangle$ werden zu viele Mathematikalphabete benutzt

In den Formeln wurden zu viele unterschiedliche Alphabetbefehle verwendet. Sollte dieser Fehler auftreten, nachdem man das `bm`-Paket hinzugefügt hat, dann sollte man vor Laden des Paketes die Definition `\newcommand\bmmax{0}` verwenden. Dadurch weist das Paket den Formellayouts nicht bereits im Voraus Mathematikalphabete zu.

Too many unprocessed floats

Zu viele aufgelaufene Gleitobjekte

Gleitobjekte, die nicht sofort gesetzt werden können, werden von $\LaTeX$ zurückgestellt. Dadurch werden möglicherweise auch darauf folgende Gleitobjekte aufgeschoben. $\LaTeX$ kann bis zu 18 Gleitobjekte zurückstellen, danach erhält man diese Fehlermeldung. Mithilfe des Paketes `morefloats` kann man diese Grenze auf 36 erhöhen. Lässt sich ein Gleitobjekt jedoch aus irgendeinem Grund einfach nicht platzieren, so bewirkt diese Änderung lediglich einen Aufschub, bis man die Fehlermeldung erhält. In Kapitel 6 sind Möglichkeiten erläutert, mit diesem Problem umzugehen.

Der Fehler kann auch durch zu viele `\marginpar`-Befehle in einem einzelnen Absatz ausgelöst werden. Jeder `\marginpar`-Befehl belegt zwei Zwischenspeicher für zurückgestellte Gleitobjekte, solange der aktuelle Absatz noch nicht gesetzt ist. Daher kann ein Absatz höchstens neun Marginalien enthalten, oder weniger, wenn bereits Gleitobjekte aufgelaufen sind.

Two `\documentclass` or `\documentstyle` commands

Zwei `\documentclass`- oder `\documentstyle`-Befehle

Jedes Dokument darf nur einen solchen Befehl enthalten. Das fragliche Dokument enthält jedoch mehr als einen, vielleicht weil hier zwei ursprünglich eigenständige Dokumente zusammengefasst wurden.

Two `\LoadClass` commands

Zwei `\LoadClass`-Befehle

Eine Klasse kann auf höchstens *einer* anderen Klasse aufgebaut sein und kann damit maximal eine Klasse laden. In Abschnitt A.4 wird erklärt, wie Klassen aufgebaut sind.

Undefined color $\langle name \rangle$ color

Nicht definierte Farbe $\langle name \rangle$

Mit `\color` oder einem ähnlichen Befehl des `color`-Paketes wurde eine Farbe angefordert, die noch nicht mithilfe von `\definecolor` definiert wurde. Nähere Erläuterungen hierzu bieten [58] oder die Dokumentation des `color`-Paketes.

Undefined control sequence TeX

Nicht definierter Befehl

Dies ist der vielleicht gängigste $\LaTeX$ -Fehler, obwohl er sich als $\TeX$ -Fehlermeldung zeigt: Es wurde ein noch nicht definierter Befehlsname

benutzt. Häufig hat man nur den Namen im Dokument falsch geschrieben, wie z.B. `\bmox` anstelle von `\mbox`. Um fortzufahren gibt man an dieser Stelle einfach `i\mbox`, also den richtigen Namen, ein. Später kann man dann das Quelldokument korrigieren. Dieser Fehler kann aber auch durch einen zerbrechlichen Befehl in einem bewegten Argument hervorgerufen werden.

Undefined font size function $\langle name \rangle$

Nicht definierte Schriftgrößenfunktion $\langle name \rangle$

Eine Größenfunktion in einer `\DeclareFontShape`-Deklaration wurde falsch geschrieben. Man sollte den Eintrag überprüfen oder den Systembetreuer verständigen.

Undefined tab position

Nicht definierte Tabulatorposition

Dieser Fehler wird ausgelöst, wenn man versucht, in einer tabbing-Umgebung mit `\>`, `\+`, `\-` oder `\<` auf eine Tabulatorposition vorzurücken, die nicht zuvor mit `\=` eingerichtet wurde. Entweder fehlt wirklich der `\=`-Befehl, oder vielleicht hat man `\+` oder `\pushtabs` verwendet und ist dann bei der Angabe der Tabulatorposition, zu der man eigentlich vorrücken wollte, durcheinander geraten.

Unknown graphics extension: $\langle ext \rangle$ graphics/graphics

Unbekannte Graphikerweiterung: $\langle erw \rangle$

Diesen Fehler erhält man, wenn man versucht, eine Graphikdatei mit der Erweiterung $\langle erw \rangle$ mit vollem Namen zu laden, wenn diese Erweiterung dem Graphiktreiber nicht bekannt und auch keine Verfahrensregel für einen solchen Fall vorgegeben ist. Das Programm `dvips` beispielsweise interpretiert jede unbekannte Erweiterung als EPS, so dass diese Fehlermeldung bei diesem Treiber nicht auftreten wird – dafür aber wahrscheinlich andere.

Unknown option ' $\langle option \rangle$ ' for package ' $\langle name \rangle$ '

Unbekannte Option „ $\langle option \rangle$ “ für Paket „ $\langle name \rangle$ “

Für das Paket $\langle name \rangle$ wurde eine $\langle option \rangle$ angegeben, die von diesem Paket nicht deklariert wird. Die verfügbaren Optionen können in der Paketdokumentation nachgeschlagen werden.

Use of $\langle command \rangle$ doesn't match its definition TeX

Verwendung von $\langle befehl \rangle$ passt nicht zu seiner Definition

Low-Level-Definitionen für Makros, die mit `\def` anstelle von `\newcommand` und dessen Verwandten vorgenommen werden, verwenden manchmal besondere Begrenzungszeichen für ihre Argumente (wie etwa `(...)` bei den Bildbefehlen). Handelt es sich bei $\langle befehl \rangle$ um einen $\text{\LaTeX}$ -Befehl, so sollte man seine Syntax überprüfen. Andernfalls liegt hier wahrscheinlich eine irreführende Fehlermeldung vor, die daher rührt, dass man in einem bewegten Argument einen zerbrechlichen Befehl ohne `\protect` verwendet hat.

`\usepackage before \documentclass``\usepackage vor \documentclass`

Die `\usepackage`-Deklaration darf erst verwendet werden, wenn man die Hauptklasse mit `\documentclass` geladen hat. Innerhalb einer Klassen- datei muss man stattdessen den Befehl `\RequirePackage` verwenden¹.

`UTF-8 string \u8:<8-bit-sequence> not set up for LaTeX use` inputenc

Der UTF-8-String `\u8:<8-Bit-Zeichenfolge>` ist für $\text{\LaTeX}$ nicht eingerichtet

Das mit UTF-8 `<8-Bit-Zeichenfolge>` benannte Unicode-Zeichen ist $\text{\LaTeX}$ unbekannt. Sofern es in einer im Dokument benutzten Fontkodierung verfügbar ist, muss es zunächst mithilfe der `\DeclareUnicodeCharacter`- Deklaration eingerichtet werden (siehe Abschnitt 7.11.3 auf Seite 456).

`\verb ended by end of line``\verb durch Zeilenende beendet`

Damit Fehler leichter ersichtlich sind, muss das Argument des Befehls `\verb` in einer einzigen Zeile stehen. Dieser Fehler zeigt also entweder an, dass man das schließende Begrenzungszeichen des Argumentes vergessen hat, oder dass das Argument in der Quelle auf mehrere Zeilen umbrochen wurde. Bei sehr langen Argumenten kann es helfen, diese auf mehrere `\verb`-Befehle zu verteilen und Zeilenumbrüche in der Quelldatei gegebenenfalls mit einem `%`-Zeichen zu maskieren.

`\verb illegal in command argument``\verb im Befehlsargument unzulässig`

Abgesehen von ganz besonderen Umständen (die in diesem Buch ausdrücklich erwähnt werden), darf der Befehl `\verb` (ebenso wie `verbatim`) nicht in Argumenten anderer Befehle vorkommen. Wortwörtliche Ausgabe an solchen Stellen erzielt man beispielsweise mithilfe von `\SaveVerb` und `\UseVerb` des in Abschnitt 3.4.3 beschriebenen Paketes `fancyvrb`.

`You already have nine parameters` TeX

Es sind bereits neun Argumente vorhanden

$\text{\LaTeX}$ unterstützt Umgebungs- beziehungsweise Befehlsdefinitionen mit höchstens neun Argumenten. Der vorliegende `\newenvironment`- oder `\newcommand`-Befehl verfügt aber über zehn oder mehr.

`You can't use 'macro parameter #' in <some> mode` TeX

Der „Makroparameter #“ ist in Modus `<modus>` nicht zulässig

$\text{\TeX}$ ist irgendwo auf ein einzelnes `#`-Zeichen gestoßen, das nicht auf ein Argument eines Befehls zu verweisen scheint. Zum Setzen dieses Zeichens muss man den Befehl `\#` verwenden.

`You can't use '\spacefactor' in vertical mode` TeX

Der `\spacefactor`-Befehl lässt sich nicht im vertikalen Modus verwenden

$\text{\TeX}$ erlaubt nur beim Erstellen horizontaler Listen Verweise auf den Befehl `\spacefactor`. Man erhält diesen Fehler, wenn man den $\text{\LaTeX}$ -Befehl

¹Rein technisch ist es zwar möglich ein Paket mithilfe von `\RequirePackage` vor einer Klasse zu laden, man sollte es jedoch vermeiden, solange man nicht ganz genau weiß, was man da tut.

`\@` außerhalb eines Absatzes benutzt. Da viele interne Befehle mit einem `@` im Namen beginnen, kann dieser Fehler auch auftreten, wenn man solche Befehle (wie etwa `\@startsection`) in der Dokumentenpräambel benutzt, ohne sie zwischen `\makeatletter` und `\makeatother` zu setzen. In diesem Falle sieht $\text{\TeX}$ ein `\@`, dem die Buchstaben `startsection` folgen. Bei der späteren Verwendung dieses Programmcodes wird dann der Befehl `\@` ausgeführt, wodurch wiederum diese Fehlermeldung auftritt.

You can't use '`\prevdepth`' in horizontal mode $\text{\TeX}$

Die Länge „`\prevdepth`“ kann nicht im horizontalen Modus verwendet werden

Die Länge `\prevdepth` lässt sich nur im vertikalen Modus (also zwischen Absätzen) verwenden.

You can't use '`\end`' in internal vertical mode $\text{\TeX}$

Der Befehl `\end` ist im internen vertikalen Modus nicht zulässig

Hier handelt es sich um eine der eher irreführenden Fehlermeldungen, da sie sich auf den $\text{\TeX}$ -Basisbefehl beruft, der einen $\text{\TeX}$ -Lauf beendet. Dieser wurde von $\text{\LaTeX}$ als Ende-Tag für Umgebungen umdefiniert. Der Fehler weist eigentlich darauf hin, dass die $\text{\LaTeX}$ -Anweisung `\end{document}` oder der Befehl `\stop` angetroffen wurde, während $\text{\LaTeX}$ gerade eine Box aufbaute. Er würde zum Beispiel durch `\begin{figure}...\stop` hervorgerufen.

You can't use '`\langle command \rangle`' in `\langle some \rangle` mode $\text{\TeX}$

`\langle befehl \rangle` ist in Modus `\langle modus \rangle` nicht zulässig

$\text{\TeX}$ reklamiert, dass der fragliche `\langle befehl \rangle` in einem seiner Modi nicht zulässig ist. Einige spezielle Varianten dieses Themas wurden bereits besprochen. Falls der `\langle befehl \rangle` nicht direkt verwendet wurde, ist die Ursache sehr wahrscheinlich ein zerbrechlicher Befehl in einem bewegten Argument.

You haven't defined output directory for '`\langle path \rangle`' `\docstrip`

Für „`\langle pfad \rangle`“ wurde kein Ausgabeverzeichnis definiert

Die Konfigurationsdatei `docstrip.cfg` enthält eine `\BaseDirectory`-Deklaration, aber für den internen `\langle pfad \rangle` im `\DOCSTRIP`-Skript liegt keine Übersetzung in ein lokales Verzeichnis vor. Mithilfe eines der Befehle `\DeclareDirectory` oder `\UseTDS` lässt sich in der Datei `docstrip.cfg` eine entsprechende Übersetzung festlegen. Das Verfahren ist in Abschnitt 14.2.3 auf Seite 863 beschrieben.

You haven't defined the language `\langle language \rangle` yet `\babel`

Die Sprache `\langle sprache \rangle` wurde noch nicht definiert

Verschiedene Befehle der Benutzerschnittstelle des `\babel`-Paketes überprüfen, ob es sich bei ihrem Argument um eine Sprache handelt, die beim Laden von `\babel` in der Optionsliste enthalten war. War die betreffende `\langle sprache \rangle` dort nicht aufgeführt, dann wird die Verarbeitung gestoppt und diese Fehlermeldung erscheint.

You haven't specified a language option `\babel`

Es wurde keine Sprachoption angegeben

Diese Meldung wird angezeigt, wenn für das `\babel`-Paket keine bekannte

Sprache angegeben wurde – also weder in der `babel`-Optionsliste noch in der Liste der globalen Optionen. Die Ursache ist normalerweise ein Schreibfehler. Es ist jedoch zu erwarten, dass bei der Verarbeitung des Dokumentes noch viele weitere Fehler auftreten.

B.1.1 Tod durch zu wenig Speicher

Das $\text{\TeX}$ -Programm enthält eine Reihe interner Tabellen mit fester Größe, die dem Speichern von zur Laufzeit benötigten Daten dienen. Sobald eine dieser Tabellen überläuft, unterbricht $\text{\LaTeX}$ die Verarbeitung mit der Fehlermeldung „`TeX capacity exceeded`“ ($\text{\TeX}$ -Kapazität überschritten).

Bis Mitte der 90er konnten Speicherprobleme tatsächlich durch die Größe eines Dokumentes hervorgerufen werden. In manchen Fällen ließ sich ein Dokument nicht als Ganzes verarbeiten.¹ Diese Begrenzungen gibt es heute nicht mehr oder zumindest nicht mehr in diesem Maße. Zum einen sind durchschnittliche $\text{\TeX}$ -Installationen heutzutage bereits mit sehr großzügigen Tabellen ausgestattet; zum anderen lassen sich die Tabellengrößen der meisten Versionen über Konfigurationsdateien ändern, ohne dass man $\text{\TeX}$ manuell neu kompilieren müsste. In einigen Fällen muss man möglicherweise ein neues $\text{\LaTeX}$ -Format erzeugen; genaueren Aufschluss hierüber bietet die Dokumentation der verwendeten $\text{\TeX}$ -Distribution.²

Trotzdem begegnet man diesem hässlichen Fehler hin und wieder, normalerweise aufgrund einer fehlerhaften Befehlsdefinition. Es folgen vier Kandidaten, die zum Besprechen des vorliegenden Themas auf das Wesentliche reduziert wurden. In der Praxis lauern derartige Probleme gewöhnlich in eher komplexeren Definitionen.

```
\newcommand\FAILa{.\FAILa}           \newcommand\FAILb{\FAILb x}
\newcommand\FAILc{\typeout{.}\FAILc} \newcommand\FAILd{.\par\FAILd}
```

Führt man `\FAILa` wie in diesem Beispiel definiert aus, dann erhält man nach kurzer Zeit die folgende Ausgabe (die Größe des gemeldeten Speichers kann dabei unterschiedlich sein):

```
! TeX capacity exceeded, sorry [main memory size=1500001].
\FAILa ->.
\FAILa
```

Der Hauptspeicher (main memory) ist der Teil von $\text{\TeX}$, in dem Makrodefinitionen und das Material für die aktuelle Seite abgelegt sind. Ein Blick auf die zuvor formulierte rekursive Definition macht deutlich, dass diese eine unendliche Folge von Punkten erzeugt. Da der Absatzumbruch zurückgestellt

¹Die erste Ausgabe dieses Buches erforderte eine speziell kompilierte $\text{\TeX}$ -Version, bei der alle derartigen Tabellen auf das 10fache vergrößert waren, und konnte nur auf einer großen UNIX-Workstation verarbeitet werden.

²Die diesem Buch beiliegende $\text{\TeX}$ Live-Distribution ermöglicht eine Änderung der meisten Tabellengrößen mithilfe der Konfigurationsdatei `texmf.cnf`. Das $\text{\TeX}$ Live-Handbuch enthält nähere Einzelheiten hierzu.

wird, bis $\text{\TeX}$ auf einen $\text{\par}$ -Befehl oder eine Leerzeile stößt, um die Zeilenumbrüche global optimieren zu können, wartet $\text{\TeX}$ hier vergebens auf eine Gelegenheit, das Absatzmaterial in Zeilen zu umbrechen.

Ein nicht ausreichender Speicher aufgrund zu zahlreicher Makrodefinitionen ist heutzutage eher unwahrscheinlich. Trotzdem kann es (theoretisch) auch dazu kommen, wenn dieser Speicher klein ist und man viele Pakete lädt, eine Vielzahl von größeren Gleitobjekten aufgelaufen ist oder man Makropakete¹ verwendet, die ihrerseits im laufenden Betrieb neue Makros erzeugen.

Ein solcher Fall liegt möglicherweise vor, wenn der Fehler nur bei größeren Dokumenten auftritt und $\text{\LaTeX}$ die Arbeit erst nach einigen Seiten einstellt. Dann kann man mithilfe des Befehls $\text{\tracingstats=2}$ in der Dokumentenpräambel prüfen, ob der Speicherbedarf plötzlich rapide ansteigt oder sich der Speicher allmählich füllt. Letzteres würde auf ein Problem mit der Tabellengröße hindeuten. $\text{\TeX}$ meldet dann nach jeder erstellten Seite den Hauptspeicherstatus mit in etwa folgender Ausgabe:

```
[765]
Memory usage before: 4262&161788; after: 1286&157691; still untouched: 1323176
[766]
Memory usage before: 3825&160983; after: 636&156520; still untouched: 1323176
[767]
Memory usage before: 3652&160222; after: 771&156307; still untouched: 1323176
```

Die gemeldete Zahl links des $\&$ -Zeichens gibt den Speicher an, der großen Objekten wie etwa Boxen gewidmet ist; die Zahl rechts nennt den von Makrodefinitionen und Zeichendaten verbrauchten Speicher. Man sollte also erwarten, dass beide Werte kleiner werden, sobald eine Seite fertig gestellt ist (also die jeweiligen after: -Werte). Wird der Wert rechts jedoch allmählich immer größer, dann fügt wahrscheinlich irgendetwas immer mehr Definitionen hinzu.

Mit $\text{\FAILb}$ wird eine andere Tabelle überfüllt. Hier erfolgt die Rekursion, bevor $\text{\LaTeX}$ das Makro vollständig expandiert hat, so dass der noch nicht verarbeitete Teil der Expansion gespeichert werden muss.

```
! TeX capacity exceeded, sorry [input stack size=1500].
\FAILb ->\FAILb
x
```

Aufgrund der Größe heutiger Stapelspeicher für die Eingabe (input stacks) erfolgt diese Fehlermeldung gewöhnlich nur, wenn eine Rekursion wie die hier definierte dafür sorgt, dass der Stapel mit beängstigender Geschwindigkeit wächst. In normalen $\text{\LaTeX}$ -Dokumenten stößt man selten auf verschachtelte Definitionen, die diesen Stapel auf einen Wert über 50 wachsen lassen (der Maximalwert für dieses Buch betrug 35).

Was geschieht, wenn man $\text{\FAILc}$ oder $\text{\FAILd}$ ausführt? Beide ähneln $\text{\FAILa}$, bringen jedoch keine interne $\text{\TeX}$ -Tabelle zum Überlaufen. Stattdessen schreiben beide einfach die Festplatte voll. Die einzige Aktivität von

¹Das Paket `varioref` beispielsweise definiert intern zwei Label für jeden benutzten $\text{\vref}$ -Befehl. In umfangreichen Dokumenten kann dies zu einem erheblichen Speicherverbrauch führen.

\FAILc besteht darin, Punkte auf dem Bildschirm und in der Protokolldatei anzuzeigen, so dass die Festplatte ganz allmählich mit einer riesigen Protokolldatei gefüllt wird. Die Definition von \FAILd hingegen enthält einen \par-Befehl, so dass hier Absätze gesetzt werden können (die jeweils aus einem einzelnen Punkt bestehen). Dadurch werden in schneller Folge Absätze erzeugt. Ein solches Experiment auf dem Computer des Verfassers endete mit einem 22279 Seiten langen Dokument und der folgenden Meldung:

```
tex: fwrite: No space left on device
```

Auf dem eigenen Rechner ist das eigentlich einfach nur ärgerlich und lässt sich schnell beheben. Auf Systemen mit gemeinsam genutzten Ressourcen sollte man jedoch sehr vorsichtig sein, wenn man $\text{\LaTeX}$ unbeaufsichtigt arbeiten lässt. Ein Fehler dieser Art hat einmal einen Studenten ziemlich hart getroffen: Er verarbeitete ein solches Dokument als Batchjob ohne Zeit- oder Größengrenzung auf einem Großrechner und erhielt daraufhin für die Verarbeitungszeit auf dem Computer eine Rechnung über mehrere tausend Dollar.

Im Prinzip können noch einige andere interne Tabellen überlaufen. Im Folgenden werden die noch nicht besprochenen Tabellen aufgeführt, mit einer Erklärung, wodurch ein Überlauf am ehesten ausgelöst werden könnte. Einige weitere Informationen lassen sich in [87, S. 300] nachlesen.

buffer size (Puffergröße) die Zeichen der Zeilen, die aus einer Datei gelesen werden; da der vorgegebene Wert normalerweise ziemlich hoch ist, liegt ein Überlauf am ehesten an Zeilenumbrüchen, die bei der fehlerhaften Konvertierung einer Datei von einem Betriebssystem zu einem anderen verloren gegangen sind. Ein Pufferüberlauf kann auch von einigen Textverarbeitungsprogrammen auf PCs verursacht werden, die intern einen ganzen Absatz in eine einzelne Zeile schreiben, auch wenn der Text auf dem Bildschirm in mehrere Zeilen umbrochen erscheint.

exception dictionary (Ausnahmewörterbuch) die Anzahl der mithilfe von \hyphenation festgelegten Ausnahmeregelungen für die Silbentrennung; in $\text{\LaTeX}$ sind einige Ausnahmen für die englische Sprache definiert und einige Sprachpakete legen weitere Ausnahmen fest. Um diese Tabelle zum Überlaufen zu bringen, muss man allerdings schon sehr gründlich sein.

font memory (Zeichensatzspeicher) die von $\text{\LaTeX}$ geladenen Fontmetrikdaten; heutzutage ist ein Überlauf dieser Tabelle äußerst unwahrscheinlich. Sollte es trotzdem dazu kommen, dann hat $\text{\LaTeX}$ zu viele Fonts geladen – wahrscheinlich, weil viele unterschiedliche Schriftgrade verwendet wurden und $\text{\LaTeX}$ für all diese Größen Mathematikfonts berechnet und geladen hat. Soweit möglich sollte man hier die Tabelle vergrößern. Oder man kann in Kapitel 7 nachlesen, wie sich die Anzahl der Fonts verringern lässt.

grouping levels (Gruppierungsebenen) die Anzahl nicht beendeter Gruppen, welche den Geltungsbereich für das Setzen von Parametern, Definitionen und anderen Elementen begrenzen; dazu gehören etwa geschweifte Klammern, der Beginn von Umgebungen oder Begrenzungen für den

Formelmodus. Ein Überlauf an dieser Stelle deutet normalerweise auf einen Programmierfehler hin (wie z.B. auf eine Definition, die mehr Gruppen öffnet, als sie schließt). Es ist manchmal schwierig, diese Art von Fehlern zu finden. Eine gute Unterstützung bietet hier das Programm eTeX ¹ mit dem Befehl `\showgroups`, der eine Liste der verschachtelten Gruppen anzeigt, beginnend mit der Innersten. Platziert man ihn beispielsweise in der Fußnote auf der aktuellen Seite, so erzeugt er folgende Meldung:

```
### semi simple group (level 3) entered at line 2955 (\begingroup)
### insert group (level 2) entered at line 2955 (\insert0{)
### semi simple group (level 1) entered at line 2921 (\begingroup)
### bottom level
```

Die Meldung „semi simple group on level 1“ rührt daher, dass dieser Text in einer `description`-Umgebung gesetzt ist (der Befehl `\begin` ruft intern einen `\begingroup`-Befehl auf). Der `\footnote`-Befehl wird mithilfe des TeX -Basisbefehls `\insert` ausgeführt, der die zweite Ebene („level 2“) beiträgt. Tatsächlich erzeugt der Befehl `\footnote` noch eine weitere Gruppe („level 3“), durch die sichergestellt wird, dass sich Farbbänderungen nur lokal auswirken.

Dieses Beispiel zeigt, dass die Beziehungen zwischen den Dokumentenbefehlen auf der oberen Ebene und den internen Gruppen alles andere als offensichtlich oder einfach sind. Die Zeilennummern, welche den Beginn der jeweiligen Gruppe anzeigen, sind jedoch durchaus hilfreich, da weitreichende Gruppen eher selten sind.

Alternativ bietet eTeX den internen Zähler `\tracinggroups`. Hat dieser Zähler einen positiven Wert, so werden der Beginn und das Ende von Gruppen in der Protokolldatei nachgehalten; wenn auch der Befehl `\tracingonline` einen positiven Wert hat, erscheinen die Daten zusätzlich auf dem Bildschirm.

hash size (Hash-Größe) die Anzahl der TeX bekannten Befehlsnamen; die meisten Pakete bringen eine feste Anzahl an Befehlsnamen mit sich. Jedoch erzeugt jeder `\label`- oder `\bibitem`-Befehl in einem Dokument einen neuen internen Befehlsnamen. Pakete (wie z.B. `varioref`), die intern den `\label`-Befehl verwenden, können in großen Dokumenten also erheblich dazu beitragen, dass diese Tabelle aufgefüllt wird.

number of strings (Anzahl der Strings) die Anzahl der Zeichenfolgen – Befehlsnamen, Dateinamen und eingebaute Fehlermeldungen –, die TeX sich merken muss; in manchen Fällen kann TeX nicht mehr benötigten Platz wieder frei geben, aber normalerweise bleiben solche Strings erhalten, selbst wenn sie nur lokal benutzt werden. Ein möglicher Grund für ein Überlaufen dieser Tabelle ist das Verwenden zu vieler Dateien in einer Anwendung. Jedes Öffnen einer Datei zum Lesen oder Schreiben schlägt sich in ihr nieder, selbst wenn es sich mehrmals um die gleiche Datei handelt.

¹Bei neueren Distributionen arbeitet $\text{L}^{\text{A}}\text{TeX}$ automatisch mit eTeX . Bei älteren Installationen muss man möglicherweise ein anderes Programm (z.B. `elatex` anstelle von `latex`) aufrufen, um ein Dokument zu verarbeiten.

Aus historischen Gründen benutzt $\TeX$ ein etwas ungewöhnliches Verfahren zum Umgang mit Strings, das mehrere Tabellen einschließt, von denen jede einzelne überlaufen kann. Vergrößert man die `hash size`-Werte, um mehr Befehle zuzulassen, so muss man daher möglicherweise auch die `number of strings`- sowie sehr wahrscheinlich die `pool size`-Einstellungen anpassen und umgekehrt.

`parameter stack size` (Größe des Argumentstapels) die Gesamtzahl der Befehlsargumente verschachtelter Befehle, die bereits expandiert, aber noch nicht völlig abgearbeitet wurden; ruft beispielsweise ein Befehl mit 4 Argumenten einen Befehl mit 5 Argumenten auf, welcher wiederum einen Befehl mit 3 Argumenten aufruft, dann werden auf diese Weise 12 Plätze in dieser Tabelle belegt. Sobald $\TeX$ das Ende eines Ersetzungstextes für ein Makro erreicht, wird der Stack wieder freigegeben. Daher lässt sich diese Grenze bei den heutigen Implementierungen kaum erreichen, es sei denn, man verwendet eine fahrige rekursive Definition mit Argumenten, wie beispielsweise die folgende:

```
\newcommand\FAIL[3]{\typeout{Habe #1, #2 und #3 aber
\FAIL ist voll daneben }\DO}
```

Problem erkannt? Da die `\typeout`-Anweisung fälschlicherweise den Befehl `\FAIL` enthält, wird dieser erneut aufgerufen, bevor der Ersetzungstext vollständig abgearbeitet werden kann. (Dabei übernimmt er `i`, `s` und `t` als Argumente.) Folglich wird der Befehl `\DO` niemals ausgeführt und man erhält schließlich die Fehlermeldung:

```
! TeX capacity exceeded, sorry [parameter stack size=1500].
\FAIL #1#2#3->
\typeout {Habe #1, #2 und #3 aber \FAIL ist voll daneben}\DO
1.18 \FAIL 123
```

Dieser Fall ähnelt dem `\FAILb`-Beispiel von Seite 957, nur dass hier der `parameter stack` zuerst überläuft, da drei Argumente beteiligt sind.

`pattern memory` (Musterspeicher) der für Silbentrennungsmuster verfügbare Speicher; diese Tabelle kann nicht während der Verarbeitung eines Dokumentes überlaufen, da die entsprechenden Muster nur beim Erzeugen eines Formates geladen werden. Tritt der Fehler in diesem Prozess auf, dann sollte man die Anzahl der Sprachen reduzieren, für die Silbentrennungsmuster in das Format geladen werden. Die entsprechenden Einstellungen sind heutzutage normalerweise in der Datei `language.dat` definiert.

`pool size` (Poolgröße) die in Strings – Befehlsnamen und Dateinamen (bei manchen Versionen mit vollständigem Pfad) – enthaltenen Zeichen; der Grund für ein Überlaufen dieser Tabelle sind meistens zu viele Dateien, insbesondere, wenn diese über lange absolute Pfade verfügen. Das kann beispielsweise geschehen, wenn ein Dokument viele Graphiken enthält und $\LaTeX$ mit `\graphicspath` angewiesen wird, diese nur in verschiedenen Verzeichnissen zu suchen. Jeder Versuch, eine Datei zu öffnen, wirkt sich auf diesen Stringpool aus.

`save size` (Größe des Speichers für Wiederherstellungsinformationen) Sammlung von Werten, die wiederhergestellt werden müssen, wenn eine Gruppe endet; auch dieser Speicher lässt sich bei den heutzutage voreingestellten Grenzwerten nur schwer zum Überlaufen bringen. Die wahrscheinlichste Ursache liegt bei gleichzeitigen lokalen und globalen Zuweisungen für das gleiche Objekt. Dies lässt sich nur durch eine Low-Level-Programmierung in $\text{\TeX}$ hervorrufen, da $\text{\LaTeX}$ -Zuweisungen immer entweder lokaler (meistens) oder globaler (z.B. bei Zählern) Natur sind.

Um ein unnötiges Anwachsen des `save stack` zu vermeiden, ist die `document`-Umgebung auf besondere Weise implementiert¹: Sie erzeugt im Gegensatz zu herkömmlichen Umgebungen keine eigene Gruppe. Andernfalls würde jede neue Definition unnötigerweise einen Wert auf dem `save stack` ablegen – unnötig deshalb, weil mit dem Ende dieser Gruppe die Verarbeitung des Dokumentes beendet ist.

`semantic nest size` (semantische Verschachtelungstiefe) die Anzahl der gleichzeitig in Verarbeitung befindlichen Tokens: Boxen, Formeln und andere Elemente eröffnen jedes Mal eine neue Liste und unterbrechen dabei die Arbeit am bisherigen Element. Sobald ein Element beendet ist, muss $\text{\TeX}$ sich wieder dem nächsten zurückgestellten Objekt zuwenden und merkt sich dazu alle noch nicht vollständig verarbeiteten Objekte im `semantic nest`-Stack.

Bei einem voreingestellten Wert von mehreren hundert Objekten ist es sehr schwierig, sich dieser Grenze in einem normalen Dokument auch nur anzunähern². Zur Not verfügt $\text{\TeX}$ über den Befehl `\showlists`, der alle noch unvollendeten Listen anzeigt, an denen $\text{\TeX}$ gerade arbeitet.

`text input levels` (Eingabequellen-Ebenen) die Anzahl der gleichzeitig geöffneten Eingabequellen (zum Beispiel mit `\include`, `\input` oder `\usepackage` geöffnete Dateien); bei der $\text{\TeX}$ -Version der Autoren müsste man 1500 Dateien verschachteln um diese Grenze zu erreichen.

B.2 Warnungen und informelle Meldungen

Während Fehlermeldungen dazu führen, dass $\text{\LaTeX}$ anhält und auf Anweisungen seitens des Anwenders wartet, werden Warnungen nur auf dem Bildschirm und in der Protokolldatei ausgegeben, während die Verarbeitung weiterläuft. Soweit möglich gibt $\text{\LaTeX}$ auch die Zeile des Quelltextes an, in welcher die Warnung ausgelöst wurde.

Vor der eigentlichen Warnung steht „ $\text{\LaTeX}$ Warning:“ bzw. „ $\text{\LaTeX}$ Font Warning:“, sofern es sich um eine Warnung des $\text{\LaTeX}$ -Kerns handelt. Ansonsten werden das Paket bzw. die Klasse, welche die Warnung ausgeben, genannt: „Package *<name>* Warning:“ bzw. „Class *<name>* Warning:“. $\text{\TeX}$ -Warnungen wie „Overfull...“ geht kein Standardtext voraus.

Außer den Warnungen schreibt $\text{\LaTeX}$ noch informelle Meldungen, die nicht auf dem Bildschirm erscheinen, in die Protokolldatei. Um Warnungen

¹Als Nebeneffekt lässt sich die Anweisung `\begin{document}` in keiner anderen Umgebung verwenden, da sie sich nicht entsprechend der Gruppenstruktur verhält.

²Dem Verfasser ist keine problematische Definition eingefallen, die nicht zuerst einen der anderen Grenzwerte sprengen würde.

und informelle Meldungen besser voneinander unterscheiden zu können, sind die **Warnungen** in der folgenden alphabetischen Liste blau dargestellt.

Calculating math sizes for size $\langle text\ size\rangle$

Formelschriftgrade für den Schriftgrad $\langle textschriftgröße\rangle$ werden berechnet

L^AT_EX muss die richtigen Schriftgrade für Hoch- und Tiefstellungen schätzen, da es die entsprechenden Daten für die aktuelle $\langle textschriftgröße\rangle$ in den internen Tabellen nicht finden konnte. Dieser Meldung folgen normalerweise noch mehrere Warnungen über veränderte Schriftgrößen für die Berechnung der einzelnen Formelstile. Diese Situation kann eintreten, wenn man mithilfe des `\fontsize`-Befehls einen unüblichen Schriftgrad festlegt; Abschnitt 7.10.7 enthält weitere Informationen für den Fall, dass die Formeln seltsam aussehen sollten.

Checking defaults for $\langle encoding\rangle/\langle font\ shape\rangle$

Voreinstellungen für $\langle kodierung\rangle/\langle schriftform\rangle$ werden geprüft

Diese Meldung wird bei der Verarbeitung von `\begin{document}` in die Protokolldatei geschrieben, während L^AT_EX überprüft, ob vernünftige Ersetzungsvorgaben für die Kodierung $\langle kodierung\rangle$ vorliegen. Darauf folgt entweder „... okay“ oder eine Fehlermeldung, die erzeugt wird, wenn die mittels `\DeclareFontEncoding` angegebene $\langle schriftform\rangle$ gruppe L^AT_EX nicht bekannt ist.

Citation ‘ $\langle key\rangle$ ’ on page $\langle number\rangle$ undefined

Quellenverweis „ $\langle schlüssel\rangle$ “ auf Seite $\langle seitenzahl\rangle$ ist nicht definiert

Entweder wurde der im Argument von `\cite` oder `\nocite` aufgeführte $\langle schlüssel\rangle$ nicht mithilfe eines `\bibitem`-Befehls definiert, oder man benötigt einen weiteren L^AT_EX-Lauf (vielleicht auch noch einen B^BT_EX-Lauf) um ihn L^AT_EX bekannt zu machen. Auf letzteren Fall wird durch die zusätzliche Warnung „Label(s) may have changed...“ hingewiesen, die auf Seite 967 besprochen wird. Löst `\nocite` die Warnung aus, dann erscheint keine Seitenzahl.

Command $\langle name\rangle$ invalid in math mode

Befehl $\langle name\rangle$ im Formelmodus ungültig

Hier handelt es sich entweder um eine Warnung oder eine Fehlermeldung, die anzeigt, dass man im Formelmodus einen Befehl verwendet hat, der nur in normalem Text zulässig ist. Wenn man eine veraltete, aber noch gültige, Konstruktion verwendet, wird eine Warnung erzeugt.

Document Class: $\langle name\rangle$ $\langle date\rangle$ $\langle additional-info\rangle$

Dokumentenklasse: $\langle name\rangle$ $\langle datum\rangle$ $\langle zusatzinfo\rangle$

Diese Zeile wird von einem `\ProvidesClass`-Befehl im Programmcode der Dokumentenklasse erzeugt. Obwohl es sich nicht wirklich um eine Warnung handelt, erscheint sie sowohl auf dem Bildschirm als auch in der Protokolldatei. Wenn ein Dokument auf verschiedenen Installationen unterschiedliche Ausgaben erzeugt, dann sollte man die Meldungen „Document Class:“, „File:“ und „Package:“ darauf überprüfen, ob hier unterschiedliche Versionen vorliegen.

Empty ‘thebibliography’ environment

Leere „thebibliography“-Umgebung

Diese Warnung wird ausgegeben, wenn in einer thebibliography-Umgebung keine \bibitem-Befehle vorhanden sind. Sie ist häufig ein Hinweis darauf, dass ein Problem bei einem BibTeX-Lauf vorliegt. Möglicherweise konnte das BibTeX-Programm keinen einzigen Quellenverweis auflösen.

Encoding <name> has changed to <new name> for ...

Kodierung <name> für ... wurde zu <neuer name> geändert

Diese Warnung erfolgt, wenn in der Deklaration eines Symbolfonts für verschiedene Formellayouts unterschiedliche Kodierungen verwendet wurden. Dies kann zur Folge haben, dass die \DeclareMathSymbol-Befehle für diesen Symbolfont nicht in allen Formellayouts gültig sind.

(\end occurred <when>) TeX

(\end eingetreten <während>)

Diese Warnung erhält man am Ende eines Verarbeitungslaufs, wenn TeX der Meinung ist, dass \end{document} oder der Befehl \stop verfrüht erfolgte. Als Warnung ist diese Meldung leider irreführend, da sie auf den TeX-Basisbefehl \end verweist, der in L^AT_EX als Ende-Tag für Umgebungen umdefiniert wurde. Unter <während> erscheint einer der beiden folgenden Fälle:

inside a group at level <number>) TeX

innerhalb einer Gruppe auf Ebene <nummer>)

In diesem Falle endete der L^AT_EX-Lauf, obwohl einige Gruppen noch nicht geschlossen waren. Zu diesen Gruppen gehören auch explizite Klammern, die nicht geschlossen wurden (z.B. {\itshape...}), in Makros verwendete \bgroup- und \begingroup-Befehle ohne ihre jeweiligen Gegenstücke sowie nicht geschlossene Umgebungen im Quelltext. Der letztere Fall löst normalerweise zunächst eine passende L^AT_EX-Fehlermeldung aus (also „\begin{env} on...“), es sei denn man hat den Lauf mit \stop beendet; dann erfolgt keine Überprüfung auf nicht geschlossene Umgebungen.

when <condition> on line <line number> was incomplete) TeX

bei noch nicht vollständiger <bedingung> in Zeile <zeilennummer>)

Hier beendete L^AT_EX den Lauf, während eine Low-Level-Bedingung von TeX noch nicht abgeschlossen war. Da L^AT_EX-Dokumente nur Standardbefehle verwenden, sollte dieses Problem eigentlich nicht auftreten, außer man hat das Dokument innerhalb einer mit \include geladenen Datei beendet. Andernfalls weist diese Meldung wohl auf ein fehlerhaft programmiertes Paket hin. Man sollte anhand der <zeilennummer> die Stelle suchen, an welcher die Bedingung aufgerufen wurde, um den aufrufenden Befehl ausfindig zu machen. Dabei bezieht sich die <zeilennummer> möglicherweise nicht auf die aktuelle Datei – TeX verrät leider nicht den jeweiligen Dateinamen. Bei sehr schwierigen Fällen kann man versuchen, die genaue Ursache mittels

der erweiterten eTeX-Überwachungsfunktionen zu bestimmen: Wenn man den internen Zähler `\tracingifs` auf 1 setzt, wird die Ausführung verschachtelter Bedingungen in allen Einzelheiten protokolliert.

External font `<name>` loaded for size `<size>`

Externer Font `<name>` für Schriftgrad `<schriftgrad>` geladen

L^AT_EX hat die Anweisung, eine Schriftform in der Größe `<schriftgrad>` zu laden, ignoriert und stattdessen den externen Font `<name>` geladen. (Diese Meldung wird durch die Größenfunktion `fixed` ausgelöst.)

Faking `<command>` for font family `<name>` in TS1 encoding textcomp
`<befehl>` für die Schriftfamilie `<name>` in TS1-Kodierung wurde nachgeahmt

Die Glyphe `<befehl>` ist in der TS1-Kodierung der aktuellen Schriftfamilie nicht vorhanden. L^AT_EX hat sie deshalb irgendwie „nachgeahmt“. Dieses Verfahren wird z.B. für die `\texteuro`-Glyphe (€) verwendet, wenn diese nicht verfügbar ist. In Abschnitt 7.8.7 ist beschrieben, wie man ein echtes Euro-Zeichen erhält.

File '`<name>`' already exists on the system.
 Not generating it from this source

Die Datei „`<name>`“ ist auf dem System bereits vorhanden.
 Sie wird nicht aus dieser Quelle erzeugt

Diese Warnung wird von einer `filecontents`-Umgebung ausgelöst, wenn die Datei `<name>` bereits irgendwo im Suchpfad von L^AT_EX existiert. Um die Datei trotzdem zu entpacken, muss man die von L^AT_EX gefundene Version entweder löschen oder umbenennen oder die Datei mithilfe eines Editors manuell extrahieren.

File: `<name>` `<date>` `<additional-info>`

Datei: `<name>` `<datum>` `<zusatzinfo>`

Diese Zeile wird von einem `\ProvidesFile`-Befehl erzeugt, der verwendet wird, um eine Datei und das Datum ihrer letzten Änderung zu kennzeichnen. Üblicherweise beginnt `<zusatzinfo>` mit einer Versionsnummer. Das ist jedoch nicht zwingend erforderlich. Obwohl diese Informationen genauso wichtig sind, wie diejenigen des `\ProvidesClass`-Befehls, werden sie nur in die Protokolldatei geschrieben, um den Bildschirm nicht mit Meldungen zu überfluten. Wenn ein Dokument auf verschiedenen Installationen unterschiedliche Ausgaben erzeugt, dann sollte man die Meldungen „Document Class:“, „File:“ und „Package:“ darauf überprüfen, ob hier unterschiedliche Versionen vorliegen.

File: `<encoding>``<family>`.fd `<date>` `<additional-info>`

Datei: `<kodierung>``<familie>`.fd `<datum>` `<zusatzinfo>`

Dieser wichtige Sonderfall des vorigen Hinweises besagt, dass eine Fontdefinitionsdatei für eine bestimmte Kombination von `<kodierung>` (die normalerweise kleingeschrieben wird) und `<familie>` geladen wurde. Solche Dateien enthalten Deklarationen für Schriftformgruppen und wurden in Abschnitt 7.10.6 beschrieben.

Float too large for page by $\langle value \rangle$ *Gleitobjekt um $\langle wert \rangle$ zu groß für die Seite*

Ein Gleitobjekt ist für die aktuelle, mit `\textheight` festgelegte Höhe des Textbereiches um $\langle wert \rangle$ zu groß. Es wird (sofern zulässig) auf eine eigene Seite gesetzt und erstreckt sich dort möglicherweise in den unteren Rand. Sollte das Gleitobjekt nicht auf einer eigenen Seite platziert werden dürfen, dann verhindert es dadurch, dass weitere Gleitobjekte der gleichen Klasse gesetzt werden können.

Font shape $\langle font\ shape \rangle$ in size $\langle size \rangle$ not available*Schriftform $\langle schriftform \rangle$ in Schriftgrad $\langle schriftgrad \rangle$ nicht verfügbar*

L^AT_EX gibt diesen Hinweis bei dem Versuch aus, eine Schrift auszuwählen, für welche die gewünschte Kombination von Zeichensatzattributen nicht vorliegt und für die in den internen Tabellen eine Fontersetzung definiert ist. Je nach dem Inhalt dieser Tabellen erfolgt eine der folgenden zusätzlichen Meldungen:

external font $\langle name \rangle$ used*externer Font $\langle name \rangle$ verwendet*

L^AT_EX hat in diesem besonderen Fall den externen Font $\langle name \rangle$ ausgewählt und weiß nicht, zu welcher Schriftformgruppe er gehört. (Diese Meldung wird durch die Größenfunktion `subf` ausgelöst.)

size $\langle schriftgrad \rangle$ substituted*Schriftgrad $\langle schriftgrad \rangle$ ersetzt*

L^AT_EX hat die richtige Form ausgewählt, aber da der gewünschte Schriftgrad nicht zur Verfügung steht, wurde die nächste Größe $\langle schriftgrad \rangle$ verwendet. Diese Auswahl erfolgt automatisch, wenn keine passende einfache Größe bzw. kein entsprechender Größenbereich in der Deklaration für die $\langle schriftform \rangle$ gruppe vorliegt.

shape $\langle font\ shape \rangle$ tried*Schriftform $\langle schriftform \rangle$ ausprobiert*

L^AT_EX hat eine andere $\langle schriftform \rangle$ gruppe ausgewählt, da die gewünschte nicht in der angeforderten Größe vorliegt. (Diese Meldung wird durch die Größenfunktion `sub` ausgelöst.)

Font shape $\langle font\ shape \rangle$ undefined. Using ' $\langle other\ shape \rangle$ ' instead*Schriftform $\langle schriftform \rangle$ nicht definiert. Verwende stattdessen „ $\langle andere\ form \rangle$ “*

Diese Warnung erfolgt, wenn eine Kombination von Zeichensatzattributen angegeben wurde, für die L^AT_EX keine Schriftformdefinition vorliegt. So würde zum Beispiel der Befehl `\fontseries{b}\ttfamily` normalerweise diese Warnung auslösen, da Computer Modern-Fonts weder über fette noch über breitfette Schreibmaschinenschriften verfügen. Wird die letztere Kombination angefordert, so erhält man statt dieser Warnung jedoch nur einen Hinweis in der Protokolldatei, da die `.fd`-Dateien über eine explizite Ersetzungsregel für `\textbf{\texttt{. .}}` verfügen.

Wenn L^AT_EX auf ein bestimmtes Symbol trifft, das nicht im gewünschten Schriftschnitt gesetzt werden kann, dann folgt der betreffenden Warnung noch der Text „for symbol $\langle name \rangle$ “ (für Symbol $\langle name \rangle$).

Font shape $\langle font\ shape \rangle$ will be scaled to size $\langle size \rangle$

Schriftform $\langle schriftform \rangle$ wird auf Schriftgrad $\langle schriftgrad \rangle$ skaliert

L^AT_EX konnte den angeforderten Zeichensatz durch Skalieren auf die gewünschte Größe laden. Um ein Dokument mit größenangepassten Fonts drucken zu können, muss der Druckertreiber über entsprechende Zeichensätze in den passenden Größen verfügen oder fähig sein, diese automatisch zu skalieren.

Foreign command $\langle command \rangle$;

amsmath

$\backslash frac$ or $\backslash genfrac$ should be used instead

Fremder Befehl $\langle befehl \rangle$;

stattdessen sollten $\backslash frac$ oder $\backslash genfrac$ verwendet werden

Auch wenn der Einsatz des Befehls $\langle befehl \rangle$ nicht falsch ist, wird doch nachdrücklich davon abgeraten, diese alte Form für (verallgemeinerte) Brüche in L^AT_EX zu verwenden. Stattdessen sollte man mit den amsmath-Befehlen arbeiten.

Form feed has been converted to Blank Line

Seitenvorschub wurde in Leerzeile umgewandelt

Die Umgebung `filecontents` hat ein „Seitenvorschub“-Zeichen (^^L) in der Quelle entdeckt und schreibt dieses als Leerzeile (die von L^AT_EX als $\backslash par$ -Befehl interpretiert würde) in die externe Datei. Da `filecontents` zum Verteilen von Textdaten entwickelt wurde, lassen sich mit der Umgebung keine beliebigen Binärdateien verarbeiten.

'h' float specifier changed to 'ht' or

'!h' float specifier changed to '!ht'

Positionsargument „h“ für Gleitobjekte wurde zu „ht“ geändert oder

Positionsargument „!h“ für Gleitobjekte wurde zu „!ht“ geändert

Zum Positionieren eines Gleitobjektes wurde nur `h` oder `!h` ohne jede weitere Option angegeben. L^AT_EX benötigt eine Alternative, wenn die Platzierung „hier“ nicht möglich sein sollte, da auf der aktuellen Seite nicht mehr genug Platz ist. Will man Gleitobjekte wirklich vom Gleiten abhalten, so sollte man das in Abschnitt 6.3.1 beschriebene float-Paket in Betracht ziehen.

Ignoring text $\langle text \rangle$ after $\backslash end\{\langle env \rangle\}$

Text „ $\langle text \rangle$ “ hinter $\backslash end\{\langle env \rangle\}$ wird ignoriert

Diese Warnung wird von der `filecontents`- oder der `filecontents*`-Umgebung ausgelöst, wenn sich hinter dem $\backslash end$ -Tag noch Textmaterial befindet.

Label $\langle key \rangle$ multiply defined

Label „ $\langle schlüssel \rangle$ “ mehrfach definiert

Das Dokument enthält mindestens zwei $\backslash label$ -Befehle mit gleichem $\langle schlüssel \rangle$. Verweise auf diesen $\langle schlüssel \rangle$ beziehen sich immer auf den zuletzt definierten $\backslash label$ -Befehl. Man muss sicherstellen, dass alle $\langle schlüssel \rangle$ unterschiedlich sind.

Label(s) may have changed. Rerun to get cross-references right

Label(s) können sich geändert haben. Um korrekte Querverweise zu erhalten, ist ein weiterer Lauf erforderlich.

L^AT_EX hat entdeckt, dass sich die Labeldefinitionen gegenüber dem vorherigen Lauf geändert haben, so dass (mindestens) ein weiterer Verarbeitungslauf erforderlich ist, um die Querverweise korrekt aufzulösen.

Theoretisch kann diese Meldung immer wieder auftauchen, egal wie viele L^AT_EX-Läufe durchgeführt werden.¹ In diesem Falle muss man die .aux-Dateien verschiedener Läufe vergleichen, um herauszufinden, welches Label zwischen verschiedenen Zuständen hin- und herwechselt, und das Problem dann manuell beheben.

Loose \hbox (badness <number>) <somewhere> 

\hbox mit relativ viel Weißraum (Badness <zahl>) <irgendwo>

T_EX hat eine horizontale Box mit einem höheren Badness-Wert als 13 erzeugt (was bedeutet, dass mindestens 50% ihrer verfügbaren Dehnbarkeit in Anspruch genommen wurde). Wer kein Perfektionist ist, kann diese Warnung einfach ignorieren; eigentlich wird sie überhaupt erst ausgegeben, wenn man die Voreinstellung für \hbadness geändert hat. Im Zusammenhang mit der Meldung „Underfull \hbox...“ auf Seite 973 werden nähere Einzelheiten erläutert.

Loose \vbox (badness <number>) <somewhere> 

\vbox mit relativ viel Weißraum (Badness <zahl>) <irgendwo>

T_EX hat eine vertikale Box mit einem höheren Badness-Wert als 13 erzeugt (was bedeutet, dass mindestens 50% ihrer verfügbaren Dehnbarkeit in Anspruch genommen wurde). Diese Warnung erfolgt nur, wenn \vbadness auf einen Wert unter 100 gesetzt wurde. Im Zusammenhang mit der Meldung „Underfull \vbox...“ auf Seite 974 werden nähere Einzelheiten erläutert.

Making <char> an active character 

<zeichen> wird in ein aktives Zeichen umgewandelt

Diese Meldung wird für jedes Zeichen, das in ein Kurzzeichen umgewandelt wird, in die Protokolldatei geschrieben. Diese Information kann hilfreich sein, wenn ein Dokument anders aussieht als erwartet, sofern die Ursache dafür der versehentliche Gebrauch eines Kurzzeichens ist.

Marginpar on page <number> moved

Marginalie auf Seite <zahl> verschoben

Ein \marginpar-Befehl konnte nicht bündig zu der Zeile, mit der er ursprünglich verknüpft war, ausgegeben werden, da dieser Platz bereits von einer vorhergehenden Marginalie belegt ist.

¹ Sollte beispielsweise ein \label-Befehl direkt an einem Seitenumbruch zwischen Seite „iii“ und „iv“ stehen, so kann ein vorhergehender \pageref-Befehl dazu führen, dass das Label auf Seite „iv“ gelangt, wenn der Querverweistext die Seitenzahl „iii“ enthält und umgekehrt.

Missing character: There is no $\langle char \rangle$ in font $\langle name \rangle$! TeX

Fehlendes Zeichen: $\langle zeichen \rangle$ ist in Zeichensatz $\langle name \rangle$ nicht vorhanden!

Auf diese
Meldung in der
Protokolldatei Acht
geben!

Obwohl diese Meldung normalerweise auf ein ernstes Problem hinweist, wird sie leider nur in die Protokolldatei geschrieben (es sei denn der Wert für `\tracingonline` ist positiv). Sie bedeutet, dass irgendwo in der Quelldatei ein Symbol $\langle zeichen \rangle$ angefordert wird, für das an der entsprechenden Position im aktuellen Zeichensatz ($\langle name \rangle$ ist der externe Name) keine Glyphe existiert. Das angezeigte $\langle zeichen \rangle$ kann auf unterschiedlichen T_EX-Installationen variieren¹. Diese Warnung kann z.B. durch einen `\symbol`-Befehl ausgelöst werden, da man mit seiner Hilfe jede beliebige Position in einem Zeichensatz ansteuern kann. Die in Abschnitt 7.11.4 auf Seite 467 besprochenen fontkodierungsspezifischen Befehle sollten diese Warnung jedoch nie erzeugen.

No `\author` given

`\author`-Befehl fehlt

Man hat den Befehl `\maketitle` benutzt, ohne zuvor einen Autor anzugeben. Im Gegensatz zu einem fehlenden `\title`-Befehl erzeugt dieses Versäumnis eine Warnung.

No auxiliary output files

Keine Hilfs-Ausgabedateien

Diese Meldung wird angezeigt, wenn man in der Dokumentenpräambel den Befehl `\nofiles` verwendet. Die Meldung ist etwas irreführend, denn es werden lediglich keine *neuen* Hilfsdateien geschrieben – existierende werden aber benutzt.

No characters defined by input encoding change to $\langle name \rangle$

Beim Wechsel der Eingabekodierung auf $\langle name \rangle$ wurden keine Zeichen definiert

Die Eingabekodierungsdatei $\langle name \rangle$.def enthält anscheinend keine Deklarationen für die Eingabekodierung. Bei der `ascii`-Kodierung ist das zu erwarten; bei allen anderen Kodierungen deutet es auf ein Problem hin.

No file $\langle name \rangle$

Datei $\langle name \rangle$ nicht vorhanden

L^AT_EX gibt diese Warnung immer dann aus, wenn eine Hilfsdatei (z.B. mit der Endung `.aux` oder `.toc`) gelesen werden soll, diese aber nicht zu finden ist. Das wird nicht als Fehler betrachtet, da diese Dateien erst beim ersten Verarbeitungslauf erzeugt werden. Der Befehl `\include` benutzt jedoch die gleiche Routine, so dass eine fehlende „include“-Datei leider auch diese scheinbar harmlose Warnung auslöst.

No hyphenation patterns were loaded for the language ' $\langle language \rangle$ ' babel

Für die Sprache „ $\langle sprache \rangle$ “ wurden keine Trennmuster geladen

Alle Sprachdefinitionsdateien überprüfen, ob die Silbentrennungsmuster

¹Manchmal wird etwas wie `^^G` ausgegeben, manchmal werden echte Zeichen angezeigt. Leider gibt es keine Garantie, dass sie wirklich dem entsprechen, was man eingegeben hat: Es können je nach Betriebssystem Übersetzungen erfolgen, wenn die Zeichen in die Protokolldatei geschrieben werden.

für die ausgewählte Sprache in das L^AT_EX-Format geladen wurden. Wurden sie nicht geladen, so erfolgt diese Meldung und es wird ein voreingestellter Satz von Trennmustern verwendet. Dabei handelt es sich um die Muster, die in das Musterregister 0 geladen wurden (typischerweise diejenigen für amerikanisches Englisch).

No input encoding specified for $\langle language \rangle$ language babel

Für die Sprache $\langle sprache \rangle$ wurde keine Eingabekodierung angegeben

Diese Warnung erfolgt, wenn im Dokument eine der unterstützten Sprachen zum Setzen des kyrillischen Alphabets benötigt und keine besondere Eingabekodierung festgelegt wurde. Für diese Sprachen gibt es mehrere gängige Eingabekodierungen, so dass die Sprachdefinitionsdatei darauf besteht, dass die verwendete Kodierung explizit genannt werden muss.

No positions in optional float specifier. Default added ...

Optionales Positionsargument enthielt keine Parameter.

Voreingestellte Positionierungsdaten hinzugefügt ...

Eine Gleitobjektumgebung (wie etwa `figure` oder `table`) wurde mit optionalem Positionsargument verwendet, das jedoch keine verwertbaren Informationen enthält. L^AT_EX hat deshalb die voreingestellten Regeln für die Positionierung benutzt.

Oldstyle digits unavailable for family $\langle name \rangle$ textcomp

Mediävalziffern für Schriftfamilie $\langle name \rangle$ nicht verfügbar

Der Befehl `\oldstylenums` wurde für eine Schriftfamilie verwendet, die keine Mediävalziffern enthält. Als Notlösung hat L^AT_EX stattdessen die regulären Zahlen (der aktuellen Schriftfamilie) benutzt. Abschnitt 7.5.4 enthält nähere Einzelheiten hierzu.

Optional argument of `\twocolumn` too tall on page $\langle number \rangle$

Optionales Argument von `\twocolumn` auf Seite $\langle zahl \rangle$ zu hoch

Das Material im optionalen Argument des `\twocolumn`-Befehls hat eine so große Höhe, dass weniger als drei Zeilen auf der Seite verbleiben. L^AT_EX beginnt daher nicht auf der aktuellen Seite mit dem Zweispaltensatz, sondern erzeugt stattdessen eine neue Seite.

`\oval`, `\circle`, or `\line` size unavailable

`\oval`, `\circle`- oder `\line`-Größe nicht verfügbar

Die genannten Befehle sind nicht in der gewünschten Größe verfügbar. L^AT_EX wird die nächstmögliche Größe auswählen. Ein Weg, wie man dieses Problem vermeiden kann, wurde beispielsweise in Abschnitt 10.4.3 aufgezeigt.

Overfull `\hbox` ($\langle number \rangle$ pt too wide) $\langle somewhere \rangle$ T_EX

Überlaufende `\hbox` ($\langle irgendwo \rangle$) ($\langle zahl \rangle$ pt zu breit)

T_EX wurde gezwungen, eine horizontale Box in einer bestimmten Breite zu erzeugen (z.B. für eine Zeile in einem Absatz oder einen `\makebox`-Befehl). Dabei war es nicht möglich, das vorhandene Material auf die vorgegebene Breite zu reduzieren, selbst nachdem der gesamte verfügbare Freiraum so weit wie möglich gestaucht worden war. Daher wird

das betreffende Material rechts in den Rand hineinragen. Meistens ist das recht auffällig, auch wenn es sich nur um einen geringen Überlauf handelt. Dieses Problem muss manuell behoben werden, da $\TeX$ es nicht lösen konnte (die Abschnitte 3.1.11 und B.3.3 enthalten Ratschläge zu diesem Thema). Bei der Warnung „Underfull $\hbox{...}$ “ auf Seite 973 sind eine Liste und Erläuterungen der möglichen Ursprünge des Problems (also des $\langle\textit{irgendwo}\rangle$) aufgeführt.

Overfull $\vbox$ ($\langle\textit{number}\rangle\text{pt too wide}$) $\langle\textit{somewhere}\rangle$ $\square_{\mathbb{X}}$

Überlaufende $\vbox$ ($\langle\textit{irgendwo}\rangle$) ($\langle\textit{zahl}\rangle\text{pt}$ zu breit)

$\TeX$ sollte eine vertikale Box fester Größe erzeugen (z.B. für einen $\parbox$ -Befehl oder eine `minipage`-Umgebung mit einem zweiten optionalen Argument; siehe Anhang A.2.2 auf Seite 899) und hat mehr Material vorgefunden, als hineinpasste. Das überschüssige Material wird unten aus der Box herausragen. Hier hängt es ganz von den Umständen ab, ob dieses Ergebnis ein Problem darstellt. Bei der Warnung „Underfull $\vbox...$ “ auf Seite 974 sind eine Liste und Erläuterungen der möglichen Ursprünge des Problems (also des $\langle\textit{irgendwo}\rangle$) aufgeführt.

Overwriting encoding scheme $\langle\textit{something}\rangle$ defaults

Kodierungsschema-Voreinstellungen für $\langle\textit{etwas}\rangle$ werden überschrieben

Diese Warnung wird von dem Befehl `\DeclareFontEncodingDefaults` ausgegeben, wenn er zuvor deklarierte Voreinstellungen für „text“ oder „math“ (Formeln) überschreibt.

Overwriting $\langle\textit{something}\rangle$ in version $\langle\textit{name}\rangle ...$

$\langle\textit{etwas}\rangle$ in Formellayout $\langle\textit{name}\rangle$ wird überschrieben ...

Eine Deklaration wie `\SetSymbolFont` oder `\DeclareMathAlphabet` hat die Zuordnung von Schriftformen zu $\langle\textit{etwas}\rangle$ (einem Zeichensatz oder einem Mathematikalphabet) im Formellayout $\langle\textit{name}\rangle$ überschrieben.

Package: $\langle\textit{name}\rangle$ $\langle\textit{date}\rangle$ $\langle\textit{additional-info}\rangle$

Paket: $\langle\textit{name}\rangle$ $\langle\textit{datum}\rangle$ $\langle\textit{zusatzinfo}\rangle$

Diese Zeile wird von einem `\ProvidesPackage`-Befehl erzeugt, der verwendet wird, um ein Paket und das Datum seiner letzten Änderung zu kennzeichnen. Üblicherweise beginnt $\langle\textit{zusatzinfo}\rangle$ mit einer Versionsnummer. Das ist jedoch nicht zwingend erforderlich. Obwohl diese Informationen genauso wichtig sind, wie diejenigen des `\ProvidesClass`-Befehls, werden sie nur in die Protokolldatei geschrieben, um den Bildschirm nicht mit Meldungen zu überfluten. Wenn ein Dokument auf verschiedenen Installationen unterschiedliche Ausgaben erzeugt, dann sollte man die Meldungen „Document Class:“, „File:“ und „Package:“ darauf überprüfen, ob hier unterschiedliche Versionen vorliegen.

Redeclaring font encoding $\langle\textit{name}\rangle$

Fontkodierung $\langle\textit{name}\rangle$ wird umdeklariert

Diese Warnung wird ausgegeben, wenn man eine bereits vorhandene Fontkodierung mithilfe des Befehls `\DeclareFontEncoding` erneut deklariert (wodurch möglicherweise ihre Voreinstellungen verändert werden).

Redeclaring math accent $\langle name \rangle$

Akzentzeichen für Formeln $\langle name \rangle$ wird umdeklariert

Diese Warnung wird ausgegeben, wenn man ein bereits vorhandenes Akzentzeichen für Formeln mithilfe des Befehls `\DeclareMathAccent` erneut deklariert. Sofern der zu deklarierende Befehl schon existiert, es sich aber nicht um einen Akzent handelt, erhält man stattdessen eine Fehlermeldung.

Redeclaring math alphabet $\langle name \rangle$

Mathematikalphabet $\langle name \rangle$ wird umdeklariert

Der $\langle name \rangle$ wurde mithilfe eines `\DeclareMathAlphabet`- oder eines `\DeclareSymbolFontAlphabet`-Befehls als Alphabetbefehl deklariert. Dabei handelte es sich jedoch bereits um einen Alphabetbefehl. Die neue Deklaration überschreibt alle bisherigen Einstellungen für $\langle name \rangle$.

Redeclaring math symbol $\langle name \rangle$

Mathematisches Symbol $\langle name \rangle$ wird umdeklariert

Der Befehl $\langle name \rangle$ war bereits als mathematisches Symbol deklariert und die neue Deklaration überschreibt die bisherige Definition.

Redeclaring math version $\langle name \rangle$

Formellayout $\langle name \rangle$ wird umdeklariert

Ein bereits deklariertes Formellayout wird mithilfe der Deklaration `\DeclareMathVersion` bearbeitet. Die neue Deklaration überschreibt alle bisherigen Einstellungen für dieses Layout mit den vorgegebenen Werten.

Redeclaring symbol font $\langle name \rangle$

Symbolfont $\langle name \rangle$ wird umdeklariert

Ein bereits deklarierter Symbolfont wird mithilfe der Deklaration `\DeclareSymbolFont` bearbeitet. Die neue Deklaration überschreibt den Symbolfont in allen bekannten Formellayouts.

Reference ' $\langle key \rangle$ ' on page $\langle number \rangle$ undefined

„ $\langle schlüssel \rangle$ “ für Querverweis auf Seite $\langle zahl \rangle$ nicht definiert

Ein mit `\ref`, `\pageref` oder einem der anderen in Kapitel 2 besprochenen Querverweisbefehle erzeugter Verweis benutzt einen $\langle schlüssel \rangle$, zu dem $\text{\LaTeX}$ keinen entsprechenden `\label`-Befehl gefunden hat. Sofern das Label im Dokument vorhanden ist, benötigt man lediglich einen weiteren Verarbeitungslauf, um es in $\text{\LaTeX}$ bekannt zu machen. Auf diesen Fall wird durch die zusätzliche Warnung „Label(s) may have changed...“ hingewiesen, die auf Seite 967 besprochen wurde.

Size substitutions with differences up to $\langle size \rangle$ have occurred

Es sind Schriftgrößenersetzungen mit einem Unterschied von bis zu $\langle größe \rangle$ erfolgt

Diese Meldung erscheint am Ende eines Laufs, wenn $\text{\LaTeX}$ mindestens einen deutlich abweichenden Schriftgrad auswählen musste, da die gewünschte Größe nicht zur Verfügung stand. Die $\langle größe \rangle$ gibt die höchste benötigte Abweichung an.

Some font shapes were not available, defaults substituted

*Einige Schriftschnitte waren nicht verfügbar;
sie wurden entsprechend der Vorgaben ersetzt*

Diese Meldung erscheint am Ende eines Laufs, wenn $\text{\LaTeX}$ einige Schriftschnitte automatisch ersetzen musste.

Tab has been converted to Blank Space

Tabulator wurde in Leerzeichen umgewandelt

Die `filecontents`-Umgebung hat ein „Tabulator“-Zeichen (`^^I`) in der Quelldatei entdeckt und wird dieses als Leerzeichen in die externe Datei schreiben. Da `filecontents` zum Verteilen von Textdaten entwickelt wurde, lassen sich mit der Umgebung keine beliebigen Binärdateien verarbeiten.

Text page <number> contains only floats

Textseite <zahl> enthält ausschließlich Gleitobjekte

Ein oder mehrere Gleitobjekte, die am Kopf („top“) oder Fuß („bottom“) einer Seite erscheinen sollten, sind zusammen so groß, dass auf der aktuellen Seite nur sehr wenig Platz (weniger als zwei Zeilen) für Text verbleibt. Daher hat $\text{\LaTeX}$ entschieden, auf der fraglichen Seite nur Gleitobjekte zu platzieren, auch wenn einige oder alle der Gleitobjekte eine solche Positionierung nicht explizit erlauben. Diese Meldung kann nur auftreten, wenn die Positionsparameter für Gleitobjekte gegenüber den voreingestellten Werten erheblich verändert wurden. Nähere Einzelheiten zu diesem Thema finden sich am Anfang von Kapitel 6.

There were multiply-defined labels

Es sind mehrfach definierte Label aufgetreten

Diese Warnung erscheint am Ende eines Verarbeitungslaufs, wenn $\text{\LaTeX}$ auf mindestens ein Paar `\label-` oder `\bibitem-`Befehle mit identischen Schlüsseln gestoßen ist. Man muss die Protokolldatei überprüfen und dafür sorgen, dass nur unterschiedliche Schlüssel verwendet werden.

There were undefined references

Es sind nicht definierte Verweise aufgetreten

Diese Meldung erscheint am Ende eines Verarbeitungslaufs, wenn $\text{\LaTeX}$ Verweise auf unbekannte Schlüssel entdeckt und gefolgert hat, dass diese sich nicht durch einen weiteren Lauf auflösen lassen würden. Man sollte die Protokolldatei auf die Meldungen „Reference <key> undefined“ und „Citation <key> undefined“ überprüfen und die Ursachen beheben – entweder durch Korrigieren etwaiger Schreibfehler oder durch Hinzufügen der erforderlichen `\label-` oder `\bibitem-`Befehle. Bei fehlenden <schlüssel>n für Quellenverweise benötigt man möglicherweise nur einen erneuten $\text{\LaTeX}$ - und danach einen $\text{\LaTeX}$ -Lauf.

Tight \hbox (badness <number>) <somewhere> $\text{\TeX}$

\hbox mit relativ wenig Weißraum (Badness <zahl>) <irgendwo>

$\text{\TeX}$ hat eine horizontale Box erzeugt und musste die Leerräume ihres Inhaltes stauchen. Diese Meldung erscheint nur, wenn man den Befehl

`\hbadness` auf einen Wert kleiner 100 gesetzt hat. Im Zusammenhang mit der Meldung „Underfull `\hbox...`“ auf dieser Seite werden nähere Einzelheiten erläutert.

Tight `\vbox` (`badness` *`<number>`*) *`<somewhere>`* TeX

*`\vbox` mit relativ wenig Weißraum (`Badness` *`<zahl>`*) *`<irgendwo>`**

TeX hat eine vertikale Box erzeugt und musste die Leerräume ihres Inhaltes stauchen. Diese Meldung erscheint nur, wenn man den Befehl `\vbadness` auf einen Wert kleiner 100 gesetzt hat. Im Zusammenhang mit der Meldung „Underfull `\vbox...`“ auf Seite 974 werden nähere Einzelheiten erläutert.

Try loading font information for *`<encoding>`*+*`<family>`*

Versuche Zeichensatzdaten für `<kodierung>`+`<familie>` zu laden

Diese Meldung findet sich jedes Mal dann in der Protokolldatei, wenn L^AT_EX nach einer `.fd`-Datei für die Kombination aus Kodierung und Familie *`<kodierung>`*+*`<familie>`* sucht.

Unable to redefine math accent *`<accent>`* amsmath

Mathematischer Akzent `<akzent>` lässt sich nicht umdefinieren

Diese Warnung kommt sehr selten vor. Sie kann jedoch erscheinen, wenn man das `amsmath`-Paket mit `Mathematicafonts` lädt, die nicht dem Standard entsprechen.

Underfull `\hbox` (`badness` *`<number>`*) *`<somewhere>`* TeX

*`\hbox` mit zu viel Weißraum (`Badness` *`<zahl>`*) *`<irgendwo>`**

TeX war gezwungen, eine horizontale Box (z.B. für eine Absatzeile oder einen `\makebox`-Befehl) mit einer bestimmten Breite zu erzeugen. Dabei musste der Weißraum in dieser Box über das vorgegebene Maß hinaus gedehnt werden. Das heißt, die Dehnung betrug an den erweiterbaren Abständen mehr als 100% der verfügbaren plus-Zugaben. Intern wird dies durch einen `Badness`-Grad ausgedrückt, der über 100 liegt. Ein Grad von 800 bedeutet, dass der Weißraum auf das Doppelte des erlaubten Wertes gedehnt werden musste, um die erforderliche Breite zu erreichen.¹

Ob eine solche unzureichend gefüllte Box tatsächlich ein merkliches Problem darstellt, muss man gegebenenfalls im erzeugten Dokument visuell prüfen. Bei einem `Badness`-Grad von 10000 kann die Box beliebig hässlich aussehen. Da der TeX-Wert für unendlich mit 10000 recht niedrig liegt, kann es vorkommen, dass eine einzelne extrem leere Zeile in Kauf genommen wurde, obwohl auch mehrere nicht gut gefüllte, aber doch noch akzeptable aufeinander folgende Zeilen möglich gewesen wären. In einem solchen Fall kann sich der Längenparameter `\emergencystretch` als hilfreich erweisen (siehe Abschnitt 3.1.11).

Ab welchem `Badness`-Grad eine solche Warnung erfolgt, wird mithilfe des ganzzahligen Parameters `\hbadness` gesteuert. In L^AT_EX ist der

¹Die genaue Formel lautet $\min(100r^3, 10000)$ wobei r das Verhältnis von „verwendeter Dehnung“ zu „verfügbarer Dehnung“ ist. Wenn allerdings ein unbegrenzt dehnbarer Abstand vorhanden ist (zum Beispiel aufgrund eines Befehls wie `\hfill`), beträgt der `Badness`-Grad null.

Wert 1000 voreingestellt, so dass nur vor wirklich unschönen Boxen gewarnt wird. Bei einem wichtigen Dokument kann man durchaus einen anspruchsvolleren Wert wie etwa `\hbadness=10` testen, um zu sehen, wie viele Zeilen $\TeX$ wirklich als unausgewogen erachtet.

Da die Warnung direkt von $\TeX$ erzeugt wird, meldet sie übrigens immer eine `\hbox`, egal welcher Boxbefehl tatsächlich im Quelldokument verwendet wurde. Der Ort, an dem das Problem aufgetreten ist, wird mit *<irgendwo>* angegeben und kann eine der folgenden vier Möglichkeiten sein:

detected at line *<line number>* $\TeX$

*in Zeile *<zeilennummer>* entdeckt*

Das Problem ist eine explizit konstruierte Box (sie endet im Quelldokument in Zeile *<zeilennummer>*). Es kann sich beispielsweise um einen `\makebox`-Befehl mit einem expliziten Argument für die Breite handeln oder um einen anderen $\LaTeX$ -Befehl, der Boxen erzeugt.

has occurred while *\output* is active $\TeX$

*trat bei aktivem *\output*-Befehl auf*

$\TeX$ war gerade damit beschäftigt, eine Seite aufzubauen, und stieß beim Hinzufügen lebender Kolumnentitel und ähnlichem auf das Problem. Da dieser Prozess asynchron verläuft, wird keine Zeilennummer genannt. Ein Blick auf die Seite, die am nächsten lag, als die Warnung ausgegeben wurde, zeigt, ob hier manuell korrigiert werden muss.

in alignment at lines *<line numbers>* $\TeX$

*in ausgerichteter Struktur in Zeilen *<zeilennummern>**

Die Box gehört zu einer `tabular`- oder einer ausgerichteten Formelumgebung. Die *<zeilennummern>* geben die Position der gesamten ausgerichteten Struktur im Quelldokument an, da $\TeX$ zu dem Zeitpunkt, an dem es auf das Problem stößt, bereits keine Möglichkeit mehr hat, dessen Ursprungsort genauer festzustellen.

in paragraph at lines *<line numbers>* $\TeX$

*in Absatz in Zeilen *<zeilennummern>**

Die „Underfull Box“ ist eine unausgewogene Zeile im genannten Absatz (bei den angegebenen *<zeilennummern>* im Quelldokument). Die zusätzliche symbolische Anzeige der fraglichen Zeile sollte dabei helfen, den genauen Ursprung des Problems ausfindig zu machen.

Underfull *\vbox* (*badness <number>*) *<somewhere>* $\TeX$

*\vbox mit zu viel Weißraum (*Badness <zahl>*) *<irgendwo>**

$\TeX$ war gezwungen, eine vertikale Box (z.B. für einen `\parbox`-Befehl oder eine `minipage`-Umgebung) mit einer bestimmten Höhe zu erzeugen. Dabei musste der Weißraum in dieser Box über das vorgesehene Maß hinaus gedehnt werden (siehe die Erläuterungen zu `Badness` und `Dehnbarkeit` in der Beschreibung zur Warnung „Underfull `\hbox`...“). Mit dem Befehl `\vbadness=<wert>` kann man alle Warnungen für `Badness`-Grade unter dem angegebenen Grenzwert unterdrücken. $\LaTeX$ gibt dann nur

noch Warnungen für Boxen mit einem höheren Badness-Grad als `<wert>` aus (voreingestellt ist der Wert 1000). Der Ursprungsort des Problems wird mit `<irgendwo>` angegeben und kann einer der folgenden sein:

detected at line `<line number>` TeX

in Zeile `<zeilennummer>` entdeckt

Die Box wurde explizit konstruiert (die `<zeilennummer>` verweist auf das Ende der Konstruktion) und innerhalb der Box steht nicht ausreichend dehnbarer Leerraum zur Verfügung. Das Beispiel

```
\parbox[c][2in][s]{4cm}{Test Test}
```

würde diese Warnung erzeugen, da die Box zwei Zoll hoch sein soll und die Inhalte diese Höhe ausfüllen sollen (Argument `[s]`), wobei jedoch kein dehnbarer Leerraum zur Verfügung steht, wie etwa `\par\vfill` zwischen den beiden Wörtern. In Anhang A.2.2 wurden nähere Einzelheiten zu Absatzboxen erläutert.

has occurred while `\output` is active TeX

trat bei aktivem `\output`-Befehl auf

In den meisten Fällen musste der Leerraum auf der aktuellen Seite nach Einschätzung von TeX über ein annehmbares Maß hinaus gedehnt werden. Ob sich dieses Problem auch wirklich visuell auswirkt, hängt von vielerlei Faktoren ab, z.B. davon, welche Art von Leerräumen auf der Seite vorhanden ist. So ist etwa ein größerer Abstand oberhalb einer Überschrift weniger nachteilig als eine zu weit gedehnte Liste. Darum empfiehlt es sich, solche Seiten manuell zu überprüfen. Häufig sind hier die Befehle `\enlargethispage` oder `\pagebreak` hilfreich.

Sofern dieses Problem ungewöhnlich häufig auftritt, sollte man prüfen, ob die Längenparameter für Listen, Absätze und Überschriften zu starr eingestellt sind (siehe die Kapiteln 2 bis 4). Außerdem sollte man überprüfen, ob der Längenparameter `\textheight` einer ganzzahligen Anzahl von Textzeilen entspricht (vgl. die Erläuterungen auf Seite 206).

in alignment at lines `<line numbers>` TeX

in ausgerichteter Struktur in Zeilen `<zeilennummern>`

In Standard-L^AT_EX sollte diese Warnung nicht auftreten. Sie kann jedoch in einigen speziellen Anwendungen erscheinen. In einem solchen Fall kann man anhand der `<zeilennummern>` den Ursprungsort im Quelldokument identifizieren.

Unused global option(s): `[<option-list>]`

Nicht verwendete globale Option(en): `[<optionsliste>]`

Einige der mit `\documentclass` deklarierten Optionen wurden weder von der Klasse noch von irgendeinem Paket in der Präambel verwendet. Sehr wahrscheinlich wurden die Optionsnamen falsch geschrieben. Man sollte auch bedenken, dass einige Pakete keine globalen sondern

nur explizit beim Laden des Paketes angegebene Optionen verarbeiten. Nähere Erläuterungen hierzu finden sich in Anhang A.4.

Writing file ‘ $\langle name \rangle$ ’

Datei „ $\langle name \rangle$ “ wird geschrieben

Diese informelle Meldung wird sowohl von `filecontents` als auch von `filecontents*` erzeugt, wenn sie ihren Inhalt in eine externe Datei $\langle name \rangle$ schreiben.

Writing text ‘ $\langle text \rangle$ ’ before $\backslash end\{\langle env \rangle\}$ as last line of $\langle file \rangle$

Text „ $\langle text \rangle$ “ vor $\backslash end\{\langle env \rangle\}$ wird als letzte Zeile der $\langle datei \rangle$ geschrieben

Diese Warnung wird von der `filecontents`- oder der `filecontents*`-Umgebung ausgelöst, wenn sie direkt vor dem $\backslash end$ -Tag noch Textmaterial entdecken.

You have more than once selected the attribute ‘ $\langle attrib \rangle$ ’ babel for language $\langle language \rangle$

Das Attribut $\langle attrib \rangle$ für $\langle sprache \rangle$ wurde mehr als einmal ausgewählt

Diese Meldung wird angezeigt, wenn im zweiten Argument des Befehls $\backslash languageattribute$ mehrmals das gleiche Attribut erscheint.

You have requested $\langle package-or-class \rangle$ ‘ $\langle name \rangle$ ’, but the $\langle package-or-class \rangle$ provides ‘ $\langle alternate-name \rangle$ ’

$\langle paket-oder-klasse \rangle$ „ $\langle name \rangle$ “ wurde angefordert, aber $\langle paket-oder-klasse \rangle$ stellt „ $\langle anderer-name \rangle$ “ bereit

Mithilfe eines der Befehle $\backslash usepackage$ oder $\backslash RequirePackage$ (im Falle eines Paketes) bzw. $\backslash documentclass$ oder $\backslash LoadClass$ (im Falle einer Klasse) sollte $\langle name \rangle$ geladen werden. Das Paket bzw. die Klasse stellt jedoch eine Variante des Originals mit dem internen Namen $\langle anderer-name \rangle$ zur Verfügung. Sofern es sich dabei nicht um einen Schreibfehler im Paket oder in der Klasse handelt, verfügt die gegebene Installation über eine Klassen- oder Paketvariante, die sich wahrscheinlich anders verhalten wird als das Original. Demzufolge wird das Dokument möglicherweise anders formatiert als auf anderen Installationen. Ob dieses Verhalten richtig ist, lässt sich nur durch eine Untersuchung des fraglichen Paketes bzw. der Klasse feststellen.

Die Angabe eines relativen oder absoluten Pfadnamens löst im Nebeneffekt ebenfalls diese Warnung aus.

You have requested release ‘ $\langle date \rangle$ ’ of LaTeX, but only release ‘ $\langle old-date \rangle$ ’ is available

Die Version „ $\langle datum \rangle$ “ von LaTeX wurde angefordert, es steht jedoch nur die Version „ $\langle altes-datum \rangle$ “ zur Verfügung

Ein $\backslash NeedsTeXFormat$ -Befehl hat eine $\text{\LaTeX}$ -Version mit dem frühesten Datum $\langle datum \rangle$ angefordert. Es steht jedoch nur ein Format von $\langle altes-datum \rangle$ zur Verfügung. Mit einer solchen Forderung soll normalerweise sichergestellt werden, dass das $\text{\LaTeX}$ -Format über bestimmte Funktionen verfügt. Daher kann es sein, dass das betreffende Dokument im weiteren Verlauf einige zusätzliche Fehler erzeugt oder seltsame Formatierung enthält. Hier sollte man auf eine neuere $\text{\LaTeX}$ -Version umsteigen.

You have requested, on line $\langle num \rangle$, version ' $\langle date \rangle$ ' of $\langle name \rangle$, but only version ' $\langle old-date \rangle$ ' is available

In Zeile $\langle num \rangle$ wurde Version „ $\langle datum \rangle$ “ von $\langle name \rangle$ angefordert, es steht aber nur Version „ $\langle altes-datum \rangle$ “ zur Verfügung

Eine angeforderte Klasse oder ein Paket darf nicht älter sein als $\langle datum \rangle$, die vorliegende Version stammt jedoch von $\langle altes-datum \rangle$. Das fragliche Paket bzw. die Klasse muss aktualisiert werden.

B.3 T_EX und L^AT_EX-Tracingbefehle

In diesem Abschnitt werden Werkzeuge und Verfahren für das so genannte Tracing (Überwachen und Verfolgen von Programmabläufen) sowie zum Darstellen von Statusinformationen besprochen. Man kann sie beispielsweise nutzen, um herauszufinden, warum Leerräume auf einer Seite seltsam verteilt sind oder warum eine Befehlsdefinition nicht macht, was sie soll.

B.3.1 Befehlsdefinitionen und Registerwerte anzeigen

In vielen Fällen sind Informationen über den aktuellen internen Status von L^AT_EX, die genaue Definition von Befehlen oder die Werte von Registern sehr aufschlussreich. Gibt beispielsweise ein `\newcommand`-Befehl die Meldung aus, dass der zu definierende neue Befehl bereits vorhanden ist, dann möchte man vielleicht die vorhandene Definition überprüfen, damit man nicht versehentlich einen wichtigen Befehl umdefiniert.

T_EX verfügt zu diesem Zweck über den Befehl `\show`, welcher die Definition des ihm folgenden Elementes anzeigt, danach die Verarbeitung anhält und den Benutzer mit einem Fragezeichen auffordert, das weitere Vorgehen zu bestimmen. Beispielsweise kann man nach dem Definieren von `\xvec` im Beispiel A-1-4 auf Seite 877 die Definition des Befehls folgendermaßen anzeigen lassen:

Befehlsdefinitionen anzeigen

```
\newcommand\xvec[1]{\ensuremath{x_1,\ldots,x_{\#1}}}  
\show\xvec
```

Dadurch erhält man die folgenden Informationen auf dem Bildschirm und in der Protokolldatei:

```
> \xvec=\long macro:  
#1->\ensuremath {x_1,\ldots ,x_{\#1}}.  
1.6 \show\xvec  
  
?
```

In der ersten Zeile, die mit `>` beginnt, ist zu sehen, welches Element angezeigt wird (`\xvec`) und um welche Art von Element es sich handelt. `\xvec` ist demzufolge ein Makro, in dessen Argument `\par`-Befehle stehen dürfen. Das heißt, es wurde mit `\newcommand` und nicht mit `\newcommand*` definiert. Die zweite Zeile (bis `->`) gibt Aufschluss über die Argumentstruktur des Befehls: In diesem Falle verfügt der Befehl über ein Argument (`#1`). Man beachte, dass das

Argument in der `\newcommand`-Deklaration zwar als `[1]` angegeben wurde, hier jedoch anders dargestellt wird. Der Rest dieser und etwaige weitere Zeilen enthalten die Definition. Das Ende des Programmcodes wird durch einen Punkt gekennzeichnet, der nicht zur Definition gehört, jedoch gegebenenfalls (überflüssige) Leerzeichen am Ende der Definition sichtbar macht. Man beachte, dass der Code normalisiert dargestellt wird. So steht nach einem Befehl, der normalerweise auf ihn folgende Leerzeichen unterdrückt, immer ein Leerzeichen, unabhängig davon ob dieses in der Originaldefinition enthalten ist.

Nach der Definition wird die entsprechende Zeile des Quelldokumentes angezeigt (zusammen mit der Nummer der Zeile in der Eingabedatei). Danach hält $\text{\LaTeX}$ mit einem Fragezeichen an. Um fortzufahren drückt man die Eingabetaste. Durch Eingabe von `h` kann man sich alternativ die weiteren Möglichkeiten anzeigen lassen.

Nicht alle Befehle erzeugen eine so leicht verständliche Ausgabe. Angenommen man möchte einen Befehl ansehen, der mit einem optionalen Argument definiert wurde, wie etwa `\lvec` in Beispielen A-1-5 auf Seite 877:

```
\newcommand\lvec[2][n]{\ensuremath{\#2_1+\cdots + \#2_{\#1}}}  
\show\lvec
```

In diesem Falle erhält man das folgende Ergebnis:

```
> \lvec=macro:  
-> \@protected@testopt \lvec \\\lvec {n}.
```

Offenbar verfügt der Befehl `\lvec` über keinerlei Argumente (diese werden nämlich erst im Laufe der weiteren Verarbeitung hinzugenommen). Noch etwas anderes an dieser Ausgabe ist seltsam: Was bedeutet `\\lvec`? Handelt es sich um den Befehl `\\`, dem die Buchstaben `lvec` folgen oder liegt hier ein seltsamer Befehl `\\lvec` vor, zu dessen Name zwei Backslash-Zeichen gehören? Tatsächlich stimmt Letzteres, auch wenn es keine Möglichkeit gibt, dies anhand der Angaben des `\show`-Befehls festzustellen. Derartige seltsame Befehlsnamen, die Anwender nicht ohne Weiteres generieren können, werden manchmal intern von $\text{\LaTeX}$ verwendet, um aus existierenden Befehlsnamen mithilfe von `\csname` und `\endcsname` sowie weiteren Low-Level-Verfahren von $\text{\TeX}$ neue Befehlsnamen zu bilden.

*Interne Befehle mit
seltsamen Namen
anzeigen*

Wie kann man sich dann die Definition von `\\lvec` anzeigen lassen? Es sollte klar sein, dass `\show` vor einem solchen Befehl nicht weiterhilft, da $\text{\TeX}$ hier normalerweise nur `\\` liest und entsprechend annimmt, dass genau dieser Befehl angezeigt werden soll. Aus diesem Grund muss man zunächst eben diese Low-Level-Verfahren benutzen, um den Befehlsnamen in der Weise zu generieren, dass er von $\text{\TeX}$ als einzelnes Element betrachtet wird. Dieses Element kann man dann an `\show` übergeben:

```
\expandafter\show\csname \string\lvec \endcsname
```

Technisch gesehen wird hier aus dem, was zwischen `\csname` und `\endcsname` steht, ein Befehlsname erzeugt. Innerhalb des Konstruktes wandelt der Befehl `\string` den Befehl `\lvec` in eine Zeichenfolge um, die mit einem Backslash beginnt, der aber nicht mehr länger den Beginn eines Befehls anzeigt. Darum stehen am Anfang des erzeugten Befehlsnamens zwei Backslash-Zeichen. Der Befehl

`\expandafter` verschiebt die Ausführung des auf ihn folgenden `\show`-Befehls, damit zuerst `\csname` abgearbeitet wird, bevor dann `\show` das Ergebnis weiterverwendet.

Das ist eine ganze Menge Low-Level-T_EX, aber nach dieser Eingabe wird man mit dem folgenden Ergebnis belohnt:

```
> \lvec=\long macro:
[#1]#2->\ensuremath {#2_1+\cdots + #2_{#1}}.
<recently read> \lvec
```

Dieses Mal folgt dem Befehl keine Zeilennummer, sondern der Hinweistext `<recently read>` (kürzlich gelesen). Er besagt, dass T_EX das Element `\lvec` im Speicher zusammengesetzt und nicht direkt aus einer Datei gelesen hat.

Was geschieht, wenn man den Befehl `\expandafter` am Anfang des letzten Beispiels vergisst? Man erhält folgendes Ergebnis:

```
> \csname=\csname.
1.5 \show\csname
      \string\lvec \endcsname
?
! Extra \endcsname.
1.5 \show\csname \string\lvec \endcsname
?

```

Als Erstes erfolgt die Angabe, dass es sich bei `\csname` um den Befehl `\csname` handelt. Diese Information erscheint zunächst nutzlos. Tatsächlich bedeutet sie jedoch, dass `\csname` ein Basisbefehl oder ein Register ist, das fest zu T_EX gehört – im Gegensatz zu `\lvec`, bei dem es sich um ein mit `\newcommand` definiertes Makro handelt. L^AT_EX gibt zudem an, bis wohin die Eingabezeile gelesen wurde, indem es die nicht gelesenen Elemente (`\string` usw.) in die nächste Zeile schreibt. Da die Verarbeitung danach fortgesetzt wird, hält T_EX kurz darauf (nachdem es die ganze Zeile gelesen hat) wieder an und beklagt sich über einen falschen `\endcsname`-Befehl: Der passende `\csname`-Befehl wurde ja nur angezeigt und nicht ausgeführt.

*Einen Basisbefehl
ausfindig machen*

Der Befehl `\show` eignet sich als Informationsquelle zu Befehlen und ihren Definitionen oder um herauszufinden, ob es sich um T_EX-Basisbefehle handelt. Mit seiner Hilfe kann man jedoch nicht die aktuellen Werte von Längen- oder Zählerregistern auslesen. Beispielsweise erzeugt die Anweisung

*Registerwerte
anzeigen*

```
\show\parskip \show\topmargin \show\topsep
```

folgendes Ergebnis:

```
> \parskip=\parskip.
1.5 \show\parskip
      \show\topmargin \show\topsep
?
> \topmargin=\dimen73.
1.5 \show\parskip \show\topmargin
      \show\topsep
?
> \topsep=\skip23.
1.5 \show\parskip \show\topmargin \show\topsep
```

Aus diesen Angaben kann man schließen, dass `\parskip` ein $\TeX$ -Basisbefehl ist (dass es sich tatsächlich um eine elastische Länge handelt, wird verschwiegen), dass `\topmargin` eigentlich das `\dimen`-Register (feste Länge) mit der Registernummer 73 und `\topsep` das `\skip`-Register (elastische Länge) Nummer 23 ist.

Der Wert dieser Register lässt sich anstatt mit `\show` mittels eines anderen $\TeX$ -Basisbefehls namens `\showthe` anzeigen. Er erzeugt die folgende Ausgabe auf dem Bildschirm und beweist, dass es sich bei `\parskip` wirklich um eine elastische Länge handelt:

```
> 0.0pt plus 1.0pt.
1.5 \showthe\parskip
```

Auf diese Weise kann man mithilfe von `\showthe` die Werte anzeigen, die Längenregistern mittels `\newlength` zugewiesen wurden sowie diejenigen interner $\TeX$ -Register wie `\baselineskip` und `\tolerance`. Die Werte von $\LaTeX$ -Zählern, die durch den Befehl `\newcounter` zugewiesen wurden, lassen sich so jedoch nicht direkt anzeigen. Dazu muss man die $\LaTeX$ -Zähler zusätzlich mithilfe eines `\value`-Befehls in eine von `\showthe` akzeptierte Form bringen. Die Anweisung

```
\showthe\value{footnote}
```

beispielsweise würde den aktuellen Wert des Fußnotenzählers auf dem Bildschirm anzeigen.

*Befehlsdefinitionen
und Registerwerte
formatieren*

Anstatt die Bedeutung eines Makros oder den Wert eines Registers auf dem Bildschirm anzuzeigen, kann man diese Art von Daten auch mithilfe der Befehle `\meaning` anstelle von `\show` und `\the` anstelle von `\showthe` formatiert ausgeben. Die Ausgabe ist etwas anders gestaltet: `\meaning` zeigt nicht den Namen des Elementes, sondern nur seinen Typ und seine „Bedeutung“. Zum Vergleich mit den bisherigen Ausgaben hier ein Beispiel mit diesen Befehlen:

<pre>\long macro:#1->\ensuremath {x_1,\ldots ,x_{\#1}} 0.0pt plus 1.0pt 16.0pt 8.0pt plus 2.0pt minus 4.0pt footnote=0</pre>	<pre>\newcommand\xvec[1]{\ensuremath{x_1,\ldots ,x_{\#1}}}\n \ttfamily % Schreibmaschinenschrift \raggedright \meaning\xvec \par \the\parskip\par \the\topmargin \par \the\topsep \par footnote=\the\value{footnote}</pre>

Bsp.
B-3-1

Sollte das Anzeigen von Befehlsdefinitionen und Registerwerten zum Identifizieren eines Problems nicht ausreichen, so kann man als andere Möglichkeit das Verhalten der aktiven Befehle überwachen (siehe Abschnitt B.3.5 auf Seite 992).

B.3.2 Diagnose bei problematischen Seitenumbrüchen

Hin und wieder erzeugt $\LaTeX$ unerwartete Seitenumbrüche oder seltsame vertikale Abstände und man wüsste gerne, woher diese kommen und welche Län-

gen eigentlich genau daran beteiligt sind. Zu diesem Zweck bietet T_EX einige Low-Level-Tracingbefehle.

Symbolische Darstellung des Seiteninhalts

Durch den Befehl `\showoutput` an beliebiger Stelle im Dokument, gibt T_EX (ab der aktuellen Seite) symbolische Darstellungen ganzer Seiten auf dem Bildschirm und in der Protokolldatei aus. Das Ergebnis sind große Mengen von Daten. Als Beispiele werden hier Auszüge einer solchen Darstellung gezeigt, die durch separates Verarbeiten eines einzelnen Absatzes erzeugt wurden.

Jede Seite beginnt mit dem Text `Completed box being shipped out` (vollständige Box wird ausgegeben), gefolgt von der aktuellen Seitennummer in eckigen Klammern. Dann folgen viele Zeilen, in denen die Boxen, aus welchen die Seite besteht, wiedergegeben werden. Sie beginnen mit einem `\vbox`-Befehl (vertikale Box), der die ganze Seite umfasst, und ihren Maßen in pt. Um anzuzeigen, welcher Inhalt zu welcher Box gehört, werden die jeweiligen Inhalte mit Punkten anstelle von Leerzeichen rekursiv eingerückt. Abstände, auch solche die fest sind, werden durch das Schlüsselwort `\glue` („Leim“) angezeigt (siehe Zeile 3 und 6); in dehnbaren Abständen sind, wie noch gezeigt wird, immer Plus- und/oder Minusbestandteile enthalten. Ob es sich um horizontale oder vertikale Abstände handelt, ergibt sich aus der Art der Box, in der sie sich befinden. So ist der `\glue` in Zeile 3 mit dem Wert `16.0pt` beispielsweise ein vertikaler Abstand, der von `\topmargin` herrührt (vgl. Beispiel B-3-1 auf der gegenüberliegenden Seite). In dem Auszug sieht man außerdem einen leeren `\vbox`-Befehl der Höhe `12pt` (Zeile 5 bis 7): den Kolumnentitel. Ihm folgt in Zeile 8 der `\headsep`-Abstand (`25pt`) und in Zeile 10 beginnt dann die Box, welche den eigentlichen Textbereich der Seite enthält. In Zeile 15 und den folgenden Zeilen ist zu sehen, wie einzelne Zeichen dargestellt werden; `\T1/cmr/m/n/10` steht hier für den Font jedes einzelnen Zeichens. Der dazwischen angezeigte `\glue`-Befehl (z.B. in Zeile 19) kennzeichnet einen Wortzwischenraum mit seinen dehn- und stauchbaren Anteilen.

```

1 Completed box being shipped out [1]
2 \vbox(633.0+0.0)x407.0
3 .\glue 16.0
4 .\vbox(617.0+0.0)x345.0, shifted 62.0
5 ..\vbox(12.0+0.0)x345.0, glue set 12.0fil
6 ... \glue 0.0 plus 1.0fil
7 ... \hbox(0.0+0.0)x345.0
8 .. \glue 25.0
9 .. \glue(\lineskip) 0.0
10 .. \vbox(550.0+0.0)x345.0, glue set 502.00241fil
11 ... \write-{}
12 ... \glue(\topskip) 3.1128
13 ... \hbox(6.8872+2.15225)x345.0, glue set - 0.17497
14 .... \hbox(0.0+0.0)x15.0
15 .... \T1/cmr/m/n/10 0
16 .... \T1/cmr/m/n/10 n
17 .... \T1/cmr/m/n/10 c
18 .... \T1/cmr/m/n/10 e
19 .... \glue 3.33252 plus 1.66626 minus 1.11084

```



```

20 ....\T1/cmr/m/n/10 i
21 ....\T1/cmr/m/n/10 n
22 ....\glue 3.33252 plus 1.66626 minus 1.11084
23 ....\T1/cmr/m/n/10 a

```

Das zweite Beispiel zeigt die symbolische Seitendarstellung in der Nähe eines Zeilenumbruchs. Man sieht den Abstand, den $\TeX$ am rechten Ende einer Textzeile hinzufügt (`\rightskip` in Zeile 5) und die Box, welche die Zeile enthält. Daher erscheint Zeile 6 wieder ausgerückt. Der `\penalty`-Wert entscheidet hier, wie schwierig (oder einfach) es für $\TeX$ wäre, nach dieser Zeile die Seite zu umbrechen. Der im Beispiel gezeigte Wert leitet sich aus dem Parameter `\clubpenalty` ab.¹ In Zeile 7 wird zwischen den Zeilen ein vertikaler Abstand eingefügt. $\TeX$ errechnet diesen aus dem Wert von `\baselineskip` minus der Tiefe der vorhergehenden Zeilenbox und der Höhe der folgenden Zeilenbox, die in Zeile 8 beginnt.

```

1 ....\T1/cmr/m/n/10 s
2 ....\T1/cmr/m/n/10 o
3 ....\T1/cmr/m/n/10 m
4 ....\T1/cmr/m/n/10 e
5 ....\glue(\rightskip) 0.0
6 ... \penalty 150
7 ... \glue(\baselineskip) 2.96054
8 ... \hbox(6.8872+1.94397)x345.0, glue set 0.55421
9 ....\T1/cmr/m/n/10 s
10 ....\T1/cmr/m/n/10 t

```

Das letzte Beispiel zeigt einen Teil einer symbolischen Seitenausgabe, die aus einer Zeile, wie der folgenden erzeugt wird:

```
\begin{itemize} \item Test \end{itemize} \section{Test}
```

Besonders interessant ist dabei der Teil, der durch `\end{itemize}` und `\section{Test}` erzeugt wird. In den Zeilen 1 bis 7 ist hier eine merkwürdige Ansammlung von `\glue`-Anweisungen zu sehen, die sich größtenteils gegenseitig aufheben. Dazwischen sind einige Penalty-Werte eingestreut:

```

1 ... \penalty -51
2 ... \glue 10.0 plus 3.0 minus 5.0
3 ... \glue -10.0 plus -3.0 minus -5.0
4 ... \penalty -300
5 ... \glue 10.0 plus 3.0 minus 5.0
6 ... \glue -10.0 plus -3.0 minus -5.0
7 ... \glue 15.0694 plus 4.30554 minus 0.86108
8 ... \glue(\parskip) 0.0 plus 1.0
9 ... \glue(\baselineskip) 8.12001
10 ... \hbox(9.87999+0.0)x345.0, glue set 290.70172fil

```

¹Der Penalty-Wert für den Umbruch nach der ersten Zeile eines Absatzes ergibt sich aus dem ganzzahligen Parameter `\clubpenalty`, für den Umbruch vor der letzten Zeile aus `\widowpenalty`. Beide sind auf 150 voreingestellt, sollen einen Seitenumbruch also nur ein wenig erschweren.

Diese Zeilen werden durch verschiedene `\addpenalty`- und `\addvspace`-Befehle generiert. Die Zeilen 1 und 2 beispielsweise zeigen den Penalty-Wert und den elastischen Abstand, die durch `\end{itemize}` eingefügt werden. Der Befehl `\section` fügt dann einen Umbruchpunkt ein, wodurch er (mit einem `\secpenalty`-Wert von `-300`) signalisiert, dass sich die Stelle vor dem Abschnitt gut für einen Seitenumbruch eignet. Der Umbruch sollte eigentlich sogar vor dem `\glue`-Befehl in Zeile 2 erfolgen, weil ansonsten ein seltsamer Leerraum am Fuß der Seite verbleibt. Da es technisch nicht möglich ist, Material wieder von der Druckfahne zu entfernen, bedient sich `\addpenalty` eines Tricks: Der Befehl fügt zunächst einen negativen Abstand ein (Zeile 3), gefolgt von dem Penalty-Wert (Zeile 4), und gibt dann wieder den `\glue`-Befehl aus (Zeile 5). In den Zeilen 6 und 7 verwendet `\addvspace` die gleiche Methode, um einen vertikalen Abstand vor der Überschrift einzufügen.

T_EX fügt die Zeilen 8 und 9 hinzu, wenn der eigentliche Überschriftstext (Zeile 10) auf der Druckfahne positioniert wird. Man beachte, dass die Überschrift technisch als „Absatz“ betrachtet wird; daher kommt noch ein `\parskip`-Abstand hinzu. Aus diesem Grund sollte ein Vergrößern dieses Parameters sehr sorgfältig geplant werden. Das trifft auch für andere Parameter zu (wie etwa denjenigen, der in Zeile 7 eingefügt wird).

Der Befehl `\showoutput` erzeugt auch symbolische Darstellungen überlaufender Boxen. Die Überwachung endet dabei am Ende der aktuellen Umgebung oder der nächsten schließenden Klammer. Um also die Ausgabe ganzer Seiten zu sehen, muss man dafür sorgen, dass der Seitenumbruch erfolgt, bevor die nächste Gruppe endet.

*Nebeneffekt von
`\showoutput`*

Seitenumbrüche überwachen

T_EX bietet auch Daten zum Überwachen von Seitenumbrüchen. Um sie anzeigen zu lassen, muss man den internen Zähler `\tracingpages` auf eine positive ganze Zahl setzen:

```
\tracingonline=1 \tracingpages=1
```

Durch den positiven Wert für `\tracingonline` werden die Überwachungsdaten nicht nur in die Protokolldatei (Vorgabe) sondern auch auf dem Bildschirm ausgegeben.

Wenn man den vorigen Absatz, der mit „`\TeX{}` bietet auch...“ beginnt, als eigenständiges Dokument verarbeitet, erzeugt das Seiten-Tracing die folgenden Zeilen:

```
1 %% goal height=522.0, max depth=4.0
2 % t=10.0 g=522.0 b=10000 p=150 c=100000#
3 % t=22.0 g=522.0 b=10000 p=0 c=100000#
4 % t=55.0 plus 4.0 g=522.0 b=10000 p=-51 c=100000#
5 % t=77.0 plus 8.0 g=522.0 b=10000 p=300 c=100000#
6 % t=89.0 plus 8.0 g=522.0 b=10000 p=0 c=100000#
7 % t=90.94397 plus 8.0 plus 1.0fil g=522.0 b=0 p=-10000 c=-10000#
```

Die mit zwei Prozentzeichen beginnende erste Zeile gibt die anvisierte Seitenhöhe an („goal height“; hier 522pt), aus der sich bei einem `\baselineskip`

Wert von 12pt insgesamt 43 Zeilen ergeben. Der in dieser Rechnung verbleibende Fehlbetrag von 2pt rührt daher, dass der Abstand `\topskip` zum Positionieren der ersten Zeile einen Wert von lediglich 10pt hat. Lässt sich aus der Seitenhöhe keine glatte Anzahl von Zeilen errechnen, so können leichter Probleme wie unzureichend gefüllte vertikale Boxen (`\vbox`) entstehen.

Kandidaten für einen Umbruch

Die verbleibenden Zeilen, die alle mit nur einem Prozentzeichen beginnen, zeigen jeweils eine neue Position, die $\text{\TeX}$ für einen Seitenumbruch in Betracht gezogen hat. Sie lassen sich folgendermaßen interpretieren: Der Wert `t=` gibt die bisherige Länge der Druckfahne an, zuzüglich der Gesamtsumme der Dehn- und Stauchwerte etwaig vorhandener vertikaler elastischer Längen. Zeile 4 zeigt beispielsweise, dass abgesetzte Verbatim-Umgebungen im Layout dieses Buches mit einem zusätzlichen Abstand von 10pt plus einer dehnbaren Länge von 4pt versehen sind. (Die Verbatim-Zeilen werden in einer kleineren Schrift mit einem `\baselineskip`-Wert von nur 11pt gesetzt.) Zwischen Zeile 4 und 5 wird noch einmal der gleiche Betrag hinzugefügt.

Anvisierte Seitenhöhe

Der Wert `g=` gibt die an diesem Punkt anvisierte Seitenhöhe an. Er ändert sich nur, wenn der verfügbare Raum auf der Druckfahne zwischenzeitlich durch Gleitobjekte oder ähnliche Elemente reduziert wurde.

Badness-Wert für Seitenumbruch

Mit `b=` gibt $\text{\TeX}$ den Badness-Wert für einen Seitenumbruch an dieser Position an. Dieser Wert errechnet sich aus dem Faktor, mit welchem die zur Verfügung stehenden dehn- oder stauchbaren Abstände von `t=` multipliziert werden müssen, um die unter `g=` anvisierte Seitenhöhe zu erhalten. Bei diesem Beispiel ist die Seite fast leer, so dass sich immer ein Badness-Wert von 10000 (unendlich schlecht) ergibt. Erst in Zeile 7, in der die elastische Länge `fil` eingefügt wird, betrachtet $\text{\TeX}$ die Seite plötzlich als optimal (`b=0`).

Penalty-Wert für Seitenumbruch

$\text{\TeX}$ verknüpft mit jeder Umbruchposition einen numerischen `\penalty`-Befehl, der angibt, wie günstig ein Seitenumbruch an dieser Stelle wäre. Dieser Wert ist als `p=` aufgeführt. So darf z.B. direkt vor einer abgesetzten Verbatim-Umgebung kein Seitenumbruch erfolgen. Daher ist der Wert `t=` in Zeile 4 so viel höher als in Zeile 3, da hier der Seite zwei nicht trennbare Zeilen hinzugefügt wurden. Nach der Umgebung wird ein Umbruch hingegen durch einen Bonus (`p=-51`) als besonders günstig gekennzeichnet. Zeile 5 zeigt, dass ein Umbruch nach der ersten Zeile des zweizeiligen Absatzfragmentes als sehr ungünstig angesehen wird (`p=300`). Er würde zu verwaisten einzelnen Zeilen am Ende der aktuellen und am Anfang der nächsten Seite führen. Der erste dieser Umbruchfehler, auch Waisenkind oder Schusterjunge genannt, wird mit dem Penalty-Wert `\clubpenalty` belegt; der zweite Fall, auch Witwe oder Hurkenkind genannt, mit dem Penalty-Wert `\widowpenalty`. Beide Penalty-Werte betragen jeweils 150, wodurch sich die Summe von 300 ergibt.

Kosten für einen Seitenumbruch

Der Wert `c=` schließlich beschreibt die errechneten „Kosten“ für einen Seitenumbruch an der jeweiligen Stelle. Er wird mittels einer Formel berechnet, die sowohl den Badness-Wert der fertigen Seite (`b=`) berücksichtigt als auch die Frage, wie günstig (`p=`) ein Umbruch an dieser Stelle wäre. $\text{\TeX}$ vergleicht diese Kostenwerte und nimmt den Seitenumbruch letztendlich an der kostengünstigsten Stelle vor. Endet eine Zeile auf `#`, dann betrachtet $\text{\TeX}$ diese Position von allen bisher ausgewerteten Stellen als die beste für einen Seitenumbruch. Im Beispiel sind alle Zeilen mit einem `#`-Zeichen versehen – das ist nicht weiter überraschend, da $\text{\TeX}$ bei gleichen „Kosten“ spätere Umbruchpositionen als besser ansieht und alle Positionen außer der letzten als gleich schlecht eingestuft sind.

Für den Fall, dass Seiten bei einem Umbruch an einer bestimmten Position überfüllt würden, kennzeichnet T_EX die betreffenden Stellen mit `b=*`. Ab dieser Zeile sucht T_EX dann nicht mehr nach weiteren Umbruchpositionen, sondern umbricht die Seite an der bisher günstigsten Stelle.

Wer an weiteren Erläuterungen zu den Daten, die von diesen Low-Level-Anzeigefunktionen ausgegeben werden, interessiert ist, sollte [87, S. 112] zu Rate ziehen.

B.3.3 Diagnose und Lösung von Absatzproblemen

Für den Fall, dass T_EX Zeilenumbrüche in einem Absatz nicht in geeigneter Weise anordnen kann, lässt das Programm als letztes Mittel ein oder mehrere „überlaufende“ Zeilen zu. Jede dieser Zeilen erzeugt eine Warnung wie die folgende auf dem Bildschirm und in der Protokolldatei:

```
Overfull \hbox (6.762pt too wide) in paragraph at lines 4267--4285
/hlhr8t@8.50006pt/Hier wer-den die Text-zei-le und die
Zei-len-num-mern des Ab-sat-zes, der sie enthält,|
```

Hier werden die Textzeile und die Zeilennummern des Absatzes, der sie enthält symbolisch dargestellt. Durch einen Blick auf die symbolische Darstellung findet man leicht heraus, dass sich das Problem ergibt, weil T_EX das Wort „enthält“ nicht trennen kann.¹ Damit solche Zeilen im Dokument extra gekennzeichnet werden, muss man den Parameter `\overfullrule` auf einen positiven Wert setzen. Für den hier gezeigten Absatz wurde er auf 5pt eingestellt. Dadurch erzeugt er den schwarzen Tintenklecks, der die überlaufende Zeile deutlich hervorhebt. In den Standarddokumentenklassen lässt sich dieses Verhalten mit der Option `draft` aktivieren. Andererseits kann es auch sein, dass man nur gering überlaufende Zeilen nicht als störend ansieht. In diesem Falle kann man den auf 0.1pt voreingestellten Parameter `\hfuzz` ändern; es werden nur Zeilen gemeldet, die um mehr als den Wert dieses Parameters in den Rand hineinragen.

Wenn T_EX einen Absatz nicht zufrieden stellend umbricht, so kann dies verschiedene Ursachen haben: Oftmals liegt es an der Silbentrennung im Zusammenhang mit nicht trennbaren Worten, wie im letzten Beispiel, oder an den Parametereinstellungen für den Absatzalgorithmus. Oder der Text passt, unter Berücksichtigung ästhetischer Gesichtspunkte wie etwa eines akzeptablen Grauwertes der einzelnen Zeilen, einfach nicht in die Begrenzungen, welche durch die Spaltenbreite oder andere Parameter vorgegeben sind. In letzterem Falle bleibt einem normalerweise nichts anderes übrig, als den Text teilweise umzuformulieren.

Probleme mit der Silbentrennung

Wenn die benötigten Trennmuster geladen sind, kann T_EX viele Sprachen recht gut verarbeiten [117]. Das Programm findet jedoch normalerweise nicht alle möglichen Stellen, an denen eine Trennung erfolgen kann, so dass man hier hin und wieder nachhelfen muss. Besonders im Deutschen kann es bei

¹T_EX kann das Wort „ent-hält“ natürlich sehr wohl trennen; für dieses Beispiel wurde das Programm jedoch explizit daran gehindert. Ansonsten wäre der Absatz fehlerfrei gesetzt worden.

langen zusammengesetzten Wörtern auch manchmal zu Fehltrennungen kommen. Um herauszufinden, welche Trennpositionen $\TeX$ für ein Wort wie „Gleitobjektumgebungen“ kennt, übergibt man das Wort oder den Ausdruck einfach als Argument an den Befehl `\showhyphens`:

```
\showhyphens{Gleitobjektumgebungen}
```

Dieser Ausdruck schreibt bei der Verarbeitung mit $\LaTeX$ einige Protokolldaten auf den Bildschirm und in die Protokolldatei. Die von $\TeX$ bestimmten Trennpositionen werden durch Trennstriche gekennzeichnet:

```
[] \T1/cmr/m/n/10 Gleit-ob-jek-tum-ge-bun-gen
```

Wie leider zu erkennen ist, wird das Wort an der Nahtstelle falsch getrennt. Mithilfe des Befehls `\-` kann man fehlende oder falsche Trennpositionen lokal im Text ergänzen bzw. korrigieren, indem man, wie im folgenden Beispiel, *alle* erlaubten bzw. gewünschten Trennungen für das Wort angibt:

```
Gleit\-\ob\-\jekt\-\um\-\ge\-\bun\-\gen
```

Stattdessen kann man auch, wie für dieses Buch mehrfach geschehen, eine `\hyphenation`-Deklaration in der Präambel vornehmen:

```
\hyphenation{Gleit-ob-jekt-um-ge-bun-gen Gleit-ob-jekt-um-ge-bung}
```

Das letztere Verfahren ist besonders nützlich, wenn man eine falsche Trennung entdeckt oder häufig ein Wort verwendet, von dem man weiß, dass $\TeX$ dort wichtige Trennpositionen nicht kennt. Man muss jedoch berücksichtigen, dass $\TeX$ diese expliziten Angaben nur verwendet, um genau in dieser Form erscheinende Wörter zu trennen. Für die Singularform „Gleitobjektumgebung“ musste also eine eigene Auflistung der Trennpositionen erfolgen.

Die `\hyphenation`-Deklarationen gelten immer nur für die aktuelle Sprache. Werden in einem Dokument verschiedene Sprachen benutzt – beispielsweise mithilfe der Funktionen des `babel`-Systems –, dann muss man vor der jeweiligen Deklaration immer zunächst auf die betreffende Sprache umschalten.

Überwachen des Absatzalgorithmus

Da $\TeX$ zum Optimieren von Absatzumbrüchen einen global optimierenden Algorithmus verwendet, ist es nicht immer so einfach nachzuvollziehen, warum gerade eine bestimmte Lösung gewählt wurde. Gegebenenfalls kann man die Entscheidungen im Verlauf des Absatzumbruchs mithilfe der folgenden Deklarationen nachverfolgen:¹

```
\tracingparagraphs=1 \tracingonline=1
```

Für Leser, die wirklich verstehen möchten, auf welcher Grundlage bestimmte Entscheidungen gefällt werden, folgen hier einige Beispiele mit detaillierten Erläuterungen.

¹Diese Parameter werden auch durch einen `\tracingall`-Befehl eingeschaltet, so dass man in diesem Falle viele Zeilen an Daten zu Absatzumbrüchen erhält, obwohl man vielleicht ganz andere Informationen benötigt.

Die Absatzüberwachung gibt Daten aus, die zunächst etwas einschüchternd wirken. Der Beispieltext „The \hyphenation declarations apply to the current language, so if a document uses several languages—for example, by using the methods provided by the babel system—then you need to switch to the right language before issuing the relevant declarations.“ erzeugt zum Beispiel Daten, die folgendermaßen beginnen:

```
1 @firstpass
2 @secondpass
3 []\T1/cmr/m/n/10 The [] dec-la-ra-tions ap-ply to the cur-rent lan-guage, so
4 @ via @@0 b=3219 p=0 d=10436441
```

Zeile 2 besagt, dass T_EX sofort davon abgesehen hat, den Absatz ohne Silbentrennung zu setzen. Dies liegt daran, dass der Wert von `\pretolerance` im Quelltext für dieses Buch auf 100 gesetzt wurde. Andernfalls wäre T_EX vielleicht weiter gekommen oder hätte den Absatz sogar setzen können. In englischen Texten lässt sich normalerweise ein Großteil der Absätze ohne Silbentrennung vernünftigerweise setzen – in deutschen Texten mit durchschnittlich längeren Wörtern ist dies eher unwahrscheinlich. Zusätzlich zu `@secondpass` erfolgt manchmal noch der Eintrag `@emergencypass`, der bedeutet, dass T_EX trotz Silbentrennung keine geeignete Lösung finden konnte und daher einen weiteren Verarbeitungslauf unter Zuhilfenahme der Länge `\emergencystretch` durchgeführt hat¹. Zeile 3 zeigt, wie weit T_EX lesen musste, um den ersten potentiellen Zeilenumbruch mit einem Badness-Wert unter $\infty = 10000$ zu finden. In Zeile 4 folgen nähere Einzelheiten zu diesem möglichen Umbruch. Derartige Zeilen beginnen mit einem einzelnen @-Zeichen. Der Eintrag `via` verweist auf die vorhergehende Umbruchposition (in diesem Falle @@0 für den Absatzanfang), den Badness-Wert der Zeile (`b=`), den Penalty-Wert für einen Umbruch an dieser Stelle (`p=`) und weitere Strafpunkte (englisch: demerits; `d=`), die mit dem Umbrechen in Verbindung stehen. Bei letzteren handelt es sich um „Kosten“, welche außer dem Badness- und dem Penalty-Wert noch weitere Kontextinformationen berücksichtigen, wie etwa das Umbrechen an einem Trennstrich oder die visuelle Verträglichkeit mit der vorhergehenden Zeile.

*Bis zu drei Durchläufe
bei Absatzdaten*

```
5 @@1: line 1.0 t=10436441 -> @@0
```

In Zeile 5 teilt T_EX mit, dass über die in Zeile 3 genannte Umbruchposition eine Zeile mit sehr weiten Wortzwischenräumen gebildet werden könnte. Die Gesamtkosten (`t=`) entsprächen dabei den Strafpunkten aus Zeile 4. Diese Zeile würde an der Umbruchposition @@0 beginnen. Der Text `line 1.0` nennt die Nummer der zu erzeugenden Zeile, wobei die Suffixe `.0`, `.1`, `.2`, `.3` für sehr weite, weite, passende bzw. enge Wortzwischenräume in der Zeile stehen. Diese Klassifizierung ist wichtig, um die visuelle Verträglichkeit aufeinander folgender Zeilen zu vergleichen.

T_EX findet nun immer weitere potenzielle Zeilenumbrüche, wie etwa nach dem „if“ in Zeile 6 und nach dem „a“ in Zeile 9. Jedes Mal teilt T_EX mit, welche Art von Zeile sich bei einem Umbruch an dem betreffenden Punkt ergeben würde. Ein `b=*` weist darauf hin, dass T_EX keine passende Stelle für einen Zeilenumbruch finden konnte und daher eine unpassende Lösung gewählt hat (welche also den `\tolerance`-Wert für die betreffende Zeile überschreitet).

```
6 if
7 @ via @@0 b=1087 p=0 d=1213409
8 @@2: line 1.0 t=1213409 -> @@0
9 a
```

¹Damit es dazu kommt, muss `\emergencystretch` auf einen positiven Wert eingestellt sein (vgl. die Erläuterungen in Abschnitt 3.1.11).

```

10 @ via @@0 b=334 p=0 d=128336
11 @@3: line 1.0 t=128336 -> @@0
12 doc-
13 \@discretionary via @@0 b=0 p=50 d=2600
14 @@4: line 1.2- t=2600 -> @@0
15 u-
16 \@discretionary via @@0 b=1 p=50 d=2621
17 @@5: line 1.2- t=2621 -> @@0

```

Durch Trennen des Wortes doc-u-ment findet T_EX noch zwei weitere Umbruchpositionen (in Zeile 12 und 15). Der Penalty-Wert für diese Umbrüche beträgt 50, was dem Wert des Parameters \hyphenpenalty (Umbruch an einem Trennstrich) entspricht. Zeile 15 gibt den letzten möglichen Umbruch für die erste Zeile des Absatzes an. Alle weiteren Umbruchpositionen würden eine überlaufende Zeile erzeugen. Daher zeigt die nächste Protokollzeile wieder weiteren Text; keine der möglichen Umbruchpositionen in diesem Text lässt sich verwenden, da die zweite Zeile sonst außerhalb des \tolerance-Wertes liegen würde.

```

18 ment uses sev-eral languages---for ex-am-ple, by us-ing the meth-
19 \@discretionary via @@1 b=1194 p=50 d=1452116
20 \@discretionary via @@2 b=2875 p=50 d=8325725
21 @@6: line 2.0- t=9539134 -> @@2

```

Es gibt nun zwei verschiedene Möglichkeiten für den Umbruch zur zweiten Zeile: Er kann an Position @@1 (Zeile 19) oder an Position @@2 (Zeile 20) liegen. Vergleicht man nur diese beiden Lösungsmöglichkeiten für die zweite Zeile des Absatzes, dann wäre die erste die bessere Wahl: Ihr Badness-Wert beträgt 1194, während die zweite Möglichkeit einen Badness-Wert von 2875 aufweist, der als ein Faktor von 5 bei den „Kosten“ (d=) zu Buche schlägt. T_EX betrachtet trotzdem die zweite Position als die bessere Lösung, da ein Ende der ersten Zeile an Position @@1 so viel schlechter wäre als ein Zeilenende an Position @@2, dass die Gesamtkosten für einen Umbruch bei der zweiten Alternative immer noch geringer sind. T_EX hält daher in Zeile 21 fest, dass der beste Weg zu der in Zeile 18 genannten Umbruchposition über @@2 führt und mit Gesamtkosten von t=9539134 einhergeht. Für den weiteren Verlauf muss T_EX nicht mehr berücksichtigen, dass es verschiedene Möglichkeiten gab, Position @@6 zu erreichen; das Programm muss nur den besten Weg dorthin festhalten.

Genauer gesagt, muss sich T_EX die jeweils beste Möglichkeit eines Zeilenumbruchs für jeden der vier Zeilentypen (sehr weit, weit, passend, eng) merken, da der Algorithmus andere Strafpunkte verhängt, wenn aufeinander folgende Zeilen optisch nicht zueinander passen (also etwa eine weit gesetzte Zeile auf eine enge folgt). Daher erfolgt später im Verlauf des Tracings (die Zeilen 22–40 werden hier nicht angezeigt) folgende Ausgabe:

```

41 by
42 @ via @@3 b=19 p=0 d=10841
43 @ via @@4 b=9 p=0 d=361
44 @ via @@5 b=42 p=0 d=2704
45 @@10: line 2.1 t=5325 -> @@5
46 @@11: line 2.2 t=2961 -> @@4

```

Diese Daten deuten darauf hin, dass es drei Möglichkeiten gibt, die Zeile an dem Wort „by“ zu umbrechen: indem man die Zeile an Position @@3, @@4 oder @@5 beginnt. Eine Zeile mit einem Badness-Wert bis einschließlich 12 wird als passend angesehen (Suffix .2); eine gedehnte Zeile, deren Badness nicht über 100 liegt, gilt als weit (Suffix .1). T_EX merkt sich hier also zwei günstige Umbruchpositionen für seine weiteren Berechnungen – eine an Position @@5 und die andere an Position @@4.

Wo die Umbrüche letztendlich tatsächlich erfolgen, wird erst entschieden, wenn das Ende des Absatzes erreicht ist. Daher können selbst kleine Änderungen irgendwo im Absatz dazu führen, dass T_EX eine andere Anordnung von Umbrüchen als beste Lösung für die aktuelle Umbruchaufgabe wählt, da diese die geringsten Kosten erzeugt. Da der Algorithmus sehr komplex ist, haben kleinere Änderungen manchmal überraschende Wirkungen. So kann das Löschen eines Wortes einen Absatz durchaus um eine Zeile verlängern. T_EX kann nämlich in diesem Moment entscheiden, dass gleichmäßig weit gesetzte Zeilen oder ein nicht getrenntes Wort jeder anderen Möglichkeit, den Absatz zu umbrechen, vorzuziehen sind. In [87, S. 98] sind alle Parameter, die den Zeilenumbruch beeinflussen, ausführlich beschrieben. Soweit notwendig kann man mit `\linebreak` Umbrüche an einer bestimmten Stelle erzwingen oder sie mit `\nolinebreak` bzw. einem `~`-Zeichen anstelle eines Leerzeichens verhindern. Am Anfang eines Absatzes ist die Auswahl natürlich ziemlich klein und es kann durchaus sein, dass man einen Satz umformulieren muss, um einen ungünstigen Umbruch zu vermeiden. Später in einem Absatz kann man jedoch nahezu jede mögliche Umbruchposition verwenden, ohne dass man dazu den angegebenen `\tolerance`-Wert überschreiten müsste.

Kürzen oder Verlängern von Absätzen

Als weiteres Low-Level-Werkzeug lässt sich der interne Zähler `\looseness` verwenden. Setzt man diesen auf eine ganze Zahl n , die nicht null ist, so versucht T_EX den nächsten Absatz unter Berücksichtigung aller weiteren Rahmenbedingungen (wie etwa des erlaubten `\tolerance`-Wertes) um n Zeilen zu verlängern (positiver Wert) oder zu verkürzen (negativer Wert). Die letzten zwei Absätze des vorigen Abschnitts wurden durch die folgenden Befehle künstlich um eine Zeile verkürzt bzw. verlängert:

```
\looseness=-1
Diese Daten deuten darauf hin, dass es drei Möglichkeiten ...

\looseness=1
Wo die Umbrüche letztendlich tatsächlich erfolgen, wird erst ...
```

Nicht immer, wenn man einen `\looseness`-Wert festlegt, wirkt sich dieser auch aus. Für T_EX ist es viel schwieriger, einen Absatz zu verkürzen als ihn zu verlängern, da sich Wortzwischenräume im Vergleich zu ihrer Dehnbarkeit nur wenig stauchen lassen. Die besten Ergebnisse erzielt man bei langen Absätzen, deren letzte Zeile sehr kurz ist (der gewählte Absatz enthielt nur „@@4.“ in seiner letzten Zeile). Entsprechend lässt sich ein Absatz leichter verlängern, wenn er bereits länger und die letzte Zeile fast voll ist. Möglicherweise muss man jedoch die letzten Wörter des Absatzes mit einem `\mbox`-Befehl zusammenfassen, damit nicht nur ein einzelnes Wort in der letzten Zeile erscheint.

B.3.4 Weitere Low-Level-Überwachungsfunktionen

T_EX verfügt über eine Reihe weiterer interner ganzzahliger Parameter, die manchmal helfen können, den Ursprung eines Problems ausfindig zu machen. Sie sind hier im Folgenden mit einer kurzen Erläuterung zu ihrem Gebrauch aufgeführt.

*Bildschirm-
überwachung*

Der Befehl `\tracingonline` wurde ja bereits angesprochen. Mit einem positiven Wert sorgt er dafür, dass alle Protokolldaten auch auf dem Bildschirm erscheinen; andernfalls werden die meisten nur in die Protokolldatei geschrieben. Dieser Parameter wird von `\tracingall` automatisch eingeschaltet.

Durch den Befehl `\tracingoutput` wird der Seiteninhalt protokolliert. Was gezeigt wird, hängt dabei von zwei weiteren Parametern ab: `\showboxdepth` bestimmt, bis zu welcher Ebene verschachtelte Boxen dargestellt werden und `\showboxbreadth` legt fest, wie viel Material für jede Ebene angezeigt wird. Alles, was diese beiden Werte übersteigt, wird in der symbolischen Darstellung mit `etc.` oder `[]` (bei einer verschachtelten Box) abgekürzt. Der $\text{\LaTeX}$ -Befehl `\showoutput` setzt diese Werte auf ihr Maximum und `\tracingoutput` auf 1, so dass man so ausführliche Informationen wie möglich erhält. Der Befehl `\showoutput` wird von `\tracingall` automatisch aufgerufen.

Boxeninhalte

Mithilfe des Befehls `\showbox` kann man den Inhalt einer mit `\sbox` oder `\savebox` erzeugten Box anzeigen lassen.

```
\newsavebox\test\sbox\test{1 Test}
{\tracingonline=1\showbox\test }
```

Das Ergebnis ist jedoch nicht besonders hilfreich, wenn man nicht gleichzeitig auch `\showboxdepth` und `\showboxbreadth` anpasst. Darum ist der $\text{\LaTeX}$ -Befehl `\showoutput` hier die bessere Alternative:

```
{\showoutput\showbox\test }
```

Man beachte, dass der Geltungsbereich von `\showoutput` durch die Klammern begrenzt wird. Ohne die Klammern würden alle weiteren Seitenboxen angezeigt, was vielleicht nicht so interessant wäre. Auf dem Bildschirm wird die gleiche symbolische Darstellung verwendet, wie sie in Abschnitt B.3.2 besprochen wurde.

```
> \box26=
\hbox(6.83331+0.0)x27.00003
.\OT1/cmr/m/n/10 1
.\glue 3.33333 plus 1.66498 minus 1.11221
.\OT1/cmr/m/n/10 T
.\OT1/cmr/m/n/10 e
.\OT1/cmr/m/n/10 s
.\OT1/cmr/m/n/10 t
```

Durch Einfügen von `\scrollmode` oder `\batchmode` vor dem `\showbox`-Befehl hält $\text{\LaTeX}$ an diesem Punkt nicht an. Man kann die Daten dann in der Protokolldatei untersuchen.

*Lokal zurückgesetzte
Werte*

Um zu sehen, welche Werte und Definitionen $\text{\TeX}$ wiederherstellt, wenn eine Gruppe endet, kann man `\tracingrestores` auf einen positiven Wert setzen. Dieser Parameter wird von `\tracingall` automatisch eingeschaltet.

Der List-Stack von $\text{\TeX}$

Mithilfe des Befehls `\showlists` kann man $\text{\TeX}$ anweisen, den List-Stack (Listen-Stapelspeicher; vertikal, horizontal) anzuzeigen, mit dem das Pro-

gramm gerade arbeitet. Schreibt man beispielsweise den Befehl `\showlists` in die Fußnote¹ des aktuellen Absatzes, dann erhält man die folgenden Einträge in der Protokolldatei:

```
### horizontal mode entered at line 4840 (language3:hyphenmin2,2) []
spacefactor 1000, current language 3
### internal vertical mode entered at line 4840
prevdepth ignored
### horizontal mode entered at line 4824 (language3:hyphenmin2,2) []
spacefactor 1000, current language 3
### vertical mode entered at line 0
### current page: []
total height 546.50224 plus 24.0
goal height 572.56006
prevdepth 2.22221
```

Der Text der Fußnote beginnt in Zeile 4840 und der Befehl `\spacefactor` wird an ihrem Anfang auf 1000 gesetzt. Der Text verwendet die Silbentrennungsmuster für neue deutsche Rechtschreibung (im Buch `language3`) mit einem Wert für `\lefthyphenmin` und `\righthyphenmin` von jeweils 2 (siehe Abschnitt 9.5.1 auf Seite 596). Die Fußnote selbst beginnt in der gleichen Zeile und steuert den „internen vertikalen Modus“ bei. T_EX ignoriert dabei völlig zu Recht den äußeren Wert von `\prevdepth`. Die Fußnote ist Teil eines Absatzes, der in der Quellcodezeile 4824 beginnt. Dieser wiederum ist in eine vertikale Liste eingebettet, die in Zeile 0 anfängt, was darauf hindeutet, dass es sich um den Fließtext der Kolumne handelt. Und schließlich enthalten die ausgegebenen Daten noch Informationen über die gerade im Aufbau befindliche Seitenliste, einschließlich der aktuellen Seitenlänge, der anvisierten Höhe und des Wertes von `\prevdepth` (also der Tiefe der momentan letzten Zeile auf der Seite).

Aufgrund der für `\showboxbreadth` und `\showboxdepth` voreingestellten Werte sind die Inhalte aller Listen als `[]` abgekürzt. Um weitere Einzelheiten zu sehen, muss man die Werte entsprechend anpassen oder mit `\showoutput\showlists` alle Daten anzeigen lassen.

Der Befehl `\tracingcommands`, der alle während der Verarbeitung von T_EX aufgerufenen Basisbefehle zeigt, ist für sich allein genommen nicht besonders nützlich. In Kombination mit anderen Überwachungsoptionen kann er sich jedoch als hilfreich erweisen. Ein mit ihm verwandter interner ganz-zahliger Befehl ist `\tracingmacros`, der alle von T_EX durchgeführten Makroexpansionen anzeigt. Auf den Wert 2 eingestellt zeigt er auch die Verarbeitung von Bedingungen. Beide Parameter werden von `\tracingall` automatisch eingeschaltet.

*Überwachen der
Verarbeitung*

Sofern man alles fehlerfrei eingerichtet hat, ist es unwahrscheinlich, dass T_EX jemals auf eine Position im aktuellen Zeichensatz stößt, die nicht mit einer Glyphie verknüpft ist. Einige Befehle, wie etwa `\symbol`, können jedoch explizit auf eine bestimmte Position verweisen, so dass sich Fehler nicht ganz ausschließen lassen. Leider betrachtet T_EX ein solches Ereignis nicht als Fehler

*Fehlende Zeichen
aufspüren*

¹Eine Fußnote beginnt eine neue vertikale Liste und darin eine neue horizontale Liste für den Fußnotentext.

(was es aber tun sollte). Das Programm protokolliert derartige fehlende Zeichen lediglich in Form unauffälliger Einträge in der Protokolldatei, und auch nur dann, wenn der Befehl `\tracinglostchars` auf einen positiven Wert eingestellt ist. $\text{\LaTeX}$ versucht hier zu unterstützen, indem es diesen ganzzahligen Parameter auf 1 setzt.

*Dokumente
schrittweise
verarbeiten*

Abschließend kann man $\text{\TeX}$ noch anweisen, eine Datei zeilenweise zu verarbeiten. Mit einem Wert von 1 für `\pausing` wird jede Quellcodezeile zunächst einmal nur angezeigt (mit `=>` als Suffix). $\text{\TeX}$ wartet dann auf Anweisungen, wie diese verarbeitet werden soll. Durch Drücken der Eingabetaste übernimmt $\text{\TeX}$ die Zeile unverändert; jede andere Eingabe bedeutet, dass $\text{\TeX}$ die Angaben des Benutzers anstelle der aktuellen Zeile verwenden soll. $\text{\TeX}$ führt dann den ihm übergebenen Code aus und formatiert ihn. Danach gibt das Programm die nächste Zeile aus und hält erneut an. Um mit der normalen Verarbeitung fortzufahren, muss man `\pausing=0` eingeben. Dabei darf man allerdings nicht vergessen, dass die Eingabe anstelle der Quellcodezeile verwendet wird, so dass man das entsprechende Material gegebenenfalls auch noch einmal wiederholen muss.

B.3.5 trace – Selektives Überwachen von Befehlen

Der $\text{\LaTeX}$ -Befehl `\tracingall` (der aus Plain- $\text{\TeX}$ übernommen wurde) dient dazu, die Überwachung sämtlicher Abläufe einzuschalten. Es gibt jedoch einige Probleme mit diesem Befehl:

1. Es gibt keinen entsprechenden Befehl, um die Überwachung wieder abzuschalten. Entsprechend muss man den Geltungsbereich einschränken, was nicht immer hilfreich und manchmal sogar unmöglich ist.
2. Einige Teile von $\text{\LaTeX}$ erzeugen sehr umfangreiche Protokolldaten, die von nur geringem oder gar keinem Interesse für das aktuelle Problem sind.

Lädt $\text{\LaTeX}$ beispielsweise einen neuen Zeichensatz, dann werden einige interne Routinen des NFSS ausgeführt, die Fontdefinitionstabellen durchsuchen oder andere Prozesse abwickeln. Und in 99,9% der Fälle ist man gar nicht an diesem Teil der Verarbeitung interessiert, sondern nur an den zwei Zeilen davor und den fünf Zeilen danach. Trotzdem muss man einige hundert Ausgabezeilen nach den benötigten Daten durchsuchen (wenn man sie überhaupt findet).

Ein weiteres, recht anschauliches Beispiel ist eine Anweisung wie etwa `\setlength\linewidth{1cm}`. Standard- $\text{\LaTeX}$ erzeugt hierzu fünf Zeilen Protokolldaten. Hat man jedoch das `calc`-Paket geladen, so erhält man hier ein Datenvolumen von ungefähr 60 Zeilen. Das ist wahrscheinlich nicht ganz das, was man erwartet und auch nicht wirklich hilfreich, es sei denn, man möchte die Parsing-Routinen des `calc`-Paketes auf Fehler untersuchen (was idealerweise nicht notwendig sein sollte).

Zum Lösen des ersten Problems stellt das `trace`-Paket [133] von Frank Mittelbach das Befehlspaar `\traceon` und `\traceoff` zur Verfügung. Verwendet man $\text{\LaTeX}$ auf Basis von $\text{\TeX}$, dann ist der Befehl `\traceon` eigentlich nur eine andere Bezeichnung für `\tracingall`: Er aktiviert die gleichen Überwachungsschalter (wenn auch in anderer Reihenfolge, um sich nicht

selbst zu überwachen). Betreibt man L^AT_EX auf Basis der eT_EX-Engine, so wird auch die Protokollierung von Zuweisungen und Gruppen eingeschaltet. Die entsprechenden eT_EX-Schalter sind `\tracingassigns` und `\tracinggroups` (Details findet man im *eT_EX Manual* [27]).

eT_EX stellt weitere Überwachungsdaten bereit

Ein weiterer Unterschied zwischen `\traceon` und `\tracingall` besteht darin, dass letzterer Befehl die Überwachungsdaten immer auf dem Bildschirm ausgibt, während man `\traceon` mit der Option `logonly` anweisen kann, diese nur in die Protokolldatei zu schreiben. Das ist besonders nützlich, wenn die Ausgabe auf den Bildschirm sehr langsam ist (z.B. in einem Shellfenster unter `emacs`).

Um das zweite Problem zu lösen, verfügt das `trace`-Paket über eine Reihe interner Befehle, mit denen sich die Überwachung zeitweise unterbinden lässt. Es definiert die sehr wortreichen der internen L^AT_EX-Funktionen so um, dass die Überwachung während ihrer Ausführung abgeschaltet ist. Dies trifft beispielsweise auf die Funktion zum Laden neuer Zeichensätze zu. Beginnt ein Dokument mit den beiden Formeln

```
$a \neq b$ \small $A = \mathcal{A}$
```

dann lädt L^AT_EX an dieser Stelle 22 neue Zeichensätze.¹ Der Standardbefehl `\tracingall` führt bei dieser Quellcodezeile zu einer Bildschirmausgabe von ungefähr 7500 Zeilen. Verwendet man `\traceon`, so werden hingegen nur 350 Zeilen ausgegeben (hauptsächlich zu dem Befehl `\small`).

Es sind nur wenige Befehle von der Überwachung ausgenommen und auch nur solche, bei denen es unwahrscheinlich ist, dass sie etwas mit dem aktuellen Problem zu tun haben. Benötigt man jedoch eine vollständige Überwachung, so kann man entweder `\tracingall` oder die Option `full` verwenden. In letzterem Falle verfolgt `\traceon` alle Vorgänge. Man kann aber immer noch festlegen, dass die Überwachungsdaten nur in die Protokolldatei geschrieben werden.

¹Das lässt sich mittels der Option `loading` des `tracefmt`-Paketes überprüfen.

A n h a n g C

L^AT_EX-Software und Usergroups

C.1 Wie erhält man Hilfe?	995
C.2 Wo bekommt man diese T _E X-Dateien?	996
C.3 Wie verwendet man CTAN?	998
C.4 Dokumentation auf dem eigenen System	1002
C.5 T _E X-Usergroups	1004

Die in diesem Buch beschriebenen Dateien sind in den meisten T_EX-Distributionen, wie der T_EX Live-DVD oder -CD-ROM (die mit diesem Buch mitgeliefert wird), oder auf den CTAN-DVDs von DANTE verfügbar. Die neuesten Versionen können außerdem direkt aus dem Internet heruntergeladen werden. Ziel dieses Anhangs ist es, Informationen bereitzustellen, die man benötigt, um die aktuellen Versionen dieser DVDs zu erhalten (auf Anfrage sind auch CD-ROMs erhältlich). Außerdem werden Hinweise erteilt, wie man die benötigten Dateien im Internet findet und herunterlädt.

C.1 Wie erhält man Hilfe?

Die Autoren hoffen zwar, dass in den vorangegangenen Kapiteln bereits alle Fragen beantwortet wurden, aber ein paar Fragen werden wohl trotzdem noch offen sein. Bei Fragen, die sich auf spezielle, in diesem Buch besprochene Pakete beziehen, kann es hilfreich sein, die Originaldokumentation zu lesen, die mit dem Paket mitgeliefert wird. Anhang C.4 gibt Hinweise, wie man eine solche Dokumentation auf dem eigenen System findet.

Eine sehr wertvolle Quelle sind die bestehenden „Frequently Asked Questions“ (häufig gestellte Fragen)-Dokumente, kurz FAQs. Die wichtigsten unter ihnen sind die UK-TUG FAQ von Robin Fairbairns, die unter <http://www.tex.ac.uk/faq> (oder <http://faq.tug.org>) verfügbar ist, und die DANTE FAQ

von Bernd Raichle u.a., die unter <http://www.dante.de/faq/de-tex-faq> (in Deutsch) verfügbar ist. Robins FAQ gibt es auch im HTML-Format auf der beigefügten CD-ROM im Verzeichnis `/texmf/doc/html/faq/index.html`. Da sich beide Dokumente jedoch permanent weiterentwickeln, empfiehlt es sich, wenn möglich, die Online-Version zu benutzen.

Für den Fall, dass die Standardantworten nicht ausreichen, gibt es diverse Newsgroups, die sich allgemeinen Fragen zu T_EX und L^AT_EX widmen: die wichtigste ist wahrscheinlich `news://comp.text.tex`. Sie erhält pro Tag mehr als 100 Anfragen. Viele der Autoren, die in diesem Buch erwähnt werden, verfassen regelmäßig Beiträge in diesen Newsgroups und helfen bei der Beantwortung der Fragen und Bitten. Auf diese Art gibt es riesige Mengen an hilfreichen Material auf dem Internet. Es kann bequem über Suchmaschinen durchsucht werden, die solche Beiträge in ihren Index aufnehmen.

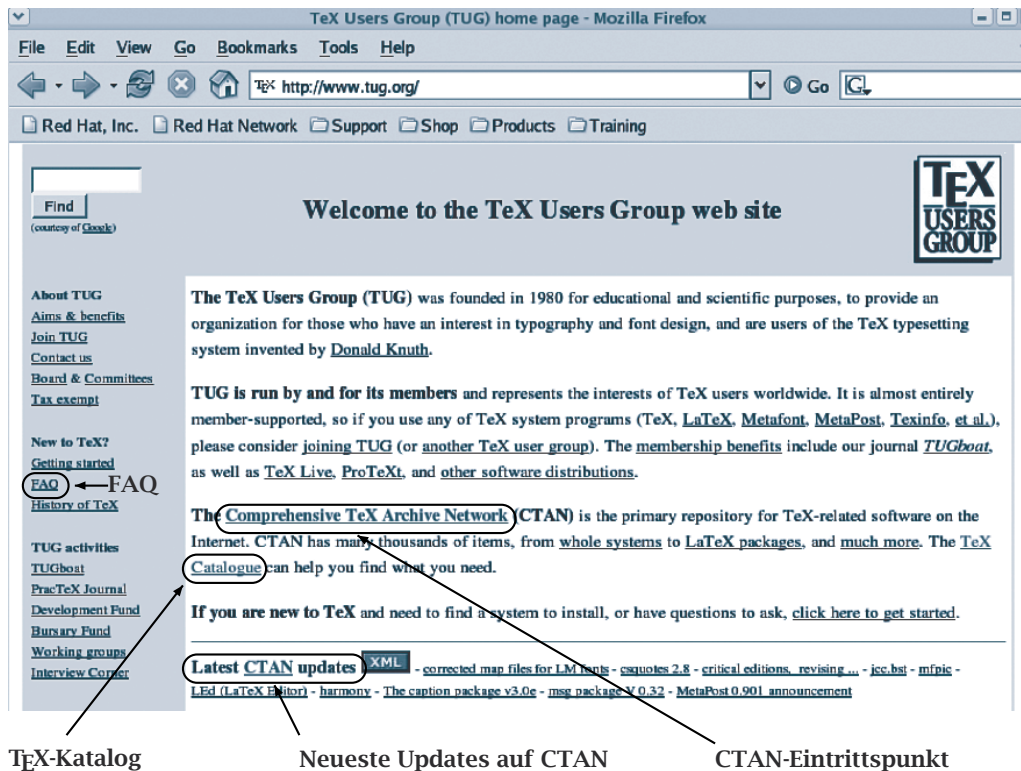
Jeder, der eine Nachricht an eine dieser Newsgroups schickt, sollte sich an die grundlegende Netiquette halten. Die Benutzergemeinschaft ist hilfsbereit, aber manchmal auch sehr direkt und erwartet, dass der Hilfesuchende vorher selbst einige Recherchen angestellt hat (z.B. zunächst die FAQs gelesen und dann die archivierten News durchsucht hat) und keine Fragen stellt, die schon hundert Mal beantwortet wurden. Eine gute Idee ist, sich vorab Eric Raymonds „*How To Ask Questions The Smart Way*“ durchzulesen, das unter <http://www.catb.org/~esr/faqs/smart-questions.html> verfügbar ist. Wenn möglich, sollte man stets ein kleines, aussagekräftiges *und* vollständiges Beispiel des anstehenden Problems beifügen, anhand dessen andere ohne Aufwand die aufgetretenen Symptome nachvollziehen können – dies spart den Helfern viel Zeit und führt möglicherweise zu einer schnelleren Antwort.

C.2 Wo bekommt man diese T_EX-Dateien?

Ein nützlicher Eintrittspunkt in die T_EX-Welt ist die Homepage der T_EX Users Group (<http://www.tug.org>; siehe Abbildung C.1). Von dort erreicht man die meisten Informationsquellen zu T_EX und zugehörigen Programmen, die weltweit verfügbar sind.

Von dieser Homepage gelangt man besonders gut auf einen der CTAN (Comprehensive T_EX Archive Network)-Knoten. CTAN ist eine Gemeinschaftsarbeit, die 1992 von der technischen Arbeitsgruppe ins Leben gerufen wurde, die sich bei der T_EX Users Group mit den T_EX-Archivrichtlinien befasste. CTAN wurde ursprünglich von George Greenwade koordiniert, und zwar auf der Grundlage früherer Arbeiten von Peter Abbott (für einen historischen Abriss siehe [63]), und wird inzwischen von Jim Hefferon, Robin Fairbairns, Rainer Schöpf und Reinhard Zierke gepflegt (Stand: Sommer 2005). Das Archivnetzwerk dient hauptsächlich dazu, einen leichten Zugang zu aktuellen Versionen von T_EX, L^AT_EX, METAFONT und Hilfsprogrammen sowie zugehörigen Dateien zu bieten.

Derzeit gibt es drei Hauptknoten, die als FTP-Server fungieren: in Großbritannien (cam.ctan.org), in Deutschland (dante.ctan.org) und in den Vereinigten Staaten (tug.ctan.org). Zudem werden die Archive weltweit gespiegelt, und alle haben eine Webschnittstelle (siehe Abbildung C.2).

Abbildung C.1: Die Homepage der T_EX Users Group im Internet

Das Material auf dem CTAN wird regelmäßig (derzeit einmal pro Jahr) auf DVDs bereitgestellt (bei Bedarf wird das Material auch auf mehreren CD-ROMs bereitgestellt, die auf Anfrage über eine der T_EX Usergroups bezogen werden, siehe unten). Eine der Distributionen ist die T_EX Live-DVD bzw. CD-ROM ([158]; siehe auch <http://tug.org/texlive>), die eine „lauffähige“ Version von T_EX für verschiedene Plattformen bereitstellt. T_EX Live CD-ROMs werden seit 1996 durch Zusammenarbeit der T_EX Users Group (TUG; Vereinigte Staaten) und den T_EX-Usergroups u.a. in Deutschland, Frankreich, Großbritannien, Indien, den Niederlanden, Polen, der Slowakei und der Tschechischen Republik entwickelt. Diese Usergroups verteilen die T_EX Live-Distribution an ihre Mitglieder, bei Bedarf sollte man sich direkt an sie wenden (ihre Adressen sind in Abschnitt C.5 angegeben).

Eine weitere Distribution wird von der deutschsprachigen T_EX-Usergroup DANTE (siehe Abschnitt C.5) bereitgestellt. Sie enthält auf einer DVD ein Image der kompletten CTAN-Verzeichnisstruktur (mehr als 4GB Daten). Ähnlich wie die T_EX Live-DVD wird auch diese DANTE CTAN-DVD von den meisten Usergroups an ihre Mitglieder verteilt. Bei Interesse an dieser CTAN-DVD sollte man sich also auch hier wieder direkt an eine der Usergroups wenden.

C.3 Wie verwendet man CTAN?

Im vorangegangenen Abschnitt wurden die CTAN-DVDs (ehemals CD-ROMs) von T_EX Live und DANTE beschrieben. Sich die neueste Version dieser DVDs zu besorgen, ist eine optimale Möglichkeit, um Zugriff auf aktuelle Versionen des L^AT_EX-Programms zu erhalten.

Für Leser mit einer Internetverbindung ist es trotzdem sinnvoll, ab und zu einen der CTAN-Knoten zu besuchen, um zu überprüfen, ob eine der L^AT_EX-Komponenten, die sie benötigen, aktualisiert wurde. Insbesondere auf der Homepage der T_EX Users Group gibt es einen Bereich, in dem die neuesten Updates aufgelistet sind, die auf dem CTAN verfügbar sind (siehe Abbildung C.1, unteres Oval). Updates, die sich auf diesen Internetseiten befinden, können heruntergeladen werden, indem man die neueste Version des angegebenen Paketes direkt aus einem CTAN-Archiv herunterlädt (zur genauen Vorgehensweise siehe Abschnitt C.3.2).

Netzwerkverbindungen werden zwar von Tag zu Tag schneller, aber es ist trotzdem empfehlenswert, eine Verbindung zu einer Internetseite aufzubauen, die geographisch nicht zu weit vom eigenen Standort entfernt ist (Eine Liste der gespiegelten Seiten der CTAN-Knoten ist unter <http://www.tug.org/tex-archive/CTAN.sites> zu finden.).

C.3.1 Verwenden des T_EX-Dateikatalogs

Ein Katalog mit T_EX- und L^AT_EX-verwandten Paketen, der von Graham Williams gepflegt wird, kann unter der Adresse <http://datamining.csiro.au/tex/catalogue.html> abgerufen werden. Einen direkten Zugriff zum Katalog bietet auch die Homepage der T_EX Users Group (siehe zweites Oval in Abbildung C.1).

Was wichtiger ist: Über die CTAN-Schnittstelle (z.B. <http://www.tug.org/ctan.html>) kann der Katalog sogar durchsucht werden. In Abbildung C.2 ist zu sehen, wie nach der Eingabe der Zeichenfolge „graphicx“ im Bereich „Search Catalogue“ die Seite angezeigt wird, die im unteren Teil der Abbildung dargestellt ist. Auf dieser zweiten Seite kann man aus den Vorschlägen direkt den Eintrag auswählen, den man näher betrachten möchte (linke Liste auf der Seite), oder man kann den Link zum zugehörigen CTAN-Verzeichnis anklicken (rechte Liste auf der Seite).

C.3.2 Suchen und Herunterladen von Archivdateien

Bei einem Blick zurück auf Abbildung C.2 stellt man fest, dass die Webschnittstelle eine einfache Möglichkeit bietet, um auf CTAN eine Datei zu finden. Tatsächlich muss man nur im CTAN-Suchbereich den gewünschten Suchbegriff eingeben. Im Beispiel wurde die Zeichenfolge „graphicx“ ausgewählt (oberes Oval in Abbildung C.2). Die Suchmaschine gibt daraufhin eine Liste aller Dateien aus, die im CTAN-Archiv dem Suchkriterium entsprechen (siehe oberen Teil der Abbildung C.3). Das Verzeichnis kann nun durchsucht und, wenn gewünscht, einzelne Dateien heruntergeladen werden. Es besteht auch die Möglichkeit, das gesamte Verzeichnis zu übertragen. Hierfür klickt man auf den Link „entire directory“ (ganz rechtes Oval in Abbildung C.3), der einen Sprung auf die Seite auslöst, die im unteren Teil der Abbildung C.3 zu

Comprehensive TeX Archive Network (CTAN)

Some years ago, a TUG [Technical Council](#) working group defined a common network archive for TeX software, the Comprehensive TeX Archive Network (CTAN). The archive holds up-to-date releases of many implementations of TeX, LaTeX, Metafont and related packages and programs. It is the primary repository for TeX-related software on the Internet.

CTAN was initiated and continues to be supported by the TeX user groups. Please consider supporting them in turn, by [joining TUG](#) or the [TeX user group](#) best for you.

The backbone machines are the ftp servers at [cam.ctan.org](#) (UK), [dante.ctan.org](#) (Germany) and [tug.ctan.org](#) (USA). Of course there is also a [web interface](#) (at each server).

CTAN is [mirrored worldwide](#). This tug.org server itself is a CTAN mirror: [browse local CTAN via http: via ftp](#). (Besides CTAN, tug.org also makes available much TeX-related and other assorted software: [all tug.org software via http: via ftp](#).)

A snapshot of the archive is distributed on physical media (both CD and DVD, these days) once a year by TUG (as a [benefit of membership](#)), and most other TeX user groups. You can also [purchase the media](#) without becoming a member.

CTAN search engines

Several hosts offer a searchable index of CTAN files:

USA from TUG (by Jim Hefferon)

Search engines: [Search CTAN files](#), [Search Catalogue](#), [Search TeX FAQ](#)

Search URL: <http://www.ctan.org/tools/cataloguesearch?filename=macros/latex>

Results of your Catalogue search for: macros/latex/required/graphics

These are the Catalogue entries that match your request.

Often, the easiest way to get a package is to download its directory. Clicking on the link in the third column gives you a filesearch for the match's directory, and also directions on the easiest way to get a whole directory.

Entry	Brief abstract	CTAN link
color	Allows text and page background colors to be set.	macros/latex/required/graphics/
epsfig	Include Encapsulated PostScript in LaTeX documents.	macros/latex/required/graphics/
graphics	none	macros/latex/required/graphics/
graphicsx	Better support for graphics.	macros/latex/required/graphics/
grfguide	none	macros/latex/required/graphics/
keyval	Process 'key=value' schemes.	macros/latex/required/graphics/
lscap	Place selected parts of a document in landscape.	macros/latex/required/graphics/
trig	none	macros/latex/required/graphics/

Abbildung C.2: CTAN-Homepage und TeX-Katalogeintrag

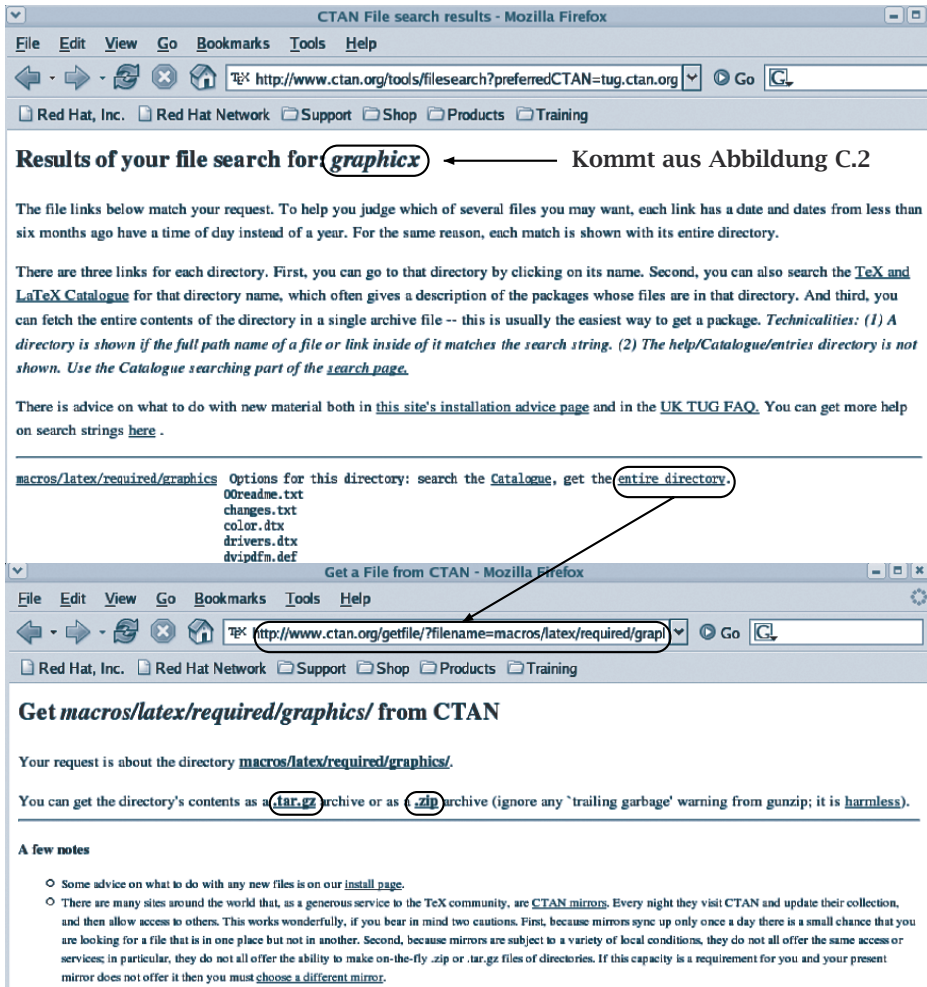


Abbildung C.3: Verwenden der CTAN-Webschnittstelle

sehen ist. Auf dieser Seite hat der Anwender die Wahl zwischen einem gezippten tar-Archiv oder einem zip-Archiv. Durch einen Rechtsklick auf einen der beiden Zeiger (untere Ovale in Abbildung C.3) wird das Archiv im gewünschten Format auf den Rechner des Anwenders heruntergeladen, und die Dateien können installiert werden.

C.3.3 Dateien über die Kommandozeile herunterladen

Wenn man die Internetadresse des Paketes kennt, das man herunterladen möchte (zum Beispiel durch eine Internetsuche), ist es möglicherweise praktischer, das Archiv direkt – das heißt ohne einen Web-Browser zu benutzen – über die Kommandozeile zu holen. In diesem Fall kann man FTP oder das Programm wget verwenden. Letzteres ermöglicht es, Dateien nicht interaktiv

aus dem Internet herunterzuladen. Es unterstützt die Protokolle HTTP, HTTPS, und FTP. Das folgende Beispiel zeigt eine typische nicht interaktive Sitzung (Befehlseingaben vom Anwender sind unterstrichen).

```
> wgetftp://ftp.dante.de/tex-archive/macros/latex/required/graphics.zip
--18:13:27--
ftp://ftp.dante.de/tex-archive/macros/latex/required/graphics.zip
      => 'graphics.zip'
Resolving ftp.dante.de... 80.237.210.73
Connecting to ftp.dante.de[80.237.210.73]:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.      ==> PWD ... done.
==> TYPE I ... done.    ==> CWD /tex-archive/macros/latex/required ... done.
==> PASV ... done.      ==> RETR graphics.zip ... done.
Length: 361,065 (unauthoritative)

100%[=====>] 361,065      378.48K/s

18:13:28 (377.84 KB/s) - 'graphics.zip' saved [361,065]
```

Alternativ kann man das FTP-Protokoll manuell verwenden. Nachdem man hier eine FTP-Verbindung zur CTAN-Seite ([ftp.dante.de](ftp://ftp.dante.de)) aufgebaut hat, meldet man sich mit Benutzernamen anonymous oder ftp an. Als Passwort *muss* die eigene E-Mail-Adresse angegeben werden. Im Beispiel soll das `graphics`-Paket übertragen werden. Deshalb besteht der erste Schritt darin, in das Verzeichnis zu wechseln, in dem sich die Datei befindet (cd tex-archive/macros/latex/required). Nach einer Anfrage über den Inhalt des Verzeichnisses (mit dem Befehl ls) kann das zip-Archiv heruntergeladen werden. Mit dem Befehl quit wird die FTP-Sitzung beendet.

```
> ftp ftp.dante.de
Connected to ftp.dante.de (80.237.210.73).
220 ProFTPD 1.2.10 Server (CTAN) [80.237.210.73]
Name (ftp.dante.de:goossens): ftp
331 Guest login ok, send your complete e-mail address as password.
Password: uuu.vvv@xxx.zz (Hier die eigene E-Mail-Adresse verwenden!)
230 Anonymous access granted, restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd tex-archive/macros/latex/required
250 CWD command successful
ftp> ls
227 Entering Passive Mode (80,237,210,73,145,185).
150 Opening ASCII mode data connection for file list
drwxrwxr-x  6 ftpmaint server      94 Oct 22  2004 amslatex
-rw-rw-r--  1 ftpmaint server    2121853 May  1 17:26 amslatex.zip
drwxrwxr-x  2 ftpmaint server     4096 Apr  1 22:03 babel
-rw-rw-r--  1 ftpmaint server   3098120 May  1 17:25 babel.zip
drwxrwsr-x  2 ftpmaint server     4096 Mar  1  2004 cyrillic
-rw-rw-r--  1 ftpmaint server    37586 May  1 17:25 cyrillic.zip
drwxrwsr-x  2 ftpmaint server     4096 Dec 20 14:43 graphics
-rw-rw-r--  1 ftpmaint server    361065 May  1 17:25 graphics.zip
drwxrwxr-x  2 ftpmaint server     4096 Apr 12 15:26 psnfss
-rw-rw-r--  1 ftpmaint server   1068096 May  1 17:25 psnfss.zip
drwxrwsr-x  2 ftpmaint server     4096 Mar  1  2004 tools
-rw-rw-r--  1 ftpmaint server    280673 May  1 17:25 tools.zip
226 Transfer complete.
```

```
ftp> get graphics.zip
local: graphics.zip remote: graphics.zip
227 Entering Passive Mode (80,237,210,73,145,193).
150 Opening BINARY mode data connection for graphics.zip (361065 bytes)
226 Transfer complete.
361065 bytes received in 0.832 secs (4.2e+02 Kbytes/sec)
ftp> quit
221 Goodbye.
```

C.4 Dokumentation auf dem eigenen System

Wenn man ein L^AT_EX-Paket verwenden möchte, wäre es schön, wenn man die Dokumentation studieren könnte, ohne sich merken zu müssen, wo die relevanten Dateien im T_EX-System abgelegt sind. Hier gibt es zwei kleine Helfer für die Suche: `texdoc` und sein Abkömmling `texdoctk`.

C.4.1 `texdoc` – Kommandozeilesuche nach Namen

Thomas Esser entwickelte das Programm `texdoc`, das Teil der T_EX Live-Distribution ist. Wenn man den Namen der Datei kennt, die ein Paket beschreibt, kann man folgendermaßen nach den entsprechenden Dokumentationsdateien suchen:

```
texdoc -l pspicture
/TeXlive/tl7/texmf/doc/latex/carlisle/pspicture.dvi
/TeXlive/tl7/texmf/doc/html/catalogue/entries/pspicture.html
```

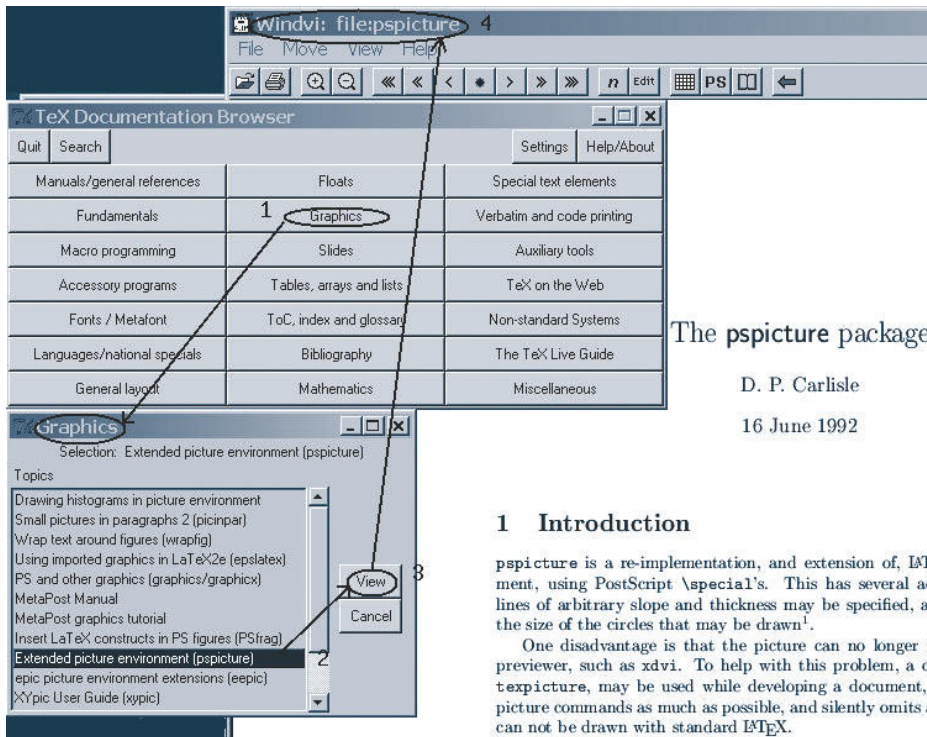
Die Option `-l` weist `texdoc` an, nur den Pfad zu den Dateien aufzuführen, die dem Suchkriterium entsprechen (in diesem Fall Dateien namens `pspicture` mit beliebiger Erweiterung). Wenn die Option `-l` nicht angegeben wird, gibt `texdoc` den Inhalt der Dokumentationsdatei aus (in diesem Fall `pspicture.dvi`). Dazu wird ein passendes Anzeigeprogramm verwendet (z.B. `xdvi` oder `Windvi`).

Wenn man den genauen Namen der Datei nicht kennt, kann man die Option `-s` verwenden und bei der Suche Platzhalter angeben, z.B. folgendermaßen:

```
texdoc -s *picture*
/TeXlive/tl7/texmf/doc/generic/mfpic/examples/lapictures.tex
/TeXlive/tl7/texmf/doc/generic/mfpic/examples/pictures.tex
/TeXlive/tl7/texmf/doc/latex/carlisle/pspicture.dvi
/TeXlive/tl7/texmf/doc/html/catalogue/entries/pspicture.html
/TeXlive/tl7/texmf/doc/html/catalogue/entries/pspicture.xml
```

Hier wurde nach Dateien gesucht, in deren Namen die Zeichenfolge `picture` vorkommt – unter anderem wurden dabei auch die `pspicture`-Dateien gefunden, die schon in der vorherigen Suche aufgelistet wurden.

Das Hilfsprogramm `texdoc` ist recht nützlich, hat aber einen Nachteil: man muss den Namen der Datei kennen, die das gewünschte Paket beschreibt. Dieser ist nicht immer identisch mit dem Namen des Paketes selbst (wie bei `pspicture` in den obigen Beispielen).

Abbildung C.4: Dokumentation mit dem `texdoctk`-Programm suchen

C.4.2 `texdoctk` – Themensuche mit graphischer Oberfläche

Thomas Ruedas wählte einen etwas anderen Ansatz, um einen einfachen Zugang zur Dokumentation von Dateien zu bieten, die im $\text{T}_{\text{E}}\text{X}$ -System vorhanden sind. Sein `texdoctk`-Programm verwendet eine graphische Benutzeroberfläche, die auf `perl` und `Tk` basiert. Das Programm verwendet eine Datenbank, die Dokumentationsdateien, die in Thomas Essers `tetex`-Distribution vorhanden sind ($\text{T}_{\text{E}}\text{X}$ Live basiert auf `tetex`), in 17 Kategorien unterteilt. Außerdem bietet es eine 18. „benutzerdefinierte“ Kategorie, in welcher Anwender bei Bedarf eigene (lokale) Dokumentationseinträge in die Datenbank vornehmen können. Wie bei `texdoc` werden zur Anzeige entsprechende Anzeige- oder Ausgabeprogramme verwendet (z.B. `xdvi` oder `dvips`).

Abbildung C.4 zeigt, wie man das `texdoctk`-System verwendet, um die Dokumentation für das `pspicture`-Paket anzuzeigen. In diesem Fall war es nicht erforderlich, den Namen des Paketes zu kennen. Die Suche wird auf dem Hauptbildschirm gestartet, auf dem (1) die Kategorie ausgewählt wird (hier `Graphics`). Dadurch wird ein Menü eingeblendet (hier `Graphics`, unten links), aus welchem der Suchbegriff (`Extended picture environment (pspicture)`) ausgewählt wird (2). Anschließend klickt man auf die Schaltfläche `View` (3), die den `.dvi`-Viewer `Windvi` aufruft (4), der den Text der Dokumentation anzeigt.

In der Abbildung sind alle verfügbaren Dokumentationskategorien zu sehen (man achte auf die Schaltfläche Miscellaneous für Sonderfälle in der unteren rechten Ecke) sowie die Schaltflächen Search und Help für die erweiterte Suche.

C.5 T_EX-Usergroups

In mehreren Ländern gründeten T_EX-Anwender T_EX-Usergroups, oftmals organisiert nach Sprachgemeinschaften, z.B. DANTE für alle deutschsprachigen T_EX-Anwender. Wer Hilfe benötigt, sollte sich zunächst an seine lokale Usergroup wenden, denn diese ist möglicherweise in der Lage, eine Lösung zu finden, die für seine sprachspezifische Arbeitsumgebung am besten geeignet ist. Nachstehend sind einige Informationen über offiziell registrierte Usergroups aufgeführt (vollständigere und möglicherweise aktuellere Listen finden sich unter <http://www.tug.org/lugs.html> oder <http://www.servalys.nl/lug/>). Sie können helfen, T_EX-bezogenes Material auf DVDs oder andere Publikationen zu beziehen.

cn: China PR <i>Name:</i> Chinese TeX Users Group <i>Sprache:</i> chinesisch <i>Homepage:</i> www.rons.net.cn <i>Kontakt:</i> Hong Feng <i>Adresse:</i> RON's Datacom Co., Ltd. 79, DongWu Ave., Wuhan, Hubei Province 430040 China P.R. <i>E-Mail:</i> info@mail.rons.net.cn <i>Telefon:</i> +862783222108 <i>Fax:</i> +862783222108	<i>Sprache:</i> dänisch <i>Kontakt:</i> Kaja Christiansen <i>Adresse:</i> Department of Computer Science Ny Munkegade, Bldg. 540 DK-8000 Århus C Dänemark <i>E-Mail:</i> board@tug.dk <i>Homepage:</i> www.tug.dk <i>Telefon:</i> +4589423220
cz: Tschechische Republik <i>Sprache:</i> tschechisch <i>Name:</i> CsTUG <i>Kontakt:</i> Petr Sojka <i>Adresse:</i> CsTUG, c/o FI MU Botanická 68a CZ-602 00 Brno Tschechische Republik <i>E-Mail:</i> cstug@cstug.cz <i>Homepage:</i> www.cstug.cz <i>Telefon:</i> +420541212352	ee: Estland <i>Name:</i> Estonian User Group <i>Adresse:</i> Astrophysical Observatory, Toravere Enn Saar, Tartu EE 2444 Estland <i>E-Mail:</i> saar@aai.ee
de: Deutschland <i>Name:</i> DANTE e.V. <i>Sprache:</i> deutsch <i>Kontakt:</i> Volker Schaa <i>Adresse:</i> Postfach 101840 D-69008 Heidelberg Deutschland <i>E-Mail:</i> dante@dante.de <i>Homepage:</i> www.dante.de <i>Telefon:</i> +49622129766 <i>Fax:</i> +496221167906	es: Spanien (CervanTeX) <i>Name:</i> CervanTeX <i>Sprache:</i> spanisch <i>E-Mail:</i> secretario@cervantex.org <i>Homepage:</i> www.cervantex.org
esc: Spanien (Katalonien) <i>Name:</i> Catalan TeX Users Group <i>Sprache:</i> katalanisch <i>Kontakt:</i> Gabriel Valiente <i>Adresse:</i> Technical University of Catalonia Jordi Girona Salgado, 1-3 E-08034 Barcelona Spanien <i>E-Mail:</i> valiente@lsi.upc.es <i>Homepage:</i> www-lsi.upc.es/~valiente/tug-catalan.html	

fr: Frankreich
Name: GUTenberg
Sprache: französisch
Adresse: c/o Irisa
 Campus Universitaire de
 Beaulieu
 F-35042 Rennes cedex
 Frankreich
E-Mail: gut@irisa.fr
Homepage: www.gutenberg.eu.org
Telefon: +33681665102
Fax: +33492579667

fra: Frankreich (Astex)
Kurzname: AsTeX
Sprache: französisch
Adresse: Association AsTeX
 BP 6532
 45066 Orléans cedex 2
 Frankreich
E-Mail: astex-admin@
 univ-orleans.fr
Homepage: www.univ-orleans.fr/EXT/
 ASTEX/astex/doc/en/web/
 html/astex000.htm
Telefon: +33238640994

gr: Griechenland
Name: Greek TeX Friends Group
Sprache: griechisch
Kontakt: Apostolos Syropoulos
 366, 28th October Str.
Adresse: GR-671 00 Xanthi
 Griechenland
E-Mail: eft@ocean1.ee.duth.gr
Homepage: obelix.ee.duth.gr/ef/
Telefon: +3054128704

hu: Ungarn
Name: MaTeX
Sprache: ungarisch
Adresse: Institute of Mathematics and
 Informatics
 University of Debrecen
 H-4010 Debrecen, P.O. Box 12
 Ungarn
E-Mail: matex@math.klte.hu
Homepage: www.math.klte.hu/~matex/

in: Indien
Name: TUGIndia
Kontakt: K.S.S. Nambooripad
Adresse: Kripa, TC 24/548, Sashta
 Gardens
 Thycaud, Trivandrum 695014
 Indien
E-Mail: tugindia@river-valley.
 com
Homepage: www.river-valley.com/
 tug/
Telefon: +91471324341
Fax: +91471333186

kr: Korea

Name: KTUG
Sprache: koreanisch
Kontakt: Kim Kangsu
E-Mail: info@mail.ktug.or.kr
Homepage: www.ktug.or.kr

lt: Litauen
Name: Lietuvos TeX'o Vartotoju
 Grupė
Kontakt: Vytas Statulevicius
Adresse: Akademijos 4
 LT-2600 Vilnius
 Litauen
E-Mail: vytass@ktl.mii.lt
Telefon: +3702359609
Fax: +3702359804

mx: Mexiko
Name: TeX México
Adresse: Rayon No. 523, Centro 58000
 Morelia, Michoacan
 Mexiko
E-Mail: tex@ciencia.dcc.umich.mx
Homepage: ciencia.dcc.umich.mx.
 /tex/
Telefon: +52143128724
Fax: +52143173945

**nl: Niederlande, Belgien
 (flämischer Teil)**
Name: NTG
Sprache: holländisch
Kontakt: Hans Hagen
Adresse: Pragma
 Ridderstraat 27
 8061 GH Hasselt
 Niederlande
E-Mail: info@ntg.nl
Homepage: www.ntg.nl
Telefon: +31384775369
Fax: +31384775374

no: Nordische Staaten
Name: NTUG
Sprache: skandinavische Sprachen
discussion: nordictex@ifi.uio.no
Kontakt: Dag Langmyhr
Adresse: University of Oslo
 PO Box 1080 Blindern
 N-0316 Oslo
 Norwegen
E-Mail: dag@ifi.uio.no
Homepage: www.ifi.uio.no/~dag/
 ntug/
Telefon: +4722852450
Fax: +4722852401

ph: Philippinen
Name: TUG-Philippines
Kontakt: Felix P. Muga II
Adresse: Ateneo de Manila University
 Loyola Heights
 Quezon City
 Philippinen

- E-Mail:* fpmuga@admu.edu.ph
Telefon: +6324266001 ext 2515
Fax: +6324266008
- pl: Polen**
Name: GUST
Sprache: polnisch
Adresse: UCI UMK
 Gagarina 7
 87-100 Toruń
 Polen
E-Mail: sekretariat@gust.org.pl
Homepage: www.GUST.org.pl
- pt: Portugal**
Name: GUTpt
Sprache: portugiesisch
Kontakt: Pedro Quaresma de Almeida
Adresse: Coimbra University
 Dep. Matemática, Largo
 D.Dinis
 Apartado 3008, 3001-454
 COIMBRA
 Portugal
E-Mail: GUTpt@hilbert.mat.uc.pt
Homepage: http://hilbert.mat.uc.pt/~GUTpt/
Telefon: +351239791181
- ru: Russland**
Name: CyrTUG
E-Mail: cyrtug@mir.msk.su
Homepage: www.cemi.rssi.ru/cyrtug/
Diskussion: CyrTeX-en@vsu.ru
Anmeldung: CyrTeX-en-on@vsu.ru
- si: Slowenien**
Name: TeXCeH
Kontakt: Vladimir Batagelj
Adresse: Jadranska 19
 SI-61111 Ljubljana
 Slowenien
E-Mail: Tex.Ceh@fmf.uni-lj.si
Homepage: vlado.fmf.uni-lj.si/texceh/texceh.htm
- uk: Großbritannien**
Name: UKTUG
Sprache: englisch
E-Mail: uktug-enquiries@tex.ac.uk
Homepage: uk.tug.org
Kontakt: Dr R.W.D. Nickalls
Adresse: Department of Anaesthesia
 Nottingham City Hospital
 NHS Trust
 Hucknall Road
 Nottingham, NG5-1PB (UK)
E-Mail: enxw1@nottingham.ac.uk
Telefon: +441159691169 (ext. 45637)
Fax: +441159627713
- us: TeX User Group (international)**
Name: TUG
Adresse: P.O. Box 2311
 Portland, OR 97208-2311
 U.S.A.
E-Mail: office@tug.org
Homepage: www.tug.org
Telefon: +15032239994
Fax: +15032233960
- vn: Vietnam**
Name: ViệtTUG
Kontakt: Nguyễn-Dai Quý
Adresse: LTAS-University of Liège
 Rue des Chevreuils, 1
 Bât B52, Local 522B
 B4000, Liège
 Belgien
E-Mail: viettug@eGroups.com
Telefon: +3243669098
Fax: +3243669311

Anhang D

LB2 T_EX-CD

Der Ursprung – Das T _E X-Live-System	1007
L ^A T _E X von der CD-ROM installieren	1008
L ^A T _E X direkt von der CD-ROM ausführen	1008
Die Beispiele zum L ^A T _E X-Begleiter	1008
Lizenzen	1009

Mithilfe der CD am Ende dieses Buches lässt sich eine L^AT_EX-Umgebung aufsetzen, die der in diesem Buch beschriebenen so weit wie möglich gleicht. Dieser Anhang beschreibt die Historie der CD und wie man sie nutzen kann.

Der Ursprung – Das T_EX-Live-System

T_EX-Live ist eine „Open Source“-Distribution von T_EX und L^AT_EX, die von einer internationalen Vereinigung von T_EX-Benutzergruppen gefördert wird. Die LB2-T_EX-CD-ROM ist sehr stark an diese Distribution angelehnt und darum sei hier ausdrücklich all jenen gedankt, die im Laufe der Jahre zur Herstellung und Pflege dieses Systems beigetragen haben.

Das 2003-Release der T_EX-Live-Distribution erschien auf drei Medien: einer DVD mit der vollständigen Distribution und einer Kopie der CTAN-Archive, einer CD mit einer vollständigen T_EX-Live-Distribution (in komprimierter Form) und einer „Demo“-CD mit einer T_EX-Live-Distribution, die sich entweder auf Festplatte installieren oder direkt von der CD starten lässt.

Damit alles auf eine CD passt, musste bei der „Demo“-CD auf einige Pakete verzichtet werden. Aus dem gleichen Grund werden auch nur die Betriebssysteme Linux, Windows und Mac OS X unterstützt

Die LB2-T_EX-CD-ROM ist eine besondere Version der T_EX-Live-„Demo“-CD. Alle Binärprogramme sind unverändert, verschiedene in diesem Buch beschriebene Pakete wurden aktualisiert oder hinzugefügt und das L^AT_EX-Format selbst entspricht dem Release vom 1.12.2003. Aufgrund des begrenzten Platzes mussten einige Pakete entfernt werden. Die Datei `readme-lb2.html` enthält eine vollständige Liste aller geänderten Pakete. Sie befindet sich im Stammverzeichnis der CD.

L^AT_EX von der CD-ROM installieren

Diese CD-ROM ist genauso zu verwenden und zu installieren, wie es für die T_EX-Live-„Demo“-CD in der Original T_EX-Live-Dokumentation beschrieben wird. Die Datei `readme.html` gibt einen Überblick über das Installationsverfahren mit Links zu ausführlicheren Beschreibungen auf der CD. (Ein Großteil der T_EX-Live-Dokumentation ist in mehreren Sprachen verfügbar.)

Bei einer Installation unter Linux oder Mac OS X sollte man das im Stammverzeichnis befindliche Installationsskript `install-tl.sh` ausführen. Unter Windows sollte das Installationsprogramm automatisch ausgeführt werden (ansonsten genügt ein Doppelklick auf die Datei `autorun.exe`). Der Prozess führt durch einige Konfigurationsmöglichkeiten und installiert demgemäß ein L^AT_EX-System auf der Festplatte. Entsprechend der gewählten Optionen werden einige eher seltener benötigte Pakete nicht direkt installiert. In der T_EX-Live-Dokumentation ist beschrieben, wie man diese nachträglich zur lokalen Installation hinzufügen kann.

Wer L^AT_EX bereits verwendet, möchte vielleicht nicht das gesamte System installieren, sondern die CD nur nutzen, um die L^AT_EX-Basisdateien und einige ausgewählte Pakete zu aktualisieren. Unter Linux und Mac OS X kann man mithilfe des Skripts `install-pkg.sh` Pakete einzeln nachinstallieren. Unter Windows dient die Option „T_EX Live/maintenance“ im Startmenü diesem Zweck.

L^AT_EX direkt von der CD-ROM ausführen

Anstatt das gesamte System auf Festplatte zu installieren, kann man die Software auch direkt von der CD starten. Es wird trotzdem Speicherplatz auf dem lokalen Rechner benötigt, damit T_EX seine Ausgabedateien schreiben und gegebenenfalls zusätzliche Schriften generieren kann.

Um das T_EX-Live-System unter Windows direkt von der CD zu starten, wählt man im Begrüßungsprogramm von T_EX-Live den Menüpunkt `Explore CD-Rom/Run TeX off CD-Rom` (CD-ROM durchsuchen/TeX von CD-ROM starten). Auf den anderen Systemen sollte man das zuvor bereits erwähnte Skript `install-tl.sh` ausführen und dort die entsprechende Option auswählen, die das Programm direkt von der CD startet.

Diese Installation stellt nicht nur ein lauffähiges T_EX-System zur Verfügung, sondern richtet außerdem `xemacs` als Umgebung zum Erstellen von Dokumenten ein. Dieser Editor enthält eine umfangreiche Sammlung von Menüoptionen, die beim Bearbeiten von L^AT_EX-Dokumenten sowie beim Ausführen von L^AT_EX und zugehörigen Programmen, wie etwa BibT_EX, nützlich sind.

Die Beispiele zum L^AT_EX-Begleiter

Die Dateien für alle im Buch abgebildeten Beispiele befinden sich auf der CD im Verzeichnis `Books/lb2/examples`. Der Dateiname setzt sich aus der Beispielenummer und der Dateierweiterung `.ltx` oder `.ltx2` (für doppelseitige Beispiele) zusammen, wie etwa `1-3-1.ltx` und `2-4-4.ltx2`.

Die meisten dieser Beispiele verwenden die Klassendatei `lb2exa.cls`, die sich im gleichen Verzeichnis wie die Beispiele befindet. Es handelt sich um

eine kleine Erweiterung der `article`-Klasse: Sie definiert einige zusätzliche Befehle, mit denen die Anzeige der Präambelbefehle im Buch gesteuert wird.

Wird das T_EX-System direkt von der CD gestartet, dann sind alle für die Beispiele benötigten Pakete verfügbar, mit Ausnahme einiger Pakete, die kommerzielle Fonts erfordern, welche naturgemäß nicht auf dieser CD-ROM verbreitet werden dürfen.

Wird die Distribution auf der Festplatte eingerichtet, so werden gemäß der Voreinstellung nicht alle Pakete mit installiert. Gegebenenfalls müssen dann noch einzelne Pakete nachinstalliert werden, um alle Beispiele ausführen zu können.

Lizenzen

Die Datei `LICENSE.TL` im Stammverzeichnis beschreibt die Lizenz und die Bedingungen zum Kopieren des T_EX-Live-Systems. Diese gelten auch für die veränderte Distribution auf der LB2-T_EX-CD-ROM. Alle auf dieser CD-ROM enthaltene Software darf (nach dem besten Wissen der Autoren) frei verbreitet werden, auch wenn die einzelnen Komponenten verschiedenen Lizenzbedingungen unterliegen, die in der jeweiligen Paketdokumentation beschrieben sind.

Viele der L^AT_EX-Pakete und alle Beispieldateien in diesem Buch werden unter der „L^AT_EX Project Public License“ (LPPL) verteilt. Der entsprechende Lizenztext ist auf der CD in der Datei `texmf/doc/latex/base/lppl.txt` gespeichert. Die LPPL erlaubt jegliche Verwendung, einschließlich Vervielfältigung und Veränderung, solange die geänderten Kopien nicht unter dem gleichen Namen verteilt werden, wie die Originaldateien.

Literaturverzeichnis

- [1] Adobe Systems Incorporated. Adobe Type 1 Font Format. Addison-Wesley, Reading, MA, USA, 1990. ISBN 0-201-57044-0.

Das „Black Book“ enthält die Spezifikationen für das Adobe Type 1-Fontformat und beschreibt, wie man ein Type 1-Fontprogramm erstellt. Das Buch erläutert die Details der Type 1-Syntax (einer Untermenge von PostScript). Dabei behandelt es auch den Aufbau von Zeichensatzprogrammen, Verfahren zum Festlegen von Outline-Schriften für Computer und den Inhalt der verschiedenen „font dictionaries“ in PostScript. Außerdem befasst es sich mit den Themen Verschlüsselung, Subroutinen und Hinting.

http://partners.adobe.com/public/developer/en/font/T1_SPEC.PDF

- [2] Adobe Systems Incorporated. *Encapsulated PostScript File Format Specification (Version 3.0)*. Technische Dokumentation 5002, 1992.

In dieser technischen Dokumentation wird das Format für Encapsulated PostScript-Dateien (EPSF) detailliert beschrieben. Mithilfe dieses Standardformates lassen sich PostScript-Dateien zwischen Anwendungen in vielen verschiedenen Umgebungen importieren und exportieren. Das EPSF-Format basiert auf und richtet sich nach den Konventionen zum Strukturieren von Dokumenten (DSC) [3].

http://partners.adobe.com/public/developer/en/ps/5002.EPSF_Spec.pdf

- [3] Adobe Systems Incorporated. *PostScript Document Structuring Conventions Specification (Version 3.0)*. Technische Dokumentation 5001, 1992.

In dieser technischen Dokumentation werden Standardkonventionen zum Strukturieren von Dokumenten (DSC, „Document Structuring Conventions“) definiert, mit deren Hilfe die Geräteunabhängigkeit von PostScript-Dokumenten gewährleistet werden soll. Auf Basis der DSC können PostScript-verarbeitende Programme Dokumentenverwaltungssystemen mitteilen, wie ihre Dokumente aufgebaut sind und was für den Ausdruck erforderlich ist, ohne dass dabei die PostScript-Seitenbeschreibung beeinträchtigt wird.

http://partners.adobe.com/public/developer/en/ps/5001.DSC_Spec.pdf

- [4] Adobe Systems Incorporated. PostScript Language Reference. Addison-Wesley, Reading, MA, USA, dritte Auflage, 1999. ISBN 0-201-37922-8.

Man kann das „Red Book“ als das maßgebliche Hilfsmittel für alle PostScript-Programmierer betrachten. Darin wird die gesamte PostScript-Sprache einschließlich der neuesten Level3-Operatoren beschrieben.

<http://www.adobe.com/products/postscript/pdfs/PLRM.pdf>

- [5] Adobe Systems Incorporated. PDF Reference, version 1.4. Addison-Wesley, Boston, MA, USA, dritte Auflage, 2002. ISBN 0-201-75839-3. Spezifikation des Adobe PDF-Formates (Portable Document Format). Das Buch stellt alle Aspekte des PDF-Formates vor und erläutert diese. Dazu gehören auch die Architektur und das Imaging-Modell (das durchsichtige und deckende Text-, Bild- und Graphikelemente ermöglicht), die Befehlssyntax, die Graphikoperatoren, Schriften und Rendering sowie das Verhältnis zwischen PostScript und PDF.
<http://partners.adobe.com/public/developer/en/pdf/PDFReference.pdf>
- [6] American Mathematical Society, Providence, Rhode Island. Instructions for Preparation of Papers and Monographs: $\mathcal{A}_M\mathcal{S}$ - $\text{\LaTeX}$, Dezember 1999.
Dieses Dokument erklärt, in welchem Stil Autoren, die mithilfe von $\text{\LaTeX}$ Artikel und Bücher für die American Mathematical Society (AMS) verfassen, ihre Dokumente gestalten müssen: Zeitschriften (amsart), Konferenzberichte (amspc) und Monographien (amsbook).
<ftp://ftp.ams.org/pub/author-info/documentation/amslatex/instr-1.pdf>
- [7] American Mathematical Society, Providence, Rhode Island. Using the amsthm Package (Version 2.07), Juni 2000.
Das amsthm-Paket bietet eine erweiterte Version des $\text{\LaTeX}$ -Befehls `\newtheorem` zum Definieren theoremmähnlicher Umgebungen. Es erkennt `\theoremstyle`-Angaben und stellt die Umgebung `proof` zur Verfügung.
<http://ftp.ams.org/pub/tex/doc/amscs/amsthdoc.pdf>
- [8] American Mathematical Society, Providence, Rhode Island. User's Guide for the amsmath Package (Version 2.0), Februar 2002.
Das amsmath-Paket, das von der American Mathematical Society entwickelt wurde, bietet viele zusätzliche Funktionen für das Setzen von Formeln.
<http://www.ams.org/tex/amslatex.html>
- [9] American Mathematical Society, Providence, Rhode Island. User's Guide to AMSFonts Version 2.2d, Januar 2002.
In diesem Dokument werden die AMSFonts der American Mathematical Society, eine Sammlung von Symbolzeichensätzen und einiger Alphabete, beschrieben.
<http://www.ams.org/tex/amsfonts.html>
- [10] J. André und Ph. Louarn. *Notes en bas de pages : comment les faire en $\text{\LaTeX}$?* *Cahiers GUTenberg*, 12:57–70, Dezember 1991.
Hier werden Spezialfälle von Fußnoten in $\text{\LaTeX}$ besprochen – z.B. wie man in einer Fußnote auf Informationen einer tabular- oder minipage-Umgebung verweist oder wie man sich mehrmals auf die gleiche Fußnote bezieht.
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/12-louarn.pdf>
- [11] Michael Barr. *A new diagram package*, 2001.
Eine Neufassung des ursprünglichen diagram-Paketes von Michael Barr, das als Frontend für Roses xypic-Paket dient (siehe [58, Kapitel 5]). Es bietet allgemeine Funktionen zum Zeichnen von Pfeilen; verschiedene gängige Diagrammformen, wie etwa Quadrate, Dreiecke, Würfel und 3×3 -Diagramme; kleine zweiseitige Pfeile, die sich in Diagrammen beliebig platzieren lassen; und Zugriff auf sämtliche xypic-Funktionen.
Auf CTAN unter: `macros/generic/diagrams/barr`
- [12] Claudio Beccari und Apostolos Syropoulos. *New Greek fonts and the greek option of the babel package*. *TUGboat*, 19(4):419–425, Dezember 1998. ISSN 0896-3207.
Hier wird ein neues vollständiges Set griechischer Schriften beschrieben und wie sich diese in Kombination mit der greek-Erweiterung zum babel-Paket verwenden lassen.
<http://www.tug.org/TUGboat/Articles/tb19-4/tb61becc.pdf>

- [13] Nelson Beebe. *Bibliography prettyprinting and syntax checking*. *TUGboat*, 14(4):395–419, Dezember 1993. ISSN 0896-3207.
In diesem Artikel werden drei Software-Tools zur Unterstützung von $\text{\LaTeX}$ beschrieben: ein Pretty-Printer, ein Programm für die Syntaxkontrolle und eines für die lexikalische Analyse von $\text{\LaTeX}$ -Dateien. Alle zusammen werden unter der Bezeichnung bibclean geführt. <http://www.tug.org/TUGboat/Articles/tb14-4/tb41beebe.pdf>
- [14] Barbara Beeton. *Mathematical symbols and Cyrillic fonts ready for distribution*. *TUGboat*, 6(2):59–63, Juli 1985. ISSN 0896-3207.
Bekanntgabe der ersten allgemeinen Ausgabe der Fonts der Euler-Serie durch die American Mathematical Society.
<http://www.tug.org/TUGboat/Articles/tb06-2/tb11beet.pdf>
- [15] Frank G. Bennett, Jr. *CAMEL: kicking over the bibliographic traces in $\text{\LaTeX}$* . *TUGboat*, 17(1):22–28, März 1996. ISSN 0896-3207.
Das camel-Paket bietet eine einfache, logisch aufgebaute Zitierschnittstelle für $\text{\LaTeX}$, mit deren Hilfe man den Bibliographiestil eines Dokumentes auf einfache Weise ohne intensive Bearbeitung ändern kann.
<http://www.tug.org/TUGboat/Articles/tb17-1/tb50benn.pdf>
- [16] Frank G. Bennett, Jr. *User's Guide to the Camel Citator*, 1997.
Dokumentation der Version 1 des camel-Paketes.
Auf CTAN unter: [macros/latex/contrib/camel](http://www.ctan.org/macros/latex/contrib/camel)
- [17] A. Berdnikov, O. Lapko, M. Kolodin, A. Janishevsky und A. Burykin. *Cyrillic encodings for $\text{\LaTeX} 2_{\epsilon}$ multi-language documents*. *TUGboat*, 19(4):403–416, Dezember 1998. ISSN 0896-3207.
Beschreibung von vier Kodierungen, die entwickelt wurden, um im Mehrsprachenmodus von $\text{\LaTeX} 2_{\epsilon}$ kyrillische Schreibsysteme zu unterstützen. Bei der „Rohkodierung“ $\text{\LaTeX} 2_{\epsilon}$ handelt es sich um eine Sammlung kyrillischer Glyphen, mit deren Hilfe man Textfragmente von Sprachen, die ein modernes kyrillisches Schreibsystem verwenden, in $\text{\LaTeX} 2_{\epsilon}$ -Dokumente einfügen kann. T2A, T2B und T2C hingegen sind echte $\text{\LaTeX} 2_{\epsilon}$ -Kodierungen, die sich gemeinsam mit anderen Sprachkodierungen in einer mehrsprachigen Umgebung verwenden lassen.
<http://www.tug.org/TUGboat/Articles/tb19-4/tb61berd.pdf>
- [18] Karl Berry. *Filenames for fonts*. *TUGboat*, 11(4):517–520, November 1990. ISSN 0896-3207.
In diesem Artikel wird das einheitliche, rationale Schema zur Benennung von Fontdateien beschrieben, das schon seit 15 Jahren Bestand hat. Jeder Name besteht aus bis zu acht Zeichen (für Hersteller, Schriftname, Strichstärke, Variante, Erweiterungsmerkmale und Entwurfsgröße), mit denen jede Schrift eindeutig gekennzeichnet werden kann.
<http://www.tug.org/TUGboat/Articles/tb11-4/tb30berry.pdf>
- [19] Karl Berry. *Fontname: Filenames for $\text{\TeX}$ fonts*, Mai 2003.
Online-Dokumentation der aktuellen Version von „Fontname“, einem Schema für Fontdateinamen in $\text{\TeX}$; hier werden für eine Reihe von Ländern einige rechtliche Fragen im Zusammenhang mit Fonts geklärt. <http://www.tug.org/fontname/html/index.html>
- [20] Javier Bezos. *The accents Package*, 2000.
Verschiedene Werkzeuge für mathematische Akzente: zum Erzeugen künstlicher Akzente aus Symbolen, die keine Akzente sind, zum Gruppieren von Akzenten und zum Platzieren von Akzenten unter Glyphen.
Auf CTAN unter: [macros/latex/contrib/bezos](http://www.ctan.org/macros/latex/contrib/bezos)
- [21] The *Bluebook*: A Uniform System of Citation. The Harvard Law Review Association, Cambridge, MA, 17. Auflage, 2000.
Das *Blue Book* besteht aus drei Hauptteilen: In Teil 1 werden die allgemeinen Zitier- und Formatierungsstandards für Rechtstexte genauer dargelegt; Teil 2 stellt die besonderen Zitierregeln für Fälle (cases), Gesetze (statutes), Bücher (books), Periodika (periodicals) und fremdsprachige sowie internationale Texte vor; und Teil 3 besteht aus einer Reihe von Tabellen, die unter anderem zeigen, welche Instanz zitiert werden soll und wie man richtig abkürzt.
Zu bestellen bei: <http://www.legalbluebook.com>

- [22] Francis Borceux. *De la construction de diagrammes*. *Cahiers GUTenberg*, 5:41–48, Mai 1990.
Die diagram-Makros setzen Diagramme, die aus mathematischen Ausdrücken bestehen, die durch verschiedenartige Pfeile miteinander verbunden sind. Die Makros berechnen automatisch Länge und Position jedes Elementes. Benutzer können für jedes Diagramm einen Skalierungsfaktor festlegen.
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/5-borceux.pdf>
- [23] Francis Borceux. *Diagram 3*, 1993.
Paket für kommutative Diagramme, das den $\text{\LaTeX}$ -Bildmodus verwendet.
Auf CTAN unter: [macros/generic/diagrams/borceux](http://www.ctan.org/macros/generic/diagrams/borceux)
- [24] Thierry Bouche. *Diversity in math fonts*. *TUGboat*, 19(2):120–134, Juni 1998. ISSN 0896-3207.
In diesem Artikel werden Fragen behandelt, die entstehen, wenn man $\text{\LaTeX}$ -Fonts in Formelumgebungen ändert. Es wird versucht, eine effektive Methode vorzuschlagen, mit der man auf eine größere Vielfalt von Fontoptionen zugreifen kann, ohne dabei typographischen Unsinn zu erzeugen.
<http://www.tug.org/TUGboat/Articles/tb19-2/tb59bouc.pdf>
- [25] Johannes Braams. *Babel, a multilingual style-option system for use with $\text{\LaTeX}$'s standard document styles*. *TUGboat*, 12(2):291–301, Juni 1991. ISSN 0896-3207.
Das babel-Paket war ursprünglich eine Sammlung von Optionen für Formatvorlagen zur Unterstützung verschiedener Sprachen. In *TUGboat*, 14(1):60–62, April 1993, wurde eine aktuellere Version veröffentlicht.
<http://www.tug.org/TUGboat/Articles/tb12-2/tb32braa.pdf>
<http://www.tug.org/TUGboat/Articles/tb14-1/tb38braa.pdf>
- [26] Neil Bradley. *The XML Companion*. Addison-Wesley, Boston, MA, USA, dritte Auflage, 2002. ISBN 0-201-77059-8.
Dieses Buch beschreibt XML-Funktionen, ohne dafür HTML- oder SGML-Kenntnisse vorzusetzen. Es behandelt auch verwandte Standards wie Xpath, W3C XML Schema, SAX, DOM, XSLT, Xlink und Xpointer.
- [27] Peter Breitenlohner et al. *The e $\text{\TeX}$ manual (Version 2)*, Februar 1998.
Aktuelles Handbuch des e $\text{\TeX}$ -Systems, das die Fähigkeiten von $\text{\TeX}$ erweitert ohne die Kompatibilität einzuschränken.
Auf CTAN unter: [systems/e-tex/v2/doc/etex_man.pdf](http://www.ctan.org/systems/e-tex/v2/doc/etex_man.pdf)
- [28] Robert Bringhurst. *The elements of typographic style*. Hartley & Marks Publishers, Point Roberts, WA, USA und Vancouver, BC, Kanada, zweite Auflage, 1996. ISBN 0-88179-133-4 (gebunden), 0-88179-132-6 (Taschenbuch).
Ein sehr gutes Buch über Typographie, das besonderen Wert auf die richtige Verwendung von Schriften legt.
- [29] Judith Butcher. *Copy-editing: The Cambridge handbook for editors, authors and publishers*. Cambridge University Press, New York, dritte Auflage, 1992. ISBN 0-521-40074-0.
Ein Referenzwerk für alle, die sich mit dem Erstellen von Manuskripten und Illustrationen für den Druck und deren Veröffentlichung beschäftigen. Das Buch behandelt alle redaktionellen Aspekte: von den Grundlagen, wie man ein Manuskript für Designer und Setzer richtig aufbereitet, über die Grundregeln für verlagsspezifische Richtlinien und Einheitlichkeit, bis zum Lesen und Korrigieren von Druckfahnen.
- [30] David Carlisle, Patrick Ion, Robert Miner und Nico Poppelier, Hg. *Mathematical Markup Language (MathML) Version 2.0*. W3C, zweite Auflage, Oktober 2003.
MathML ist ein XML-Befehlssatz für mathematische Arbeiten, der für Browser und als Mittel zur Kommunikation zwischen Computeralgebrasystemen entwickelt wurde.
<http://www.w3.org/TR/MathML2>

- [31] David Carlisle, Chris Rowley und Frank Mittelbach. *The L^AT_EX3 Programming Language—a proposed system for T_EX macro programming*. *TUGboat*, 18(4):303–308, Dezember 1997. ISSN 0896-3207.
Vorschläge für eine völlig neue L^AT_EX Kernel-Syntax und für passende Softwaretools.
<http://www.tug.org/TUGboat/Articles/tb18-4/tb57rowl.pdf>
- [32] David Carlisle. *A L^AT_EX tour, Part 1: The basic distribution*. *TUGboat*, 17(1):67–73, März 1996. ISSN 0896-3207.
Eine „Führung“ durch die Dateien der L^AT_EX-Basisdistribution. Die Dateinamen und -pfade beziehen sich auf die Dateihierarchie der CTAN-Archive.
<http://www.tug.org/TUGboat/Articles/tb17-1/tb50carl.pdf>
- [33] David Carlisle. *A L^AT_EX tour, Part 2: The tools and graphics distributions*. *TUGboat*, 17(3):321–326, September 1996. ISSN 0896-3207.
Eine „Führung“ durch die Werkzeug- und Graphik-Pakete. Man beachte, dass im L^AT_EX-Manual [106] vorausgesetzt wird, dass Standard-L^AT_EX zumindest über die Graphikdistribution verfügt. <http://www.tug.org/TUGboat/Articles/tb17-3/tb52carl.pdf>
- [34] David Carlisle. *A L^AT_EX tour, Part 3: mfnfss, psnfss and babel*. *TUGboat*, 18(1):48–55, März 1997. ISSN 0896-3207.
Eine „Führung“ durch drei weitere Distributionen, die zu Standard-L^AT_EX gehören. Die mfnfss-Distribution unterstützt einige beliebte METAFONT-Schriften, die über keine andere L^AT_EX-Schnittstelle verfügen. Die psnfss-Distribution besteht aus L^AT_EX-Paketen, die den Zugriff auf PostScript-Fonts ermöglichen. Die babel-Distribution rüstet L^AT_EX mit Funktionen zum Setzen in mehreren Sprachen aus.
<http://www.tug.org/TUGboat/Articles/tb18-1/tb54carl.pdf>
- [35] David Carlisle. *OpenMath, MathML, and XSL*. *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, 34(2):6–11, Juni 2000. ISSN 0163-5824.
Erörterung über den Einsatz von XML in der Mathematik – insbesondere für OpenMath und MathML – und die Verwendung von XSLT für Transformationen zwischen diesen Sprachen. [Eingeschränkt auf Mitglieder der ACM: http://www.acm.org/sigsam/bulletin/issues/issue132.html](http://www.acm.org/sigsam/bulletin/issues/issue132.html)
- [36] David Carlisle. *XMLTEX: A non validating (and not 100% conforming) namespace aware XML parser implemented in T_EX*. *TUGboat*, 21(3):193–199, September 2000. ISSN 0896-3207.
XMLTEX ist ein T_EX-Programm zum Parsen und Setzen, das so eingerichtet ist, dass es den Funktionsumfang des L^AT_EX-Kerns zum Setzen benutzt.
<http://www.tug.org/TUGboat/Articles/tb21-3/tb68carl.pdf>
- [37] Pehong Chen und Michael A. Harrison. *Index Preparation and Processing. Software—Practice and Experience*, 19(9):897–915, September 1988.
Originalbeschreibung des *MakeIndex*-Systems.
- [38] The Chicago Manual of Style. University of Chicago Press, Chicago, IL, USA, 15. Auflage, 2003. ISBN 0-226-10403-6.
Das Standardreferenzwerk über die Gestaltung von Veröffentlichungen in den USA für Autoren und Verleger.
- [39] Adrian F. Clark. *Practical halftoning with T_EX*. *TUGboat*, 12(1):157–165, März 1991. ISSN 0896-3207.
Überblick über die praktischen Probleme beim Setzen von Rasterbildern mit T_EX und Vergleich anderer Verfahren zum Einbinden von Graphiken. Die Vor- und Nachteile der verschiedenen Ansätze werden beschrieben und einige Versuche, Farbauszüge zu erstellen, werden besprochen.
<http://www.tug.org/TUGboat/Articles/tb12-1/tb31clark.pdf>

- [40] Matthias Clasen und Ulrik Vieth. *Towards a new Math Font Encoding for (L^A)T_EX*. *Cahiers GUTenberg*, 28-29:94-121, März 1998.
Ein Prototyp für die Einbindung von 8-Bit-kodierten Mathematikfonts in L^AT_EX. <http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-clasen.pdf>
- [41] Carl Dair. *Design with Type*. University of Toronto Press, Toronto, Ontario, Kanada, 1967. ISBN 0-8020-1426-7 (gebunden), 0-8020-6519-8 (Taschenbuch).
Ein gelungener Überblick über die traditionelle Typographie mit vielen nützlichen Faustregeln.
- [42] Emmanuel Donin de Rosière. *From stack removing in stack-based languages to BibT_EX++*. Diplomarbeit, ENSTBr, September 2003.
Beschreibung von BibT_EX++, einem Programm zum Erstellen von Bibliographieabschnitten mit L^AT_EX, das ein möglicher Nachfolger von BibT_EX sein könnte. Dieses Programm kann BibT_EX-Stildateien (.bst) in Java-Programmcode übersetzen. http://www.lit.enstb.org/~keryell/elevs/ENSTBr/2002-2003/DEA/Donin_de_Rosiere
- [43] Michael Downes. *Breaking equations*. *TUGboat*, 18(3):182-194, September 1997. ISSN 0896-3207.
T_EX hat Schwierigkeiten, Formeln, die über mehr als eine Zeile umbrochen werden müssen, abgesetzt darzustellen. Das breqn-Paket löst viele der wichtigsten Probleme, indem es bei abgesetzten Formeln automatische Zeilenumbrüche unterstützt. <http://www.tug.org/TUGboat/Articles/tb18-3/tb56down.pdf>
- [44] Michael Downes. *The amsrefs L^AT_EX package and the amsxport BibT_EX style*. *TUGboat*, 21(3):201-209, September 2000. ISSN 0896-3207.
Bibliographische Einträge auf Basis des amsrefs-Formates reichen mit ihrer ausgeprägten internen Struktur und ihrem hoch entwickelten Markup an die Qualität heran, die man traditionellerweise in BibT_EX-Datenbankdateien findet. Darüber hinaus kann man mithilfe des amsrefs-Markups den Bibliographiestil vollständig in einer L^AT_EX-Klassendatei festlegen. <http://www.tug.org/TUGboat/Articles/tb21-3/tb68down.pdf>
- [45] Dudenredaktion, Hg. *Duden, Rechtschreibung der deutschen Sprache*. Dudenverlag, Mannheim, 21. Auflage, 1996. ISBN 3-411-04011-4.
Das Standard-Referenzwerk für die richtige Schreibweise und Silbentrennung aller Wörter der modernen deutschen Sprache. Es enthält Beispiele und Erläuterungen für problematische Fälle und vergleicht die Regeln der alten und neuen Rechtschreibung.
- [46] Victor Eijkhout. *T_EX by Topic, A T_EXnician's Reference*. Addison-Wesley, Reading, MA, USA, 1991. ISBN 0-201-56882-9. Vergriffen. Vom Autor kostenlos als PDF-Datei erhältlich.
Systematisch aufgebautes Nachschlagewerk für erfahrene T_EX-Anwender. Das Buch behandelt alle Aspekte von T_EX umfassend mit detaillierten Erläuterungen der zugrundeliegenden Mechanismen sowie zahlreichen Beispielen für Programmierverfahren in T_EX. <http://www.eijkhout.net/tbt>
- [47] Robin Fairbairns. *UK List of T_EX Frequently Asked Questions on the Web*, 2003.
Diese Liste häufig gestellter Fragen zu T_EX wurde vom Committee der U.K. T_EX Users' Group ins Leben gerufen. Sie enthält weit über 300 Einträge und wird regelmäßig aktualisiert und erweitert. <http://www.tex.ac.uk/faq>
- [48] Jürgen Fenn. *T_EX Topical Index*. Onlineresource, Juni 2005.
Diese Onlineresource listet viele der auf CTAN verfügbaren Pakete nach Themen geordnet auf. Dabei richtet sie sich soweit möglich nach der in *The L^AT_EX Companion* (also der englischen Ausgabe des vorliegenden Buches) definierten Systematik. <http://texcatalogue.sarovar.org/bytopic.html>

- [49] Laurence Finston. *Spindex—Indexing with special characters*. *TUGboat*, 18(4):255–273, Dezember 1997. ISSN 0896-3207.
„Common Lisp“-Indexprogramm und unterstützende TeX-Makros für Indexe, die nicht lateinische Zeichen enthalten.
<http://www.tug.org/TUGboat/Articles/tb18-4/tb57fins.pdf>
- [50] Shinsaku Fujita und Nobuya Tanaka. *X_YTeX (Version 2.00) as implementation of the X_YM notation and the X_YM markup language*. *TUGboat*, 21(1):7–14, März 2000. ISSN 0896-3207.
Beschreibung der Version 2 des X_YTeX-Systems. Man kann X_YTeX als ein in TeX-Makros formuliertes lineares Notationssystem ansehen, das der IUPAC-Nomenklatur (International Union of Pure and Applied Chemistry) entspricht. Mit seiner Hilfe lassen sich komplizierte Strukturformeln auf komfortable Weise abbilden.
<http://www.tug.org/TUGboat/Articles/tb21-1/tb66fuji.pdf>
- [51] Shinsaku Fujita und Nobuya Tanaka. *Size reduction of chemical structural formulas in X_YTeX (Version 3.00)*. *TUGboat*, 22(4):285–289, Dezember 2001. ISSN 0896-3207.
Weitere Verbesserungen des X_YTeX-Systems, insbesondere im Bereich der Größenreduzierung von Strukturformeln.
<http://www.tug.org/TUGboat/Articles/tb22-4/tb72fuji.pdf>
- [52] Rei Fukui. *TIPA: A system for processing phonetic symbols in L^AT_EX*. *TUGboat*, 17(2):102–114, Juni 1996. ISSN 0896-3207.
TIPA ist ein System, mit dessen Hilfe sich Symbole des Internationalen Phonetischen Alphabetes (IPA) mit L^AT_EX verarbeiten lassen. Es führt eine neue Kodierung für phonetische Symbole (T3) ein, die alle Symbole und diakritischen Zeichen der neueren IPA-Versionen sowie einige nicht in IPA enthaltene Symbole umfasst. Es unterstützt L^AT_EX 2_ε und bietet eine einfache Eingabemethode in der IPA-Umgebung.
<http://www.tug.org/TUGboat/Articles/tb17-2/tb51rei.pdf>
- [53] Bernard Gaulle. *Comment peut-on personnaliser l'extension french de L^AT_EX?* *Cahiers GUTenberg*, 28–29:143–157, März 1998.
In diesem Artikel wird beschrieben, wie sich das Paket french an persönliche Erfordernisse anpassen lässt.
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-gaulle.pdf>
- [54] Maarten Gelderman. *A short introduction to font characteristics*. *TUGboat*, 20(2):96–104, Juni 1999. ISSN 0896-3207.
Diese Abhandlung erläutert die zentralen Gesichtspunkte beim Beschreiben einer Schrift, ihre Grundmerkmale, elementare numerische Maße, mit denen man sich die Eigenschaften eines Schriftentwurfs erschließen kann, sowie das Konzept „Kontrast“.
<http://www.tug.org/TUGboat/Articles/tb20-2/tb63geld.pdf>
- [55] Charles F. Goldfarb. *The SGML Handbook*. Oxford University Press, London, Oxford, New York, 1990. ISBN 0-19-853737-9.
Der vollständige Text des ISO SGML-Standards [73], vom Autor umfassend kommentiert und durch verschiedene Anleitungen ergänzt.
- [56] Norbert Golluch. *Kleinweich Büro auf Schlabberscheiben*. Eichborn, Frankfurt, 1999. ISBN 3-8218-2438-7.
Technisches Deutsch für Angefangen.
- [57] Michel Goossens, Frank Mittelbach und Alexander Samarin. *The L^AT_EX Companion. Tools and Techniques for Computer Typesetting*. Addison-Wesley, Reading, MA, USA, 1994. ISBN 0-201-54199-8.
Die englische erste Ausgabe dieses Buches.
- [58] Michel Goossens, Sebastian Rahtz und Frank Mittelbach. *The L^AT_EX Graphics Companion: Illustrating Documents with TeX and PostScript*.

Tools and Techniques for Computer Typesetting. Addison-Wesley, Reading, MA, USA, 1997. ISBN 0-201-85469-4.

In diesem Buch wird gezeigt, wie man Graphikdateien in ein $\LaTeX$ -Dokument integriert, wie man mithilfe mehrerer verschiedener Sprachen technische Diagramme programmiert, wie man farbige Bilder erzeugt, wie man mit eingebetteten PostScript-Fragmenten Spezialeffekte erzielt und wie man hochwertige Notenblätter und Spieldiagramme anfertigt. Es enthält zudem ausführliche Beschreibungen wichtiger Pakete, wie `xypic`, `pstricks` und `MetaPost`, der $\LaTeX$ -Standardpakete `color` und `graphics`, von PostScript-Fonts und wie man diese in $\LaTeX$ verwendet, sowie der Programme `dvips` und `ghostscript`.

- [59] Michel Goossens und Sebastian Rahtz. *The $\LaTeX$ Web Companion: Integrating $\TeX$, HTML, and XML*. Tools and Techniques for Computer Typesetting. Addison-Wesley Longman, Reading, MA, USA, 1999. ISBN 0-201-43311-7. Mit Eitan M. Gurari, Ross Moore und Robert S. Sutor.

Dieses Buch lehrt (wissenschaftliche) Autoren, wie man etwas im Web oder einem anderen Hypertext-Präsentationssystem veröffentlicht. Es richtet sich an erfahrene $\LaTeX$ -Anwender und berücksichtigt ihre speziellen Anforderungen in Bereichen wie der Mathematik, nicht europäischer Sprachen und algorithmischer Graphiken. Das Buch erklärt, wie man das Adobe Acrobat Format aus $\LaTeX$ heraus in vollem Umfang nutzen kann, wie man Dokumente in HTML oder XML konvertiert, wie man Formeln in Webanwendungen benutzt, wie sich mithilfe von $\LaTeX$ Webseiten erzeugen lassen, wie man einfache XML/SGML-Daten liest und schreibt und wie man mithilfe von $\TeX$ oder PDF aus XML- oder HTML-Webseiten hochwertige Druckerzeugnisse anfertigt.

- [60] Michel Goossens und Vesa Sivunen. *$\LaTeX$, SVG, Fonts*. *TUGboat*, 22(4):269–279, Dezember 2001. ISSN 0896-3207.

Ein kurzer Überblick über SVG (skalierbare Vektorgraphiken) und ihre Vorteile, wenn es um die Portierbarkeit von Graphikinhalt geht; über das Konvertieren der Outlines von PostScript-Glyphen zu SVG-Outlines; und über die Verwendung von SVG-Glyphen in $\TeX$ -Dokumenten. <http://www.tug.org/TUGboat/Articles/tb22-4/tb72goos.pdf>

- [61] Ronald L. Graham, Donald E. Knuth und Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, MA, USA, zweite Auflage, 1994. ISBN 0-201-55802-5.

Ein Mathematiklehrbuch, das mit $\TeX$ in der Schrift Concrete Roman gesetzt wurde. Siehe auch [96].

- [62] George Grätzer. *Math into $\LaTeX$* . Birkhäuser und Springer-Verlag, Cambridge, MA, USA; Berlin, Deutschland/Basel, Schweiz, and Berlin, Deutschland/Heidelberg, Deutschland/London, UK/ usw., dritte Auflage, 2000. ISBN 0-8176-4131-9, 3-7643-4131-9.

Eine allgemeine Einführung in $\LaTeX$ als Instrument zum Erstellen mathematischer Bücher und Artikel. Zusätzlich zu den grundlegenden $\LaTeX$ -Funktionen werden auch die American Mathematical Society-Dokumentenklassen und -Pakete besprochen.

- [63] George D. Greenwade. *The Comprehensive $\TeX$ Archive Network (CTAN)*. *TUGboat*, 14(3):342–351, Oktober 1993. ISSN 0896-3207.

Ein Überblick über das Konzept, die Entwicklung und die frühere Nutzung des CTAN-Archivs, das alle Dateien, die etwas mit $\TeX$ zu tun haben, im Netz zur Verfügung stellt. <http://www.tug.org/TUGboat/Articles/tb14-3/tb40green.pdf>

- [64] Yannis Haralambous. *Typesetting old German: Fraktur, Schwabacher, Gotisch and initials*. *TUGboat*, 12(1):129–138, März 1991.

Dieser Artikel zeigt, wie man mit METAFONT originalgetreue Kopien altdeutscher Schriften anfertigt. Anhand von Beispielen wird erläutert, nach welchen Regeln diese Schriften gesetzt werden.

<http://www.tug.org/TUGboat/Articles/tb12-1/tb31hara.pdf>

- [65] Horace Hart. *Hart's Rules; For Compositors and Readers at the University Press, Oxford*. Oxford University Press, London, Oxford, New York, 39. Auflage, 1991. ISBN 0-19-212983-X.
Ein weit verbreitetes Referenzwerk für Autoren und Verleger im britischen Raum. Gemeinsam mit dem *Oxford Dictionary for Writers and Editors* gibt es die maßgeblichen Layoutrichtlinien für die Oxford University Press vor (vgl. [145]).
- [66] Alan Hoenig. *T_EX Unbound: L^AT_EX and T_EX Strategies for Fonts, Graphics, & More*. Oxford University Press, London, Oxford, New York, 1998. ISBN 0-19-509686-X (Taschenbuch), 0-19-509685-1 (gebunden).
Der erste Teil dieses Buches bietet einen kurzen aber vollständigen Überblick über T_EX, L^AT_EX, METAFONT und MetaPost, mit einem besonderen Augenmerk auf ihr Zusammenspiel, den Produktionszyklus und die Arten der beteiligten Dateien. Der zweite Teil ist näheren Einzelheiten zu Schriften und ihrer Verwendung in T_EX gewidmet. Besonders interessant sind dabei 30 Beispielseiten, die zeigen, wie man verschiedene Kombinationen wohlbekannter Schriften mit der geringen Auswahl an derzeit verfügbaren Mathematikfonts verbinden kann. Im letzten Teil des Buches werden Graphikanwendungen besprochen – insbesondere T_EX-freundliche Verfahren wie etwa METAFONT und MetaPost, das pstricks-Paket, P_CT_EX und MFpic.
- [67] Berthold K. P. Horn. *The European Modern fonts*. *TUGboat*, 19(1):62–63, März 1998. ISSN 0896-3207.
Bei den European Modern (EM) Schriften handelt es sich um auf der Computer Modern basierende Type 1 Fonts. Sie verfügen über fertige Akzentbuchstaben und zusammengesetzte Zeichen, so dass in T_EX auch Wörter aus Sprachen getrennt werden können, die diese Zeichen verwenden.
<http://www.tug.org/TUGboat/Articles/tb19-1/tb58horn.pdf>
- [68] Jean-Michel Hufflen. *Typographie: les conventions, la tradition, les goûts, ..., et L^AT_EX*. *Cahiers GUTenberg*, 35–36:169–214, Mai 2000.
Dieser Artikel zeigt, dass das Erlernen typographischer Regeln – für die französische und die englische Sprache – gar nicht so schwer ist. Er erklärt auch, wie man die L^AT_EX-Pakete french (nur für Französisch) und babel (für die meisten anderen Sprachen gleich gut geeignet) verwendet. Abschließend erklärt der Autor, wie man eine neue mehrsprachige Dokumentenklasse und einen Bibliographiestil erstellt.
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/35-hufflen.pdf>
- [69] *ISO/IEC 10646-1:2000, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 1: Architecture and Basic Multilingual Plane*. Internationaler Standard ISO 10646-1 (2. Auflage), ISO Genf, 2000.
Dieser Standard beschreibt den Aufbau des Universal Multiple-Octet Coded Character Set (UCS). In praktischer Hinsicht ist dieser Standard für 32-Bit-Zeichenkodierungen mit dem Unicode-Standard (siehe [166]) identisch. Das Layout der Basic Multilingual Plane (BMP oder Ebene 0, der grundlegenden mehrsprachigen Ebene) wird detailliert beschrieben. Bei einer Änderung 2002 wurden mathematische Symbole und weitere Zeichen hinzugefügt.
- [70] *ISO/IEC 10646-2:2001, Information technology—Universal Multiple-Octet Coded Character Set (UCS)—Part 2: Supplementary Planes*. Internationaler Standard ISO 10646-2, ISO Genf, 2001.
Als Ergänzung zu [69], in dem die Ebene 0 (BMP) des UCS beschrieben wird, legt dieser Standard das Layout der zusätzlichen Ebenen dar (siehe auch [166]).
- [71] *ISO/IEC 14651:2001, Information technology—International string ordering and comparison—Method for comparing character strings and description of the common template tailorable ordering*. Internationaler Standard ISO/IEC 14651:2001, ISO Genf, 2001.
Der ISO-Standard 14651 definiert eine Methode zum Vergleichen zweier Zeichenfolgen (Strings), mit der sich bestimmen lässt, in welcher Reihenfolge diese in einer sortierten

Liste erscheinen müssten. Diese Methode lässt sich auch auf Strings anwenden, die alle Möglichkeiten des Unicode-Standards ausschöpfen. Der Standard definiert eine Common Template Table (Tabelle mit allgemeinen Vorlagen zur Sortierreihenfolge), die den Erfordernissen der jeweiligen Sprache und Kultur angepasst werden kann.

- [72] *ISO/IEC 8859-1:1998 to ISO/IEC 8859-16:2001, Information technology—8-bit single-byte coded graphic character sets, Parts 1 to 16*. Internationaler Standard ISO/IEC 8859, ISO Genf, 1998–2001.
Beschreibung verschiedener alphabetischer 8-Bit-Zeichensätze. Die Teile 1–4, 9, 10 und 13–16 beziehen sich auf zehn Zeichensätze, die man benötigt um verschiedene Sprachgruppen, die das lateinische Alphabet benutzen, zu kodieren. Teil 5 bezieht sich auf kyrillische, Teil 6 auf arabische, Teil 7 auf griechische, Teil 8 auf hebräische und Teil 11 auf thailändische Zeichen.
- [73] *ISO 8879:1986, Information Processing—Text and Office Systems—Standard Generalised Markup Language (SGML)*. Internationaler Standard ISO 8879, ISO Genf, 1986.
Nicht immer einfach zu lesender ISO-Standard, der die Sprache SGML in allen technischen Einzelheiten beschreibt. 1988 wurde eine Ergänzung und 1996 und 1999 wurden zwei Korrigenda veröffentlicht. Eine kommentierte Beschreibung befindet sich in [55].
- [74] Alan Jeffrey, Rowlan McDonnell und Lars Hellström. *fontinst: Font installation software for T_EX*, Dezember 2004.
Dieses Paket von Hilfsprogrammen bietet Unterstützung beim Erzeugen komplexer virtueller Zeichensätze für L^AT_EX in jeder beliebigen Kodierung, insbesondere aus Sammlungen von PostScript-Fonts.
Auf CTAN unter: [fonts/utilities/fontinst/doc/manual](http://ctan.org/fonts/utilities/fontinst/doc/manual)
- [75] Alan Jeffrey, Sebastian Rahtz, Ulrik Vieth und Lars Hellström. *The fontinst utility*, Juli 2003.
Technische Beschreibung des Hilfsprogramms fontinst.
Auf CTAN unter: [fonts/utilities/fontinst/source/fisource.dvi](http://ctan.org/fonts/utilities/fontinst/source/fisource.dvi)
- [76] Alan Jeffrey. *PostScript font support in L^AT_EX 2_ε*. *TUGboat*, 15(3):263–268, September 1994. ISSN 0896-3207.
Beschreibt die ursprüngliche psnfss-Distribution für den Einsatz von PostScript-Fonts in L^AT_EX.
<http://www.tug.org/TUGboat/Articles/tb15-3/tb44jeff.pdf>
- [77] Alan Jeffrey. *Tight setting with T_EX*. *TUGboat*, 16(1):78–80, März 1995. ISSN 0896-3207.
Beschreibt einige Versuche, Texte in Adobe Times, einer Schrift traditionell geringer Laufweite, mit T_EX zu setzen.
<http://www.tug.org/TUGboat/Articles/tb16-1/tb46jeff.pdf>
- [78] Roger Kehr. *xindy—A flexible indexing system*. *Cahiers GUTenberg*, 28-29:223–230, März 1998.
Hier wird ein neuer Index-Prozessor, xindy, beschrieben. Mit seiner Hilfe lassen sich Indexeinträge in einer mehrsprachigen Umgebung nach den Sortierregeln der jeweiligen Sprache sortieren. Er bietet neue Verfahren, über Seitenzahlen und römische Ziffern hinaus, weitere strukturierte Verweise auf Stellen im Text zu verarbeiten und ist für komplexe Markup-Systeme gerüstet.
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/28-29-kehr.pdf>
- [79] Brian W. Kernighan. *pic—A Graphics Language for Typesetting*. Computerwissenschaftlicher Fachbericht 116, AT&T Bell Laboratories, 1991.
Benutzerhandbuch für die Sprache pic, die zum Zeichnen einfacher Figuren auf einem Belichter dient. Die Grundobjekte dieser Sprache sind Kästen, Kreise, Ellipsen, Linien, Pfeile, Spline-Kurven und Text. Diese lassen sich an jeder beliebigen Stelle platzieren, entweder mithilfe absoluter Positionsangaben oder im Verhältnis zu vorhergehenden Objekten.
<http://cm.bell-labs.com/cm/cs/cstr/116.ps.gz>

- [80] Jörg Knappen. *Release 1.2 of the dc-fonts: Improvements to the European letters and first release of text companion symbols*. *TUGboat*, 16(4):381–387, Dezember 1995. ISSN 0896-3207.
Beschreibung der DC Fonts, der Vorgänger der EC Fonts, welche wiederum die Standardfonts für die T₁-Kodierung in L^AT_EX sind.
<http://www.tug.org/TUGboat/Articles/tb16-4/tb49knap.pdf>
- [81] Jörg Knappen. *The dc fonts 1.3: Move towards stability and completeness*. *TUGboat*, 17(2):99–101, Juni 1996. ISSN 0896-3207.
Fortsetzung von [80]. In diesem Artikel werden die in Version 1.3 erzielten Fortschritte hinsichtlich Stabilität und Vollständigkeit beschrieben.
<http://www.tug.org/TUGboat/Articles/tb17-2/tb51knap.pdf>
- [82] Donald E. Knuth und Pierre MacKay. *Mixing right-to-left texts with left-to-right texts*. In Knuth [101], S. 157–176. Ursprünglich in *TUGboat* 8(1):14–25, 1987 veröffentlicht.
T_EX wurde ursprünglich entwickelt, um Material in Dokumenten von links nach rechts und von oben nach unten zu setzen. In dieser Abhandlung wird erläutert, welche Probleme bei der Verarbeitung von Dokumenten mit gemischten Schreibrichtungen auftreten. Außerdem wird besprochen, mit welchen Änderungen man T_EX zu einem bidirektionalen Satzsystem ausbauen kann.
<http://www.tug.org/TUGboat/Articles/tb08-1/tb17knutmix.pdf>
- [83] Donald E. Knuth und Michael F. Plass. *Breaking Paragraphs Into Lines*. In Knuth [101], S. 67–155.
Dieser 1981 zum ersten Mal veröffentlichte Artikel spricht das Problem an, den Text eines Absatzes in Zeilen annähernd gleicher Länge einzuteilen. Der grundlegende Algorithmus betrachtet den Absatz als Ganzes und stellt die (heute in T_EX wohlbekannten) Konzepte von „Boxen“, „Glue“ (Leim) und „Penalty“-Werten zum Auffinden der besten Stellen für Zeilenumbrüche vor. In der Abhandlung wird das dynamische Programmierverfahren beschrieben, mit dem der Algorithmus umgesetzt wurde.
- [84] Donald E. Knuth und Hermann Zapf. *AMS Euler—A New Typeface for Mathematics*. In Knuth [101], S. 339–366.
In diesem 1989 zum ersten Mal veröffentlichten Artikel erläutern die beiden Autoren, wie eine Zusammenarbeit zwischen Wissenschaftlern und Künstlern dazu beitragen kann, ansprechende und qualitativ hochwertige mathematische Zeitschriften und Lehrbücher herzustellen.
- [85] Donald E. Knuth. *T_EX and METAFONT—New Directions in Typesetting*. Digital Press, 12 Crosby Drive, Bedford, MA 01730, USA, 1979. ISBN 0-932376-02-9.
Enthält einen Artikel über „Mathematical Typography“ (Mathematiksatz), in dem die Frühgeschichte des Computersatzes beschrieben ist und was den Autor dazu brachte, an T_EX zu arbeiten. Im Buch werden frühe (mittlerweile veraltete) Versionen von T_EX und METAFONT beschrieben.
- [86] Donald E. Knuth. *Literate Programming*. Bericht STAN-CS-83-981, Stanford University, Department of Computer Science, Stanford, CA, USA, 1983.
Eine Sammlung von Abhandlungen zu Programmier- und Dokumentationsstilen.
<http://www.literateprogramming.com/farticles.html>.
- [87] Donald E. Knuth. *The T_EXbook*, Band A von *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13447-0.
Die definitive Benutzeranleitung und vollständige Referenz zu T_EX.

- [88] Donald E. Knuth. *T_EX: The Program*, Band B von *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13437-3.
Der vollständige dokumentierte T_EX-Programmcode, mit verschiedenen Stichwortverzeichnissen versehen.
- [89] Donald E. Knuth. *The METAFONTbook*, Band C von *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13445-4 (gebunden), 0-201-13444-6 (Taschenbuch).
Benutzeranleitung und Referenz zu METAFONT, das Begleitprogramm zu T_EX zum Gestalten von Schriften.
- [90] Donald E. Knuth. *METAFONT: The Program*, Band D von *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13438-1.
Vollständiges Listing des Programmcodes von METAFONT.
- [91] Donald E. Knuth. *Computer Modern Typefaces*, Band E von *Computers and Typesetting*. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13446-2.
In diesem Buch sind über 500 griechische und römische Buchstabenformen zusammen mit ihren Satzzeichen, Ziffern und vielen mathematischen Symbolen abgebildet. Zu jeder Glyphe ist der METAFONT-Code angegeben und es wird erklärt, wie man durch Verändern der METAFONT-Parameter alle Zeichen der Schriftfamilie Computer Modern erhält.
- [92] Donald E. Knuth. *3:16 Bible texts illuminated*. A-R Editions, Inc., Madison, Wisconsin, 1990. ISBN 0-89579-252-4.
Analyse von Kapitel 3, Vers 16 jedes Buches der Bibel; enthält wunderschöne Kalligraphien verschiedener Künstler.
- [93] Donald E. Knuth. *The Art of Computer Programming*, vols 1–3. Addison-Wesley, Reading, MA, USA, 1998. ISBN 0-201-89683-4, 0-201-03822-6, 0-201-03803-X.
Ein wichtiges Werk über Algorithmen und Datenstrukturen für effektives Programmieren.
- [94] Donald E. Knuth. *Mathematical Typography*. In Knuth [101], S. 19–65.
Ausgehend von einer Vorlesung, die er 1978 gehalten hat, weist Knuth nach, dass Mathematikbücher und -zeitschriften nicht mehr so schön aussehen wie früher. Da die Ursache hauptsächlich darin liegt, dass qualitativ hochwertiges Setzen zu teuer geworden ist, schlägt er vor, das Problem mithilfe der Mathematik selbst zu lösen. Zuerst muss seiner Meinung nach eine Methode entwickelt werden, um mathematische Elemente in einem Dokument eindeutig auszuzeichnen, so dass sie leicht von Maschinen verarbeitet werden können. Der zweite Schritt besteht darin, die Formen der Buchstaben und Symbole mithilfe der Mathematik zu gestalten. Der Artikel erläutert beide Ansätze sehr ausführlich.
- [95] Donald E. Knuth. *Virtual Fonts: More Fun for Grand Wizards*. In Knuth [101], S. 247–262. Ursprünglich in *TUGboat* 11(1):13–23, 1990 veröffentlicht.
Erklärung, was virtuelle Fonts sind und wozu man sie benötigt, ergänzt um technische Einzelheiten.
Auf CTAN unter: [info/virtual-fonts.knuth](http://www.tug.org/TUGboat/Articles/tb11-1/tb27knut.pdf)
<http://www.tug.org/TUGboat/Articles/tb11-1/tb27knut.pdf>
- [96] Donald E. Knuth. *Typesetting Concrete Mathematics*. In Knuth [101], S. 367–378. Ursprünglich in *TUGboat* 10(1):31–36, 1989 veröffentlicht.
Knuth erläutert, wie er das Lehrbuch *Concrete Mathematics* verfasst hat. Er betont, dass er das Buch sowohl in mathematischer als auch in typographischer Hinsicht „ansprechend“ gestalten wollte, da hier das erste Mal die neue Schrift AMS Euler von Herman

- Zapf in größerem Umfang zum Einsatz kam. Die Fontparameter wurden so optimiert, dass der Text der besten Handschrift eines Mathematikers in nichts nachstehen sollte. Außerdem werden noch weitere Entscheidungen hinsichtlich der Gestaltung des Buches beschrieben. <http://www.tug.org/TUGboat/Articles/tb10-1/tb26knut.pdf>
- [97] Donald E. Knuth. *Fonts for digital halftones*. In Knuth [101], S. 415–448. Ursprünglich in *TUGboat* 8(2):135–160, 1987 veröffentlicht.
In diesem Artikel werden einige Versuche besprochen, in denen mithilfe von METAFONT Zeichensätze zum Erstellen von Rasterdrucken auf Laserdruckern erzeugt wurden. Die Verfahren erwiesen sich auch in einigen anderen Anwendungen als nützlich, und die Gestaltung warf verschiedene interessante Fragen auf.
<http://www.tug.org/TUGboat/Articles/tb08-2/tb18knut.pdf>
- [98] Donald E. Knuth. *Computers and Typesetting*. In Knuth [101], S. 555–562. Ursprünglich in *TUGboat* 7(2):95–98, 1986 veröffentlicht.
Ausführungen von Knuth im Computer Museum in Boston, Massachusetts, am 21. Mai 1986 bei der „Einweihungsfeier“ anlässlich der Vollendung von T_EX.
<http://www.tug.org/TUGboat/Articles/tb07-2/tb14knut.pdf>
- [99] Donald E. Knuth. *The New Versions of T_EX and METAFONT*. In Knuth [101], S. 563–570. Ursprünglich in *TUGboat* 10(3):325–328, 1989 veröffentlicht.
Knuth erklärt, wie er beim TUG Meeting in Stanford 1989 überzeugt wurde, noch einmal einige Änderungen an T_EX und METAFONT vorzunehmen, damit diese Programme auch 8-Bit-Zeichensätze unterstützten. Im Anschluss beschreibt er die verschiedenen zu diesem Zweck durchgeführten Änderungen sowie einige weitere Verbesserungen.
<http://www.tug.org/TUGboat/Articles/tb10-3/tb25knut.pdf>
- [100] Donald E. Knuth. *The future of T_EX and METAFONT*. In Knuth [101], S. 571–572. Ursprünglich in *TUGboat* 11(4):489, 1990 veröffentlicht.
In diesem Artikel erklärt Knuth, dass seine Arbeit an T_EX, METAFONT und Computer Modern nun am „Ende angelangt“ sei, und dass weitere Änderungen nur noch in der Korrektur sehr schwerwiegender Fehler bestehen würden.
<http://www.tug.org/TUGboat/Articles/tb11-4/tb30knut.pdf>
- [101] Donald E. Knuth. *Digital Typography*. CSLI Publications, Stanford, CA, USA, 1999. ISBN 1-57586-011-2 (Leinen), 1-57586-010-4 (Taschenbuch).
Sammlung von Texten von Knuth zum Thema T_EX und Typographie.
- [102] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script: eine Sammlung von Klassen und Paketen für L^AT_EX 2_ε*. DANTE, Heidelberg, Februar 2003. ISBN 3-936427-45-3.
KOMA-Script ist eine Sammlung von L^AT_EX-Klassen und -Paketen, die man anstelle der Standard-L^AT_EX-Klassen verwenden kann und die einen erweiterten Funktionsumfang bieten. Die Distribution umfasst deutsche und englische Handbücher.
Auf CTAN unter: [macros/latex/contrib/koma-script/scrguide.pdf](http://www.ctan.org/macros/latex/contrib/koma-script/scrguide.pdf)
- [103] Helmut Kopka und Patrick Daly. *Guide to L^AT_EX. Tools and Techniques for Computer Typesetting*. Addison-Wesley, Boston, MA, USA, vierte Auflage, 2004. ISBN 0-201-17385-6.
L^AT_EX-Einführung in einem anderen pädagogischen Stil als ihn Lamport in seinem *L^AT_EX-Manual* [106] benutzt.
- [104] Klaus Lagally. *ArabT_EX—Typesetting Arabic with vowels and ligatures*. In *Proceedings of the 7th European T_EX Conference*, Prague, S. 153–172. CsTUG, Prague, September 1992. ISBN 80-210-0480-0.
Makropaket zum Setzen teilweise und vollständig vokalisierter arabischer Texte, das zu Plain-T_EX und L^AT_EX kompatibel ist.

- [105] Leslie Lamport. *MakeIndex, An Index Processor For L^AT_EX*. Technischer Bericht, Electronic Document in *MakeIndex* distribution, 1987.
In diesem Dokument wird erläutert, welche Syntax man in dem L^AT_EX-Befehl `\index` verwenden kann, wenn man den Index mit *MakeIndex* generiert. Es enthält auch eine Liste der möglichen Fehlermeldungen.
[Auf CTAN unter: indexing/makeindex/doc/makeindex.dvi](http://www.ctan.org/indexing/makeindex/doc/makeindex.dvi)
- [106] Leslie Lamport. *L^AT_EX: A Document Preparation System: User's Guide and Reference Manual*. Addison-Wesley, Reading, MA, USA, zweite Auflage, 1994. ISBN 0-201-52983-1. Korrigierter Nachdruck von 1996.
Das ultimative L^AT_EX 2_ε-Grundlagenhandbuch für Anwender vom Entwickler von L^AT_EX 2.09. Es ist eine gute Ergänzung zum vorliegenden Buch.
- [107] Olga Lapko und Irina Makhovaya. *The style russianb for Babel: Problems and solutions*. *TUGboat*, 16(4):364–372, Dezember 1995. ISSN 0896-3207.
In dieser Abhandlung wird die Sprachoption `russianb` besprochen, die besondere Befehle enthält, um Legenden und alphabetische Zähler an die russische Sprache anzupassen, und russische mathematische Operatoren ermöglicht. Es wird auch auf einige Probleme hingewiesen, die beim Verwenden dieser Option (mit verschiedenen Kodierungen) auftreten können.
<http://www.tug.org/TUGboat/Articles/tb16-4/tb49olga.pdf>
- [108] L^AT_EX3 Project Team. *Default docstrip headers*. *TUGboat*, 19(2):137–138, Juni 1998.
In diesem Dokument wird das Format des Vorspanns beschrieben, den `docstrip` normalerweise am Anfang erzeugter Paketdateien einfügt. Dieser Vorspann eignet sich für Copyright-Vermerke und Distributionsbedingungen.
<http://www.tug.org/TUGboat/Articles/tb19-2/tb591tdocstrip.pdf>
- [109] L^AT_EX3 Project Team. *L^AT_EX 2_ε font selection*, September 2000.
Beschreibung der Fontauswahl in Standard-L^AT_EX für Paketentwickler, die bereits mit T_EX-Fonts und L^AT_EX vertraut sind.
<http://www.latex-project.org/guides/fntguide.pdf>
- [110] L^AT_EX3 Project Team. *Configuration options for L^AT_EX 2_ε*, August 2001.
Wie man eine L^AT_EX-Installation mithilfe der Standard-Konfigurationsdateien konfiguriert.
<http://www.latex-project.org/guides/cfgguide.pdf>
- [111] L^AT_EX3 Project Team. *The L^AT_EX Project Public License (Version 1.3)*, Dezember 2003.
Open-Source-Lizenz, die für die zentrale L^AT_EX 2_ε-Distribution und viele Zusatzpakete verwendet wird.
<http://www.latex-project.org/lppl/>
- [112] L^AT_EX3 Project Team. *L^AT_EX Bug database*.
Vom L^AT_EX3-Team als Teil der L^AT_EX 2_ε-Pflege betriebene Fehlerdatenbank.
<http://www.latex-project.org/cgi-bin/ltxbugs2html>
- [113] L^AT_EX3 Project Team. *L^AT_EX News*.
Mit jedem neuen L^AT_EX 2_ε-Release wird eine Ausgabe der *L^AT_EX News* veröffentlicht, in der die Änderungen gegenüber dem letzten Release hervorgehoben werden.
<http://www.latex-project.org/ltnews/>
- [114] John Lavagnino und Dominik Wujastyk. *An overview of EDMAC: A plain T_EX format for critical editions*. *TUGboat*, 11(4):623–643, November 1990. ISSN 0896-3207.
EDMAC dient zum Setzen „kritischer Ausgaben“ von Texten wie etwa der *Oxford Classical Texts*, *Shakespeare* und anderer Reihen. Das Format unterstützt Zeilennummerierung, Fuß- und Endnotenapparate.
<http://www.tug.org/TUGboat/Articles/tb11-4/tb30lava.pdf>

- [115] Werner Lemberg. *The CJK package: Multilingual support beyond Babel*. *TUGboat*, 18(3):214–224, September 1997. ISSN 0896-3207.
Beschreibung des $\LaTeX$ -Paketes CJK (Chinesisch/Japanisch/Koreanisch) und seiner Schnittstelle zu mule (multilingual emacs).
<http://www.tug.org/TUGboat/Articles/tb18-3/cjkintr0600.pdf>
- [116] Silvio Levy. *Using Greek fonts with $\TeX$* . *TUGboat*, 9(1):20–24, April 1988. ISSN 0896-3207.
Der Autor versucht darzulegen, dass man mit der Schriftfamilie gr in $\TeX$ griechische Texte ebenso einfach und ansprechend setzen kann, wie sonst englische Texte. Der Artikel ist als Lernhilfe gedacht. Für alle, die sich bereits näher mit der Schrift vertraut gemacht haben, enthält er jedoch auch technische Einzelheiten.
<http://www.tug.org/TUGboat/Articles/tb09-1/tb20levy.pdf>
- [117] Franklin Mark Liang. *Word Hy-phen-a-tion by Com-pu-ter*. Dissertation, Stanford University, Stanford, CA 94305, Juni 1983. Auch als Stanford University, Department of Computer Science Report Nr. STAN-CS-83-977 erhältlich.
Eine ausführliche Beschreibung des Algorithmus, den $\TeX$ zur Silbentrennung verwendet.
- [118] Ruari McLean. *The Thames and Hudson Manual of Typography*. Thames and Hudson, London, UK, 1980. ISBN 0-500-68022-1.
Eine allgemeine Einführung in die traditionelle kommerzielle Typographie.
- [119] Frank Mittelbach, David Carlisle, Chris Rowley et al. *Experimental $\LaTeX$ code for class design*.
Auf der $\TeX$ Users Group-Konferenz in Vancouver hielt das $\LaTeX$ -Projektteam einen Vortrag über Modelle für Benutzer- und Entwicklerschnittstellen in $\LaTeX$ [120]. Die meisten dieser Ideen wurden als Prototypen implementiert (z.B. für das Entwerfen von Vorlagen, die Handhabung des Vorspanns, die Ausgaberoutine, die Druckfahne und das Formatieren von Absätzen). Der Quellcode ist dokumentiert und enthält weitere Erklärungen und Beispiele (vgl. [132]).
Vortrag: <http://www.latex-project.org/papers/tug99.pdf>
Code: <http://www.latex-project.org/code/experimental>
CVS: <http://www.latex-project.org/cgi-bin/cvsweb>
- [120] Frank Mittelbach, David Carlisle und Chris Rowley. *New interfaces for $\LaTeX$ class design, Parts I and II*. *TUGboat*, 20(3):214–216, September 1999. ISSN 0896-3207.
Vorschläge für eine saubere Schnittstelle zum Einrichten und Programmieren von $\LaTeX$ -Klassen.
<http://www.tug.org/TUGboat/Articles/tb20-3/tb64carl.pdf>
CVS: <http://www.latex-project.org/cgi-bin/cvsweb>
- [121] Frank Mittelbach, Denys Duchier, Johannes Braams, Marcin Woliński und Mark Wooding. *The DocStrip program*, 2003. Teil der grundlegenden $\LaTeX$ -Distribution.
Beschreibung der Umsetzung des Programms docstrip.
Auf CTAN unter: [macros/latex/base/docstrip.dtx](http://www.ctan.org/macros/latex/base/docstrip.dtx)
- [122] Frank Mittelbach und Chris Rowley. *$\LaTeX$ 2.09 $\rightarrow$ $\LaTeX$ 3*. *TUGboat*, 13(1):96–101, April 1992. ISSN 0896-3207.
Kurzer Abriss des $\LaTeX$ 3-Projektes, in dem die Geschichte noch einmal aufgerollt und die Struktur des Systems beschrieben wird. In *TUGboat*, 13(3):390–391, April 1992, wurde eine aktuellere Version veröffentlicht. Ein Aufruf an Freiwillige, bei der Entwicklung von $\LaTeX$ 3 zu helfen, erschien zusammen mit einer Liste der verschiedenen Aufgaben in *TUGboat*, 13(4):510–515, Dezember 1992. In dem Artikel wird auch beschrieben, wie man die aktuelle Aufgabenliste sowie verschiedene Arbeitsgruppenunterlagen zu $\LaTeX$ 3 per E-Mail oder FTP erhält. Zudem wird erklärt, wie man dem $\LaTeX$ 3-Diskussionsforum beitreten kann.
<http://www.tug.org/TUGboat/Articles/tb13-1/tb34mitt13.pdf>

- [123] Frank Mittelbach und Chris Rowley. *The Pursuit of Quality: How can Automated Typesetting achieve the Highest Standards of Craft Typography?* In C. Vanoirbeek und G. Coray, Hg., EP92—Proceedings of Electronic Publishing, '92, International Conference on Electronic Publishing, Document Manipulation, and Typography, Swiss Federal Institute of Technology, Lausanne, Switzerland, April 7–10, 1992, S. 261–273. Cambridge University Press, New York, 1992. ISBN 0-521-43277-4.
- In dieser Abhandlung werden hochwertige typographische Arbeiten dem, was heute mittels automatisierter Satzverfahren technisch möglich ist, gegenübergestellt. Der erste Teil befasst sich mit verschiedenen typographischen Konventionen, die sich nicht durch zurzeit verfügbare Formatierungsmodelle umsetzen lassen. Der zweite Teil erläutert, warum die aktuellen Lösungsansätze des Computersatzes nicht zu qualitativ hochwertigen Ergebnissen führen werden. Er zeigt auf, in welcher Richtung weiter geforscht werden sollte, um die Qualität am Computer generierter Layouts zu verbessern.
- [124] Frank Mittelbach und Rainer Schöpf. *A new font selection scheme for T_EX macro packages—the basic macros*. *TUGboat*, 10(2):222–238, Juli 1989. ISSN 0896-3207.
- Beschreibung der Makros, auf deren Basis die erste Version des New Font Selection Scheme (Fontauswahlssystem) von L^AT_EX umgesetzt wurde.
<http://www.tug.org/TUGboat/Articles/tb10-2/tb24mitt.pdf>
- [125] Frank Mittelbach und Rainer Schöpf. *With L^AT_EX into the nineties*. *TUGboat*, 10(4):681–690, Dezember 1989. ISSN 0896-3207.
- In diesem Artikel wird vorgeschlagen, L^AT_EX neu aufzusetzen. Dabei sollen die wesentlichen Funktionen der aktuellen Schnittstelle erhalten bleiben und gleichzeitig die wachsenden Anforderungen der verschiedenen Benutzergruppen berücksichtigt werden. Zudem werden einige Vorschläge für zukünftige Entwicklungen formuliert. Der Artikel hat L^AT_EX 2.09 zu dem Schritt zu L^AT_EX 2_ε verholfen.
<http://www.tug.org/TUGboat/Articles/tb10-4/tb26mitt.pdf>
- [126] Frank Mittelbach und Rainer Schöpf. *Reprint: The new font family selection — User interface to standard L^AT_EX*. *TUGboat*, 11(2):297–305, Juni 1990. ISSN 0896-3207.
- Umfassende Beschreibung der Benutzerschnittstelle des L^AT_EX New Font Selection Scheme (Fontauswahlssystem) in seiner ersten Version.
<http://www.tug.org/TUGboat/Articles/tb11-2/tb28mitt.pdf>
- [127] Frank Mittelbach und Rainer Schöpf. *Towards L^AT_EX 3.0*. *TUGboat*, 12(1):74–79, März 1991. ISSN 0896-3207.
- Beschreibung der Ziele des L^AT_EX 3-Projektes. Die Autoren untersuchen, welche Verbesserungen der L^AT_EX-Schnittstellen für Benutzer und Layout-Dateien erforderlich sind, um mit modernen Entwicklungen wie etwa SGML Schritt zu halten. Sie prüfen auch einige interne Konzepte, die überarbeitet werden sollten.
<http://www.tug.org/TUGboat/Articles/tb12-1/tb31mitt.pdf>
- [128] Frank Mittelbach. *E-T_EX: Guidelines for Future T_EX Extensions*. *TUGboat*, 11(3):337–345, September 1990. ISSN 0896-3207.
- Dieser Artikel vergleicht T_EX-Ausdrucke mit von Hand gesetzten Dokumenten. Dabei zeigt sich, dass viele wichtige Elemente typographisch hochwertiger Arbeiten nicht unterstützt werden und dass weitere Untersuchungen dahin führen sollten, einen „Nachfolger“ für das Satzsystem T_EX zu entwickeln.
<http://www.tug.org/TUGboat/Articles/tb11-3/tb29mitt.pdf>
- [129] Frank Mittelbach. *Comments on “Filenames for Fonts” (TUGboat 11#4)*. *TUGboat*, 13(1):51–53, April 1992. ISSN 0896-3207.
- Hier werden einige Probleme mit der Nomenklatur von K. Berry besprochen, die insbesondere auftreten, wenn man bestimmte Fontcharakteristika unabhängig voneinander definieren will. Zudem wird die Verwendung der Namensgebung im NFSS betrachtet.
<http://www.tug.org/TUGboat/Articles/tb13-1/tb34mittfont.pdf>

- [130] Frank Mittelbach. *A regression test suite for L^AT_EX 2_ε. TUGboat*, 18(4):309–311, Dezember 1997. ISSN 0896-3207.
Beschreibung der Konzeption und Implementation des Testpaketes, mit dessen Hilfe der L^AT_EX-Kernel nach Veränderungen auf unerwartete Nebeneffekte geprüft werden kann. Eines der wertvollsten Werkzeuge, um die Stabilität von L^AT_EX 2_ε zu gewährleisten.
<http://www.tug.org/TUGboat/Articles/tb18-4/tb57mitt.pdf>
- [131] Frank Mittelbach. *Language Information in Structured Documents: Markup and rendering—Concepts and problems*. In International Symposium on Multilingual Information Processing, S. 93–104. Tsukuba, Japan, März 1997. Eingeladener Beitrag. Nachdruck in *TUGboat* 18(3):199–205, 1997.
In dieser Abhandlung werden die Struktur und die Verarbeitung mehrsprachiger Dokumente besprochen, sowohl auf allgemeiner Ebene als auch hinsichtlich einer vorge schlagenen Erweiterung von Standard-L^AT_EX.
<http://www.tug.org/TUGboat/Articles/tb18-3/tb56lang.pdf>
- [132] Frank Mittelbach. *Formatting documents with floats: A new algorithm for L^AT_EX 2_ε. TUGboat*, 21(3):278–290, September 2000. ISSN 0896-3207.
Beschreibungen von Funktionen und Konzepten für eine neue L^AT_EX-Ausgaberoutine, die in mehrspaltigen Seitenlayouts spaltenübergreifende Gleitobjekte setzen kann.
<http://www.tug.org/TUGboat/Articles/tb21-3/tb68mittel.pdf>
CVS: <http://www.latex-project.org/cgi-bin/cvsweb>
- [133] Frank Mittelbach. *The trace package. TUGboat*, 22(1/2):93–99, März 2001. ISSN 0896-3207.
Beschreibung des trace-Paketes, mit dessen Hilfe sich Debugging-Informationen bei der Ausführung von L^AT_EX-Code steuern lassen.
<http://www.tug.org/TUGboat/Articles/tb22-1-2/tb70mitt.pdf>
- [134] Gerd Neugebauer. *BIBTOOL: A Tool to Manipulate B^BT_EX Files*, 2002.
Beschreibt das bibtool-Programm mit den folgenden Funktionen: Prettyprinting, Sortieren und Zusammenführen von B^BT_EX-Datenbanken, Erzeugen einheitlicher Verweisschlüssel und Auswählen von in einer Veröffentlichung verwendeten Quellen.
Auf CTAN unter: [biblio/bibtex/utls/bibtool/bibtool.dvi](http://ctan.org/tex-archive/biblio/bibtex/utls/bibtool/bibtool.dvi)
- [135] O. Nicole, J. André und B. Gaulle. *Notes en bas de pages : commentaires. Cahiers GUTenberg*, 15:46–32, April 1993.
Kommentare, Erläuterungen und Ergänzungen zu [10].
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/15-nicole.pdf>
- [136] Scott Pakin. *The Comprehensive L^AT_EX Symbol List*, 2003.
In diesem Dokument sind über 2800 Symbole mit den sie erzeugenden L^AT_EX-Befehlen aufgeführt. Einige dieser Symbole sind garantiert auf jedem L^AT_EX 2_ε-System verfügbar; andere erfordern Zeichensätze, die nicht in jeder Distribution enthalten sind und daher erst installiert werden müssen. Alle in diesem Dokument beschriebenen Fonts und Pakete sind in den CTAN-Archiven frei verfügbar.
Auf CTAN unter: [info/symbols/comprehensive/](http://ctan.org/tex-archive/info/symbols/comprehensive/)
- [137] Oren Patashnik. *B^BT_EXing*, Februar 1988.
In diesem Dokument und in Anhang B von *The Manual* [106] wird die B^BT_EX-Benutzerschnittstelle beschrieben, ergänzt um nützliche Hinweise dazu, wie man ihr Verhalten steuert.
Auf CTAN unter: [biblio/bibtex/contrib/doc/btxdoc.pdf](http://ctan.org/tex-archive/biblio/bibtex/contrib/doc/btxdoc.pdf)
- [138] Oren Patashnik. *Designing B^BT_EX Styles*, Februar 1988.
Ausführliche Beschreibung der in B^BT_EX verwendeten Postfix-Stapelverarbeitungssprache für Entwickler von B^BT_EX-Stilen. Nach einer allgemeinen Beschreibung der Sprache werden alle Befehle und eingebauten Funktionen besprochen. Abschließend wird die B^BT_EX-Namensformatierung in allen Einzelheiten erläutert.
Auf CTAN unter: [biblio/bibtex/contrib/doc/btxhak.pdf](http://ctan.org/tex-archive/biblio/bibtex/contrib/doc/btxhak.pdf)

- [139] John Plaice, Yannis Haralambous und Chris Rowley. *A multidimensional approach to typesetting*. *TUGboat*, 24(1):105–114, 2004. ISSN 0896-3207.
Überblick über einen mikrotypographischen Ansatz, der den Lösungen von $\TeX$ und $\Omega 2.0$ deutlich überlegen ist.
<http://www.tug.org/TUGboat/Articles/tb24-1/plaice.pdf>
- [140] John Plaice und Yannis Haralambous. *The latest developments in Ω* . *TUGboat*, 17(2):181–183, Juni 1996. ISSN 0896-3207.
In diesem Artikel werden ΩTimes und $\Omega\text{Helvetica}$ beschrieben, virtuelle Times und Helvetica-Fonts (so genannte „Glyphenbehälter“), die auf echten PostScript-Fonts basieren. Sie enthalten alle Zeichen, die man benötigt, wenn man hochwertige $\TeX$ -Dokumente in allen Sprachen und Systemen setzen will, die sich der folgenden Alphabete und ihrer Derivate bedienen: Latein, Griechisch, Kyrillisch, Arabisch, Hebräisch und Tifinagh. Weitere Alphabete wie z.B. Koptisch, Armenisch und Georgisch werden ebenso folgen wie mathematische Symbole, Dingbats und andere Zeichensammlungen. Letztendlich werden die Ω -Fonts Glyphen für den gesamten Unicode-Zeichensatz enthalten, ergänzt durch einige besondere Glyphen, die man für hochwertige Typographie benötigt.
<http://www.tug.org/TUGboat/Articles/tb17-2/tb51omeg.pdf>
- [141] Sunil Podar. *Enhancements to the Picture Environment of $\LaTeX$* . Fachbericht 86-17, Department of Computer Science, S.U.N.Y., 1986. Version 1.2: July 14, 1986.
In diesem Dokument werden einige neue Befehle für die *picture*-Umgebung von $\LaTeX$ beschrieben. Es geht insbesondere um High-Level-Befehle, welche die graphischen Fähigkeiten des Paketes durch eine anwenderfreundlichere und leistungsfähigere Benutzerschnittstelle verbessern. Mit ihrer Hilfe kann man mit weniger Aufwand ausgefeiltere Bilder erstellen als im normalen $\LaTeX$.
- [142] Rama Porrat. *Developments in Hebrew $\TeX$* . In Proceedings of the 7th European $\TeX$ Conference, Prague, S. 135–147. CsTUG, Prag, September 1992. ISBN 80-210-0480-0.
Besprechung verfügbarer Software und Makropakete, die bidirektionales Setzen unterstützen, sowie ihrer zugehörigen hebräischen Fonts.
- [143] Bernd Raichle, Rolf Niepraschk und Thomas Hafner. *DE- $\TeX$ -FAQ—Fragen und Antworten über $\TeX$, $\LaTeX$ und DANTE e.V.*, 2003.
Häufig gestellte Fragen (FAQ) mit Antworten zu $\TeX$ und zu der deutschsprachigen $\TeX$ -Benutzervereinigung DANTE e.V.
<http://www.dante.de/faq/de-tex-faq>
- [144] Brian Reid. *Scribe Document Production System User Manual*. Unilogic Ltd, 1984.
Handbuch für das System, das sich in einigen Punkten inspirierend auf die Entwicklung von $\LaTeX$ ausgewirkt hat.
- [145] Robert M. Ritter, Hg. *The Oxford Style Manual*. Oxford University Press, London, Oxford, New York, 2003. ISBN 0-198-60564-1.
Referenzwerk, das auch eine Aktualisierung von *Hart’s Rules* [65] sowie das *Oxford Dictionary for Writers and Editors* umfasst.
- [146] Tomas G. Rokicki. *A proposed standard for specials*. *TUGboat*, 16(4):395–401, Dezember 1995. ISSN 0896-3207.
Entwurf eines Standards für `\special`-Befehle in $\TeX$.
<http://www.tug.org/TUGboat/Articles/tb16-4/tb49roki.pdf>
- [147] Tomas G. Rokicki. *Dvips: A DVI-to-PostScript Translator, Version 5.66a*, Februar 1997.
Benutzeranleitung für *dvips* und seine Begleitprogramme und -Pakete wie z.B. *afm2tfm*.
Auf CTAN unter: [dviware/dvips/dvips_man.pdf](http://www.ctan.org/dviware/dvips/dvips_man.pdf)

- [148] Chris A. Rowley und Frank Mittelbach. *Application-Independent Representation of Multilingual Text*. In Unicode Consortium, Hg., Europe, Software + the Internet: Going Global with Unicode: Tenth International Unicode Conference, March 10–12, 1997, Mainz, Germany. The Unicode Consortium, San Jose, CA, 1997.
Befasst sich mit dem Thema, wie sich Text in Computerdateien darstellen lässt, sowie mit den Bedürfnissen vieler unterschiedlicher Textverarbeitungsprogramme.
<http://www.latex-project.org/papers/unicode5.pdf>
- [149] Chris Rowley. *Models and languages for formatted documents*. *TUGboat*, 20(3):189–195, September 1999. ISSN 0896-3207.
In diesem Dokument werden viele Ideen zum Formatieren von Dokumenten untersucht, und wie sich diese formulieren und umsetzen lassen.
<http://www.tug.org/TUGboat/Articles/tb20-3/tb64rowl.pdf>
- [150] Chris Rowley. *The L^AT_EX legacy: 2.09 and all that*. In ACM, Hg., Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing 2001, Newport, Rhode Island, United States, S. 17–25. ACM Press, New York, NY, USA, 2001. ISBN 1-58113-383-9.
Teil einer Festschrift zum 60sten Geburtstag von Leslie Lamport; ein sehr aufschlussreicher Bericht über die technische Geschichte und die Philosophie von T_EX und L^AT_EX.
- [151] Richard Rubinstein. *Digital Typography—An Introduction to Type and Composition for Computer System Design*. Addison-Wesley, Reading, MA, USA, November 1988. ISBN 0-201-17633-5. Korrigierter Nachdruck.
Dieses Buch beschreibt einen technischen Typographieansatz. Es zeigt, wie man mithilfe von Computern graphische Elemente, die der Darstellung von Dokumenten auf Computern dienen, entwerfen, erzeugen und positionieren kann.
- [152] Joachim Schrod. *International L^AT_EX is ready to use*. *TUGboat*, 11(1):87–90, April 1990. ISSN 0896-3207.
Kündigt einige der frühen Standards für die Globalisierung von L^AT_EX an.
<http://www.tug.org/TUGboat/Articles/tb11-1/tb27schrod.pdf>
- [153] Joachim Schrod. *An International Version of MakeIndex*. *Cahiers GUTenberg*, 10–11:81–90, September 1991.
Der Indexprozessor *MakeIndex* ist eigentlich nur für englische Texte geeignet. Die Verarbeitung von Texten in anderen Sprachen stellt ein Problem dar, insbesondere wenn sie andere Alphabete als das lateinische benutzen, wie z.B. Russisch, Arabisch oder Chinesisch. In solchen Fällen erweist sich das Taggen von Indexeinträgen häufig als ein mühsamer und fehleranfälliger Prozess. Kommen im Indexschlüssel selbst Markup-Befehle vor, so muss zudem noch ein expliziter Sortierschlüssel angegeben werden. In diesem Artikel wird eine neue *MakeIndex*-Version vorgestellt, die weniger Speicher belegt, so dass sie auch zum Erstellen sehr umfangreicher Indexe geeignet ist. Mit ihrer Hilfe lassen sich Sortierschlüssel automatisch anhand benutzerdefinierter Zuordnungen aus den Indexschlüsseln erzeugen und sie unterstützt Dokumente, die nicht auf dem lateinischen Alphabet basieren.
<http://www.gutenberg.eu.org/pub/GUTenberg/publicationsPDF/10-schrod.pdf>
- [154] Joachim Schrod. *The components of T_EX*. *MAPS*, 8:81–86, 1992.
T_EX benötigt eine Vielzahl zusätzlicher Komponenten (Dateien und Programme), deren Bedeutung und Austausch untereinander vielen verborgen bleibt; hier wird der Aufbau einer vollständigen T_EX-Installation beschrieben.
http://www.ntg.nl/maps/pdf/8_18.pdf
- [155] Paul Stiff. *The end of the line: A survey of unjustified typography*. *Information Design Journal*, 8(2):125–152, 1996.
Guter Überblick über die typographischen Probleme, die zu lösen sind, wenn man hochwertige Druckvorlagen im Flattersatz (d.h. ohne Randausgleich) setzen will.

- [156] Anders Svensson. *Typesetting diagrams with kuvio.tex*, 1996.
Handbuch zum kuvio-System, das dem Setzen von Diagrammen dient; es verwendet PostScript-Code in `\special`-Befehlen.
Auf CTAN unter: [macros/generic/diagrams/kuvio](http://ctan.org/macros/generic/diagrams/kuvio)
- [157] Ellen Swanson. *Mathematics into Type*. American Mathematical Society, Providence, Rhode Island, zweite Auflage, 1999. ISBN 0-8218-1961-5. Überarbeitet von Arlene O'Sean und Antoinette Schleyer.
Ursprünglich als Handbuch zum Standardisieren der Korrekturprozesse (Copy-Editing) geschrieben, richtet sich die zweite Auflage auch an Verleger und Autoren als Anleitung, mathematisches Material für den Druck aufzubereiten.
- [158] The TUGboat Team. *T_EX Live CD 5 and the T_EX Catalogue*. TUGboat, 21(1):16–90, März 2000. ISSN 0896-3207.
T_EX Live CD ist ein direkt lauffähiges T_EX-System für die beliebtesten Betriebssysteme. Es arbeitet mit allen wichtigen T_EX-Zusatzprogrammen zusammen, enthält eine vollständige Sammlung von Fonts, Makros und anderen Elementen und unterstützt zahlreiche Sprachen. In diesem Artikel wird die Distribution T_EX Live CD 5 beschrieben, mit Querverweisen zu Graham Williams T_EX-Katalog.
<http://www.tug.org/TUGboat/Articles/tb21-1/tb66cd.pdf>
Aktuelle Version: <http://www.tug.org/texlive>
- [159] Hàn Thế Thành und Sebastian Rahtz. *The pdfT_EX user manual*. TUGboat, 18(4):249–254, Dezember 1997. ISSN 0896-3207.
Benutzerhandbuch für das pdfT_EX-System, das T_EX so erweitert, dass es PDF-Dateien direkt erzeugen kann.
<http://www.tug.org/TUGboat/Articles/tb18-4/tb57than.pdf>
- [160] Hàn Thế Thành. *Improving T_EX's typeset layout*. TUGboat, 19(3):284–288, September 1998. ISSN 0896-3207.
Dieser Ansatz, das Layout mit T_EX gesetzter Dokumente zu verbessern, konzentriert sich auf den Ausgleich der Wortzwischenräume nach erfolgtem Zeilenumbruch der Absätze. Anstatt nur die Wortzwischenräume für den Randausgleich anzupassen, wird auch die Laufweite der Schrift in der jeweiligen Zeile minimal erweitert, um ein übermäßiges Dehnen der Zwischenräume zu vermeiden. Diese Laufweitenveränderung wird mithilfe von horizontaler Skalierung in PDF realisiert. Bei sehr zurückhaltend vergrößerten Laufweiten und geeigneten Einstellungen für die Zeilenumbruch- und Abstandsparameter in T_EX kann diese Methode das Erscheinungsbild mit T_EX gesetzter Dokumente verbessern. <http://www.tug.org/TUGboat/Articles/tb19-3/tb60than.pdf>
- [161] Hàn Thế Thành. *Micro-typographic extensions to the T_EX typesetting system*. TUGboat, 21(4):317–434, Dezember 2000. ISSN 0896-3207.
Doktorarbeit an der Fakultät für Informatik, Masaryk University, Brno, Tschechien, Oktober 2000. <http://www.tug.org/TUGboat/Articles/tb21-4/tb69than.pdf>
- [162] Hàn Thế Thành. *Margin kerning and font expansion with pdfT_EX*. TUGboat, 22(3):146–148, September 2001. ISSN 0896-3207.
Unter „Rand-Unterschneidung“ (Margin Kerning) ist zu verstehen, dass die ersten und letzten Glyphen einer Textzeile so positioniert werden, dass die Ränder „gerade aussehen“. Diese Funktion wurde auch auf das vorliegende Buch angewandt. Bei der „Fonterweiterung“ (Font Expansion) werden minimal schmalere oder breitere Varianten eines Fonts verwendet, um die Wortzwischenräume gleichmäßiger zu gestalten. Diese Verfahren werden hier anhand von Beispielen erläutert. In [161] werden die zugrunde liegenden Konzepte dazu ausführlich erklärt.
<http://www.tug.org/TUGboat/Articles/tb22-3/tb72thanh.pdf>
- [163] Harold Thimbleby. *‘See also’ indexing with makeindex*. TUGboat, 12(2):290–290, Juni 1991. ISSN 0896-3207.
In diesem Artikel wird beschrieben, wie man mit *MakeIndex* „see also“-Einträge (siehe auch) erzeugt, die nach den Seitenzahlen für den jeweiligen Eintrag erscheinen (siehe auch [164]). <http://www.tug.org/TUGboat/Articles/tb12-2/tb32thim.pdf>

- [164] Harold Thimbleby. *Erratum: 'See also' indexing with makeindex, TUGboat 12, no. 2, p. 290.* *TUGboat*, 13(1):95–95, April 1992. ISSN 0896-3207.
Erratum zu [163]. <http://www.tug.org/TUGboat/Articles/tb13-1/tb34thim.pdf>
- [165] TUG Working Group on a $\TeX$ Directory Structure. *A directory structure for $\TeX$ files (Version 0.999).* *TUGboat*, 16(4):401–413, Dezember 1995.
In diesem Artikel wird die in $\TeX$ gemeinhin verwendete Standard-Verzeichnisstruktur ($\TeX$ Directory Structure, TDS) für installationsunabhängige $\TeX$ -Dateien beschrieben.
<http://www.tug.org/TUGboat/Articles/tb16-4/tb49tds.pdf>
Aktuelle Version: <http://www.tug.org/tds>
- [166] The Unicode Consortium. The Unicode Standard, Version 4.0. Addison-Wesley, Boston, MA, USA, August 2003. ISBN 0-321-18578-1.
Referenzhandbuch zum Unicode-Standard, einem universellen Zeichenkodierungsschema, das ein einheitliches Verfahren zum Kodieren mehrsprachiger Texte definiert. Unicode ist die vorgegebene Kodierung für HTML und XML. Das Buch beschreibt ihre Funktionsprinzipien und enthält Abbildungen der Glyphen aller Zeichen, die zurzeit in Unicode definiert sind.
[Zur eingeschränkten Verwendung verfügbar unter: http://www.unicode.org/versions/Unicode4.0.0](http://www.unicode.org/versions/Unicode4.0.0)
- [167] Gabriel Valiente Feruglio. *Typesetting commutative diagrams.* *TUGboat*, 15(4):466–484, Dezember 1994. ISSN 0896-3207.
Überblick über die verfügbaren Lösungen zum Setzen kommutativer Diagramme.
<http://www.tug.org/TUGboat/Articles/tb15-4/tb45vali.pdf>
- [168] Gabriel Valiente Feruglio. *Modern Catalan typographical conventions.* *TUGboat*, 16(3):329–338, September 1995. ISSN 0896-3207.
Viele Sprachen, wie etwa Deutsch, Englisch und Französisch, verfügen über eine eigene typographische Tradition. Für die katalanische Sprache wurden jedoch trotz der Existenz einer solchen bewährten Tradition für wissenschaftliche Arbeiten bisher keine Standards formuliert, die auch die typographischen Konventionen in diesem Bereich abdecken. In dieser Abhandlung werden typographische Regeln vorgeschlagen, die den Geist der alten katalanischen wissenschaftlichen Werke widerspiegeln und dabei gleichzeitig modernen typographischen Konventionen folgen. Einige dieser Regeln wurden in den $\TeX$ - und $\LaTeX$ -Erweiterungen für die katalanische Sprache berücksichtigt. Mit diesem Vorschlag soll auch ein Beitrag zur Entwicklung von Standards für wissenschaftliche Arbeiten in katalanischer Sprache geleistet werden.
<http://www.tug.org/TUGboat/Articles/tb16-3/tb48vali.pdf>
- [169] Michael Vulis. *VT $\TeX$ enhancements to the $\TeX$ language.* *TUGboat*, 11(3):429–434, September 1990. ISSN 0896-3207.
Beschreibung des kommerziellen VT $\TeX$ -Systems, das ein skalierbares Fontformat unterstützt.
<http://www.tug.org/TUGboat/Articles/tb11-3/tb29vulis.pdf>
Neuere Informationen auf <http://www.micropress-inc.com/enfeat.htm>
- [170] Hugh Williamson. *Methods of Book Design.* Yale University Press, New Haven, London, dritte Auflage, 1983. ISBN 0-300-03035-5.
Klassiker, der zu einem grundlegenden Werkzeug für professionelle Buchgestalter geworden ist. Er behandelt Themen wie: Manuskriptaufbereitung, Auswahl und Anordnung der Schrift, Rolle des Gestalters bei der Illustration eines Buches und der Gestaltung des Schutzumschlags und wirtschaftliche Gesichtspunkte der Buchproduktion. Außerdem werden Materialien und Verfahren bei der Buchproduktion und ihre Auswirkungen auf die Buchgestaltung erläutert.
- [171] Graham Williams. *Graham Williams' $\TeX$ Catalogue.* *TUGboat*, 21(1):17–90, März 2000. ISSN 0896-3207.
In diesem Katalog werden über 1500 $\TeX$ -, $\LaTeX$ - und verwandte Pakete und Werkzeuge aufgeführt, mit direkten Verknüpfungen zu den jeweiligen Elementen auf CTAN.
<http://www.tug.org/TUGboat/Articles/tb21-1/tb66catal.pdf>
Neueste Version auf CTAN unter: <http://help/Catalogue/catalogue.html>

- [172] Peter Wilson. *ledmac*—A presumptuous attempt to port EDMAC and TABMAC to $\LaTeX$, 2003.
Bei EDMAC und TABMAC handelt es sich um eine Sammlung von Plain- $\TeX$ -Makros, mit deren Hilfe sich kritische Editionen auf traditionelle Weise setzen lassen. Das *ledmac*-Paket stellt die Funktionen dieser Makros für $\LaTeX$ zur Verfügung, insbesondere die Zeilennummerierung sowie Fuß- und Endnotenapparate. Das Paket bietet außerdem die zusätzliche Möglichkeit, Indexeinträge sowohl mit Seitenzahlen als auch mit Zeilennummern zu verknüpfen. Mehrfache Folgen der vertrauten nummerierten Fußnoten sind ebenfalls verfügbar.
Auf CTAN unter: [macros/latex/contrib/ledmac/ledmac.pdf](http://www.ctan.org/macros/latex/contrib/ledmac/ledmac.pdf)
- [173] Reinhard Wonneberger und Frank Mittelbach. *Bi $\TeX$ reconsidered*. *TUGboat*, 12(1):111–124, März 1991. ISSN 0896-3207.
Besprechung von $\text{Bi}\TeX$ mit verschiedenen Verbesserungsvorschlägen.
<http://www.tug.org/TUGboat/Articles/tb12-1/tb31wonn.pdf>
- [174] Hermann Zapf. *My collaboration with Don Knuth and my font design work*. *TUGboat*, 22(1/2):26–30, März 2001. ISSN 0896-3207.
Zapf erzählt die Geschichte seiner Zusammenarbeit mit Don Knuth und äußert einige Gedanken zur Typographie.
<http://www.tug.org/TUGboat/Articles/tb22-1-2/tb70zapf.pdf>
- [175] Justin Ziegler. *Technical Report on Math Font Encoding (Version 2)*. Technischer Bericht, $\LaTeX$ 3 project, Juni 1994.
Mit diesem Dokument wurde der Boden für eine Sammlung von 8-Bit-Formelkodierungen für $\TeX$ geebnet.
Auf CTAN unter: [info/latex3pub/13d007](http://www.ctan.org/info/latex3pub/13d007).*

Befehls- und Begriffsindex

In gewisser Weise verbirgt sich hinter diesem Titel die Tatsache, dass alles außer Personennamen in diesem einen langen Index enthalten ist. Damit er sich leichter handhaben lässt, werden alle Einträge nach „Typen“ unterschieden. Der jeweilige „Typ“ wird normalerweise durch eine der folgenden „Typbezeichnungen“ am Anfang eines Haupt- oder Untereintrages gekennzeichnet:

Attribut (einer Sprache), BibTeX-Befehl, BibTeX-Feld, BibTeX-Publikationstyp, BibTeX-Stil, boolesche Var., Datei, Dateierweiterung, Dokumentenklasse, elastische Länge, Fehler, feste Länge, Folio-Stil, Font, Fontkodierung, Funktion (eines Programms), Größenfunktion (NFSS), interne BibTeX-Funktion, Kolumnentitel-Layout, Meldung, Paket, Programm, Option (Klasse oder Paket), Schlüssel (in einer Schlüssel-Wert-Beziehung), Schlüssel/Option (Schlüssel, der auch als Option erlaubt ist), Schlüsselwert, Schlüsselwort (einer Programmiersprache), Syntax, Systemvariable (im Betriebssystem), Umgebung oder Zähler.

Ist keine solche Bezeichnung vorhanden, so handelt es sich entweder um einen L^AT_EX-Befehl oder einfach um einen generellen „Begriff“.

Paketnamen in Klammern verweisen auf die Verwendung in oder mit einem bestimmten Paket, wobei der „virtuelle“ Paketname `lb2` bedeutet, dass ein Befehl in diesem Buch nur zur Veranschaulichung eingeführt wurde.

Eine blaue kursive *Seitenzahl* weist auf ein Beispiel zum jeweiligen Befehl oder Begriff hin, eine fettgedruckte **Zahl** unter mehreren Seitenzahlen auf wichtige Informationen, wie etwa eine Definition oder grundlegende Hinweise zum Gebrauch.

Haupt- und Untereinträge werden auf bis zu drei Ebenen durch Einzüge voneinander abgesetzt, danach und bei allein stehenden Einträgen gegebenenfalls mit Komma. Querverweise auf verschiedene Einträge sind durch Semikolon getrennt.

Die Zeichen \ und . werden für die Sortierung ignoriert, wenn sie am Anfang eines Befehls beziehungsweise einer Dateierweiterung stehen. Alle Symbole erscheinen vor den Buchstaben und alle Einträge, die mit einem @-Zeichen beginnen, stehen unmittelbar vor dem Buchstaben A.

Symbole

- ! Syntax, [288](#), [544](#)
 - (array), [252](#), [254](#), [255](#)
 - (babel), Kurzzeichen, [571](#)
 - (docstrip), [852](#)
 - (makeindex), [671](#), [673](#), [679](#), [680](#), [681](#)
- \!, [522](#)
 - (tipa), [417](#)
- " Syntax, [356](#), [357](#)
 - (BibTeX), [790](#), [798](#)
 - (babel), Kurzzeichen, [568](#), [569](#), [570](#), [591](#), [678](#), [684](#)
 - (makeindex), [673](#), [680](#)
- \", [466](#), [468](#), [684](#)
 - (yfonts), [405](#), [407](#)
- "" Syntax (babel), [570](#)
- "' Syntax (babel), [155](#), [569](#)
- "- Syntax (babel), [570](#)
- "< Syntax (babel), [570](#)
- "= Syntax (babel), [570](#), [573](#)
- "> Syntax (babel), [570](#)
- "(buchst.) Syntax
 - (babel), [564](#), [568](#), [569](#), [583](#), [607](#)
 - (yfonts), [406](#)
- "(8-bit-zeichen) Syntax (babel), [583](#)
- "~ Syntax (babel), [570](#)
- "` Syntax (babel), [155](#), [569](#)
- "| Syntax (babel), [570](#)
- ˆ Syntax, [356](#), [357](#)
 - (BibTeX), [798](#)
 - (babel), Kurzzeichen, [572](#), [579](#), [591](#)
- \', [249](#), [250](#), [468](#), [583](#)
 - in tabbing, [249](#), [250](#)
 - (inputenc), [457](#)
 - (tipa), [417](#)
- ˘(buchst.) Syntax (babel), [572](#)
- (Syntax, [511](#), [553](#)
 - (BibTeX), [798](#)
 - (calc), [904](#)
 - (delarray), [502](#)
 - (makeindex), [680](#)
- \(, [515](#)
 - Fehler bei, [930](#)
 - (ifthen), [910](#)
 - (soul), [95](#)
-) Syntax, [511](#), [553](#)
 - (BibTeX), [798](#)
 - (calc), [904](#)
 - (delarray), [502](#)
 - (makeindex), [680](#)
- \), [515](#)
 - Fehler bei, [930](#)
 - (ifthen), [910](#)
 - (soul), [95](#)
- * (Stern) Fehlermeldungen, [929](#)
- * Syntax, [251](#), [546](#)
 - (array), [258](#)
 - (calc), [206](#), [258](#), [904](#), [905](#), [906](#), [909](#)
 - (docstrip), [852](#)
 - (hhline), [274](#), [275](#)
- *
 - (doc), [854](#)
 - (tipa), [417](#)
- * (buchst.) Syntax (yfonts), [406](#)
- + Syntax, [546](#)
 - (calc), [138](#), [156–158](#), [206](#), [209](#), [235](#), [258](#), [883](#), [894](#), [899](#), [904](#)
 - (docstrip), [852](#)
- \+, Fehler bei, [953](#)
 - , Syntax, [552](#)
 - (BibTeX), [790](#), [798](#)
 - (lb2), [283](#)
- \., [121](#), [134](#), [521](#), [522](#), [541](#), [715](#)
 - in Formeln, [484–486](#), [498](#), [499](#), [505](#), [509](#)
- (Bindestrich)
 - geschützter, [89](#), [90](#), [100](#)
 - Trennung nach, [719](#)
- Syntax, [89](#), [546](#)
 - (calc), [258](#), [900](#), [902](#), [903](#), [904](#), [905](#), [906](#)
 - (docstrip), [852](#)
 - (hhline), [274](#), [275](#)
- \-, [249](#), [255](#), [257](#), [570](#), [986](#)
 - Fehler bei, [953](#)
 - (soul), [95](#)
 - (ulem), [93](#)
- Syntax, [89](#)
- Syntax, [89](#), [356](#)
- . Syntax, [511](#), [544](#)
 - (babel), Kurzzeichen, [574](#)
 - (lb2), [283](#)
- \., [469](#), [583](#)
 - ... (Auslassungspunkte)
 - Abstände, [87–89](#)
 - mathematisches Symbol, [508](#), [509](#)
 - .pybrc.conf Datei (pybliographic), [814](#)
 - / Syntax, [511](#), [544](#)
 - (calc), [258](#), [904](#), [906](#)
 - (docstrip), [852](#)
- \/, [351](#), [352](#), [353](#)
 - Ligatur aufbrechen, [191](#), [728](#)
 - (soul), [95](#)
- : Syntax, [551](#)
 - (arydshln), [275](#), [276](#)
 - (babel), Kurzzeichen, [571](#)
 - (hhline), [274](#), [275](#)
- \:, [521](#), [522](#), [541](#)
 - (tipa), [417](#)
- :: Syntax
 - (arydshln), [276](#)
 - (hhline), [275](#)
 - (yfonts), [406](#)
- ; Syntax, [552](#)
 - (arydshln), [276](#)
 - (babel), Kurzzeichen, [571](#), [607](#)
- \;, [521](#), [522](#), [541](#)
 - (tipa), [417](#)
- < Syntax, [548](#)
 - (array), [252](#), [254](#), [256](#)
 - (babel), Kurzzeichen, [574](#), [591](#)
 - (ifthen), [906](#), [908](#), [909](#)

- \<
 - Fehler bei, 929, 953
 - (soul), 96, 97
- << Syntax (babel), 574, 606
- = Syntax, 548
 - (BibTeX), 790, 798
 - (babel), Kurzzeichen, 573, 598
 - (hhline), 274, 275
 - (ifthen), 906, 908
- \=, 249, 250, 469
 - Fehler bei, 950
 - in tabbing, 249
 - (inputenc), 457
 - (tipa), 417
- =(<buchst.) Syntax (babel), 573
- > Syntax, 548
 - (array), 252, 253–258, 272
 - (babel), Kurzzeichen, 574, 591
 - (colortbl), 273
 - (ifthen), 908, 909
 - (tabularx), 259, 260
- \>, 249, 250
 - Fehler bei, 953
 - (soul), 96, 97
- >> Syntax (babel), 574, 606
- ? Fontkodierung, 465
- ? Syntax, 544
 - (babel), Kurzzeichen, 571
- (QED)-Zeichen, 151, 152
- [Syntax, 511, 553
- \[, 481, 493, 515, 927
 - falscher Abstand oberhalb, 492, 493
 - Fehler bei, 930
 - (amsmath), 482
- # Syntax, 157
 - in TeX-Fehlermeldungen, 943, 946, 954
 - (BibTeX), 798, 799, 800
 - (bibtool), 810
 - (hhline), 274, 275
- \#, 514, 541, 544, 545
- ## Syntax, 157, 866
- % Syntax (BibTeX), 798
- \%, 544
- %<...> Syntax (docstrip), 851, 852
- %<<...> Syntax (docstrip), 866
- %% Syntax (docstrip), 866
- & Syntax, 250
 - Fehler bei, 934, 941, 951
 - (amsmath), 482, 485, 487–490, 498–500
 - Fehler bei, 934
 - (docstrip), 852
- \&, 544, 545
 - Fehler bei, 941
- \$ Syntax, 254, 515
- \\$, 468, 544
- _ Syntax
 - Fehler bei, 944
 - (index), 704
- _, 470, 544
- \\, 111, 250, 272, 501, 893
 - Fehler bei, 951
 - in tabbing, 249, 250
 - in Überschriften, 26, 33
 - Probleme in tabular, 112
 - (amscd), 501
 - (amsmath), 482, 483, 484–491, 493, 494–500, 505
 - (array), 252, 254, 255
 - (booktabs), 279
 - (delarray), 502
 - (fancyhdr), 233
 - (longtable), 269
 - (soul), 96, 97
 - (supertabular), 264, 265
 - (tabularx), 260
- *, 269
 - (amsmath), 483, 494
 - (longtable), 269
- \(sprache)hyphenmins (babel), 596, 602, 603
- \(num)headLines Option (typearea), 213
- \{, 475, 487, 495, 511, 514, 542, 553
 - { Syntax
 - (BibTeX), 790, 795–797, 798
 - (makeindex), 680
 - { } Syntax, 86, 486, 499, 519
 - (xspace), 87
- \}, 475, 487, 495, 511, 514, 542, 553
 - } Syntax
 - (BibTeX), 790, 795–797, 798
 - (makeindex), 680
- \^, 470
 - (tipa), 417
- ^ Syntax
 - (babel), Kurzzeichen, 573
 - (index), 704
- ^(<buchst.) Syntax (babel), 572, 573
- ^| Syntax (babel), 573
- \~, 475
 - (tipa), 417
- ~ Syntax, 571, 989
 - geschütztes Leerzeichen, 96
 - (babel), Kurzzeichen, 591
 - (babel), geschütztes Leerzeichen, 568
 - (babel), in verschiedenen Sprachen, 570, 571
 - (hhline), 274, 275
- Syntax (babel), 570
- Syntax (babel), 570
- Syntax (babel), 570
- ~(<buchst.) Syntax (babel), 570, 571
- _, 86
- \], 481, 515, 927
 - Fehler bei, 930
 - (amsmath), 482
-] Syntax, 511, 553
- \', 249, 470
 - in tabbing, 249
 - (inputenc), 457
 - (textcomp), 376
 - (tipa), 417

- ‘ Syntax
 - (babel), Kurzzeichen, [571](#), [572](#), [591](#)
 - (dvips), [644](#)
- ‘(buchst.) Syntax (babel), [571](#), [572](#)
- \l, 511, 544
 - (tipa), [417](#)
- | Syntax, 251, 511, 544
 - (array), [252](#), [253-255](#), [257](#), [276](#)
 - (babel), Kurzzeichen, 591
 - (booktabs), 277
 - (docstrip), [852](#)
 - (hhline), [274](#), [275](#)
 - (ltxdoc), 867
 - (makeindex), [672](#), 679, [680](#)
 - (tabs), [277](#)
 - (tabularx), [259](#), [260](#)
 - (tabulary), [261](#), [262](#)
- | (Syntax (makeindex), [671](#), [672](#), 680, 681
- |) Syntax (makeindex), [671](#), [672](#), 680, 681
- |see Syntax (makeindex), [671](#), [673](#)
- || Syntax, 251
 - (booktabs), 278
 - (hhline), [275](#)
- 10pt Option, 207, [915](#)
 - (amsmath), 491
- 11pt Option, 18, 152, 207, 354
- 12pt Option, 207
- 1 Syntax (paralist), [140](#), [142](#), [143](#), [145](#)
- 8859-8 Option (inputenc), 594
- 88591lat.csf Datei (bibtex8), [787](#)
- 8r.enc Datei, 399, [432](#)
- € (Euro-Zeichen), [418-423](#)
- @
- @ Syntax, 251, 254, [280](#), [544](#)
 - Fehler bei, 943
 - in Befehlsnamen, 20, 875
 - (BibTeX), [790](#), 794
 - (makeindex), [672](#), [673](#), 679, [680](#)
- \@, [14](#), [87](#), 721
 - Fehler bei, 955
- @(((Syntax (amscd), 500
- @))) Syntax (amscd), 500
- @. Syntax (amscd), 501
- @<<< Syntax (amscd), 500
- @= Syntax (amscd), [501](#)
- @>>> Syntax (amscd), [500](#), [501](#)
- @{} Syntax, [233](#), [255](#), [256](#), [278](#), [280](#)
- @AAA Syntax (amscd), [500](#), [501](#)
- \@addtoreset, [16](#), 27, 119, [225](#), 497, [884](#), 885, [887](#)
- \@afterheading, [35](#), 908
- @afterindent boolesche Var., 35, [908](#)
- \@afterindentfalse, [35](#)
- \@beginparpenalty, [154](#)
- \@biblabel, [717](#)
 - (natbib), 717
- \@cite, [716](#), 717
- \@dotsep, 54
- \@dottedtocline, [53](#), 54, [55](#), [57](#)
- \@endparpenalty, 154
- \@evenfoot, 231, [232](#)
- \@evenhead, 231, [232](#)
 - @firstcolumn boolesche Var., 908
- \@float, [317](#)
- \@gobble, 918
- \@idxitem, [702](#), [703](#)
 - (doc), 856
- \@ifpackagelater, 921
- \@ifpackageloaded, 921
- \@ifpackagewith, 921
- \@include, 22
 - @inlabel boolesche Var., 908
- \@itempenalty, 154
- \@listi, 152
- \@listii, 152
- \@listiii, 152
- \@makecaption, [316](#), [317](#)
- \@makefigcaption (lb2), [317](#)
- \@makefnmark, [120](#)
- \@makefntext, [120](#), [121](#)
- \@makeschapterhead, [702](#)
- \@makewincaption (picinpar), 116
- \@mkboth, 231
 - @newlist boolesche Var., 908
 - @noskipsec boolesche Var., 908
- \@oddfloat, 231, 926
- \@oddhead, 231, [232](#)
- \@pnumwidth feste Länge, [54](#), [55](#), 65
 - @preamble BibTeX-Befehl, [800](#), 839, 879
 - (bibextract), 807
- \@ptsize, [207](#)
- \@removefromreset (remreset), [884](#)
- \@secntformat, 28, [29](#)
- \@secpenalty, 45, 983
- \@startsection, 29, 30, 31, [32-34](#), [295](#), 892
 - Fehler bei, 955
 - mit Gleitobjektbarriere, [297](#)
- \@starttoc, [58](#)
 - (notoccite), 722
- @string BibTeX-Befehl, [798](#), [799](#)
 - (BibTeXMng), 820
 - (bibextract), 807
 - (bibtool), [811](#)
- \@tabacckludge, [457](#), [458](#), [464](#)
- \@tempboxa, [316](#), 317
 - @tempswa boolesche Var., [716](#), [908](#)
- \@thefnmark, [120](#), [121](#)
- \@tocrmarg, 54, 55
- \@topcaptionfalse (supertabular), 265
- @twocolumn boolesche Var., [703](#), [908](#)
- @twoside boolesche Var., [207](#), [908](#)
- @VVV Syntax (amscd), [500](#), [501](#)
- @| Syntax (amscd), 501
- A
- A Syntax (paralist), [140](#), [143](#), [145](#)
- a Syntax (paralist), [139](#), [140](#), [145](#)
 - Probleme mit, [141](#)
- \a', [249](#), [250](#)
 - a.a.O. (am angegebenen Ort), *siehe* Ibidem
- aO Option (crop), 221

- a0paper Option, 204
 - (typearea), 212
- a0paper Schlüssel/Option (geometry), 215
- a1 Option (crop), 221
- a1paper Schlüssel/Option (geometry), 215
- a2 Option (crop), 221
- a2paper Schlüssel/Option (geometry), 215
- a3 Option (crop), 221
- a3paper Schlüssel/Option (geometry), 215
- a4 Option (crop), 221
- a4 Paket, 207, 210
- a4dutch Paket, 211
- a4paper Option, [18](#), [204](#), [913](#)
 - (typearea), 212
- a4paper Schlüssel/Option (geometry), 215
- a4wide Paket, 211
- a5 Option (crop), [221](#), [222](#), [223](#)
- a5 Paket, 211
- a5comb Paket, 211
- a5paper Option, 204
 - (typearea), [212–214](#)
- a5paper Schlüssel/Option (geometry), 214, 215
- a6 Option (crop), 221
- a6paper Option (typearea), 212
- a6paper Schlüssel/Option (geometry), 215, [217](#), [219](#)
- \a=, 249
- \a', [249](#)
- Abbildungen
 - am Dokumentende platzieren, 298–300
 - von Text umflossen, [115](#), [116](#), 307, 308, [309](#), [310](#), 311
- Abbildungsverzeichnisse, 48–69
 - in Inhaltsverzeichnissen, 51
- abbrv $\BbbTeX$ -Stil, [717](#), [796](#), 821, 822, [824](#), 837
 - (bibtopic), [780](#)
- abbrvnat $\BbbTeX$ -Stil (natbib), [709](#), [732](#), [735](#), [740](#), 821
- abgerundete Ecken, Boxen, [614](#), [615](#)
- abgesetzte Überschriften, 29, [30](#)
- Abkürzungen, *siehe* Kurzformen
- \above, 506
- above Option (placeins), 297
- \abovecaptionskip elastische Länge, [316](#), [317](#), 321
- \abovedisplayshortskip elastische Länge, [492](#)
- \abovedisplayskip elastische Länge, [491](#), [492](#)
- \aboveruleskip feste Länge (booktabs), 278
- aboveskip Schlüssel/Option (caption), [320](#), [321](#), [328](#), 329
- \abovetopsep feste Länge (booktabs), 278
- \abovewithdelims, 506
- \abs (lb2), [512](#), [514](#)
- Absätze
 - eingerahmte, [618](#)
 - Einzug nach einer Überschrift, fremdsprachige Dokumente, 581
 - Fehlersuche, 985–989
 - Flattersatz, 110, 111, [113](#), 114
 - Hurenkind, 984
 - Initiale, [106–108](#), 406, [407](#)
 - Layoutparameter, 113
 - linksbündige, 110, 111, [113](#), 114
 - mit Bildern, [115](#), [116](#)
 - Absätze (*Forts.*)
 - mit Randausgleich, 108–110, [111](#), [112](#), 113, 114
 - mit rechteckigen Löchern, [115](#), [116](#)
 - ohne Randausgleich, 110–114
 - rechtsbündiger Flattersatz, 111, 112
 - Schusterjunge, 984
 - verkürzen, 989
 - verlängern, 989
 - Waisenkind, 984
 - Witwe, 984
 - Wortzwischenräume in, 109, 110
 - Zeilenvorschub/-abstände, [114](#), 115, [354](#), 383, *siehe auch* Durchschuss
 - zentrierte, 111, 112
- Absatzboxen, 893, [896–899](#)
- Absatzdurchschuss in Legenden, 320, [321](#)
- Absatzformat für Inhaltsverzeichnisse, 65, [66](#), [67](#)
- Absatzoptionen in Tabellen, [253](#), [254](#)
- Absatzumbruch
 - Behandlung der vorletzten Zeile, [882](#)
 - Fehlersuche, 985–989
 - konfigurieren, 882
 - überwachen, 986–989
- Abschneiden von Text in Kolummentiteln, 240, [241](#)
- Abschnitte
 - drucken, 21, 22, 848, 868
 - Inhaltsverzeichnisse für, [61](#), 62
 - mit ltxdoc-Klasse erzeugen, 868
 - mit doc-Paket erzeugen, 848
- Abstände
 - Geviert, [41](#)
 - Gleitobjekte, 293
 - horizontale, in Formeln , 521, 522
 - in Inhaltsverzeichnissen, 51
 - Legenden, 321, 326
 - Mathematiksatz, 514, 515, [516](#), [518–520](#), 521
 - mathematische Symbole, [541](#), [542](#), [545](#)
 - mehrseitige Tabellen, 269
 - nach Makronamen, 86, [87](#)
 - nach Satzzeichen, fremdsprachige Dokumente, 580
 - Parameter, *siehe* Längenparameter
 - Sperren, 94, [95–99](#)
 - Tabellenspalten, [255](#), [256](#)
 - Tabellenzeilen, 252, [253](#), [277](#), 279
 - theoremähnliche Strukturen, 149
 - Überschriften, *siehe* Überschriften, Abstände
 - um Formeln, 491, [492](#), 493, 494
 - wortwörtliche Ausgabe, [167](#)
 - Wortzwischenräume, 109, 110
 - Zeilenvorschub/-abstände, [114](#), 115, [354](#), 383, *siehe auch* Durchschuss
 - zwischen Fußnoten und Text, 119
- abstract Umgebung, [36](#)
- abstract $\BbbTeX$ -Feld, 792, 821
 - (BibTeXMng), 820
 - (printbib), 805
- abstract $\BbbTeX$ -Stil, 821
- \abstractname, [36](#), [37](#)
 - (babel), 561
- acadian Option (babel), 559

- \accent, 340, 341, 348, 365, **442**, 464, 607
- \accentedsymbol (amxtra), 479
- accents Paket, **507**, 1013
- \accentset (accents), **507**
- \Acite (babel), 580
- \acite (babel), 580
- acm BibTeX-Stil, 821
- \acro (lb2), **352**
- Acrobat Distiller Programm, 661
- \active@char(zeichen) (babel), 606
- activeacute Option (babel), **571**, **572**, **598**
- activegrave Option (babel), **571**, **598**
- actual Schlüsselwort (makeindex), **680**, **683**
- \actualchar (doc), 854
- \acute, 545
- Ada Schlüsselwert (listings), **179–181**
- \add (lb2), **501**
- add.period\$ interne BibTeX-Funktion, 839, **841**
- \addcontentsline, **35**, **49**, **50**, **51**, **52**, 56, 58, **703**
 - Probleme mit \include, 52
 - (titleref), 82
- \adddialect (babel), **601**
- \addlanguage (babel), 601
- \addlinespace (booktabs), **279**, **280**
- \addpenalty, **45**, **892**, 893
 - erzeugte Ausgabe, **982**
 - address BibTeX-Feld, **714**, **742**, **791**, **792**, **801**, **808**, **809**
- \AddThinSpaceBeforeFootnotes (babel), **582**
- \AddTo (jurabib), **748**, **753**, **759**, **760**, **761**
- \addto (babel), **14**, **36**, **56**, 78, **566**, **605**, **751**, 759
- \addtocontents, 49, **51**, **52**, 62
 - Probleme mit \include, 52
- \addtocounter, 26, **884**, 885
 - Fehler bei, 945, 946
 - (calc), 904
 - Fehler bei, 929
- \addtolength, **887**, **905**
 - Fehler bei, 946
 - (calc), **904**, **905**
 - Fehler bei, 929
- \addvspace, **35**, **51**, 62, **64**, **65**, **67**, **891**, **892**, 893
 - erzeugte Ausgabe, **982**
 - Fehler bei, 949
- adjust Option (cite), 720
- \ADLdrawingmode (arydshln), 276
- Adobe Reader Programm, 82, 660, 661
- \advance, 904
- \AE, 356, **469**
- \ae, 470
 - (tipa), **417**
- ae Paket, 368
- Änderungen markieren, *siehe* Revisionsbalken
- Änderungsverlauf, erzeugen
 - doc-Paket, 849
 - ltxdoc-Klasse, 869
- Äthiopisch, 608
- affiliation BibTeX-Feld (BibTeXMng), 820
- afm2tfm Programm, 1028
- afrikaans Option (babel), **559**, **601**
- \afterpage (afterpage), 297, **298**, **304**
- afterpage Paket, 297
- aftersave Schlüssel (fancyvrb), 175, **176**
- agsm BibTeX-Stil
 - (harvard), **725**, **821**
 - (natbib), **728–731**, **733**
- agu BibTeX-Stil
 - (bibentry), **736**
 - (natbib), **730**, **731**
- Akut-Akzent (‘), Kurzzeichen, **572**
- Akzentbuchstaben, *siehe auch* Sonderzeichen; Textsymbole
 - Eingabekodierung, 369, **370**, 371, 372
 - in OT1-Kodierung, 348
 - in Befehls- und Umgebungsnamen, 874
 - in Bibliographiedatenbanken, 797, 798
 - in fremdsprachigen Dokumenten, **568**
- Akzente
 - als hochgestellte Zeichen, 479, **507**
 - als mathematische Symbole, **545**
 - in Tabellen, **249**, **250**
 - Punkte als, **507**
- \aleph, 544
- alg Paket, 177
- algorithmic Paket, 177
- \aliasshorthand (babel), **564**
- align Umgebung (amsmath), **481**, **482**, **487**, **488**, 489, **496**, **497**
 - Fehler bei, 930, 941
 - mit \minalignsep justieren, **489**
 - unterbrechen, **491**
- align* Umgebung (amsmath), **481**, **505**, **510**
- aligned Umgebung (amsmath), **481**, **489**, **490**, 498, 934
 - Fehler bei, 930, 933
 - mit \minalignsep justieren, **491**
- all Schlüsselwert (fancyvrb), 167
 - (jurabib), **745**, **746**, **747**, **748**, **749**, **760**, **761**
- \allcaps (lb2), **98**
- \allinethickness (eepicmu), 629
 - (eepic), 628
- \allletters (lb2), **398**
- \allowdisplaybreaks (amsmath), **480**, **494**
- \allowhyphens (babel), **607**
- allowmove Option (url), 100
- allreversed Schlüsselwert (jurabib), **749**, **764**
- alltt Umgebung (alltt), **160**
- alltt Paket, 160
- Almost European Fonts, 338, 367
- almostfull Option (textcomp), 375
- Alph Folio-Stil, 224
- \Alph, **28**, **35**, **137**, 140, **885**
 - Fehler bei, 932
 - (babel), 576
- \alph, 137, 140, **885**
 - Fehler bei, 932
 - (babel), 576
 - (perpage), 128
- alph Folio-Stil, 224
- \alpha, **403**, **502**, **514**, **543**

- alpha $\LaTeX$ -Stil, 821, 822, 825, 837, 838, 842
 - Labelgenerierung, 794, 797
 - (biblist), 804
 - (multibib), 782
- Alphabetbefehle, 360–362
- alphabetisch nummerierte Überschriften, 28
- \Alphfinal (babel), 576
- alpine Option (ifsym), 416
- \AlsoImplementation (doc), 849, 853, 869
- \alsoName (babel), 561
- altDescription Umgebung (lb2), 157, 883
- altdeutsche Fonts, 405, 406, 407
- alte deutsche Rechtschreibung, *siehe* Deutsch
- \AltMacroFont (doc), 856
- alwaysadjust Option (paralist), 142, 143
- \amalg, 546
 - (mathptmx), nicht verfügbar, 389
- american Option (babel), 559, 567
- American Mathematical Society (AMS), 479, 480
- AMS (American Mathematical Society), 479, 480
- amsalpha $\LaTeX$ -Stil, 821
- amsart Dokumentenklasse, 479, 725, 1012
- amsbook Dokumentenklasse, 479, 725, 1012
- amscd Paket, 479, 500, 501
- amsfonts Paket, 395–397, 480, 523
 - Bereitstellen von latexsym-Symbolen, 476
- $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$
 - Akzente als hochgestellte Zeichen, 479
 - Beweisumgebung, 151, 152
 - Dokumentation, 479
 - Dokumentenklassen, 479
 - Erstellen kommutativer Diagramme, 479
 - Fonts, 479, 480
 - Formatieren von Querverweisen, 479
 - Formatieren von Textfragmenten, 479
 - Kurzformen für Umgebungen, 480
 - Paketoptionen, 478, 479
 - theoremähnliche Strukturen, 146–152, 479
 - Unterpakete, 479
 - zerbrechliche Befehle, 480
- amsmath Paket, 89, 90, 146, 477–500, 502–522, 540, 553, 1012
 - Fehler bei, 923
 - vs. Standard- $\LaTeX$, 482, 483
- amsmath.dtx Datei (amsmath), 483, 496
- amsopn Paket, 479
- amspain $\LaTeX$ -Stil, 821
- amsproc Dokumentenklasse, 479, 1012
- amsrefs Paket, 1016
- amssymb Paket, 395–397, 403, 480, 523, 525, 540–554
 - Bereitstellen von latexsym-Symbolen, 476
- amssymb.sty Datei (amssymb), 545
- $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{T}\mathcal{E}\mathcal{X}$, 477, 478
- amstext Paket, 479
- amsthm Paket, 146–152, 479, 1012
 - Probleme mit ragged2e, 150
- amsxport Paket, 1016
- amsxtra Paket, 479, 507
- \anchor (dingbat), 411
- \and (ifthen), 910
- and Schlüsselwert (jurabib), 743, 744
- and Schlüsselwort ($\LaTeX$), 796
- and others Schlüsselwort ($\LaTeX$), 714, 796, 823–827
- \andname (jurabib), 761
- Anführungsstriche in Programmlistings, 161, 162, 163
- Anführungszeichen
 - ("), Kurzzeichen, 568–570
 - deutsche, 569
 - französische, 569, *siehe auch* Guillemets
 - in fremdsprachigen Dokumenten einfügen, 569, 570
 - spanische, 574
- angehängte Leerzeichen, Index, 670, 678, 688, 691
- \angle (amssymb), 544
- angle Option (natbib), 731
- angle Schlüssel (graphicx), 637, 640, 641
 - Fehler bei, 933
- annotate $\LaTeX$ -Feld, 842
- annotate $\LaTeX$ -Stil, 821, 841, 842
- annotation $\LaTeX$ -Stil, 821, 841
- annotatorfirstsep Schlüssel/Option (jurabib), 743, 749
- annotatorformat Schlüssel/Option (jurabib), 743, 759
- annotate Schlüssel/Option (jurabib), 766, 768
- annotate $\LaTeX$ -Feld, 792, 821, 841
 - (custom-bib), 832, 833
 - (jurabib), 766, 768
- annotate $\LaTeX$ -Stil, 792
- ansinew Option (inputenc), 96, 106–108, 371, 427, 691
- \answer (lb2), 860
- ante Schlüssel (lettrine), 108
- Antiqua (Schriftform), 343
- Anwendermeldungen erzeugen, 860, 861
- any Schlüsselwort (makeindex), 679
- Anzeigen formatierter Seiten, *siehe* Seitenbeschreibungssprachen
- apa $\LaTeX$ -Stil, 821
- \Apageref (babel), 580
- \apageref (babel), 580
- apalike $\LaTeX$ -Stil (apalike), 821, 822
- apalike Paket, 717, 821
- apalike2 $\LaTeX$ -Stil (apalike), 821
- \Appendix (lb2), 34, 35
- \appendix, 24, 34
 - (lb2), 35, 36
- \appendixname, 35, 37, 41
 - (babel), 561
- appleMac Option (inputenc), 371
- \approx, 548
- \approx (amssymb), 548
- apy Schlüssel (jurabib), 744
- \arabic, 27, 28, 137, 140, 429, 882, 884, 885, 886, 887
- arabic Folio-Stil, 224
- Arabisch, 608
- arabische Zahlen, Überschriften, 27
- \arc
 - (curves), 630
 - (eepicmu), 629
 - (eepic), 628, 629

- `\arccos`, 513
- `\arcctg` (babel), 580
- `\arch` (babel), 580
- `\arcsin`, 513
- `\arctan`, 513
- `\arctg` (babel), 580
- `\Aref` (babel), 580
- `\aref` (babel), 580
- `\arg`, 513
- `arg_close` Schlüsselwort (makeindex), 680
- `arg_open` Schlüsselwort (makeindex), 680
- Argumente, *siehe auch* Schlüssel
 - bewegte, 81, 100, 455, 927, 928
 - Einschränkungen, 878, 928
 - nicht verfügbar, 880
 - optionale, 877, 882, 883
 - wortwörtliche Ausgabe in, 174, 175, 176, 177
- arithmetische Ausdrücke, 904, 905
- `ark10.mf` Datei (dingbat), 411
- Armenisch, 608
- `array` Umgebung, 111, 112, 248, 250, 251, 255, 482, 498, 499, 501, 502, 896
 - Fehler bei, 938, 942, 943
 - Fußnoten in, 285
 - Layoutparameter, 251
 - (array), 254–256, 281, 282
 - (delarray), 502
 - (tabs), 277
- `array` Paket, 251–259, 288–1137
 - kombiniert mit `arydshln`, 275
 - kombiniert mit `booktabs`, 278
 - kombiniert mit `color`, 272
 - kombiniert mit `supertabular`, 264
 - nicht kompatibel zu `tabs`, 277
- `\arraybackslash`
 - (array), 255, 257
 - (tabularx), 259, 260
- `\arraycolsep` feste Länge, 251, 255 (amsmath), 499
- `\arraylinesep` feste Länge (tabs), 277
- `\arrayrulecolor` (colortbl), 273
- `\arrayrulewidth` feste Länge, 251, 258, 274, 275 (hhline), 275
- Arrays, äußere Begrenzungszeichen, 502
- `\arraystretch`, 251, 253, 275, 276, 277
- `\arrowlength` (pspicture), 658, 659, 660
- `\Arrownot` (stmaryrd), 551
- `\arrownot` (stmaryrd), 551
- `arrows` Umgebung (lb2), 190
- `\Arrowvert`, 511, 544
- `\arrowvert`, 511, 544
- `article` L^AT_EX-Publikationstyp, 714, 791, 799 (jurabib), 744
- `article` Dokumentenklasse, 7, 14, 122, 126, 155, 204, 231, 479, 702, 804
 - Alternativen zu, 244, 245
 - Fußnotennummerierung, 119
 - Gliederungsbefehle, 24, 25, 27, 55
- `arydshln` Paket, 275, 276
- `\Asbuk` (babel), 576
- `\asbuk` (babel), 576
- `ascii` Option (inputenc), 371, 968
- ASCII-Textdateien, 7
- `\Ask` (docstrip), 860, 861
- `\askforoverwritefalse` (docstrip), 861
- `\askforoverwritetrue` (docstrip), 861
- `askinclude` Paket, 22
- `\askonceonly` (docstrip), 861
- `\AskOptions` (optional), 24
- `asparadesc` Umgebung (paralist), 143, 146
- `asparaenum` Umgebung (paralist), 140, 141
- `asparaitem` Umgebung (paralist), 143
- `\ast`, 507, 546
- `astron` L^AT_EX-Stil, 821
- `asymmetric` Schlüssel/Option (geometry), 216, 217
- asymmetrisches Seitenlayout, 217
- `\asymp`, 548
- `\AtBeginDelayedFloats` (endfloat), 299
- `\AtBeginDocument`, 434, 868, 869, 911, 917
- `\AtBeginFigures` (endfloat), 299
- `\AtBeginTables` (endfloat), 299
- `\AtEndDocument`, 225, 869, 911, 917
- `\AtEndOfClass`, 911, 917, 920
- `\AtEndOfPackage`, 911, 917
- `\AtForty` (marvosym), 413
- `\athnum`
 - (athnum), 579
 - (babel), 579
- `\atop`, 506
- `\atopwithdelims`, 506
- aufrechte Schriftform, 344, 351
- Aufteilen von Material über mehrere Seiten, *siehe* Gleitobjekte
- Aufzählungen
 - Erweiterungen, 143
 - Layoutparameter, 136
 - Standard, 135, 136
 - Voreinstellungen, 144, 145, 146
- ausführliche Quellenverweise im Fließtext
 - Autor-Jahr-Zitierschema, 736, 737
 - Kurztitel-Zitierschema, 748, 749–751
- ausführlicher Modus, Indexerstellung, 697
- Ausgabedateien
 - Codeerzeugung, 859, 860
 - Index, 678, 690, 697
- Ausgabekodierung, *siehe* Fontkodierung
- ausgelassenen Text markieren, *siehe* Auslassungspunkte
- Ausgleichen von Spalten, 196, 197
- Auslassungspunkte (...)
 - Abstände, 87–89
 - mathematisches Symbol, 508, 509
- Ausrichtung
 - Formelgruppen
 - ausgerichtet, 487
 - nicht ausgerichtet, 487
 - Formeln
 - mehrfache Ausrichtung, 487, 488, 489
 - mehrzeilig, ausgerichtet, 485, 486
 - mehrzeilig, nicht ausgerichtet, 483, 484, 485
 - Position der Tags, 481, 482

Ausrichtung (*Forts.*)

- Inhaltsverzeichnisse, 63, [64](#), [65](#)
- Mathematiksatze, [518-520](#), 521
- Randausgleich, optischer, 1135
- Tabellen
 - am Dezimalzeichen, [280](#), 282, [283](#), [284](#)
 - horizontal, 268
 - vertikal, 254, [281](#), [282](#)
- Überschriften, 39

Ausrufezeichen (!), Kurzzeichen, [571](#)Ausschließen von Dateien, 22, *siehe auch* Einbinden von Dateien

australian Option (babel), 559

austrian Option (babel), 559, 563, [567](#), 759

\author, Warnung bei, 968

author BibTeX-Feld, [714](#), 757, 791, 792, 793, 794, [795-798](#), [801](#)(jurabib), [742](#), 744

authordate1 BibTeX-Stil (authordate1-4), 724, 820, 821

authordate1-4 Paket, 724, 821

authordate2 BibTeX-Stil (authordate1-4), [724](#), 820, 821

authordate3 BibTeX-Stil (authordate1-4), 724, 820, 821

authordate4 BibTeX-Stil (authordate1-4), 724, 820, 821

authorformat Schlüssel/Option (jurabib), [743](#), [744](#), [746](#), [749](#), [755](#), [758](#), [759](#), 761, 763, [764](#)

authoryear Option (natbib), 734, 739

auto Schlüsselwert (fancyvrb), 167, 172

Auto-Vervollständigung, Seitenlayout, [214](#), 215, 216, [217](#), 218, [219](#), 220autodotlist Umgebung (pifont), [391](#)

automatische Indexierung deaktivieren

doc-Paket, 849

ltxdoc-Klasse, 869

Autor-Jahr-Zitierschema, 722-737, *siehe auch* ZitierschemataAnpassen von Bibliographien, [732](#)Anpassen von Quellenverweisen, [730](#), [731](#)ausführliche Quellenverweise im Fließtext, [736](#), 737

Autor mit Nummer, Umschalten auf, 739

Autoren nur beim ersten Verweis auflisten, [729](#), [730](#)

Definition, 708

elektronische Veröffentlichungen, 735

erzwingen, [734](#)fehlende Autorenaten, [733](#), [734](#)

fehlende Jahresdaten, 733, 734

Geschichte, [724](#), [725](#)

kein Umbruch zwischen Autorennamen, 731

Kurztitelformat, kombinieren mit, [758](#), [759](#)mehrfache Quellenverweise, [728](#), [729](#)

numerische Verweise, Umschalten auf, 739

unterstützte Stile, 735, 736

Verweise automatisch indexieren, 734

Autor-Nummer-Zitierschema, 737, *siehe auch* ZitierschemataAnpassen von Quellenverweisen, [740](#)Autor-Nummer-Zitierschema (*Forts.*)

Beschreibung, 737

Definition, 710

Sortierreihenfolge, [739](#)Verweise komprimieren, [739](#)

Autoren, Bibliographien

fehlende Informationen, [733](#), [734](#)Geschlecht, 760, [761](#), 768

Informationsfeld, 768

kein Umbruch zwischen Autorennamen, 731

nur beim ersten Quellenverweis auflisten, [729](#), [730](#)Trennzeichen und -wörter bei Auflistungen, 761, [762](#), 764

Autorenverzeichnisse, 703, 746

.aux Dateierweiterung, **8**, **9**, 21, 138, 712, 713, 716, 772, 823(BibTeX), **9**, 786, 823

(aux2bib), 804

(bib2html), 806

(bibtopic), 780

(chapterbib), 773

(citetags), 807

(footmisc), 123

(index), 704

(longtable), 267

(mparhack), 135

(multibib), 783

(perpage), 128

aux2bib Programm, **804**, 817avant Paket, **382**, 385

Avant Garde Gothic Font, 385

awk Programm, 805, 807

\Az (babel), 580

\az (babel), 580

B\b, 465, **470**

b Syntax, 253

(array), 252, [253](#), 258(delarray), [502](#)(hline), [274](#), [275](#)

b0 Option (crop), 221

b0paper Schlüssel/Option (geometry), 215

b1 Option (crop), 221

b1paper Schlüssel/Option (geometry), 215

b2 Option (crop), 221

b2paper Schlüssel/Option (geometry), 215

b3 Option (crop), 221

b3paper Schlüssel/Option (geometry), 215

b4 Option (crop), 221

b4paper Schlüssel/Option (geometry), 215

b5 Option (crop), 221

b5paper Option, 204

(typearea), 212

b5paper Schlüssel/Option (geometry), 215

b6 Option (crop), 221

b6paper Option (typearea), 212

b6paper Schlüssel/Option (geometry), 215

Bögen zeichnen, 628

- Bücher, *siehe* Dokumente
- ba Paket, 537
- babel Paket, 14, 555, 557, **558**, 559, **560–607**, 725, 759, 775, 956
- Benutzerschnittstelle, 560–595
 - Beschreibung, 558, 559
 - Fehler bei, 923, 940, 945, 950, 955
 - in diesem Buch, 1137
 - Optionen (Sprachauswahl), 559
 - Paketdatei, 598
 - Silbentrennung in fremdsprachigen Texten, 596–598
 - Sprachdefinitionsdateien
 - Aufbau, 596, **598–607**
 - Copyright-Vermerk, 598
 - Definitionen hinzufügen, **605**
 - Dokumentation, 600
 - Dokumentationstreiber, 599
 - Identifikation der Sprache, 599
 - Initialisierung, 600
 - Interpunktion, Sonderfälle, **607**
 - Kurzformen, 606, 607
 - Lizenzdaten, 598
 - Sprachen und Dialekte definieren, 600–602
 - sprachspezifische Zeichenfolgen übersetzen, 603
 - Trennmuster einstellen, 602, 603
 - Versionsdaten, 599
 - sprachspezifische Zeichenfolgen, **561**, 565, **566**, **567**, 596, 603
 - Warnung bei, 976
- babel Paket, Optionen (Sprachauswahl)
- Erwägungen zum Layout, 580–582
 - Fontkodierung, **583**, **594**
 - OT1, 583
 - T1, 583
 - T2A, 589
 - T2B, 589
 - T2C, 589
 - Kurzformen, 568–574
 - sprachspezifische Befehle, 574–580
 - Übersetzungen, 566, 567
- babel.def Datei (babel), 596
- babel.sty Datei (babel), 598
- \backepsilon (amssymb), 551
- \backmatter, 24
- \backprime (amssymb), 544
- backref Option (hyperref), 82
- \backsim (amssymb), 548
- \backsimeq (amssymb), 548
- \backslash, 511, 544
- Badness-Grad, Zeilenumbrüche, 892
- bahasa Option (babel), 559
- Balancieren von Spalten, **196**, 197
- balancingshow Option (multicol), 197
- Balkendiagramme, **631**
- \balpha (lb2), 527
- Band, Titel in Bibliographien, 769
- \bar, 545
 - (bar), **631**
- bar Paket, 631
- barenv Umgebung (bar), **631**
- \baro (stmaryrd), 546
- \barwedge (amssymb), 546
- \BaseDirectory (docstrip), **864**, **865**, 955
- baseline Schlüssel (fancyvrb), **172**, **173**
- \baselineskip elastische Länge, 114, 115, **205**, **206**, 242, **426**, **890**, 891, **899**, **982**, 983, 984
 - Zeilenvorschub anpassen, 383
 - (ccfonts), 396
 - (geometry), 215
 - (typearea), 212
 - (yfonts), 406
- \baselinestretch, 115
 - (setspace), **114**, **115**
- baselinestretch Schlüssel (fancyvrb), **167**
- basicstyle Schlüssel (listings), 179
- Basisbefehle, $\TeX$
 - Ausführung überwachen, 991
 - darstellen, 979
 - Fehlersuche, 979, 991
- Basisschrift, *siehe* Grundschrift
- Baskerville Font in Formeln und Text, **536**
- basque Option (babel), 559
- \batchinput (docstrip), 861
- \batchmode, 990
- Baumstrukturen, **630**
- bb Schlüssel (graphicx), **636**, 637, 638, **639**
- \Bbbk (amssymb), 544
- bbding Paket, 414
- .bb1 Dateierweiterung, **9**, 712, 715, 771, 772, 823
 - (BibTeXMng), 819
 - (BibTeX), 772, 800, 823, 837, 839, 840
 - (bibentry), 736, 737
 - (chapterbib), 775
 - (jurabib), 751
- \bb1@activate (babel), **605**, **606**
- \bb1@activate<zeichen> (babel), 606
- \bb1@allowhyphens (babel), 607
- \bb1@deactivate (babel), **605**, **606**
- \bb1@declare@attribute (babel), **602**
- bb1x Schlüssel (graphicx), 636
- bb1y Schlüssel (graphicx), 636
- bblopts.cfg Datei (babel), 598
- bbs $\LaTeX$ -Stil, 821
- \bbslash (stmaryrd), 546
- bburx Schlüssel (graphicx), 636
- bbury Schlüssel (graphicx), 636
- Bcenter Umgebung (fancybox), **618**
- \bcline (lb2), **273**
- BCOR<val> Option (typearea), 213
- Bdescription Umgebung (fancybox), 618
- \because (amssymb), 551
- bedingte Formatierung, 905, **906–910**
- bedingter Code, Syntax, 851–857
- Befehle, *siehe auch* Präambel
 - Ausführung überwachen, 992, 993
 - Bibliographiestile, 836, 838, 839
 - Definitionen anzeigen, 977–979
 - Dokumentations-, Liste der, 853–857
 - erstellen
 - benennen, 874, 875

Befehle (*Forts.*)

- für Indexerstellung definieren, 673, 674
 - neue definieren, 875, [876](#), 877–880
 - Portabilität, 874
 - umdefinieren, 876, [877](#), 879
 - verschachteln, 878, [879](#)
- Fehlersuche, 978, 992, 993
- Klassen, 879, 911, 917–921
- Leerraum nach, 86, [87](#)
- ltxdoc-Klasse, 867
- Pakete, 879, 911, 917–919
- robuste, 674, 879
- zerbrechliche, 674, 906, 927, 928, 946, 953, 955
- $\mathcal{AMS-}\mathrm{TeX}$, 480
- Befehlszeilenprogramme, *siehe*
 - Kommandozeilenprogramme
- `\begin`, Fehler bei, 930, 935
- `\begingroup`, 517, 931, 934, 959, 963
 - Fehler bei, 935, 944
- Begrenzungssymbole
 - äußere, bei Matrizen, [502](#)
 - für wortwörtlichen Text (doc), 847, 848
 - mathematische, 502, 510, [511](#), 512, [517](#)
- Beispiele in diesem Buch, 15, 16, 171, 1008, 1135, 1136, *siehe auch die einzelnen Beispiele*
- `below` Option (placeins), 61, [297](#)
- `\belowbottomsep` feste Länge (booktabs), 278
- `\belowcaptionskip` elastische Länge, [316](#), [317](#), 321
- `\belowdisplayshortskip` elastische Länge, [492](#)
- `\belowdisplayskip` elastische Länge, [491](#), [492](#)
- `\belowrulessep` feste Länge (booktabs), 278
- `belowskip` Schlüssel/Option (caption), 321
- benannte Boxen, 901, [902](#), [903](#), *siehe auch* Boxen
- benannte Kolumnentitel-Layouts, 238
- Benennen von Fonts, 431
- bengali Paket, 608
- Benumerate Umgebung (fancybox), 618
- Benutzergruppen, *siehe* Hilfsquellen
- `Beqnarray` Umgebung (fancybox), [618](#)
- `Beqnarray*` Umgebung (fancybox), 618
- Berechnungen, [904](#), 905
- Berge, Symbole, 414, [415](#), [416](#)
- Beschnittlinien/-marken, 220, [221–223](#)
- beschreibende Listen
 - benutzerdefinierte, 155, [156–159](#)
 - Erweiterungen, 143, [144](#)
 - Standard, [139](#)
 - Voreinstellungen, 145, 146
- Beschriftungen
 - mathematische Symbole, 507, [508](#)
 - Pfeile, [502](#)
 - von Diagrammmachen, 624, 625
- `\beta`, 543
 - (fourier), [403](#), [404](#)
- `\beth` (amssymb), 544
- beton Paket, 395, 408
- `\between` (amssymb), 551
- bewegte Argumente, 81, 100, 455, 927, 928
- Beweisführungen, *siehe* theoremähnliche Strukturen
- Bézier-Kurven, *siehe* epic-Paket; eepic-Paket
- `\bf`, 338, [358](#), 359
 - in Formeln, 361, 476
 - (custom-bib), 833
- `bf` Option (titlesec), [39](#)
- `bf` Schlüsselwert
 - (caption), [310](#), [315](#), [319](#), [320](#), [322](#), [334](#)
 - (subfig), [326](#)
- `\bfdefault`, [357](#), [358](#), [451](#)
- `bfitemize` Umgebung (lb2), [356](#)
- `Bflushleft` Umgebung (fancybox), [618](#)
- `Bflushright` Umgebung (fancybox), [618](#)
- `\bfseries`, 351, [354](#), 355, [356](#), 357–359, [881](#)
 - in Formeln, 360, 361
 - (ulem), 93
 - ersetzt durch `\uwave`, 94
- `bfseries` Umgebung, [349](#)
- `\bga` (lb2), [480](#)
- `\bggroup`, 963
- `\bhline` (lb2), [273](#)
- `.bib` Dateierweiterung, 9, 712, 713
 - (BibTeX), 9, 790, 792, 794, 795, 798, 799, 802, 805, 837, 840
 - `\cite` in, [802](#)
 - (aux2bib), 804
 - (bibextract), 807
 - (bibtool), 811
 - Kurzformen entfernen, [811](#)
 - (bibulus), 788
 - (citefind), 807
 - (makebib), 805
 - (multibib), 782
 - `bib.html` Datei (bib2html), 806
 - `bib2html` Programm, [806](#), 813
 - `\bibAnnotatePath` (jurabib), [767](#)
 - `\bibansep` (jurabib), [763](#), [764](#)
 - `\bibapifont` (jurabib), [763](#)
 - `\bibatsep` (jurabib), [763](#), [764](#)
 - `\bibauthormultiple` (jurabib), [765](#), [766](#)
 - `\bibbdsep` (jurabib), [763](#), [764](#)
 - `\bibbfsasep` (jurabib), [764](#)
 - `\bibbfsesep` (jurabib), [764](#)
 - `\bibbstasep` (jurabib), [764](#)
 - `\bibbstesep` (jurabib), [764](#)
 - `\bibbtasep` (jurabib), [764](#)
 - `\bibbtesep` (jurabib), [764](#)
 - `\bibbtfont` (jurabib), [763](#)
 - `bibclean` Programm, [806](#), 807, 818, 1013
 - `\bibcolumnsep` (jurabib), [765](#)
 - `\bibdata`, [713](#)
 - (chapterbib), 773
 - `\bibeansp` (jurabib), [763](#)
 - `\bibefnfont` (jurabib), [763](#)
 - `\bibelntfont` (jurabib), [763](#)
 - `\bibentry` (bibentry), [736](#), 737
 - `bibentry` Schlüsselwert (jurabib), [764](#)
 - `bibentry` Paket, 736, 737
 - `bibextract` Programm, 807
 - `\bibfnfont` (jurabib), [763](#)
 - `\bibfont` (natbib), [732](#), [740](#)
 - `bibformat` Schlüssel/Option (jurabib), 761, [764](#), [765](#), [766](#), [827](#)

`\bibhang` feste Länge (natbib), [732](#), 740
 Probleme mit, [740](#)
`\bibidemPfname` (jurabib), 760
`\bibidemPfname` (jurabib), 760
`\bibidemPmname` (jurabib), 760
`\bibidemPmname` (jurabib), 760
`\bibidemPnname` (jurabib), 760
`\bibidemPnname` (jurabib), 760
`\bibidemPnname` (jurabib), 760
`\bibidemSfname` (jurabib), 760
`\bibidemSfname` (jurabib), 760
`\bibidemSmname` (jurabib), [760](#), 765
`\bibidemSmname` (jurabib), 760
`\bibidemSnname` (jurabib), 760
`\bibidemSnname` (jurabib), 760
`\bibindent` feste Länge, [717](#)
`\bibitem`, [710](#), [711](#), 715, 717, [723](#), 724, 771, 959
 Fehler bei, 928
 Warnung bei, 972
 (BibTeX), 794, 837
 (bibentry), 736
 (chicago), [723](#)
 (harvard), 725
 (jurabib), [723](#), 741, 768
 (natbib), 725, 726, 734, 739
 (showkeys), 72
`\bibjtfont` (jurabib), 763
`\bibjtsep` (jurabib), [763](#), [764](#)
 BibKeeper Programm, 819
 bibkey Programm, 805
 biblabel Option (cite), 722
`\bibleftcolumn` (jurabib), [765](#)
`\bibleftcolumnadjust` (jurabib), [765](#)
 biblikecite Schlüssel/Option (jurabib), 763
 Bibliographiedatenbanken anlegen (BibTeX), 711–715
 Bibliographien, *siehe auch* BibTeX; Quellenverweise;
 Datenbankformat;
 Datenbankmanagement-Werkzeuge,
 anpassen
 Autor-Jahr-Zitierschema, [732](#)
 Kurztitel-Zitierschema, 762, [763](#), 764, [765](#),
[766](#), 767
 Autoren
 fehlende Informationen, [733](#), [734](#)
 Geschlecht, 760, [761](#), 768
 Informationsfeld, 768
 kein Umbruch zwischen, 731
 Namen formatieren, 828
 nur beim ersten Quellenverweis auflisten,
[729](#), [730](#)
 Trennzeichen und -wörter bei Auflistungen,
 761, [762](#), 764
 Beschreibung, 785, 786
 BibTeX-Programmiersprache
 Befehle, 836, 838, 839
 Eintragsvariablen, 836
 Feldvariablen, 836
 globale Variablen, 836
 Kleinschreibung deaktivieren, 840, 841
 Kommentarzeichen, 790
 Leerzeilen, 836
 neue Felder hinzufügen, 841, 842

Bibliographien (*Forts.*)
 Sortierreihenfolge, 837
 Stildateien, 835–844
 systemimmanente Funktionen, 836, 838,
 839
 Unterstützung von mehreren Sprachen
 hinzufügen, 842–844
 Variablenarten, 836
 Verarbeitungsablauf, 837–840
 BibTeX-Varianten, 786–789
 Datenbankformat, 789–802
 Datenbankmanagement-Werkzeuge, 803–820
 Dissertationsjahr, 768
 DOI, 735
 Editionsdaten, 768
 EID, 735
 Eingabedatei für Quellenverweise anlegen,
 711–715
 Einzug, 764, [765](#)
 elektronische Veröffentlichungen, 735
 Endnotenverweise, 752, [753](#)
 Farben, [719](#)
 fehlende Jahresdaten, 733, 734
 Fonts, 762, 763
 Fußnotenverweise, [752](#), [753](#)
 Geschlechtsangabe, 760, [761](#), 768
 Gründerdaten (jurist.), 768
 Herausgeberdaten, 768
 in Inhaltsverzeichnissen, 51
 Internet-Ressourcen, 802, 803
 Interpunktion
 Kurztitel-Zitierschema, 763, [764](#)
 numerisches Zitierschema, 719, [721](#), [722](#)
 ISBN, 735
 ISSN, 735
 Klammern
 Kurztitel-Zitierschema, 761
 numerisches Zitierschema, [720](#)
 kommentierte, 746, [747](#), [766](#), 768
 Konfigurationsdateien, externe, 767
 Kurztitel-Zitierschema, 762, [763](#), 764, [765](#), [766](#),
 767
 kyrillisches Alphabet, 589
 mehrfache
 Beschreibung, 771–773
 bibtopic-Paket, 779, [780](#), [781](#)
 bibunits-Paket, [776](#)–[779](#)
 chapterbib-Paket, 773, [774](#), [775](#)
 Funktionsvergleich der Pakete, 772, 773
 je eingebundener Datei, 773, [774](#), [775](#)
 multibib-Paket, [782](#), [783](#)
 nach beliebigen Einheiten, [776](#)–[779](#)
 nach Kapiteln, 773, [774](#), [775](#)
 nach Themen, getrennte Datenbankdateien,
[779](#), [780](#), [781](#)
 nach Themen, getrennte
 Quellenverweiskehfehle, [782](#), [783](#)
 Zitierschemata, 771–783
 numerische Verweise zusammenfassen, [739](#)
 Online-Ressourcen, 802, 803

Bibliographien (*Forts.*)

Programme

- 8-Bit-Version, 787, 788
- Bi_WTeX++, 788, 789
- Bi_WTeX8, 787, 788
- bibulus, 788
- Java-Version, 788, 789
- mehrsprachige Versionen, 789
- Mi_WTeX, 789
- perl-Version, 788
- XML-Unterstützung, 788

Quellenverweise

- automatisch indexieren, 734, 746
- Autor-Jahr-Zitierschema, 732
- Fußnoten, 752, 753
- in Überschriften, 722
- in Legenden (Abbildung, Tabelle), 722

Quellenverweise, Sortierreihenfolge

- Autor-Nummer-Zitierschema, 739
- Kurztitel-Zitierschema, 769
- numerisches Zitierschema, 718, 720, 739

Querverweise zwischen Einträgen, 757, 758

Rückverweise, 768

Sammlungen, 768

Schlüssel mit Datenbankeinträgen verknüpfen, 713

Seitenanzahl-Feld, 769

Seitengrenzen ignorieren (Ibidem), 755

Sortierreihenfolge

- Autor-Nummer-Zitierschema, 739
- Kurztitel-Zitierschema, 769
- numerisches Zitierschema, 718, 720, 739

Spaltenlayout, 765

Stildateien, 8, 9

- bearbeiten, 835–844
- Beschreibung, 820
- Bi_WTeX-Programmiersprache, 835–844
- erstellen, 828–835
- Formatierung angeben, 834, 835
- Initialisierung, 829, 830
- Liste von, 820–823
- neue Felder hinzufügen, 841, 842
- unterstützte Erweiterungen festlegen, 832–834
- Unterstützung für Fremdsprachen hinzufügen, 842–844
- verändern, 835–844
- Zitierschema auswählen, 831, 832

Titel

- Formate, 744–746
- Kurztitel zuordnen, 747, 748
- Zusatzinformations-Feld, 768, 769

Titel eines Bandes (volume), 769

übersetzte Werke, 768, 769

Unterstützung

- für Gesetzestexte, 769, 770, 771
- von Fremdsprachen, 759, 760, 761

URLs, 735, 768, 769

Verarbeitungsablauf, 711–715

Zeilenumbrüche, 718

zuletzt geändert-Feld, 769

\bibliography, 709, 712, 713, 717, 772, 799, 808

Verwendung in Buchbeispielen, 715

- (bibentry), 737
- (biblist), 804
- (bibunits), 776, 777, 778
- (chapterbib), 773, 774, 775
- (jurabib), 748, 751
- (multibib), 782, 783
- (natbib), 734

\bibliography* (bibunits), 778, 779

\bibliography{typ} (multibib), 782

\bibliographylatex (lb2), 782, 783

\bibliographystyle, 712, 772, 808, 823

- (biblist), 804
- (bibtopic), 780, 781
- (bibunits), 777, 778
- (chapterbib), 773, 774
- (jurabib), 742–766
- (multibib), 782, 783
- (natbib), 730, 740

\bibliographystyle* (bibunits), 778, 779

\bibliographystyle{typ} (multibib), 782

\bibliographystylelatex (lb2), 782, 783

\bibliographysunit (bibunits), 778, 779

biblist Paket, 803, 804

\biblnfont (jurabib), 763

\bibname, 37, 775

- (babel), 561, 601
- (chapterbib), 775

\bibnotcited (jurabib), 748

\bibnumfmt (natbib), 740

\bibpreamble (natbib), 732, 740

\bibpunct (natbib), 731, 740

\bibbrightcolumn (jurabib), 765

\bibbrightcolumnadjust (jurabib), 765

\bibs{sprache} (jurabib), 759

\bibsall (jurabib), 759

\bibsection (natbib), 732, 740

\bibsep elastische Länge (natbib), 732, 740

\bibsgerman (jurabib), 759, 760, 761

\bibstyle, 713

- (chapterbib), 773

\bibstyle@{stil} (natbib), 731

Bi_WTeX Programm, 789–802, 820–844

kyrillisches Alphabet, 589

mehrsprachige Dokumente, 589

Bi_WTeX++ Programm, 788, 789Bi_WTeX-Programmiersprache

- Befehle, 836, 838, 839
- Eintragsvariablen, 836
- Feldvariablen, 836
- globale Variablen, 836
- Kleinschreibung deaktivieren, 840, 841
- Leerräume, 836
- neue Felder hinzufügen, 841, 842
- Sortierreihenfolge, 837
- systemimmanente Funktionen, 836, 838, 839
- Unterstützung für Fremdsprachen hinzufügen, 842–844
- Variablenarten, 836
- Verarbeitungsablauf, 837–840

- bibtex8 Programm, 787, 788
- BibTeXMng Programm, 819, 820
- \bibtfont (jurabib), 763
- bibttool Programm, xxix, 808–813, 817, 818, 1027
- bibttopic Paket, 772, 779–781, 782
 - Kompatibilitätsmatrix, 772
- \bibttotalpagesname (jurabib), 769
- bibulus Programm, 788
- bibulus.dtd Datei (bibulus), 788
- bibunit Umgebung (bibunits), 776, 777–779
- bibunits Paket, xxix, 772, 776–779
 - Kompatibilitätsmatrix, 772
 - nicht kompatibel zu bibttopic, 781
- \Bicycle (marvosym), 413
- \Big, 501, 517
 - Fehler bei, 942
- \big, 517
 - Fehler bei, 942
- big Option (titlesec), 39
- big-g (große) Begrenzungssymbole, 517
- \bigbox (stmaryrd), 552
- \bigcap, 552
- \bigcirc, 547
- \bigcup, 487, 552
- \bigcurlyvee (stmaryrd), 552
- \bigcurlywedge (stmaryrd), 552
- bigfoot Paket, 124, 129
- \Bigg, 517
 - Fehler bei, 942
- \bigg, 517
 - Fehler bei, 942
- \Biggl, 495, 517, 526
 - Fehler bei, 942
- \biggl, 484–486, 517, 524–526
 - Fehler bei, 942
- \Biggm, 517
 - Fehler bei, 942
- \biggm, 517
 - Fehler bei, 942
- \Biggr, 495, 517, 526
 - Fehler bei, 942
- \biggr, 484–486, 517, 524–526
 - Fehler bei, 942
- \biginterleave (stmaryrd), 552
- \Bigl, 517, 526, 542
 - Fehler bei, 942
- \bigl, 517, 526
 - Fehler bei, 942
- \Bigm, 517
 - Fehler bei, 942
- \bigm, 517
 - Fehler bei, 942
- \bignplus (stmaryrd), 552
- \bigodot, 552
- \bigoplus, 504, 552
- \bigotimes, 504, 552
- \bigparallel (stmaryrd), 552
- \Bigr, 517, 526, 542
 - Fehler bei, 942
- \bigr, 517, 526
 - Fehler bei, 942
- \bigskip, 891
- \bigskipamount elastische Länge, 269, 891
- \bigsqcap (stmaryrd), 552
- \bigsqcup, 552
- \bigstar (amssymb), 544
- \bigstrutjot feste Länge (multirow), 281
- \bigtriangledown, 546, 552
 - (stmaryrd), 552
- \bigtriangleup, 546, 552
 - (stmaryrd), 552
- \biguplus, 552
- \bigvee, 552
- \bigwedge, 552
 - Bilddateien einbinden, 634–641
 - Bildgröße, 638, 644
 - Dateiarten, 643
 - Dateierweiterungen
 - angeben, 643
 - Suchreihenfolge, 642, 643
 - Dateinamen parsen, 638
 - drehen, 638
 - Kapselung, 645, 646
 - skalieren, 638
 - Speicherort, 642
 - vollständige Dateinamen erzwingen, 643
 - Bilder in Absätzen, 115, 116
 - Bildgröße, 638, 644
 - Bildschirm, Seitenlayout für, 215
 - Bildschirmüberwachung, 989, 990
 - Bildunterschriften, *siehe* Legenden
 - Bildzeichen, *siehe* Sonderzeichen, Textsymbole
 - binäre Operatorzeichen, 547
- \binampersand (stmaryrd), 554
- Binary, Symbolklasse, 540
- bind Option (lb2), 919, 920
- Bindestriche (–)
 - geschützte, 89, 90, 100
 - Trennung nach, 719
- bindingoffset Schlüssel/Option (geometry), 215, 217
- \bindnasrepma (stmaryrd), 554
 - Bindung und Bundsteg, 215
- \binom (amsmath), 401, 402, 506
- Bitemize Umgebung (fancybox), 618
- Bitstream Charter Font, 386
 - in Formeln und Text, 536
- Bjarne Option (fncychap), 37
- bk11.clo Datei, 18
- Blackboard Bold-Alphabet, 390, 523, 535
- \blacklozenge (amssymb), 544
- \blacksquare (amssymb), 544
- \blacktriangle (amssymb), 544
- \blacktriangledown (amssymb), 544
- \blacktriangleleft (amssymb), 549
- \blacktriangleright (amssymb), 549
- .blg Dateierweiterung (BibTeX), 9, 712
- block Schlüssel (titlesec), 40, 41, 42, 43, 44, 46, 47
- \bluefbox (lb2), 634
- \bm (bm), 364, 388, 389, 517, 524, 525–527

- bm Paket, 524–528
 - Fehler bei, 952
 - Probleme mit fourier, 404
 - Probleme mit mathptmx, 388
- bmargin Schlüssel/Option (geometry), 216
- Bmatrix Umgebung (amsmath), 499
- bmatrix Umgebung (amsmath), 499
- \bmdefine (bm), 524, 525, 527
- \bmod, 505
- \bneg (lb2), 545
- body Schlüssel/Option (geometry), 219
- \boldmath, 363, 525
 - (bm), 528
 - (fourier), 404
 - (mathpazo), 389
 - (mathptmx), 388
- boldsans Option (ccfonts), 396, 531
- \boldsymbol (amsmath), 524
- book B^BT_EX-Publikationstyp, 714, 742, 791, 801
 - (jurabib), 769
- book Dokumentenklasse, 7, 14, 24, 122, 127, 204, 224, 231, 479, 702
 - Alternativen zu, 244, 245
 - Fußnotennummerierung, 119
 - Gliederungsbefehle, 24, 25, 55
- booklet B^BT_EX-Publikationstyp, 791
- bookman Paket, 214, 382
- Bookman Font, 385
- booktabs Paket, 277–280
- booktitle B^BT_EX-Feld, 714, 763, 768, 791, 792, 801
- booktitleaddon B^BT_EX-Feld (jurabib), 768
- \boolean (ifthen), 207, 703, 716, 907, 908, 920
- \born (lb2), 378, 379
- \bot, 544
- \botfigrule, 294
- \bothIfFirst (caption), 323
- \bothIfSecond (caption), 323
- \botmark, 226, 230
- bottom Option (footmisc), 127
- bottom Schlüssel/Option (geometry), 216
- bottom Schlüsselwert
 - (caption), 321
 - (subfig), 326, 327
- \bottomcaption (supertabular), 265
- \bottomfraction, 293, 294, 296
- bottomline Schlüsselwert (fancyvrb), 166, 167
- bottomnumber Zähler, 292
- \bottomrule (booktabs), 278, 280
- \bottomtitlespace (titlesec), 43
- Bounding Box-Kommentare, 633, 634
- \bowtie, 551
- \Box (latexsym), 476
- \boxast (stmaryrd), 546
- \boxbar (stmaryrd), 546
- \boxbox (stmaryrd), 546
- \boxbslash (stmaryrd), 546
- \boxcircle (stmaryrd), 546
- \boxdot (amssymb), 546
- \boxed (amsmath), 503
- boxed Schlüssel
 - (float), 301, 302, 303, 318, 320
 - (rotfloat), 307
- boxedminipage Umgebung
 - (boxedminipage), 613, 902
 - (lb2), 903
- boxedminipage Paket, 613
- \boxempty (stmaryrd), 546
- Boxen, *siehe auch* Rahmen; Linien
 - abgerundete Ecken, 614, 615
 - Absatzboxen, 893, 896–899
 - als mathematische Symbole, 546
 - Arten von, 893
 - Beschreibung, 893
 - Breite, 650
 - dekorative, 614–618
 - Dimensionen, 650
 - doppelte Rahmenlinie, 615
 - Fehlersuche, 990
 - Probleme mit Farben, 903, 904
 - gerahmte, 613
 - Höhe, 650
 - Inhalte darstellen, 990
 - Linienboxen, 893, 899–901
 - LR-Boxen, 893, 894, 895
 - mit Namen erzeugen, 901, 902, 903
 - ovale, 614
 - Referenzpunkte, 650
 - schattierte, 613, 614, 615
 - Tiefe, 650
 - um Kapitälchen, 579
 - verändern, 901–903
 - wortwörtliche Ausgabe, 172, 173
- \boxlength (picins), 314
- \boxminus (amssymb), 546
- \boxplus (amssymb), 546
- \boxslash (stmaryrd), 546
- \boxtimes (amssymb), 546
- boxwidth Schlüssel (fancyvrb), 172, 173
- \bpi (lb2), 527
- Brüche, mathematische Symbole, 505, 506
- braces.rsc Datei (bibtool), 810
- \bracevert, 511, 544
- brazil Option (babel), 559
- brazilian Option (babel), 559
- breakall Option (truncate), 241
- breakautoindent Schlüssel (listings), 182
- breakindent Schlüssel (listings), 182
- breaklines Schlüssel (listings), 182
- breakwords Option (truncate), 241
- Breite, *siehe* Längenparameter
- Breite der Nummerierung, Inhaltsverzeichnisse, 54
- Breite einer Box, 650
- breqn Paket, 483, 1016
- breton Option (babel), 559
- \breve, 545
- british Option (babel), 559, 567
- Brotschrift, *siehe* Grundschrift
- \bs (lb2), 674
- \bsc (babel), 579
- \bslash (doc), 853

- .bst Dateierweiterung
 - (BibTeX), 9, 712, 713, 1016
 - (custom-bib), 828, 829, 833, 834
 - (natbib), 734
- btAuxfile Zähler (bibtopic), 781
- \btPrintAll (bibtopic), 780
- \btPrintCited (bibtopic), [780](#), [781](#)
- \btPrintNotCited (bibtopic), 780
- btSect Umgebung (bibtopic), [779](#), [780](#), [781](#)
- btUnit Umgebung (bibtopic), 781
- btXbst.doc Datei (BibTeX), [837](#), 840
- bu(*num*).aux Datei (bibunits), 776
- Buch, dieses, *siehe* L^AT_EX-Begleiter
- Buchstaben, *siehe* Sonderzeichen, Textsymbole
- Buchstaben, mathematische Symbole, 542–545
- buchstabenförmige mathematische Symbole, [544](#)
- buchstabengetreue Ausgabe, *siehe* wortwörtliche Ausgabe
- Buchstabengruppen, Index, 684, 699
- buffer size Fehler, 958
- bulgarian Option (babel), 559, [567](#), [574](#), 584
- \bullet, [547](#), [565](#)
- \Bumpeq (amssymb), 548
- \bumpeq (amssymb), 548
- bundle Umgebung (ecrtree), [630](#)
- Bundsteg (innerer Rand), 201, 202, 207, 212–216, 219
- \BUseVerbatim (fancyvrb), 176
- BVerbatim Umgebung (fancyvrb), [172](#), [173](#)
- \BVerbatim* (fancyvrb), 173
- \BVerbatimInput (fancyvrb), [172](#), 173
- \BVerbatimInput* (fancyvrb), 173
- bychapter Folio-Stil (chappg), [225](#)
- \bye (nfssfont.tex), [381](#)
- C**
- C Schlüsselwert (listings), [179](#), [180](#)
- C Syntax
 - (fancyhdr), [234](#), [235](#), [236](#)
 - (lb2), [256](#)
 - (tabulary), [261](#), [262](#)
- \c, 465, 470
- c Syntax, [251](#), [252](#), [253](#)
 - (array), 258
 - (tabulary), 262
- c5paper Option (typearea), 212
- calc Paket, 904, 905
 - Fehler bei, 923, 929
 - mit geometry kombiniert, 218
 - von jurabib geladen, 765
- calcwidth Option (titlesec), [44](#), [45](#)
- call.type\$ interne BibTeX-Funktion, [837](#), 839, 840
- \calQ (lb2), [514](#)
- cam Option (crop), [221](#)
- camel Paket, xxix, 704, [769–771](#), 1013
- camel.ist Datei (makeindex), [771](#)
- canadian Option (babel), 559
- canadien Option (babel), 559
- \Cancer (marvosym), [413](#)
- \Cap (amssymb), 546
- \cap, 546
- \capitalacute (textcomp), 374, 470
- \capitalbreve (textcomp), [95](#), 374
- \capitalcaron (textcomp), 374, 470
- \capitalcedilla (textcomp), 374
- \capitalcircumflex (textcomp), 374
- \capitaldieresis (textcomp), 374, 470
- \capitaldotaccent (textcomp), 374
- \capitalgrave (textcomp), [95](#), 374, [376](#), 470
- \capitalhungarumlaut (textcomp), 374
- \capitalmacron (textcomp), 374, 470
- \capitalnewtie (textcomp), 374
- \capitalogonek (textcomp), 374, 470
- \capitalring (textcomp), 374, 470
- \capitaltie (textcomp), 374
- \capitaltilde (textcomp), 374, 470
- \caps (soul), [95](#), [97](#), [98](#), [99](#)
- \capsdef (soul), [97](#), [98](#), [99](#)
 - capsdefault Option (soul), 98
- \capsreset (soul), [98](#), [99](#)
- \capssave (soul), [98](#), [99](#)
- \capselect (soul), [98](#), [99](#)
- captcont Paket, 324
- \caption, 49, 50, 56, 304, [315](#), [316](#), 321, 772
 - Fehler bei, 316, 927, 932
 - mit Randausgleich, 111
 - Querverweis auf, 70, [71](#)
 - (caption), [270](#), [318–322](#), [324](#), [331](#)
 - (float), [301](#), [302](#), [303](#)
 - (ftpage), [336](#)
 - (longtable), 267, [270](#)
 - (pics), 314
 - (rotating), [306](#)
 - (sidecap), [334](#), [335](#)
 - (subfig), 325, 330
 - (subfloat), [332](#)
 - (supertabular), [270](#)
 - (threeparttable), [286](#)
 - (wrapfig), [309](#), [310](#)
- caption Schlüssel (listings), 183
- caption Paket, xxviii, 304, 305, 306, 317–324, 325, 326, 333
 - kombiniert mit pics, 315
 - kombiniert mit sidecap, 333
- \caption*
 - (caption), 324
 - (longtable), 270
 - (subfig), 330
- caption2 Paket, [317](#), 325
- CaptionAfterwards Option (ftpage), 335
- CaptionBefore Option (ftpage), 335
- \captionof (caption), [304](#), [305](#)
- captionpos Schlüssel (listings), 183
- \captions(*sprache*) (babel), 567, 596, 603, 605
- \captionsetup
 - (caption), 320, 321, 322, [323](#), 324
 - (subfig), [326](#), [328](#), [329](#), [331](#)
- captionskip Schlüssel/Option (subfig), 327, [329](#), [331](#)
- \captionsngerman (babel), [56](#), [751](#)
 - in diesem Buch, [14](#)
- \captionsrussian (babel), [605](#)
- \carriagereturn (dingbat), [411](#)

- cases Umgebung (amsmath), [496](#), [498](#), [519](#)
 - Fehler bei, [941](#), [946](#)
 - catalan Option (babel), [559](#), [567](#), [569](#), [571](#)
 - catalan.ldf Datei (babel), [598](#)
 - \catcode, [101](#), [356](#), [565](#), [591](#), [606](#)
 - \cb (lb2), [623](#)
 - \cbcolor (changebar), [200](#)
 - \cbdelete (changebar), [199](#), [200](#)
 - cbe BibTeX-Stil, [821](#)
 - \cbend (changebar), [199](#), [200](#)
 - \cbinput (chapterbib), [773](#)
 - \cbstart (changebar), [199](#), [200](#)
 - cbunit Umgebung (chapterbib), [773](#)
 - ccfonts Paket, [395](#), [396](#), [410](#), [531](#)
 - \ccname (babel), [561](#)
 - CD Umgebung (amscd), [479](#), [500](#), [501](#)
 - \cd (lb2), [623](#), [624](#)
 - CD-ROM, CTAN, [997](#)
 - \cdashline (arydshln), [275](#)
 - \cdot, [283](#), [487](#), [490](#), [512](#), [547](#)
 - \cdots, [499](#), [508](#), [552](#), [877](#), [879](#), [978](#)
 - cell BibTeX-Stil (jmb), [821](#)
 - \cellcolor (colortbl), [273](#)
 - Center Umgebung (ragged2e), [112](#)
 - center Option
 - (crop), [221](#)
 - (titlesec), [39](#)
 - center Umgebung, [111](#), [154](#), [881](#)
 - (ragged2e), [112](#)
 - centerbody Option (sidecap), [333](#)
 - \centerdot (amssymb), [547](#)
 - centerfirst Schlüsselwert (caption), [320](#), [321](#)
 - Centering Schlüsselwert (caption), [320](#)
 - \Centering (ragged2e), [112](#), [113](#)
 - \centering, [111](#), [383](#), [895](#)
 - in Überschriften, [33](#)
 - (array), in Tabellen, [255](#), [257](#), [258](#)
 - (multirow), in Tabellen, [282](#)
 - (ragged2e), [112](#)
 - centering Schlüssel/Option (geometry), [217](#)
 - centering Schlüsselwert (caption), [320](#)
 - \CenteringLeftskip elastische Länge (ragged2e), [113](#)
 - \CenteringParfillskip elastische Länge (ragged2e), [113](#)
 - \CenteringParindent feste Länge (ragged2e), [113](#)
 - \CenteringRightskip elastische Länge (ragged2e), [113](#)
 - centerlast Schlüsselwert (caption), [310](#), [320](#)
 - \centerline, [316](#)
 - centertags Option (amsmath), [485](#)
 - \cf (lb2), [501](#)
 - .cfg Dateierweiterung, [9](#), [441](#), [443](#), [861](#)
 - (babel), [598](#), [605](#), [606](#)
 - (caption), [324](#)
 - (color), [945](#)
 - (docstrip), [863](#), [864](#), [865](#), [955](#)
 - (endfloat), [299](#)
 - (euro), [104](#)
 - (graphics), [632](#), [945](#)
 - (jurabib), [767](#)
 - (ltxdoc), [868](#)
 - .cfg Dateierweiterung (*Forts.*)
 - (natbib), [731](#), [735](#)
 - (paralist), [146](#)
 - (subfig), [331](#)
 - (textcomp), [379](#)
 - (typearea), [212](#)
 - cfgguide.tex Datei, [441](#), [443](#)
 - \cfoot (fancyhdr), [229](#), [232](#), [233](#), [239](#), [240](#), [616](#)
 - \cfrac (amsmath), [503](#)
 - \ch (babel), [580](#)
 - Chaikins Kurven, [628](#)
 - chams Paket, [537](#)
 - chancery Paket, [382](#)
 - change.case\$ interne BibTeX-Funktion, [839](#), [840](#), [841](#), [843](#)
 - changebar Umgebung (changebar), [199](#), [200](#)
 - changebar Paket, [198](#)–[200](#)
 - changebargrey Zähler (changebar), [200](#)
 - \changebarsep feste Länge (changebar), [200](#)
 - \changebarwidth feste Länge (changebar), [200](#)
 - \changes (doc), [849](#), [855](#)
 - \chapnumfont (quotchap), [38](#)
 - chappg Paket, [225](#), [226](#)
 - \chappgsep (chappg), [226](#)
 - \chapter, [25](#), [27](#), [34](#), [36](#), [227](#), [231](#), [237](#)
 - erzeugt unerwünschte Seitenzahl, [230](#), [231](#), [238](#)
 - Querverweis auf, [70](#)
 - zusätzliche Leerräume in .lof- und .lot-Dateien, [51](#)
 - (bibunits), [778](#)
 - (chappg), [225](#)
 - (chapterbib), [774](#)
 - (fncychap), [36](#), [37](#)
 - (minitoc), partielle Verzeichnisse, [60](#)
 - (quotchap), [38](#)
 - (titlesec), [42](#), [47](#)
 - (titletoc), partielle Inhaltsverzeichnisse, [68](#)
- chapter Schlüsselwert (jurabib), [749](#), [757](#)
- chapter Zähler, [27](#), [883](#)
 - in \parts nummeriert, [27](#)
- chapter BibTeX-Feld, [791](#), [792](#), [793](#)
- \chapter*, [25](#), [231](#), [703](#), [732](#), [774](#)
 - im Inhaltsverzeichnis, [50](#)
- chapterbib Paket, [725](#), [732](#), [772](#), [773](#)–[775](#), [800](#)
 - kombiniert mit babel, [775](#)
 - Kompatibilitätsmatrix, [772](#)
 - nicht kompatibel zu bibtopic, [781](#)
- \chapterheadendvskip (quotchap), [38](#)
- \chapterheadstartvskip (quotchap), [37](#), [38](#)
- \chaptermark, [227](#), [228](#), [231](#), [775](#)
 - (fancyhdr), [237](#)
- \chaptername, [37](#), [41](#), [228](#)
 - (babel), [561](#), [562](#)
- \chapterpagestyle (KOMA), [238](#)
- \chaptertitle (titlesec), [41](#)
- \CharacterTable (doc), [853](#)
- Charakterziffern, *siehe* Mediävalziffern
- charter Option (quotchap), [38](#)
- charter Paket, [382](#)
- Charter Font, [386](#), [536](#)
- \thead (fancyhdr), [232](#), [233](#), [239](#), [240](#)

- `\check`, 545
 - `check.rule` Funktion (bibtool), 810
- `\CheckCommand`, 880, 916
- `\CheckCommand*`, 880
- `\checkmark` (amssymb), 544
- `\CheckModules` (doc), 853
- `\CheckSum` (doc), 853
- chemische Strukturformeln, 631
- `\chi`, 543
 - chicago $\text{\LaTeX}$ -Stil
 - (chicago), 709, 724
 - (natbib), 730, 731, 734, 826
 - chicago Paket, 717, 723, 724
 - chicagoa $\text{\LaTeX}$ -Stil (chicago), 724
 - Chinesisch, 608
 - chmath Paket, 537
 - `chr.to.int` interne $\text{\LaTeX}$ -Funktion, 839
- `\chunk` (ecltree), 630
- `\circ`, 495, 547, 649
- `\circeq` (amssymb), 548
- `\circle`, 625, 626
 - Warnung bei, 969
 - (eepicmu), 629
 - (eepic), 627, 628
 - (epic), 626
 - (pspicture), 657, 658, 660
 - (texpicture), 659
- `\circle*`, 625, 626
 - (eepicmu), 629
 - (eepic), 627, 628
 - (epic), 626
 - (pspicture), 657, 658
 - (texpicture), 659
- `\circlearrowleft` (amssymb), 550
- `\circlearrowright` (amssymb), 550
- `\circledast` (amssymb), 547
- `\circledcirc` (amssymb), 547
- `\circleddash` (amssymb), 547
- `\circledR` (amssymb), 544
- `\circledS` (amssymb), 544
- Citation Umgebung (lb2), 880, 881, 882
- `\citation`, 712, 713, 772, 811
 - (notoccite), 722
- `\citationdata` (camel), 770
 - `citationreversed` Schlüsselwert (jurabib), 749, 758
- `\citationstyle` (camel), 770
- `\citationsubject` (camel), 770, 771
- `\cite`, 711, 713, 715, 716, 718, 723, 726, 771, 772, 790
 - Einschränkungen für Schlüssel, 874
 - in .bib-Dateien, 802
 - Warnung bei, 962
 - ($\text{\LaTeX}$), 839
 - (authdate1-4), 724
 - (biblist), 804
 - (bibtopic), 779, 780, 781
 - (bibunits), 776, 777, 778, 779
 - (chapterbib), 774, 775
 - (chicago), 724
 - (cite), 718-722
 - Probleme mit, 721
 - (harvard), 725
- `\cite` (Forts.)
 - (jurabib), 741, 742, 743-745, 746, 747-762
 - (multibib), 782, 783
 - (natbib), 709, 726, 728, 732, 738
 - (showkeys), 72
 - (textcase), 92
 - Probleme mit, 92
 - cite Paket, xxix, 718-722
 - Kompatibilitätsmatrix, 772
 - nicht kompatibel zu natbib, 725, 739
- `\cite*`
 - (bibunits), 777
 - (harvard), 725
 - (jurabib), 745
 - (natbib), 777
 - `cite` interne $\text{\LaTeX}$ -Funktion, 839, 841
- `\cite{typ}` (multibib), 782
- `\citeA` (chicago), 724
- `\Citealp` (natbib), 727
- `\citealp`
 - (jurabib), 758
 - (natbib), 727, 738
- `\citealp*` (natbib), 727
- `\Citealt` (natbib), 727
- `\citealt`
 - (jurabib), 758
 - (natbib), 727, 738
- `\citealt*` (natbib), 727, 738
- `\citeasnoun` (harvard), 725
- `\Citeauthor` (natbib), 727
- `\citeauthor`
 - (jurabib), 758
 - (natbib), 727-729, 738
- `\citeauthor*` (natbib), 727, 738
- `\citeauthoryear` (chicago), 723
- `\citedash` (cite), 719, 720
- `\citefield` (jurabib), 744, 760
- `citefind` Programm, 807
- `\citeform` (cite), 719, 720, 721
- `\citefull` Schlüssel/Option (jurabib), 749, 750, 751, 752, 753, 755, 757, 758
- `\citefullfirstfortype` (jurabib), 749, 750
- `citehack` Paket, 590
- `\citeindexfalse` (natbib), 734
- `\citeindextrue` (natbib), 734
- `\citeindextype` (natbib), 735
- `\citelatem` (lb2), 782, 783
- `\citeleft` (cite), 718, 719, 720, 721, 722
- `\citemid` (cite), 718, 719, 720
- `\CiteMoveChars` (cite), 721
- `\citeN` (chicago), 724
- `\citen` (cite), 719, 720
- `\citenam` (harvard), 725
- `\citenotitlefortype` (jurabib), 745, 746
- `\citeNP` (chicago), 724
- `\citenum` (cite), 719
- `\citenumfont` (natbib), 740
- `\citeonline` (cite), 719
- `\Citep` (natbib), 727

- `\citep`
 - (jurabib), [758](#)
 - (natbib), [723](#), [726](#), [728–731](#), [733](#), [734](#), [737](#), [738](#), [739](#)
 - Probleme mit, [729](#), [733](#), [734](#), [739](#)
- `\citep* (natbib)`, [726](#), [729](#), [730](#), [738](#)
- `\citepalias (natbib)`, [728](#)
- `\citepunct (cite)`, [719](#), [720](#)
- `\citeright (cite)`, [718](#), [719](#), [720](#), [721](#)
- CiteSeer, [803](#)
- `\citestyle (natbib)`, [730](#), [731](#), [740](#)
- `\citeswithoutentry (jurabib)`, [751](#)
- `\Citet (natbib)`, [727](#)
- `\citet`
 - (jurabib), [758](#)
 - (natbib), [723](#), [726](#), [728–731](#), [733](#), [734](#), [736](#), [737](#), [738](#)
 - Probleme mit, [729](#), [733](#), [734](#), [739](#)
- `\Citet* (natbib)`, [727](#)
- `\citet* (natbib)`, [726](#), [727](#), [738](#)
- citetag Programm, [807](#)
- `\citetalias (natbib)`, [728](#)
- `\citetext (natbib)`, [727](#), [738](#)
- `\citetitle (jurabib)`, [745](#), [752](#), [761](#)
- `\citetitlefortype (jurabib)`, [745](#)
- `\citetitleonly (jurabib)`, [745](#)
- `\citeyear`
 - (chicago), [724](#)
 - (jurabib), [758](#)
 - (natbib), [727](#), [738](#)
- `\citeyearNP (chicago)`, [724](#)
- `\citeyearpar`
 - (jurabib), [758](#)
 - (natbib), [727](#), [728](#), [738](#)
- cjk Paket, [608](#)
- `\ClassError`, [919](#)
- classes.dtx Datei, [354](#)
- classes.ins Datei, [861](#)
- `\ClassInfo`, [919](#)
- `\ClassWarning`, [919](#)
- `\ClassWarningNoLine`, [919](#)
- `\cleardoublepage`, [243](#)
 - (endfloat), [299](#)
- `\clearpage`, [21](#), [242](#), [243](#), [271](#), [292](#), [297](#), [298](#), [304](#), [702](#), [703](#)
 - (endfloat), [299](#)
 - (lscap), [220](#)
- `\cleartoevenpage (nextpage)`, [244](#)
- `\cleartooddpage (nextpage)`, [244](#)
- `\cline`, [251](#), [280](#), [281](#), [282](#), [284](#), [290](#)
 - (booktabs), [278](#), [279](#)
 - (tabs), [277](#)
- clip Schlüssel (graphicx), [636](#), [637](#), [638](#), [639](#)
- .clo Dateierweiterung, [7](#), [9](#), [18](#)
- clock Option (ifsym), [415](#)
- `\closecurve (curves)`, [630](#)
- closeFloats Option (ftpage), [335](#)
- Closing, Symbolklasse, [540](#)
- .cls Dateierweiterung, [7](#), [9](#), [18](#)
- `\clubpenalty`, [982](#), [984](#)
- `\clubsuit`, [544](#)
- CM Bright Fonts, [396](#), [397](#)
 - in Formeln und Text, [538](#)
- CM-Super Fonts, [366](#), [367](#), [586](#)
- cm-super-t1.enc Datei, [366](#)
- cmbright Paket, [396](#), [397](#), [539](#)
- `\cmd (ltxdoc)`, [867](#)
- `\cmidrule (booktabs)`, [278](#), [279](#), [280](#)
- `\cmidrulekern feste Länge (booktabs)`, [279](#)
- `\cmidrulesep feste Länge (booktabs)`, [279](#)
- `\cmidrulewidth feste Länge (booktabs)`, [279](#)
- Code, *siehe* Computercode
- `\CodelineFont (doc)`, [429](#)
- `\CodelineIndex (doc)`, [850](#), [853](#), [869](#)
- CodelineNo Zähler (doc), [429](#)
- `\CodelineNumbered (doc)`, [853](#)
- codes Schlüssel (fancyvrb), [170](#)
- `\Coffeecup (marvosym)`, [413](#)
- collectmore Zähler (multicol), [195](#), [197](#), [198](#)
- `\colon`, [552](#), [553](#)
 - (amsmath), [514](#), [552](#)
- colon Option (natbib), [731](#)
- colon Schlüsselwert (caption), [320](#)
- colonsep Schlüsselwert (jurabib), [741](#), [745](#), [767](#)
- `\color (color)`, [106](#), [200](#), [272](#), [273](#)
 - Fehler bei, [952](#)
 - Probleme mit, [903](#)
- color Option
 - (changebar), [200](#)
 - (showkeys), [72](#)
- color Paket, [222](#), [1018](#)
 - Fehler bei, [923](#), [945](#), [952](#)
 - Kompatibilität zu anderen Paketen, [903](#)
- `\colorbox (color)`, [166](#)
- colorlinks Option (hyperref), [82](#)
- colortbl Paket, [273](#)
- columnbadness Zähler (multicol), [195–197](#)
- `\columnbreak (multicol)`, [198](#)
- `\columncolor (colortbl)`, [273](#)
- `\columnsep feste Länge`, [203](#), [205](#), [702](#), [703](#), [904](#)
 - (multicol), [194](#), [195](#), [196](#)
 - (wrapfig), [309](#)
- columnsep Schlüssel/Option (geometry), [215](#)
- `\columnseprule feste Länge`, [203](#), [205](#), [702](#), [703](#)
 - (multicol), [194](#), [195](#)
- `\columnwidth feste Länge`, [119](#), [120](#), [203](#), [642](#)
 - (multicol), [195](#)
- `\Com (lb2)`, [674](#)
- `\combinemarks (lb2)`, [240](#)
- comma Option (natbib), [731](#), [737](#)
- comma Schlüsselwert (jurabib), [743](#)
- commabeforefirst Schlüssel/Option (jurabib), [741](#), [767](#)
- command Schlüssel (graphicx), [638](#)
- commandchars Schlüssel (fancyvrb), [160](#), [169](#), [170](#), [176](#)
- commasep Schlüsselwert (jurabib), [745](#)
- comment Umgebung (verbatim), [161](#)
- commentchar Schlüssel (fancyvrb), [170](#)
- commented Schlüsselwert (jurabib), [761](#)
- commented B^BT_EX-Publikationstyp (jurabib), [761](#), [768](#), [769](#)

- commentstyle Schlüssel (listings), [179](#), [180](#), [184](#)
- compact Option (titlesec), [39](#), [40](#)
- compactdesc Umgebung (paralist), [143](#), [144](#), [146](#)
- compactenum Umgebung (paralist), [140](#), [141–143](#), [145](#)
- compactitem Umgebung (paralist), [143](#), [144](#)
- compare Schlüsselwert (jurabib), [747](#)
- \complement (amssymb), [544](#)
- Comprehensive T_EX Archive Network, *siehe* CTAN
- compress Option (cite), [720](#)
- compress Schlüsselwert (jurabib), [764](#), [765](#), [766](#)
- Computer Modern (CM) Fonts, [528](#)
 - kyrillisches Alphabet, [586](#)
 - L^AT_EX-Standardfonts, [365](#), [366](#), [367](#), [368](#)
 - Mediävalziffern, [394](#), [395](#)
- Computerbildschirm, Seitenlayout für, [215](#)
- Computercode setzen, [177](#), [178](#), [179](#), [184](#), *siehe auch*
 - wortwörtliche Ausgabe
 - als Gleitobjekt, [183](#)
 - Anführungsstriche in, [161](#), [162](#), [163](#)
 - Eingabekodierung, [184](#)
 - Einzug, [181](#)
 - Fragmente in normalem Text, [180](#)
 - Kommentare entfernen, *siehe* Kommentare aus dem Code entfernen
 - Legenden, [183](#)
 - Linien/Rahmen um Listings, [182](#), [183](#)
 - nummerierte Zeilen, [180](#), [181](#)
 - Programm-Schlüsselwörter formatieren, [179](#), [180](#)
 - Rahmen um Listings, [182](#)
 - unterstützte Programmiersprachen, [178](#)
 - Zeilenumbrüche, [182](#)
- \ComputerMouse (marvosym), [413](#)
- Concrete Font, [395](#), [396](#), [530](#)
 - in Formeln und Text, [530](#)
- Concurrent Versions System (CVS), [869](#)
- config Schlüssel/Option
 - (caption), [324](#)
 - (jurabib), [767](#)
 - (subfig), [331](#)
- config.ps Datei (dvips), [655](#)
- \cong, [548](#)
- Conny Option (fncychap), [37](#)
- contents BibT_EX-Feld (BibTexMng), [820](#)
- \contentsfinish (titletoc), [62](#), [64–67](#)
- \contentslabel (titletoc), [63](#), [64](#), [67](#)
- \contentsline, [52](#), [53](#), [54](#), [55](#)
 - (titletoc), [62](#), [65](#), [66](#)
- \contentsmargin (titletoc), [64](#), [65](#), [66](#), [67](#), [69](#)
- \contentsname, [37](#)
 - (babel), [561](#)
- \contentspage (titletoc), [63](#), [64](#), [65](#)
- \contentspush (titletoc), [64](#), [65](#)
- \contentsuse (titletoc), [62](#)
- \ContinuedFloat (caption), [324](#), [331](#)
- convert Programm, [661](#)
- \coprod, [504](#), [552](#)
 - (mathptmx), nicht verfügbar, [389](#)
- \copyright, [544](#)
 - (textcomp), [470](#)
- copyright BibT_EX-Feld (BibTexMng), [820](#)
- Copyright-Vermerk, Sprachdefinitionsdateien, [598](#)
- Cork (T1)-Kodierung, [348](#), *siehe auch*
 - T1-Fontkodierung
- \cornersize (fancybox), [614](#), [615](#)
- \cornersize* (fancybox), [614](#)
- \cos, [513](#), [519](#)
- \cosec (babel), [580](#)
- \cosh, [513](#)
- \cot, [513](#)
- \coth, [513](#)
- \count, [945](#)
- countmax Option (subfloat), [332](#)
- courier Schlüsselwert (fancyvrb), [163](#), [176](#), [177](#)
- courier Paket, [382](#)
- Courier Font, [386](#)
- \cov (lb2), [501](#)
- cp1250 Option (inputenc), [371](#)
- cp1251 Option (inputenc), [587](#)
- cp1252 Option (inputenc), [369](#), [371](#)
- cp1255 Option (inputenc), [595](#)
- cp1257 Option (inputenc), [371](#)
- cp437 Option (inputenc), [371](#)
- cp437de Option (inputenc), [371](#)
- cp850 Option (inputenc), [371](#)
- cp852 Option (inputenc), [371](#)
- cp855 Option (inputenc), [587](#)
- cp858 Option (inputenc), [371](#)
- cp862 Option (inputenc), [595](#)
- cp865 Option (inputenc), [371](#)
- cp866 Option (inputenc), [587](#)
- cp866av Option (inputenc), [587](#)
- cp866mav Option (inputenc), [587](#)
- cp866nav Option (inputenc), [587](#)
- cp866tat Option (inputenc), [587](#)
- \cr, [928](#), [934](#), [941](#), [942](#)
- \crrcr, [941](#)
- croatian Option (babel), [559](#)
- crop Paket, [220–223](#)
 - in diesem Buch, [1137](#)
- cropmarks Option (lb2), [919](#), [920](#)
- cross Option (crop), [221](#), [222](#)
- crossref Schlüssel/Option (jurabib), [757](#), [758](#)
- crossref BibT_EX-Feld, [714](#), [757](#), [792](#), [801](#), [809](#), [838](#)
 - (biblist), [804](#)
- \cs (ltxdoc), [867](#)
- \csc, [513](#)
 - (lb2), [514](#)
- .csf Dateierweiterung (bibtex8), [787](#)
- \csname, [29](#), [978](#), [979](#)
- \Csub (lb2), [33](#)
- CTAN (Comprehensive T_EX Archive Network)
 - CD/DVD, [997](#)
 - FTP-Befehle, [1000–1002](#)
 - FTP-Server, Liste der, [996](#)
 - Inhalt, [996](#)
 - Internetzugang, [998](#)
- \ctg (babel), [580](#)
- \cth (babel), [580](#)
 - ctt Option (inputenc), [587](#)
- \Cube (ifsym), [416](#)
- \Cup (amssymb), [546](#)

`\cup`, 546
`\curly Option` (natbib), [731](#)
`\curlyeqprec` (amssymb), 548
`\curlyeqsucc` (amssymb), 548
`\curlyvee` (amssymb), 546
`\curlyveedownarrow` (stmaryrd), 550
`\curlyveeuparrow` (stmaryrd), 550
`\curlywedge` (amssymb), 546
`\curlywedgedownarrow` (stmaryrd), 550
`\curlywedgeuparrow` (stmaryrd), 550
`\CurrentOption`, [911](#), [914](#), [920](#), [921](#)
`\currentpage` (layouts), [209](#), [211](#)
`\currenttitle` (titleref), [81](#)
`\Curve` (pspicture), [659](#), [660](#)
`\curve` (curves), [630](#)
`\curvearrowleft` (amssymb), 550
`\curvearrowright` (amssymb), 550
`curves` Paket, 629
`custom-bib` Paket, xxix, 801, 820, 821, [828–835](#)
 in diesem Buch, 1137
`\CustomVerbatimCommand` (fancyvrb), 173, 174, [176](#)
`\CustomVerbatimEnvironment` (fancyvrb), 173, [174](#)
`CVS` (Concurrent Versions System), 869
`Cyr` Umgebung (lb2), [427](#)
`\cyrillicencoding` (babel), [583](#), [584](#)
`\cyrillictext` (babel), [584](#), [605](#)
`czech` Option (babel), 559

D

`D` Syntax (dcolumn), [282](#), [283](#), [284](#), [576](#), [579](#)
`\d`, [465](#), [470](#)
`d` Syntax (lb2), [283](#)
`\dag`, 546
 (textcomp), 471
`\dagger`, 546
`\daleth` (amssymb), 544
`danish` Option (babel), 559
`DANTE` FAQ, 995
`DANTE`, deutschsprachige Usergroup, 1004
`\dasharrow` (amssymb), 550
`\dashbox` (pspicture), 658
`dashjoin` Umgebung
 (eepic), 627
 (epic), [622](#), [623](#), [624](#)
`\dashleftarrow` (amssymb), 550
`\dashlength` (picins), [314](#)
`\dashline`
 (eepic), 627
 (epic), [621](#), [622](#)
`\dashlinedash` feste Länge (arydshln), [276](#)
`\dashlinegap` feste Länge (arydshln), [276](#)
`\dashlinestretch` (epic), [621](#), [622](#)
`\dashrightarrow` (amssymb), 550
`\dashv`, 551
`\date`, [871](#), [945](#)
`\date<sprache>` (babel), 596, 604
 Dateien, *siehe auch* Konfigurationsdateien;
 Dokumente
 Bibliographiestile, 9, 820–823
 Eingabequellen, 7
 Fontdefinition, 7

Dateien (Forts.)

Graphiken, *siehe* Bilddateien
 Hilfsdateien, 8
 Index, 8
 interne, 8
 Klassen, 7
 Kodierspezifikation, 7
 $\LaTeX$ -Format, 7
 Pakete, 7
 Protokolldateien, 8
 Quellen, *siehe* Quelldateien
 reine Textdateien, 7
 Sprachdefinitionsdateien, 7
 $\TeX$ - und $\LaTeX$ -Übersicht, 9
 $\TeX$ -Fontmetriken, 7
 Übersicht, 6–10
 Dateierweiterungen, Bilddateien
 angeben, 643
 Suchreihenfolge, 642, 643
 Dateistruktur (Klassen und Pakete)
 Befehle, 911, 917–919, 921
 Beschreibung, 910
 Hauptteil, 916
 Initialisierungsteil, 913
 Kenndaten, 910–913
 Laden von Paketen, 916
 Mindestanforderungen (Klassen), 922
 Optionen
 ausführen, 914–916
 deklarieren, 913, 914
 Daten auf allen Seiten wiederholen, [617](#)
 Datenbankformat, Bibliographien
 Abkürzungen erzeugen, 798, 799
 Akzentbuchstaben, 797, 798
 Daten, definierte, 789
 Felder, 790–794
 erforderliche, 791, 792
 ignorierte, 792
 optionale, 791, 792
 Groß-/Kleinschreibung, Unterscheidung von,
 790
 Kommentarzeichen, 790
 Kurzformen
 erzeugen, 798, 799
 vordefinierte, 800
 Leerzeichen, 790
 Namen angeben, 795–797
 Präambel, 800, 801
 Publikationstypen, 789–794
 Querverweise, 801, 802
 Schlüssel
 Definition, 789
 Groß-/Kleinschreibung, Unterscheidung
 von, 790
 Sonderzeichen, 797, 798
 Sortierreihenfolge, 794
 Titel, 797
 Trennzeichen, 790
 vordefinierte Zeichenfolgen, 800

- Datenbankmanagement-Werkzeuge, Bibliographien
- aux2bib, 804
 - bib2html, [806](#)
 - bibclean, 806
 - bibextract, 807
 - bibkey, 805
 - biblist, 803, [804](#)
 - BibTexMng, 819, 820
 - bibttool, 808–813
 - bibttools, 804–806
 - citefind, 807
 - citetags, 807
 - doppelte Schlüssel entfernen, 809, 817
 - durchsuchen, 805, 807, [814](#), 817
 - Einträge
 - bearbeiten, 815
 - drucken, 803, [804](#), [805](#), 806
 - extrahieren, 807, 811, 812
 - nach Zeichenfolgen durchsuchen, 805, 807
 - Fehlermeldungen, Farben, 815
 - graphisches Frontend, 813–817
 - HTML-Dateien erzeugen, [806](#), 819
 - Internet-Ressourcen, 803
 - Java Datenbankmanager, 817–819
 - JBibtexManager, 817–819
 - Kommandozeilenwerkzeuge, 804–813, 816
 - lexikalische Analyse, 806
 - looktex, 805
 - makebib, 805
 - normalisieren, 809–811, 816
 - Online-Ressourcen, 803
 - portable Dateien erzeugen, 804
 - Pretty-Printing (formatierte Ausgabe), 806, 808, 809
 - printbib, [805](#)
 - pybliographer, 813–817
 - Quellenverweise exportieren, 805
 - Regeln zur Großschreibung, 816
 - Schlüssel
 - automatisch erzeugen, 812, 813
 - Duplikate entfernen, 809, 817
 - nach Zeichenfolgen durchsuchen, 805
 - zur Bibliographie-Auflistung hinzufügen, 807, [808](#)
 - showtags, 807
 - sortieren, 808, 809
 - umschreiben, 809–811
 - Windows Datenbankmanager, 819, 820
 - Zeichenfolgen
 - alle Einträge durchsuchen nach, 805, 807
 - Schlüssel durchsuchen nach, 805
 - zusammenführen, 808, 809
- Datenfluss
- LaTeX, 712
 - Indexerstellung, 668
 - LaTeX, 10
- \datesdmy (babel), [576](#)
- \datesymd (babel), 575, [576](#)
- Datumsangaben in fremdsprachigen Dokumenten, [574–576](#)
- \DavidStar (bbding), [414](#)
- \DavidStarSolid (bbding), [414](#)
- \dbinom (amsmath), [506](#)
- .dbj Dateierweiterung (custom-bib), 829, 834, [835](#)
- dbk Option (inputenc), 587
- \dblfigrule, 294
- \dblfloatpagefraction, 293
- \dblfloatsep elastische Länge, 293
- \dbltextfloatsep elastische Länge, 293
- \dbltopfraction, 293
- dbltopnumber Zähler, 293
- DC Fonts, 365
- dcolumn Paket, 282–284
- dcu LaTeX-Stil (harvard), 725
- \ddag, 546
- (textcomp), 471
- \ddagger, 546
- \ddddot (amsmath), [507](#), 545
- \dddot (amsmath), [507](#), 545
- \ddot, [507](#), 545, [607](#)
- \ddots, [499](#), 552
- \de (lb2), [578](#)
- debugshow Option (tracefmt), 380
- \decaheterov (hetarom), [631](#)
- \decimalcomma (babel), [574](#)
- \decimalpoint (babel), [574](#)
- \decimalsep (babel), [576](#), [579](#)
- \declare@shorthand (babel), [607](#)
- \DeclareCaptionFormat (caption), 323
- \DeclareCaptionJustification (caption), 320, 323
- \DeclareCaptionLabelFormat (caption), 319, [322](#), [323](#)
- \DeclareCaptionLabelSeparator (caption), [320](#), [323](#)
- \DeclareCaptionListOffFormat (subfig), 330
- \DeclareCaptionStyle (caption), [322](#), 323
- \DeclareDir (docstrip), [864](#), [865](#)
- \DeclareDir* (docstrip), [864](#)
- \DeclareDirectory (docstrip), 955
- \DeclareEncodingSubset (textcomp), [379](#)
- \DeclareErrorFont, 951
- \DeclareFixedFont, [429](#)
- \DeclareFontEncoding, 427, [442](#), 443, 451, [461](#)
- Fehler bei, 933, 962
- Warnung bei, 970
- \DeclareFontEncodingDefaults
- Warnung bei, 970
- \DeclareFontFamily, [414](#), [432](#), 438, [439](#), [441](#), [444](#), [445](#), [449](#), [450](#), 451
- \DeclareFontShape, [414](#), 432, [433](#), [434](#), [435–437](#), 438, 439, [440](#), [441](#), 443, [444](#), [445](#), [449](#), [450](#), 451
- Fehler bei, 936, 937, 945, 953
- Leerzeichen in, 433, 434
- \DeclareFontSubstitution, 443, [461](#)
- Fehler bei, 951
- \DeclareGraphicsExtensions
- (graphics), [642](#), [643](#)
- (graphicx), 642
- \DeclareGraphicsRule
- (graphics), 638, [643](#), [644](#), 645
- Fehler bei, 931

- `\DeclareGraphicsRule` (*Forts.*)
 - (`graphicx`), 645
 - Fehler bei, 931
- `\DeclareInputMath` (`inputenc`), [456](#), 459
- `\DeclareInputText` (`inputenc`), [456](#), [457](#), [458](#), 459
- `\DeclareMathAccent`, [410](#), 447
 - Fehler bei, 971
 - Warnung bei, 971
- `\DeclareMathAlphabet`, [362](#), [364](#), [448](#), 451, 452, [523](#)
 - wann nicht verwenden, 447
 - Warnung bei, 970, 971
- `\DeclareMathDelimiter`, 447
- `\DeclareMathOperator`
 - (`amsmath`), [501](#), [513](#), [514](#)
 - (`amsopn`), 479
- `\DeclareMathOperator*` (`amsmath`), [513](#), [514](#)
- `\DeclareMathRadical`, 447
- `\DeclareMathSizes`, 427, [444](#), [445](#)
- `\DeclareMathSymbol`, [362](#), [446](#), [447](#), 448, 452, [545](#)
 - Fehler bei, 949
 - Warnung bei, 963
- `\DeclareMathVersion`, [448](#), [449](#), 451
 - Warnung bei, 971
- `\DeclareNewFootnote` (`manyfoot`), [129](#), [130–132](#)
- `\DeclareOption`, [911](#), [913](#), 914, 915, [920](#), [921](#)
- `\DeclareOption*`, [911](#), [914](#), 915, [920](#), [921](#)
 - ignoriert globale Optionen, 915
- `\declarepostamble` (`docstrip`), 862
- `\declarepreamble` (`docstrip`), 862
- `\DeclareRobustCommand`, 879
- `\DeclareRobustCommand*`, 879
- `\DeclareSymbolFont`, [445](#), [447](#), 448, [449](#), 451
 - Warnung bei, 971
- `\DeclareSymbolFontAlphabet`, [362](#), [448](#), 451, 452
 - Warnung bei, 971
- `\DeclareTextAccent`, [463](#)
- `\DeclareTextAccentDefault`, [466](#), 467
- `\DeclareTextCommand`, [464](#), [465](#)
- `\DeclareTextCommandDefault`, [377](#), [378](#), [465](#), [466](#)
- `\DeclareTextComposite`, [464](#)
- `\DeclareTextCompositeCommand`, [464](#)
- `\DeclareTextSymbol`, [463](#), 466
- `\DeclareTextSymbolDefault`, [376](#), [466](#), 467
- `\DeclareUnicodeCharacter` (`inputenc`), [457](#), 459, 954
- `\DeclareUrlCommand` (`url`), [102](#), 103
- `decmulti` Option (`inputenc`), 371
- `\def`, [148](#), [878](#), 925, 948, 953
 - in $\TeX$ -Fehlermeldungen, 926
- `.def` Dateierweiterung, 7, 9, 461
 - (`graphics`), 632
 - (`inputenc`), 459
- `default` Schlüsselwert (`caption`), 318, 319, 322
- `default.type` Bib $\TeX$ -Publikationstyp, 837
- `\defaultaddspace` feste Länge (`booktabs`), 279
- `\defaultbibliography` (`bibunits`), 776
- `\defaultbibliographystyle` (`bibunits`), [776](#)
- `\DefaultFindent` (`lettrine`), 108
- `\defaultthyphenchar`, 439
- `\DefaultLhang` (`lettrine`), 108
- `\DefaultLines` (`lettrine`), 107
- `\DefaultLoversize` (`lettrine`), 107
- `\DefaultLraise` (`lettrine`), 107
- `\DefaultNindent` (`lettrine`), 108
- `\DefaultSlope` (`lettrine`), 108
- `\defcitealias` (`natbib`), [728](#)
- `define-alphabet` Funktion (`xindy`), [700](#)
- `define-attributes` Funktion (`xindy`), [701](#)
- `define-letter-group` Funktion (`xindy`), [699](#), [700](#)
- `define-location-class` Funktion (`xindy`), [700](#)
- `defineactive` Schlüssel (`fancyvrb`), 170
- `\DefineFNSymbols` (`footmisc`), [123](#), [124](#)
- `\defineshorthand` (`babel`), [564](#)
- `\DefineShortVerb` (`fancyvrb`), 176, [177](#)
- Definitionslisten, *siehe* theoremmähnliche Strukturen
- `\DEFIvec` (`lb2`), [878](#), [879](#)
 - `defn` Umgebung (`lb2`), [148](#)
- `\deg`, 513
- Deklarationen vs. High-Level-Fontbefehle, 355, [356](#)
- dekorative Boxen, 614–618
- `delarray` Paket, 502
- `\deletebarwidth` feste Länge (`changebar`), 200
- `\DeleteShortVerb`
 - (`doc`), [848](#), [853](#), 867
 - (`shortvrb`), [160](#)
- `delim_0` Schlüsselwort (`makeindex`), [682](#), [684](#)
- `delim_1` Schlüsselwort (`makeindex`), [682](#), [684](#)
- `delim_2` Schlüsselwort (`makeindex`), [682](#), [684](#)
- `delim_n` Schlüsselwort (`makeindex`), 682
- `delim_r` Schlüsselwort (`makeindex`), 682
- `\delimitershortfall`, [403](#)
- `\Delta`, [403](#), [502](#), [512](#), 543
- `\delta`, [510](#), 543
- `\Denarius` (`marvosym`), [423](#)
- `\depth`, [894](#), [895](#)
 - (`graphics`), 648
- `depth` Schlüssel (`graphicx`), 637
- `depth` Syntax, 900, 901
- `\DescribeEnv` (`doc`), [847](#), [850](#), 853
- `\DescribeMacro` (`doc`), [847](#), [850](#), 853
- `Description` Umgebung (`lb2`), [156](#), 157, [158](#), 159
- `description` Umgebung, 85, [139](#), 143, 144, 146, [155](#), [156](#), [176](#), 618, 881
- `\Descriptionlabel` (`lb2`), [156](#), 157, [158](#), 159
- `\descriptionlabel`, [139](#), 146, [155](#), [156](#)
 - (`paralist`), 146
- `\det`, 504, [513](#)
- Deutsch
 - Abstände nach Satzzeichen, 581
 - alte Rechtschreibung, 569, 597
 - Anführungszeichen, 569
 - Index-Sortierreihenfolge, 678, 690, 692
 - neue Rechtschreibung, 19, 560, 597
 - Silbentrennung, 569
 - Usergroup, DANTE, 1004
- `devnag` Paket, 608
- Dezimalzahlen in Tabellen ausrichten, [280](#), 282, [283](#), [284](#)
- `\dfrac` (`amsmath`), [505](#)
- `.dfu` Dateierweiterung (`inputenc`), 459, 460
- `\DH`, 469
- `\dh`, 471
- `\diagdown` (`amssymb`), 544

- diagram Paket, 500, 1014
- Diagramme, *siehe* Graphen
- \diagup (amssymb), 544
- diakritische Zeichen, *siehe* Akzentbuchstaben
- Dialekte definieren, 600–602
- \Diamond (latexsym), [476](#)
- \diamond, [507](#), [546](#)
- \diamondsuit, 544
- dicktengleiche Schriften, [342](#), [343](#), 350, *siehe auch*
 - wortwörtliche Ausgabe;
 - Schreibmaschinenschrift
- Courier, 386
- dieses Buch
 - Autoren, 1129–1133
 - Beispiele in, 15, 16
 - Entstehungsgeschichte, 1135–1137
 - Errata für, xxvi
 - typographische Konventionen, 12–15
- \digamma (amssymb), 543
- Digital Object Identifier (DOI), 735
- \dim, 513
- \dimen, 945, [979](#), 980
- \dimen73 feste Länge, [979](#)
- \ding (pifont), [136](#), [138](#), [390](#), 393
- dingautolist Umgebung (pifont), 138, [390](#)
- dingbat Paket, 411
- dingbat.mf Datei (dingbat), 411
- \dingfill (pifont), [392](#), 393
- \dingline (pifont), [392](#), 393
- dinglist Umgebung (pifont), [390](#)
- directivestyle Schlüssel (listings), 179
- \DisableCrossrefs (doc), [849](#), 850, [853](#)
- \discretionary, 182, 938, [988](#)
- display Schlüssel (titlesec), [40](#), 41, 42, 44, [45](#)
- \displaybreak (amsmath), 493, 494
 - Fehler bei, 933
- \displaycaps (lb2), [99](#)
- \displaylimits, [504](#)
- displaymath Option (lineno), [187](#)
- displaymath Umgebung (amsmath), 483
- \displaystyle, 91, 444, 506, [515](#), [516](#)
 - (relsize), 91
- Dissertationsjahr in Bibliographien, 768
- dissyear $\LaTeX$ -Feld (jurabib), 768
- \div, 546
- DIVn Option (typearea), 212
- DIV7 Option (typearea), 212
- DIVcalc Option (typearea), [212](#), [213](#)
- DIVclassic Option (typearea), 212
- \divide, [905](#)
- \divideontimes (amssymb), 546
 - Divis, *siehe* Trennzeichen; Silbentrennung
- \DJ, 469
- \dj, 471
- doc Paket, 160, 599, [845–857](#), 867
- doc.dtx Datei (doc), [846](#), 859
- doc.sty Datei (doc), 859
- \docdate (doc), 855
- \DocInclude (ltxdoc), 868
- \DocInput (doc), [850](#), 852, [853](#), 868
- docstrip Paket, 24, [857–866](#), 1024, 1025
 - Fehler bei, 923, 955
- docstrip.cfg Datei (docstrip), 863, [864](#), 955
- \DocstyleParms (doc), 856
- document Option (ragged2e), [113](#), [405](#)
- document Umgebung, 14, 15, [18](#), 20, 911, 917
 - Überprüfen der Fontkonfiguration, 452
 - Fehler bei, 931, 955
 - Probleme mit, 961
- \documentclass, 14, 15, [18](#), [20](#), [21](#), 22, [912](#), 915
 - Fehler bei, 952
 - globale Optionen, 19, 560
 - Versionsdaten, [912](#)
 - Warnung bei, 975, 976
- \documentstyle, 476
 - Fehler bei, 952
- doi $\LaTeX$ -Feld
 - (custom-bib), 833
 - (natbib), 735
- DOI (Digital Object Identifier), 735
- Dokumentation suchen, 1002
- Dokumentationsbefehle, Liste der, 853–857
- Dokumentationsklasse (ltxdoc)
 - Befehle, 867
 - Beschreibung, 867
 - Erweiterungen, 867
 - Formatierungsoptionen, 868, 869
 - Konfigurationsdateien anlegen, 868, 869
- Dokumentationstreiber, 599, 846, [850](#)
- Dokumentationswerkzeuge
 - Abschnitte drucken, 848, 868
 - Änderungsverlauf, erzeugen, 849, 869
 - automatische Indexierung deaktivieren, 849, 869
 - bedingter Code, Syntax, 851–857
 - Befehle, Liste der, 853–857
 - Begrenzungssymbole für wortwörtlichen Text
 - definieren, 848
 - Syntax, 847
 - Beschreibung, 846
 - CVS, 869
 - Dokumentationsbefehle, Liste der, 853–857
 - Dokumentationsklasse (ltxdoc), 867–869
 - Einbinden von Dateien, 868
 - Eingabebefehle, Liste der, 857
 - Formatierungsbefehle, Liste der, 853–857
 - Indexbefehle, Liste der, 853–857
 - Indexeinträge automatisch erzeugen, 849, 869
 - Kommentare aus der Quelldatei entfernen,
 - 857–866
 - Kommentarzeichen, 847
 - Layoutparameter, Liste der, 853–857
 - Leerzeichen, 847
 - Makrobefehle erstellen, 847, 848
 - Präambelbefehle, Liste der, 853–857
 - Querverweise, 849, 850
 - RCS, 869
 - rct Paket, [870](#), 871
 - rcsinfo Paket, [872](#)
 - Schlüssel
 - $\$Id\$$ -Schlüsselwort parsen, [872](#)
 - RCS-Daten extrahieren, [870](#), 871

Dokumentationswerkzeuge (*Forts.*)

- Softwareversionskontrolle, 869, [870](#), [872](#)
- Syntax, 846, 847
- Syntaxdiagramme erzeugen, 867
- Treiberdateien
 - erstellen, 850
 - in bedingten Code einbinden, 852
- Umgebungsbeschreibungen erstellen, 847, 848
- Verlaufsbeefhle, Liste der, 853–857
- Versionsverwaltung, 869, [870](#), [871](#), [872](#)

Dokumente

- anzeigen, *siehe* Seitenbeschreibungssprachen
- aufteilen, [21](#), [22](#)
- Gliederung, 24–26
- in Teilen neu formatieren, 21, 22
- letzte Seite, Verweis auf, 224, [234](#)
- PDF-Dokumente durchsuchen, 367, [368](#)
- Quelldateien, *siehe* Quelldateien
- Rückwärtskompatibilität, 475, 476
- schrittweise verarbeiten, 992, *siehe auch*
 - Fehlersuche
- Versionen für den Ausdruck auswählen, [23](#), [24](#)
- Versionsverwaltung, 23, 24, 869–872
- zu groß für einen einzigen Lauf, [21](#), [22](#), *siehe auch*
 - Quelldateien aufteilen

Dokumentenklassen

- $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$, 479
- Definition, 18
- Name, 18
- Standard, *siehe* article; book; report
- verändern, 20

Dokumentenpräambel, *siehe* Präambel

- `\dominilof` (minitoc), 59
- `\dominilot` (minitoc), 59
- `\dominitoc` (minitoc), 59
- `\DoNotIndex` (doc), 850, 854
- `\DontCheckModules` (doc), 853
- `\doparttoc` (minitoc), 61
- Doppelpunkt (:), Kurzzeichen, [571](#)
- doppelseitiger Ausdruck
 - einschalten, 207
 - Kolumnentitel-Layouts, 231, 232, [234](#)
- doppelte Anführungsstriche, *siehe*
 - Anführungszeichen
- doppelte Boxen, [615](#)
- doppelte Linien, 278
- `\dosecttoc` (minitoc), [61](#)
- `\dot`, [507](#), 545
- `\dotafter` Schlüssel/Option (jurabib), 753, [764](#)
- `\Doteq` (amssymb), 548
- `\doteq`, 548
- `\doteqdot` (amssymb), 548
- `\dotfill`, 392, 684, [889](#)
- `\dotinlabels` Option (titletoc), 63, [64](#)
- `\dotplus` (amssymb), 546
- `\dots`, [87](#), [88](#), [471](#), 508
 - (amsmath), [505](#), [508](#), [509](#)
 - (ellipsis), [88](#)
- `\dots` Option (euro), [103](#), [104](#)
- `\dotsb` (amsmath), [503](#), [509](#)
- `\dotsc` (amsmath), [509](#)

- `\dotsi` (amsmath), [509](#)
- `\dotsm` (amsmath), [509](#)
- `\dotso` (amsmath), 509
- `\dotted` Option (minitoc), 59
- `\dottedjoin` Umgebung
 - (eepic), 627
 - (epic), [622](#), [623](#)
- `\dottedline`
 - (eepic), 627
 - (epic), [620](#), 621, 622
- `\doublebox` (fancybox), [615](#)
- `\doublecap` (amssymb), 546
- `\doublecup` (amssymb), 546
- `\DoubleperCent` (docstrip), [866](#)
- `\doublerulesep` feste Länge, [251](#), 279
- `\doublerulesepcolor` (colortbl), 273
- `\doubleSPACE` Umgebung (setspace), [114](#), 115
- `\doublespacing` (setspace), 114
- `\Downarrow`, 511, 550
- `\downarrow`, [501](#), [511](#), [550](#)
- `\downdownarrows` (amssymb), 550
- `\downharpoonright` (amssymb), 550
- `\draft` Option, 985
 - (graphics), 633
 - (graphicx), 633
 - (showkeys), 72
 - (varioref), 77
- `\draft` Schlüssel (graphicx), [638](#), [639](#)
- `\drawdimensionsfalse` (layouts), [209](#)
- `\drawdimensionstrue` (layouts), 210
- `\drawjoin` Umgebung
 - (eepic), 627
 - (epic), [622](#), [623](#)
- `\drawline`
 - (eepic), [627](#), 628, 629
 - (epic), [621](#), [622](#), 629
- `\drawlinestretch` (epic), 622
- `\drawwith` (ecltree), [630](#)
- dreher
 - Bilddateien, 638
 - Gleitobjekte, 305, [306](#), 307
 - graphische Objekte, [648–652](#)
- `\drop` Schlüssel (titlesec), [40](#), [42](#), [43](#)
- drucken
 - ausgewählte Dokumentversionen, [23](#), [24](#)
 - Benutzerdokumentation, 848
 - Bibliographien, 803, [804](#), [805](#), 806
 - Code-Dokumentation, 868
 - Computercode, *siehe* Computercode setzen
 - doc-Paket, 845, 846
 - doppelseitig, *siehe auch* Recto-Verso-Layout
 - einschalten, 207
 - Kolumnentitel-Layouts, 231, 232, [234](#)
- `.dtx` Dateierweiterung, 9
 - (doc), 7
 - (ltxdoc), 868
- `\duplicate` Option (chapterbib), 774
- `\duplicate$` interne BibTeX-Funktion, 839
- Durchschuss, 205, [426](#), *siehe auch* Zeilenvorschub
- `\dutch` Option (babel), 559, [568–570](#), [601](#)
- DVD, CTAN, 997

- .dvi Dateierweiterung, 8, 9, 10, 338, 611, 683
 - dvi2svg Programm, 664
 - dviaw Option (graphics), 633
 - dviaw Programm, 633
 - dvlaser Option (graphics), 633
 - dvlaser/PS Programm, 633
 - dvipdf Option (graphics), 633
 - dvipdf Programm, 633
 - dvipdfm Schlüssel/Option (geometry), 218
 - dvipdfm Programm, 661
 - dvips Option
 - (changebar), 199
 - (crop), 222
 - (graphics), 632, 633, 953
 - dvips Schlüssel/Option (geometry), 218
 - dvips Programm, 199, 432, 632, 633, 655–657, 665, 1018
 - dvips.def Datei (graphics), 632
 - dvipsnames Paket, 200
 - dvipsone Option (graphics), 633
 - dvipsone Programm, 632, 633
 - dvitoln03 Option (changebar), 199, 200
 - dvitops Option
 - (changebar), 199
 - (graphics), 633
 - dvitops Programm, 633
 - dviwin Option (graphics), 633
 - dviwin Programm, 633
 - dviwindo Option (graphics), 633
 - dviwindo Programm, 633
 - dynamic Schlüsselwert (jurabib), 743, 758
- E**
- E Syntax (fancyhdr), 234, 235–238
 - E-Mail-Adressen formatieren, 99, 100, 101, 102, 103
 - E... Fontkodierung, 442
 - ebenda-Verweise, *siehe* *ibidem*
 - \Ebena (lb2), 48
 - EC (European Computer Modern) Fonts, 365, 366, 367
 - \ecaption (lb2), 58
 - ectree Paket, 630
 - eco Option (euro), 103, 104
 - eco Paket, 67, 395
 - \edef, 138
 - Probleme mit, 927
 - edition BibTeX-Feld, 742, 791, 792
 - Editionsdaten, Bibliographien, 768
 - editor BibTeX-Feld, 714, 757, 768, 791, 792, 793–795, 796
 - \editorname (jurabib), 760
 - editortype BibTeX-Feld (jurabib), 768
 - eeepic Paket, 622, 625–629, 655, 657, *siehe auch* epic-Paket
 - eeepicmu Paket, 629
 - \efloatseparator (endfloat), 299
 - efxmp1.cfg Datei (endfloat), 300
 - \eg (lb2), 87
 - \ega (lb2), 480
 - egrep Programm, 805
 - eid BibTeX-Feld
 - (custom-bib), 833
 - (natbib), 735
 - EID, Bibliographien, 735
 - Eigenschaften, *siehe* Optionen
 - Einbinden von Dateien, *siehe auch* Ausschließen von Dateien
 - Bilddateien, *siehe* Bilddateien einbinden
 - Dokumente teilweise neu formatieren, 21
 - Gründe, 21
 - Quellcode/Dokumentation, 868
 - Eingabebefehle (Dokumentation), Liste der, 853–857
 - Eingabedateien
 - anlegen (BibTeX), 711–715
 - Codeerzeugung, 859, 860
 - Index, 678, 690
 - Quelldateien, 7
 - eingabegetreue Ausgabe, *siehe* wortwörtliche Ausgabe
 - Eingabekodierung, 339–341, 369, 370, 371, 372, 456–460
 - eingebettete Überschriften, 29, 31, 32
 - eingebundene Dateien
 - Kandidaten für, 21
 - einleitender Text, Quellenverweise, 746, 747
 - Eintragsvariablen, Bibliographien, 836
 - Einzüge
 - bei wortwörtlicher Ausgabe entfernen, 165
 - Bibliographien, 764, 765
 - Inhaltsverzeichnisse, 54, 62
 - nach Überschriften, 34, 43, 581
 - unterdrücken, 34, 42
 - nummerierte Listen, 145
 - Programmlistings, 181
 - theoremähnliche Strukturen, 149
 - von Überschriften, 30, 41
 - Einzelbyte-Zeichen, *siehe* Single-Byte-Zeichen
 - elastische Längen, 887
 - elektronische Veröffentlichungen, Bibliographien, 735
 - \ell, 544
 - \ellipse
 - (eeepicmu), 629
 - (eeepic), 628
 - \ellipse*
 - (eeepicmu), 629
 - (eeepic), 628
 - Ellipsen
 - ausfüllen, 628, 629
 - zeichnen, 628
 - ellipsis Paket, xxix, 88, *siehe auch* lips-Paket
 - \ellipsisgap (ellipsis), 88
 - \ellipsispunctuation (ellipsis), 88
 - \ELSE (algorithmic), 177
 - \em, 352, 353, 355
 - Verwendung von Kapitälchen, 353
 - (ulem), 93
 - emacs Programm, 817, 993, 1025
 - \email (lb2), 102
 - \emdash, 461
 - emdash Option (euro), 104
 - \emergencystretch feste Länge, 109, 110, 973, 987
 - \emminershape (fixltx2e), 353

- \emph, 175, 352, [353](#), 355, [356](#), [882](#)
 - Fehler bei, 947
 - Verwendung von Kapitalchen, [353](#)
 - (ulem), [93](#), 94
 - (yfonts), 405
- emph Schlüssel (listings), 180
- emphstyle Schlüssel (listings), 180
- empty Kolumnentitel-Layout, 230
 - erzeugt unerwünschte Seitenzahl, 231
- empty Schlüsselwert
 - (caption), 319
 - (subfig), 330
- empty\$ interne BibTeX-Funktion, 839, [840-843](#)
- \emptyset, 544
- emTeX Programm, 632, 633
- emtex Option
 - (changebar), 199, 200
 - (graphics), 633
- \EnableCrossrefs (doc), [849](#), [853](#), [869](#)
- encap Schlüsselwort (makeindex), 680
- encap_infix Schlüsselwort (makeindex), 682
- encap_prefix Schlüsselwort (makeindex), 682
- encap_suffix Schlüsselwort (makeindex), 682
- \encapchar (doc), 854
- \enclname (babel), 561
- \encodingdefault, 357, 358, [429](#)
- \End (lb2), [501](#)
- \end, in TeX-Fehlermeldungen, 947, 955, 963
- endash Option (euro), [104](#)
- \endbatchfile (docstrip), [858](#), [859](#)
- \endcsname, [29](#), 942, [978](#), [979](#)
- \endfirsthead (longtable), [268](#), [269](#), 270
- endfloat Paket, xxix, [298-300](#)
 - kombiniert mit rotating, 300
- endfloat.cfg Datei (endfloat), 299, 300
- \endfoot (longtable), [268](#), [269](#)
- \endgraf, 932
- \endgroup, 517, 931, 944
 - Fehler bei, 934, 935
- \endhead (longtable), [268](#), [269](#), 270
- \ENDIF (algorithmic), [177](#)
- \endinput, 860, 936
- \endlastfoot (longtable), [268](#), [269](#)
- \endnote (endnotes), 132, [133](#), [134](#)
 - endnote Schlüsselwert (jurabib), 753
 - endnote Zähler (endnotes), 133
- \endnotemark (endnotes), [133](#), [134](#)
 - Endnoten, 132, [133](#), [134](#), *siehe auch* Fußnoten; Marginalien
 - Endnotenverweise, Bibliographien, 752, [753](#)
- endnotes Paket, xxix, 132-134
- \endnotetext (endnotes), 133
- \endpostamble (docstrip), 862
- \endpreamble (docstrip), 862
- \eng (lb2), [579](#)
 - english Option (babel), 559, [562-564](#), [760](#)
- enjbib.ldf Datei (jurabib), 759
- \enlargethispage, [242](#), 975
 - Fehler bei, 947, 949
- \enlargethispage*, [242](#), [243](#)
- \enoteformat (endnotes), 133, [134](#)
- \enoteheading (endnotes), 133
- \notesize (endnotes), 133, [134](#)
- \enskip, 522
- \enspace, [39](#), [889](#)
- \ensuremath, [459](#), [876](#), [877](#), [879](#), [977](#), [978](#)
- .ent Dateierweiterung (endnotes), 133
- Entfernen von Kommentaren aus der Quelldatei, *siehe*
 - Kommentare aus dem Code entfernen
- ENTRY BibTeX-Befehl, 836, 837, [838](#), 841, 842
- Entwurfsmodus, 72, 77, 633
- enumerate Umgebung, 85, 136, [137](#), [138](#), 140, [142](#), [143](#), 618
 - Fehler bei, 951
 - Querverweis auf, 70
 - Stilparameter, 137
 - (enumerate), 142
 - (paralist), 142
- enumerate Paket, 141
- enumi Zähler, [137](#), [138](#), 883
- enumii Zähler, [137](#), 883
- enumiii Zähler, 137, 883
- enumiv Zähler, 137, 883
- environment, *siehe* Umgebung
- environment Umgebung (doc), [848](#), 854, 856
- epic Paket, 619-625, 627, 629, 630, *siehe auch*
 - eeepic-Paket
- \EPS (lb2), [876](#)
- .eps Dateierweiterung, 9, 643, 644, 931
- .eps.gz Dateierweiterung, 644
- eps2pdf Programm, 661
- \epsilon, 543
- \eqcirc (amssymb), 548
- eqnarray Umgebung, [482](#), 483, 618
 - falsche Abstände in, [482](#)
 - Fehler bei, 934, 951
- eqnarray* Umgebung, [482](#), 618
- \eqref (amsmath), [74](#), [495](#), [497](#)
- \eqsim (amssymb), 548
- \eqslantgtr (amssymb), 548
- \eqslantless (amssymb), 548
- \equal (ifthen), [76](#), [77](#), [240](#), [906](#), [907](#), [910](#)
- equation Umgebung, [16](#)
 - falsche Abstände um, [492](#), [493](#)
 - Querverweis auf, 70
 - (amsmath), [481-483](#), 485, 496
 - Fehler bei, 930
- equation Zähler, [883](#), 886, [887](#)
 - (amsmath), 494, 497
- equation* Umgebung (amsmath), [481](#), [483](#), 485, [490](#)
- \equiv, [487](#), [505](#), 548
- \eqvref (lb2), [74](#)
 - erforderliche Felder, Bibliographiedatenbank, 791, 792
 - Ergebnisdateien, Codeerzeugung, 859, 860
 - Erhöhen von Zählern, [885](#)
- errorcontextlines Zähler, 926
- errorshow Option
 - (multicol), 197
 - (tracefmt), 380
- Erweiterungen
 - Übersicht, 6-10
 - unterstützte, bei Bibliographien, 832-834

- escape Schlüsselwort (makeindex), 680
 - Escape-Zeichen, 169, [170](#)
 - \Esper (babel), 576
 - \esper (babel), 576
 - esperanto Option (babel), 559, [573](#), [575](#)
 - \esssup (lb2), 479, [514](#)
 - estonian Option (babel), 559
 - \eta, 543
 - \etc
 - (lb2), [87](#)
 - (yfonts), [407](#)
 - eTeX, T_EX-Erweiterung, 228, 458, 459, 510, 517, 556, 582, 945, 959, 964
 - etex Paket, 945
 - \eth (amssymb), 544
 - ethiop Paket, 608
 - eucal Option (mathscr), [408](#)
 - eucal Paket, 408, 480
 - \EuFrak (eufrak), 408
 - eufrak Paket, [408](#), 409, [480](#)
 - euler Paket, 408, 409
 - falsche Ziffern, 410
 - Euler Font, [407](#), [408–410](#), 480, [530](#)
 - Euler Fraktur Font, 480, [523](#)
 - euler-digits Option (eulervm), [410](#), [531](#)
 - euler-hat-accent Option (eulervm), [410](#)
 - eulervm Paket, [408–410](#), 447, 531
 - \EUR
 - (europs), [422](#)
 - (eurosym), [420](#)
 - (marvosym), [423](#)
 - \EURcf (marvosym), [423](#)
 - \EURcr (europs), 422
 - \EURdig (marvosym), [423](#)
 - \EURhvv
 - (europs), 422
 - (marvosym), [423](#)
 - \EURO (euro), [103](#), [104–106](#)
 - \euro
 - (eurosans), [104](#), [106](#), [421](#), [422](#)
 - (eurosym), [419](#), 420
 - (lb2), [422](#)
 - euro Option (textcomp), [374](#), [399](#)
 - euro Paket, xxviii, [103–106](#)
 - kombiniert mit color, 106
 - Euro-Währung formatieren, [103–106](#)
 - Euro-Zeichen (€), [418–423](#)
 - euro.cfg Datei (euro), 104
 - \EUROADD (euro), [104](#)
 - \EUROfc (europs), 422
 - \EUROFORMAT (euro), [104](#), [105](#), [106](#)
 - European Computer Modern (EC) Fonts, 365, 366, [367](#)
 - European Modern Fonts, 350, 366
 - europs Paket, 422
 - eurosans Paket, 104, 106, [421](#), [422](#)
 - \EUROSYM (euro), [104](#), [106](#)
 - eurosym Paket, 419, 420
 - \EURtm
 - (europs), 422
 - (marvosym), [423](#)
 - \EuScript (eucal), 408
 - even Schlüsselwort (titlesec), [46](#)
 - even Schlüsselwort (makeindex), 679
 - \evensidemargin feste Länge, [203–205](#), [207](#), 920
 - \everypar, 263
 - EX Umgebung (lb2), [885](#)
 - exa Umgebung (lb2), [147](#), [150](#)
 - example Umgebung (lb2), [172](#)
 - exception dictionary Fehler, 958
 - \excludeonly (excludeonly), [22](#)
 - excludeonly Paket, 22
 - EXECUTE B_IB_TE_X-Befehl, [837](#), [838](#)
 - \ExecuteOptions, 632, [911](#), [915](#)
 - executive Option (crop), 221
 - executivepaper Option, 204
 - (typearea), 212
 - executivepaper Schlüssel/Option (geometry), 215
 - \exists, 544
 - \exp, 513
 - expand.macros Funktion (bibtool), [811](#)
 - \expandafter, [978](#), 979
 - expert Option (fourier), 404
 - \ExplainOptions (optional), 24
 - Exportieren von Quellenverweisen, 805
 - exscale Option (ccfonts), 396
 - exscale Paket, 91, 379
 - durch amsmath bereitgestellt, 517
 - durch ccfonts bereitgestellt, 396
 - durch eulervm bereitgestellt, 409
 - durch mathpazo bereitgestellt, 389
 - durch mathptmx bereitgestellt, 388
 - kombiniert mit relseize, 91
 - ext Schlüssel (graphic), 638
 - \ext@figure, [56](#), [57](#)
 - \ext@table, [56](#), [57](#)
 - extendedchars Schlüssel (listings), [179](#), [184](#)
 - \externaldocument (xr), 82, 83
 - externe Dokumente, Querverweise auf, 82, 83
 - extra Option (tipa), 416
 - \extracolsep, [254](#), 281, [287](#), [288](#)
 - (array), 254
 - (longtable), [269](#)
 - \extrafootnoterule (manyfoot), 131
 - \extramarks (extramarks), [229](#), 230
 - extramarks Paket, xxix, 227, [228–230](#)
 - \extrarowheight feste Länge (array), 252, [253](#), [254](#), [276](#), 277
 - \extrarulesep feste Länge (tabls), 277
 - \extras⟨sprache⟩ (babel), 596, 604
 - \extrasrussian (babel), [605](#)
 - \extratabsurround feste Länge (array), [288](#), [289](#)
 - \eye (dingbat), [411](#)
 - \EyesDollar (marvosym), [423](#)
 - EZ Fonts, 367
- ## F
- F Syntax (fancyhdr), 234, 235
 - führende Leerzeichen
 - in wortwörtlicher Ausgabe entfernen, [165](#)
 - Index, [670](#), 678, 688, 691
 - Führungslinien, *siehe* Leitpunkte
 - Füllen von Kreisen/Ellipsen, 628, 629

- Füllmaterial, *siehe* Leitpunkte
- \FAIL (lb2), [960](#)
- \FAILa (lb2), [956](#), [957](#)
- \FAILb (lb2), [956](#), [957](#)
- \FAILc (lb2), [956](#), [957](#)
- \FAILd (lb2), [956](#), [957](#)
- \fakelistoffigures (minitoc), [59](#)
- \fakelistoftables (minitoc), [59](#)
- \faketableofcontents (minitoc), [59](#), [61](#)
- \fallingdotseq (amssymb), [548](#)
- false Schlüsselwert
- (caption), [318](#), [320](#)
 - (fancyvrb), [168](#)
 - (geometry), [215](#)
 - (jurabib), [749](#)
 - (listings), [180](#), [181](#), [182](#)
 - (subfig), [328](#)
 - (titlesec), [46](#)
- false Syntax, [907](#), [908](#)
- Familien, Schrift-, *siehe* Fontfamilien
- \familydefault, [357](#), [358](#), [385](#), [429](#)
- (yfonts), [405](#)
- fancy Kolumnentitel-Layout (fancyhdr), [229](#), [232](#), [233–241](#), [616](#), [872](#)
- fancybox Paket, [614–618](#)
- \fancyfoot (fancyhdr), [234](#), [235–238](#), [241](#)
- \fancyfootoffset (fancyhdr), [235](#)
- fancyhdr Option (rcsinfo), [872](#)
- fancyhdr Paket, [xxix](#), [228](#), [229](#), [232–240](#)
- durch rcsinfo bereitgestellt, [872](#)
 - in diesem Buch, [1137](#)
- \fancyhead (fancyhdr), [234](#), [235–238](#), [241](#), [306](#)
- fancyheadings Paket, [232](#)
- \fancyheadoffset (fancyhdr), [235](#)
- \fancyhf (fancyhdr), [234](#), [235](#), [238–241](#)
- \fancyhfoffset (fancyhdr), [235](#), [616](#)
- \fancyoval (fancybox), [615](#)
- \fancypage (fancybox), [615](#), [616](#), [617](#)
- \fancypagestyle (fancyhdr), [238](#)
- \fancyput (fancybox), [617](#)
- \fancyput* (fancybox), [617](#)
- fancyref Paket, [80](#)
- \FancyVerbFormatLine (fancyvrb), [164](#), [165](#), [166](#)
- FancyVerbLine Zähler (fancyvrb), [165](#), [168](#)
- \FancyVerbStartString (fancyvrb), [170](#), [171](#)
- \FancyVerbStopString (fancyvrb), [170](#), [171](#)
- \FancyVerbTab (fancyvrb), [169](#)
- fancyvrb Paket, [160](#), [161](#), [163–177](#), [178](#), [180](#), [182](#), [183](#)
- kombiniert mit color, [166](#), [172](#)
- fancyvrb.cfg Datei (fancyvrb), [177](#)
- FAQ (Frequently Asked Questions), [995](#)
- Farben
- Bibliographien, [719](#)
 - Fehlermeldungen, Bibliographien, [815](#)
 - Fehlersuche, [903](#), [904](#)
 - Hintergrund, [166](#)
 - Linien, [273](#)
 - numerische Quellenverweise, [719](#)
 - Rahmenlinien, [166](#)
 - Tabellen, [272](#), [273](#)
 - Tabellenlinien, [273](#)
- Farben (Forts.)
- wortwörtliche Ausgabe
 - Hintergrund, [166](#)
 - Rahmenlinien, [166](#)
 - Text, [164](#), [165](#)
 - farskip Schlüssel/Option (subfig), [327](#)
 - \fatbslash (stmaryd), [546](#)
 - \fatsemi (stmaryd), [546](#)
 - \fatslash (stmaryd), [546](#)
 - \Faxmachine (marvosym), [413](#)
 - \fbox, [316](#), [503](#), [893](#), [894](#), [899](#), [902](#), [903](#)
 - (shadow), [613](#)
 - \fboxrule feste Länge, [894](#), [902](#), [903](#), [905](#)
 - (boxedminipage), [613](#)
 - (fancybox), [615](#)
 - \fboxsep feste Länge, [166](#), [336](#), [894](#), [902](#), [903](#), [905](#)
 - (boxedminipage), [613](#)
 - (fancybox), [614–616](#)
 - \fcolorbox (color), [273](#)
 - fcolumn Umgebung (lb2), [902](#)
 - \fcolwidth feste Länge (lb2), [905](#)
 - .fd Dateierweiterung, [7](#), [9](#), [366](#), [441](#), [443](#), [444](#), [445](#), [523](#), [965](#), [973](#)
 - definieren, [449–451](#)
 - .fdd Dateierweiterung (ltxdoc), [868](#)
 - Fehler in diesem Buch, [xxvi](#)
 - Fehlermeldungen, *siehe* Meldungen, Fehler; Fehlersuche
 - Fehlersuche
 - Absatzumbrüche, [985–989](#)
 - Basisbefehle, $\TeX$
 - Ausführung überwachen, [991](#)
 - darstellen, [979](#)
 - Befehlsausführung überwachen, [992](#), [993](#)
 - Befehlsdefinitionen anzeigen, [977–979](#)
 - Befehlsnamen, seltsame, [978](#)
 - Beschreibung, [923](#), [924](#)
 - bewegte Argumente, [100](#), [927](#), [928](#)
 - Bildschirmüberwachung, [989](#), [990](#)
 - Boxen, Inhalte darstellen, [990](#)
 - buffer size Fehler, [958](#)
 - Debug-Meldungen bei Indexen, [697](#)
 - Dokumente schrittweise verarbeiten, [992](#)
 - exception dictionary Fehler, [958](#)
 - Farben, [903](#), [904](#)
 - fehlende Zeichen aufspüren, [991](#)
 - Fehlermeldungen
 - Liste der, [928–956](#)
 - MakeIndex, [679–681](#)
 - nur Stern, [929](#)
 - Quellcodezeile finden, [924–928](#)
 - Syntax, [924](#)
 - font memory Fehler, [958](#)
 - Fontauswahl, [379](#)
 - Fußnoten, [990](#), [991](#)
 - Glyphen, [380](#), [381](#)
 - grouping levels Fehler, [958](#), [959](#)
 - hartnäckige Fehler, [927](#)
 - hash size Fehler, [959](#)
 - Indexerstellung, [687](#), [688](#)
 - informelle Meldungen, [961–977](#)

Fehlersuche (*Forts.*)

- interne Tabellen, Überlauf, 958–961
- List-Stack (T_EX Programm), anzeigen, 990
- MakeIndex*, 687, 688
- Makro-Stack (T_EX Programm), anzeigen, 926
- memory exceeded Meldung, 956–961
- number of strings Fehler, 959
- Overfull box Meldung, 55, **109**, 969, 970, **985**
- parameter stack size Fehler, 960
- pattern memory Fehler, 960
- pool size Fehler, 960
- Protokolldatei, hineinschreiben in die, 990
- Querverweisfehler, 928
- Registerwerte anzeigen, 979, 980
- Restore-Werte anzeigen, 990
- save size Fehler, 961
- Seiteninhalte symbolisch darstellen, 980–983
- Seitenumbrüche, 980–985
- semantic nest size Fehler, 961
- Silbentrennung, 985, 986
- TeX capacity exceeded Fehler, 956–961
- text input levels Fehler, 961
- trace Paket, 992, 993
- Underfull box Meldung, **110**, 215, 383, 440, 973–975
- vertikale Abstände, 980–985
- Warnungen, 961–977
- zerbrechliche Befehle, 906, 927, 928, 946, 953, 955
- $\mathcal{M}\mathcal{S}\text{-}\text{\LaTeX}$, 480
- Felder, Bibliographien, 790–794, 841, 842
- Feldvariablen, Bibliographien, 836
- `\FEMALE` (marvosym), **413**
- `\Female` (marvosym), **413**
- Fenster in Absätzen, 115, 116
- feste Längen, 887
- fette Schriften
 - Beschreibung, 345
 - in Formeln, **524–527**, 528
- .fff Dateierweiterung (endfloat), 299
- `\fg` (babel), **569**, **571**
- `\fi`, 940
- fighead Option (endfloat), 298
- figlist Option (endfloat), 298
- figure Umgebung, 50, 116, 300, 315, **316**, 317, **318–321**
 - Fehler bei, 936, 939, 946
 - gleitet zu weit, 297
 - Label in, **71**
 - Layoutparameter, 292, 294, 295
 - Querverweis auf, 70
 - Warnung bei, 969
 - (caption), 322
 - (float), **303**
 - (multicol), nicht unterstützt, 198
 - (rotfloat), 307
 - (subfig), 329
- figure Zähler, 883
- figure* Umgebung (multicol), 198
- `\figurename` (babel), 561
 - in diesem Buch, **14**
- `\figureplace` (endfloat), **299**
- `\figuresection` (endfloat), 298
- figuresfirst Option (endfloat), 299
- figuresleft Option (rotating), 306
- figuresright Option (rotating), **306**
- figwindow Umgebung (picipar), **116**
- `\filcenter`
 - (titlesec), **42**, **47**, **69**
 - (titletoc), 63
- `\file` (docstrip), **859**, **860**, **863**, **864**
 - filecontents Umgebung, **22**, **23**, **414**, **624**
 - Fehler bei, 942
 - Warnung bei, 964, 966, 972, 976
- filecontents* Umgebung, **23**, **180**
 - Warnung bei, 966, 976
- `\filedate` (doc), 855
- `\filename` (doc), 855
- `\fileversion` (doc), 855
- `\filinner` (titlesec), 42, 46
- `\fill` elastische Länge, 268, **269**, **882**, **888**, 889, 890
- `\fillast`
 - (titlesec), 42
 - (titletoc), 63, **69**
- fillcolor Schlüssel (fancyvrb), **166**
- `\filleft`
 - (titlesec), **42**, **43**, **46**
 - (titletoc), 63
- `\filltype` (eepic), 628
- `\filouter` (titlesec), 42, 46
- `\filright`
 - (titlesec), **42**, **45–47**
 - (titletoc), 63, **64**, **65**
- final Option
 - (graphics), 633
 - (graphicx), 633
 - (showkeys), 72
 - (varioref), 77
- finalcolumnbadness Zähler (multicol), 195–197
- `\Finale` (doc), 849, 854
- `\finalhyphendemerits`, **882**
- `\finallinebreak` (lb2), **109**
- Finalmodus, 72, 77, 633
- findent Schlüssel (lettrine), **108**
- finnish Option (babel), 559
- `\Finv` (amssymb), 544
- `\Fire` (ifsym), **416**
 - first Schlüsselwert (jurabib), **749**, **750**, **751**, 752, **753**, 755, **757**
- `\firstdashline` (arydshln), 275, 276
- `\firsthline` (array), 276, **288**, **289**
- `\firstleftmark` (extramarks), **228**, **237**
- `\firstleftxmark` (extramarks), **229**
- firstline Schlüssel
 - (fancyvrb), **170**, **172**
 - (listings), 181
- `\firstmark`, 226
 - firstnotreversed Schlüsselwert (jurabib), 749
- firstnumber Schlüssel
 - (fancyvrb), **168**, **172**
 - (listings), **181**
- `\firstrightmark` (extramarks), **228**, 239, **240**

- `\firstrightxmark` (extramarks), 229
- `fit` Option (truncate), 241
- `\FiveFlowerPetal` (bbding), 414
- `fix-cm` Paket, xxix, 366, 367, 368
- fixed Größenfunktion, 438
- `\Fixedbearing` (marvosym), 413
- `fixltx2e` Paket, 240, 353
- `fixltx2e.dtx` Datei (`fixltx2e`), 868
- `flafter` Paket, 74, 295
- `\Flag` (ifsym), 416
- `flalign` Umgebung (`amsmath`), 481, 488, 489
 - Fehler bei, 930
 - mit `\minalignsep` justieren, 489
- `flalign*` Umgebung (`amsmath`), 481
- `flanguage` $\LaTeX$ -Feld (jurabib), 768
- `\flat`, 544
- Flattersatz, Absätze im, 110, 111, 113, 114
- `fleqn` Option, 72
 - (`amsmath`), 479, 481, 482, 484, 512
- `float` Schlüssel (listings), 183
- `float` Paket, 300–304, 966
- `\FloatBarrier` (placeins), 297, 304
- `\floatdesign` (layouts), 210
- `\floatdiagram` (layouts), 210
- `floatfig` Paket, 308
- `floatflt` Paket, 308
- `\floatname`
 - (float), 301, 302
 - (rofloat), 307
- `\floatpagedesign` (layouts), 210
- `\floatpagediagram` (layouts), 210
- `\floatpagefraction`, 293, 294
- `\floatplacement` (float), 303
- `\floatsep` elastische Länge, 293
- `\floatstyle`
 - (float), 301, 302, 303, 318–320
 - (rofloat), 307
- `fltpage` Paket, 335, 336
- `\flushbottom`, 127, 242
- `\flushcolumns` (multicol), 196, 198
- `FlushLeft` Umgebung (`ragged2e`), 112
- `flushleft` Option
 - (paralist), 145
 - (threeparttable), 286, 287
- `flushleft` Umgebung, 110, 111, 154
 - (`ragged2e`), 112
- `flushmargin` Option (`footmisc`), 125, 126, 130, 756
- `FlushRight` Umgebung (`ragged2e`), 112
- `flushright` Umgebung, 111, 154, 890
 - (`ragged2e`), 112
- FML Fontkodierung (`fourier`), 402
- FMS Fontkodierung (`fourier`), 402
- `.fmt` Dateierweiterung, 9
- `fnchap` Paket, 36, 37, 38
- `fnclab` Paket, 75
- `fnpara` Paket, 125
- `\fnref` (lb2), 118
- `\fnsymbol`, 117, 885
 - Fehler bei, 932
 - (perpage), 128
- `fntguide.tex` Datei, 435
- Folio, 202, 213, *siehe auch* Kolumnentitel
- `\font`, 32, 88, 398, 439, 440, 441
- `font` Schlüssel/Option
 - (caption), 319, 326
 - (subfig), 326, 328, 329
- `font memory` Fehler, 958
- Fontbefehle
 - High-Level
 - Definition, 349
 - dicktengleiche Schriften, 350
 - für ausgewählte Wörter oder Phrasen, 349
 - für Hervorhebungen, 352
 - Gesamtgestaltung eines Dokumentes, 357, 358
 - Grundschrift, ändern, 357, 358
 - Grundschrift, Beschreibung, 349, 350
 - kombinieren, 354, 355
 - $\LaTeX$ 2.09, 353, 358, 361
 - Schreibmaschinenschriften, 350
 - Schriftgrößen einstellen, 353, 354
 - serifenlose Schriften, 350
 - Serifenschriften, 350
 - Sonderzeichen, 356, 357
 - Standardfamilien, 350
 - Standardformen, 351, 352, 353
 - Standardserien, 351
 - Text unterstreichen, 353
 - vs. Deklarationen, 355, 356
 - in Formeln, 363
 - Low-Level
 - automatische Fontersetzung, 429
 - Definition, 423, 424
 - einzelne Attribute setzen, 424–428
 - Fontkodierung, 427, 428
 - innerhalb eines Dokumentes, 430
 - mehrere Attribute setzen, 428
 - Schriftfamilien, 424
 - Schriftformen, 425
 - Schriftgrade, 426
 - Schriftserien, 425
- `fontdef.cfg` Datei, 443
- Fontdefinitionsdateien, 7, *siehe auch* `.fd`
- `\fontdimen`, 32, 88, 439–441, 449
- `\fontdimen1`, 439
- `\fontdimen2`, 32, 54, 439, 440, 441
- `\fontdimen3`, 32, 88, 440
- `\fontdimen4`, 32, 440
- `\fontdimen5`, 398, 440
- `\fontdimen6`, 440
- `\fontdimen7`, 440
- `fontenc` Paket, 7, 14, 164, 372, 373, 921
 - Ändern von `\encodingdefault`, 358
 - Fehler bei, 923, 933
- `\fontencoding`, 164, 357, 366, 378, 423, 424, 427, 430, 442, 467, 587
 - Fehler bei, 933
 - (array), erzeugt falsches Ergebnis, 253
- `\fontfamily`, 101, 366, 422, 423, 424, 428, 430
- `fontfamily` Schlüssel (`fancyvrb`), 163, 164, 176
- `fontinst` Paket, 94, 388, 431, 432, 449, 450, 1020

- Fontkodierung, **341**, 372, 373, 460–475, *siehe*
Ausgabekodierung
- Fontladeoptionen, 438–441
- `fontmath.ltx` Datei, 545
- Fonts, *siehe auch* Mathematikfonts; mathematische
Symbole; Text
ändern, *siehe* Fontbefehle
Akzentbuchstaben, 348, 369, **370**, 371, 372
Almost European Fonts, 338, 367
altdeutsche, **405**, **406**, 407
aufrechte, **344**, 351
automatische Ersetzung, 429
CM Bright, 396, **397**
CM-Super, 366, 367, 586
Computer Modern (CM)
 $\LaTeX$ -Standardfonts, 365, 366, **367**, 368
 Mediävalziffern, 394, **395**
Concrete, **395**, **396**
DC Fonts, 365
deklarieren, 432
dicktengleiche, 342, 343, 350, *siehe auch*
 wortwörtliche Ausgabe
EC Fonts, 365, 366, **367**
einbinden
 `.fd`-Dateien definieren, **449–451**
 Beispiel, **449–451**
 Deklarations-Reihenfolge, 451
 Dimensionsparameter, 439–441
 Familien deklarieren, 432
 Familien verändern, 441
 `fixed` Größenfunktion, 438
 Fontladeoptionen, 438–441
 `gen` Größenfunktion, 436
 `genb` Größenfunktion, 436
 Größenbereiche, 434
 Größenfunktionen, 434–438
 interne Dateistruktur, 443, 444
 Kodierschemata deklarieren, 442
 „leere“ Größenfunktion, 435
 Nomenklatur, 431
 `s` Größenfunktion, 436
 Schriftformgruppen, 432–441
 Schriftgrade, 434, 444
 `sfixed` Größenfunktion, 438
 `sgen` Größenfunktion, 436
 `sgenb` Größenfunktion, 437
 Silbentrennzeichen, 439
 `ssub` Größenfunktion, 437
 `ssubf` Größenfunktion, 438
 `sub` Größenfunktion, 437
 `subf` Größenfunktion, 438
 Symbolfonts, 445–449
 Übersicht, 430, 431
 zur Verwendung in Formeln, 444–449
einzelne Attribute setzen, 424–428
European Modern, 350, 366
EZ Fonts, 367
für ein Dokument definieren, *siehe* Fontbefehle,
 High-Level
für Zeilennummern, 188, **189**
- Fonts (*Forts.*)
- Familien
 deklarieren, 432
 Klassifizierung, 384
 Kodierungen, 347, 348
 Low-Level-Befehle, 424
 Schriftformen, 343–346
 Schriftgrade, 346, 347
 verändern, 441
Fontauswahl überwachen, 379
Fourier-GUTenberg, 402, **403**, **404**
Fraktur, 405–407
geneigte, 344, 351
Glyphen, 340
Gothic, 405–407
Größenbereiche, 434
Größenfunktionen, 434–438
große Operatoren skalieren, 379
Grundschrift
 ändern, 357, 358
 Beschreibung, 349, 350
hervorheben, 352
in Bibliographien, 762, 763
in der Präambel definieren, *siehe* Fontbefehle,
 Low-Level
in diesem Buch, 1135
in einem Paket definieren, *siehe* Fontbefehle,
 Low-Level
in Legenden von Gleitobjekten, 318, **319**
in Tabellen, **252**, 253
in wortwörtlicher Ausgabe, 163, **164**
Kapitalchen, **344**, **345**, 352, **579**
Klassifizierung, 384
Kodierung, *siehe* Kodierung
kursiv, 344
Kursivkorrektur, 351, **352**, **353**
 $\LaTeX$ 2.09, 358
Latin Modern Fonts, **368**
Low-Level-Schnittstelle, *siehe* Fontbefehle,
 Low-Level
- Mathematik-
 Alphabetbefehle, **360–362**
 automatischer Wechsel, 359, 360
 Baskerville Math, **536**
 Bitstream Charter Math, **536**
 CM Bright, **538**
 Computer Modern (CM), **528**
 Concrete, **530**
 Euler, 407, **408–410**, **530**
 Fontbefehle, **363**
 Formellayouts, **363**, 364
 Fourier-GUTenberg, 402, **403**, **404**, **531**
 große Operatoren skalieren, 379
 Helvetica Math, **538**
 Info Math, **539**
 Lucida Math, **537**
 Palatino, **389**, **390**, 401, **402**, **534**, **535**
 Pazo, **534**
 Pi, 390, 391, **392**, 393
 PX Fonts, **534**
 Times Roman, 387, **388**, 389, **399–401**, **532**

Fonts (*Forts.*)

- TM Math, [533](#)
- TX Fonts, [532](#)
- mehrere Attribute setzen, 428
- METAFONT, 345
- mit Serifen, 343, 350
- NFSS, 338, 339, *siehe auch* PSNFSS
- normale, 349
- oblique, 344
- Outline-, 345
- PDF-Dokumente durchsuchen, 367, [368](#)
- Pi, [392](#)
- Pica-Punkte, 346
- PostScript Fonts, *siehe auch* PSNFSS
- PostScript-Fonts, 366
- proportionale, 342, 343
- Rückwärtskompatibilität, 475, 476
- schattierte, 345
- Schmuckinitialen, 406, [407](#)
- schräg gestellte, 344
- Schreibmaschinenschrift, 350, [398](#), [399](#), 867
- Schriftbreite, 345, 346
- Schriftformen, [343–345](#), 351, [352](#), [353](#), 425
- Schriftformgruppen, 432–441
- Schriftgrad, [353](#)
 - Beschreibung, 346, 347
 - einstellen, 434, 444
 - Fußnoten, 119
 - Low-Level-Befehle, 426
 - messen, 346, 347
 - Standardgrößen, 353, 354
- Schriftstärke, 345, 346
- Schwabacher, 405–407
- Serien, 351, 425
- serifenlose, 343, 350
- skalieren, relativ zum Original, [90](#), [91](#)
- Sonderzeichen, 356, [357](#), *siehe auch* Textsymbole
- Symbole, *siehe* Textsymbole; mathematische Symbole
- Text unterstreichen, 353
- Textkörper, 349
- Textsymbole, *siehe* Textsymbole
- theoremähnliche Strukturen, 149
- Trennzeichen, 438
- unnötiges Laden von, 354
- URW Antiqua, 404, 405
- URW Grotesk, 404, 405
- verändern, 441
- Weißraum, 351, [352](#), [353](#)
- Zeichensatztabellen anzeigen, 380, 381
- `\fontseries`, 164, 351, 423, [424](#), [425](#), [430](#)
- `fontseries` Schlüssel (`fancyvrb`), 164
- `\fontshape`, 164, [422](#), 423, [424](#), [425](#), [430](#)
- `fontshape` Schlüssel (`fancyvrb`), [164](#)
- `\fontsize`, [43](#), 90, 354, 366, [383](#), 385, [419](#), 423, [424](#), [426](#), [429](#), [430](#), [476](#), 962
- `fontsize` Schlüssel (`fancyvrb`), [164](#), [175](#), [176](#)
- `fonttext.cfg` Datei, 861
- `fonttext.ltx` Datei, [443](#), 444
- `\Football` (`marvosym`), [413](#)
- `\footcite` (`jurabib`), 752, 753
- `\footcitet` (`jurabib`), 758
- `\footcitetitle` (`jurabib`), 752
- `footexclude` Option (`typearea`), 213
- `\footfullcite` (`jurabib`), [752](#), 757
- `footinclude` Option (`typearea`), 213
- `footmisc` Paket, [xxix](#), [121–127](#), 129, 130
- `\footnote`, [117](#), [118](#), 120, 129, 130, [285](#)
 - mit Randausgleich, 111
 - Querverweis auf, 70
 - Stilparameter, [119–121](#) (`babel`), [582](#) (`fancyvrb`), [176](#) (`footmisc`), [118](#), [125](#), [126](#)
 - als Marginalie formatiert, [125](#), [126](#), [128](#)
 - eingebettet formatiert, [125–127](#)
 - nummeriert mit Sternen, [124](#)
 - Probleme mit nachfolgenden, [127](#)
 - (`longtable`), 270
 - (`manyfoot`), [130–132](#)
 - (`multicol`), 198
 - (`perpage`), seitenweise nummeriert, [128](#)
 - (`supertabular`), 264
- `footnote` Zähler, 117, 128, 883, 884, [980](#) (`longtable`), 270
- `\Footnote{suffix}` (`manyfoot`), 129, [130–132](#)
- `\footnote{suffix}` (`manyfoot`), 129
- `footnote{suffix}` Zähler (`manyfoot`), 129
- `\footnotedesign` (`layouts`), 210
- `\footnotediagram` (`layouts`), 210
- `\footnotemargin` feste Länge (`footmisc`), 125, [126](#)
- `\footnotemark`, [117](#), [118](#), 129, [285](#)
- `\Footnotemark{suffix}` (`manyfoot`), 129
- `\footnotemark{suffix}` (`manyfoot`), 129
- `\footnoterule`, [119](#), 120, 126, 131, 293 (`manyfoot`), 131
- `\footnotesep` feste Länge, 119, 120
- `footnotesep` Schlüssel/Option (`geometry`), 215
- `\footnotesize`, 119, 133, [154](#), [353](#), 354, 383
- `footnotesize` Schlüsselwert (`caption`), 319 (`subfig`), 326
- `\footnotetext`, [117](#), [118](#), 129
- `\Footnotetext{suffix}` (`manyfoot`), 129, 131
- `\footnotetext{suffix}` (`manyfoot`), 129
- `footnpag` Paket, 122
- `\footrule` (`fancyhdr`), 233
- `\footrulewidth` (`fancyhdr`), [232](#), [233](#), [234](#), [236](#)
- `\footskip` elastische Länge, 203, 205
- `footskip` Schlüssel/Option (`geometry`), 218
- `\forall`, [514](#), [523](#), [544](#)
- `force` Option (`textcomp`), 375
- `\forcefootnotes` (`camel`), [770](#)
- `\foreignlanguage` (`babel`), [561](#), [562](#), [563](#), [571–573](#), [576](#), [579](#)
- `\form` (`euro`), [105](#), 106
- formale Linien, 277, [278](#), 279, [280](#)
- `format` Schlüssel/Option (`caption`), [318](#) (`subfig`), [326](#), [328](#)
- `format.ins` Datei, 861, 863

- `format.name$` interne $\text{\LaTeX}$ -Funktion, 839
- `formatcom` Schlüssel (`fancyvrb`), 164, 172
- Formatdateien, $\text{\LaTeX}$, 7
- formatieren
 - E-Mail-Adressen, 99, 100, 101, 102, 103
 - Euro-Währung, 103–106
 - Formeln, *siehe* Mathematikfont; mathematische Symbole; Mathematiksatz
 - Inhaltsverzeichnisse, 52, 53, 54, 55
 - Pfade, 99, 100, 101, 102, 103
 - URLs, 99, 100, 101, 102, 103
 - Verzeichnisnamen, 99, 100, 101, 102, 103
 - Währungen, 103–106
- Formatierungsparameter, *siehe* Layoutparameter
- Formelgruppen
 - ausgerichtet, 487
 - nicht ausgerichtet, 487
- Formeln, *siehe auch* Mathematikfont; mathematische Symbole; Mathematiksatz
 - abgesetzte Formeln unterbrechen, 491
 - als Miniseiten, 489, 490, 491
 - `amsmath`-Paket vs. Standard- $\text{\LaTeX}$, 482, 483
 - ausrichten, 481, 482
 - einzeilige, 483
 - formatieren, *siehe* Mathematikfont; mathematische Symbole; Mathematiksatz
 - Labels, *siehe* Formeln, Nummerierung; Tags
 - Leerzeilen, 492, 493
 - mehrfache Ausrichtung, 487, 488, 489
 - mehrzeilige
 - ausgerichtet, 485, 486
 - nicht ausgerichtet, 483, 484, 485
 - Nummerierung, *siehe auch* Formeln, Tags
 - Untersequenzen, 497
 - Zähler zurücksetzen, 497
 - Seitenumbrüche, 491–494
 - Tags, 481, 482, *siehe auch* Formeln, Nummerierung
 - Definition, 481
 - Formeln nummerieren, 494, 495
 - Platzierung, 495, 496, 497
 - vertikale Abstände, 491, 492, 493, 494
- Formen für Überschriften, 40
- Formen, Schrift-, *siehe* Fonts, Schriftformen
- `\ForwardToIndex` (`marvosym`), 413
- `founder` $\text{\LaTeX}$ -Feld (`jurabib`), 768
- `\foundername` (`jurabib`), 768
- `fourier` Paket, xxix, 382, 402–404, 531
- Fourier-GUTenberg Font, 402, 403, 404, 531
- `\FourStar` (`bbding`), 414
- `fp` Paket, 103
- `FPfigure` Umgebung (`fltpage`), 335, 336
- `FPtable` Umgebung (`fltpage`), 335
- `\frac`, 486, 505, 506, 516, 519
- `\fracwithdelims` (`amsmath`), 479
- Fragezeichen (?), Kurzzeichen, 571
- `\frail` (`lb2`), 927
- `\frakdefault` (`yfonts`), 407
- `\frakfamily` (`yfonts`), 405, 406, 407
- `\fraklines` (`yfonts`), 406, 407
- Fraktur Font, 405–407
 - Laufweite, 407
- `\frame`, 423
 - `frame` Option (`crop`), 221, 223
 - `frame` Schlüssel
 - (`fancyvrb`), 165, 166, 167, 174
 - (listings), 182, 183, 184
 - (`titlesec`), 40, 41, 42, 69
- `\framebox`, 336, 893, 894, 899
 - (`pspicture`), 658
- `frameround` Schlüssel (listings), 183
- `framerule` Schlüssel
 - (`fancyvrb`), 166, 174
 - (listings), 183, 184
- `framesep` Schlüssel
 - (`fancyvrb`), 166, 167
 - (listings), 183, 184
- `francais` Option (`babel`), 559
- Französisch, 571, 576, 581, 606
 - Anführungszeichen, 569
 - Layoutvorlage, 581
 - Namen, 579
 - Wörter, Index-Sortierreihenfolge, 692
- fremdsprachige Dokumente, *siehe auch* `babel`-Paket
 - ! (Ausrufezeichen), Kurzzeichen, 571
 - " (Anführungszeichen), Kurzzeichen, 568–570
 - ˆ (Akut-Akzent), Kurzzeichen, 572
 - . (Punkt), Kurzzeichen, 574
 - : (Doppelpunkt), Kurzzeichen, 571
 - ; (Semikolon), Kurzzeichen, 571
 - < (Kleinerzeichen), Kurzzeichen, 574
 - = (Gleichheitszeichen), Kurzzeichen, 573
 - > (Größerzeichen), Kurzzeichen, 574
 - ? (Fragezeichen), Kurzzeichen, 571
 - ^ (Zirkumflex), Kurzzeichen, 573
 - ` (Gravis-Akzent), Kurzzeichen, 571, 572
 - ~ (Tilde)
 - geschütztes Leerzeichen, 568
 - in verschiedenen Sprachen, 570, 571
- Abstände nach Satzzeichen, 580
- aktuelle Sprache einstellen/abfragen, 560, 561, 562, 563
- Anführungszeichen einfügen, 569, 570
- bestimmte Artikel, Ungarisch, 579
- Bibliographien, Sprachunterstützung, 759, 760, 761, 842–844
- $\text{\LaTeX}$, 589
- Buchstaben mit Akzent, 568
- Datum formatieren, 574–576
- Einzug nach einer Überschrift, 581
- $\text{\LaTeX}$, $\text{\TeX}$ -Erweiterung, 556, 582
- Fontkodierung
 - griechisches Alphabet, 590, 591, 593
 - hebräisches Alphabet, 591–593, 594, 595
 - kyrillisches Alphabet, 585–589
 - Optionen (Sprachauswahl), 582–585
- französische Namen, 579
- Fußnoten, 582
- Hcaption Font, 593
- Hclassic Font, 593
- Index, 688, 691–693

fremdsprachige Dokumente (*Forts.*)

Interpunktion, Sonderfälle, [607](#)
 Konzepte/Übersicht, 555–557
 kulturspezifische typographische Regeln, 558
 Kurzformen, 563, [564](#), [565](#)
 Listen, [581](#), 582
MakeIndex, 589
 mathematische Operatoren, [580](#)
 nicht lateinische Alphabete
 Äthiopisch, 608
 Arabisch, 608
 Armenisch, 608
 Chinesisch, 608
 Griechisch, 590, 591, [592](#), [593](#)
 Hebräisch, [591–593](#), [594](#), [595](#), 608
 Indisch, 608
 Japanisch, 608
 Koreanisch, 608
 Kyrillisch, 585–587, [588](#), 589, 590
 Mandschu (Mongolisch), 608
 Nummerierung, [576–579](#), 580
 Omega, $\TeX$ -Erweiterung, 556, 586, 608, 609, 655
 OT1-Erweiterungen, 583
 Polnisch, [583](#)
 Quellenverweise, Ungarisch, 580
 Schreibrichtung, 582, [594](#)
 Shalom-Fonts, 593
 Silbentrennung
 dynamisch festlegen, 558
 in mehreren Sprachen, 562, [596–598](#)
 Italienisch, 579
 kulturelle Gesichtspunkte, 558
 Niederländisch, [568](#)
 Sonderregelungen, [569](#), [570](#)
 sprachliche Gesichtspunkte, 557
 Trennmuster anpassen, 602, 603
 Trennmuster anwenden, [562](#)
 verhindern, [562](#)
 Sonderzeichen, [569](#)
 Sprachattribute, 565, [566](#)
 Sprache und Formatierung, 557, 558
 sprachspezifische Zeichenfolgen, 558, [561](#), 565,
 [566](#), [567](#), 603
 T1-Erweiterungen, 583
 Übersichtstabelle, 558, 559
 Zeichensätze, 557

french Option (*babel*), 19, [106–108](#), [559](#), [562](#), [565](#),
 [569](#), [571](#), [576](#), [579](#), [581](#), [582](#)

french Paket, 608, 1019

frenchb Option (*babel*), 559

frenchb.cfg Datei (*babel*), 605, 606

frenchb.ldf Datei (*babel*), 564, 565, **607**

$\backslash$ **FrenchFootnotes** (*babel*), [582](#)

$\backslash$ **FrenchLayout** (*babel*), 581

$\backslash$ **frenchspacing**, 581

 Frequently Asked Questions (FAQ), 995

$\backslash$ **from** (*docstrip*), 859, [860](#), [863](#), [864](#), 866

$\backslash$ **frontmatter**, 24, 224

$\backslash$ **frown**, 551

$\backslash$ **Frown** (*marvosym*), [413](#)

fnright Paket, 121, 185

FTP-Server

Download-Befehle, 1000–1002

Liste der, 996

Fußnoten, *siehe auch* Endnoten; Marginalien

 Absatzformat, 124, [125](#)

 Abstand vom Text, 119

 anpassen, 119–121

 Fehlersuche, 990, 991

 fremdsprachige Dokumente, [582](#)

 Haupttext vs. *minipage*-Umgebung, [117](#), [118](#),
 119–121

 hochgestellte Markierungen, [120](#), [121](#)

 im Rand, [125](#), [126](#)

 in mehrseitigen Tabellen, 270

 in Tabellen, 270, [285](#), [286](#), 287

 Layoutparameter, 120

 Linien, 119, [126](#), 127

 mehrere an derselben Stelle, [127](#)

 Nummerierung, 119, 122, [123](#), 129, [130–132](#)

 schematische Darstellung, 120

 Schriftgrad, 119

 Seitenlayout, 215

 Spalten, [121](#), [192](#), 198

 Standard, [117](#), [118](#), 119–121

 Stile, 121, [122](#), 123–127

 Textsymbole für, 123, [124](#)

 unabhängige, 129, [130–132](#)

 vertikale Abstände, 119

 wortwörtliche Ausgabe in, [176](#)

 Zähler, je Seite zurücksetzen, 127, [128](#)

 zweispaltige Umgebung, 121, [122](#)

Fußnotenverweise, Bibliographien, [752](#), [753](#)

Fußsteg (unterer Rand), 201, 202, 212, 216, 219

Fußzeilen, *siehe* Kolumnentitel

Fußzeilenhöhe, 209

full Option

 (*textcomp*), [374](#), [375](#), [396](#), [399](#), [402](#)
 (*trace*), 993

$\backslash$ **fullcite** (*jurabib*), [749](#), [755](#), 757

$\backslash$ **fulloldstyle** Option (*fourier*), 404

$\backslash$ **fullref**

 (*lb2*), [72](#)
 (*varioref*), 79

FUNCTION $\BbbTeX$ -Befehl, 836, [838](#), [840–844](#)

Funktionsnamen, *siehe* Operatoren; Operatornamen

$\backslash$ **fussy**, 110

fvr-b-ex Paket, 172

$\backslash$ **fvset** (*fancyvrb*), [173](#), [177](#), 178

G

galician Option (*babel*), 559, [572](#)

$\backslash$ **Game** (*amssymb*), 544

$\backslash$ **Gamma**, [509](#), [543](#)

$\backslash$ **gamma**, 543

gather Option (*chapterbib*), 773, 774, [775](#)

gather Umgebung (*amsmath*), 481, 485, [487](#), [496](#),
 [500](#), [512](#)

 Fehler bei, 930

gather* Umgebung (*amsmath*), 481, 485, 487, [499](#),
 [505](#), [514](#)

- gathered Umgebung (amsmath), 481, 489, 490
 Fehler bei, 930, 933
- \GBP (lb2), 104
- \gcd, 513
- \ge, 514, 548
 gedrehtes Material verbergen, 634
 gekapselte Seitenzahlen, Index, 672, 695
 Geldsymbole, *siehe* Währungen, formatieren
 gemeine Ziffern, *siehe* Mediävalziffern
- gen Option (eurosym), 420
- gen Größenfunktion, 436
- genb Größenfunktion, 436
- gender BibTeX-Feld (jurabib), 714, 760, 768
- geneigte Fonts, 344, 351
- \generalname elastische Länge (doc), 856
- \generate (docstrip), 859, 860, 863, 864
- \geneuro (eurosym), 420
- \geneuronarrow (eurosym), 420
- \geneurowide (eurosym), 420
- \genfrac (amsmath), 506
- gennarrow Option (eurosym), 420
- genwide Option (eurosym), 420
- \geometry (geometry), 219, 220
 geometry Option (ifsym), 416
 geometry Paket, xxix, 209, 214–220
 in diesem Buch, 1137
 mit calc kombiniert, 218
- gepunktete Linien, 620, 889
- \geq, 514, 548
 (eulervm), 410
- \geqq (amssymb), 548
- \geqslant (amssymb), 548
- Gerätetreiber, 632, 633
- geräteunabhängige Dateien, 8
- german Option
 (babel), 20, 559, 569, 570, 678, 684, 694
 (biblist), 804
 (varioref), 20
- german.ldf Datei (babel), 565, 601
- germanb Option (babel), 559
- \germanhyphenmins (babel), 603
- geschützte Bindestriche (–), 89, 90, 100
- Geschlechtsangabe, Bibliographien, 760, 761, 768
- geschweifte Klammern weglassen, 876
- gestrichelte Linien
 arydshin-Paket, 275, 276
 \dashline-Befehl, 621
- \gets, 550
- Geviert, 41
- \gg, 548
- \ggg (amssymb), 548
- \gggtr (amssymb), 548
- gglo.ist Datei (doc), 859
- ghostscript Programm, 381, 653, 661, 1018
- ghostview Programm, 222, 646, 653
- .gif Dateierweiterung, 9, 663, 931
- \gimel (amssymb), 544
- gind.ist Datei (doc), 859
- Gleichheits- und Relationszeichen – negierte, mathematische Symbole, 548
- Gleichheits- und Relationszeichen, mathematische Symbole, 548
- Gleichheitszeichen (=), Kurzzeichen, 573
- Gleichungen, *siehe* Formeln
- Gleitobjekte
 Abbildungen, 325–331
 angepasste Layouts, 300, 301, 302–304, 305
 Bilder im Text, 311, 312–315
 Definition, 291, 292
 drehen, 305, 306, 307
 falsche Verweise, 70, 71
 Gleitobjektseite, Kolumnentitel-Layout, 239
 höchste zulässige Anzahl festlegen, 292, 293
 halbleere Seiten/Spalten, 294
 in Fließtext eingebettete, 307–315
 Layoutparameter, 292–294
- Legenden
 über mehrere Gleitobjekte, 324
 Absatzdurchschuss, 320, 321
 Abstände, 321, 326
 anpassen, 314, 317–324
 auf separater Seite, 335, 336
 für bestimmte Gleitobjektarten, 314, 321, 322
 Fonts, 318, 319
 Form, 317, 318
 gruppenweise nummeriert, 331, 332
 Labels, 319, 320, 322–324
 mehrere Gleitobjekte, 324
 mehrseitige Tabellen, 265, 270
 Platzierung, 333, 334, 335, 336
 Randausgleich in, 320, 321
 seitliche, 315, 333, 334, 335
 Standard-L^AT_EX, 315, 316, 317
 untergeordnete, 325–329, 330, 331
- Linien, 293, 294
- Markierungen im Text für, 299, 300
- mehrseitige Tabellen, 270–272, 298
- nicht gleitende Tabellen und Abbildungen, 304, 305
- nicht positionierte ausgeben, 297, 298
- Parameter, 292–295
- Positionierung, 294–300
 am Ende eines Dokumentes, 297, 298–300
 an Begrenzungen gebunden, 297
 an einem bestimmten Punkt, 295, 304, 305
 gebündelt am Ende eines Kapitels, 294
 nach ihrem Textverweis, 295
 von Legenden, 333, 334–336
 zurückgleiten, 295
- Seitenanteil einstellen, 292, 293
- Spalten, 198
- Tabellen, 325–331
- Text um eine Abbildung fließen lassen, 115, 116
- Trennelemente, 293, 294
- unterteilte Abbildungen, 325, 326–328, 329, 330, 331
- unterteilte Tabellen, 325, 326, 327, 328, 329–331
- vertikale Abstände, 293
- von Text umflossen, 307, 308, 309, 310, 311

- Gleitobjekte (*Forts.*)
 - vorgezogene Ausgabe, 299
 - wortwörtliche Ausgabe als, 183
- Gleitobjektklassen
 - benennen, 301, [302](#)
 - Legenden auflisten, 302
- Gleitobjektseiten, Kolumnentitel-Layout, [239](#)
- Glenn Option (fncychap), 36
- Gliederungsbefehle, 24–26
 - neu definieren, [31](#), [32](#), 33
 - Standardverhalten, [34](#)
- Gliederungsebenen
 - Inhaltsverzeichnisse, 54
 - Überschriften, 26, 30
- .glo Dateierweiterung, 673
 - (doc), 869
- \global, [274](#)
 - global.max\$ interne $\TeX$ -Funktion, [844](#)
 - globalcitecopy Option (bibunits), 777
 - globale Optionen, 19, 914–916, 919
 - globale Variablen, Bibliographien, 836
 - Glossareinträge, *MakeIndex*-Verarbeitung, 673
- \glossary, 673
 - (doc), 849
- \glossaryentry, [673](#)
- \GlossaryMin feste Länge (doc), 856
- \glossaryname (babel), 561
- \GlossaryParms (doc), 856
- \GlossaryPrologue (doc), 856
- \glue, 981, [982](#), 983
 - Glyphen, 340, *siehe auch* Sonderzeichen; Textsymbole
- \gnapprox (amssymb), 548
- .gnd Dateierweiterung, 9
- \gneq (amssymb), 548
- \gneqq (amssymb), 548
- \gnsim (amssymb), 548
- gobble Schlüssel
 - (fancyvrb), [165](#), [173](#)
 - (listings), [181](#), [184](#)
- \gothfamily (yfonts), [405](#), [406](#)
- Gothic Font, 405–407
- gpik Programm, 626
- Größen einstellen
 - Fonts, [353](#), 354
 - Mathematiksatz, 514, 515, [516](#)
- Größerzeichen (>), Kurzzeichen, [574](#)
- Gründerdaten, Bibliographien, 768
- Graphen
 - Balkendiagramme, [631](#)
 - beschriftete Achsen, 624, 625
 - erstellen, 622–624
 - externe Daten laden, 623, 624
 - Kurven- und Linienarten kombinieren, [623](#)
- graphics Paket, 305, [631–636](#), 638, [642–649](#), 1001, 1018
 - Fehler bei, 923, 931, 933, 945, 948, 949
 - von lscape geladen, 220
- graphics.cfg Datei (graphics), 632, 633
- \graphicspath
 - (graphics), [642](#), 960
 - (graphicx), [642](#), 960
- graphicx Paket, 631–634, 636–642, 649–651
 - Fehler bei, 923, 931, 933, 945, 948, 949
- Graphiken, geräteabhängige Unterstützung
 - Bilddateien einbinden
 - Bildgröße, 638, 644
 - Dateiarten, 643
 - Dateierweiterungen, angeben, 643
 - Dateierweiterungen, Suchreihenfolge, 642, 643
 - Dateinamen parsen, 638
 - drehen, 638
 - \includegraphics (graphics), [634–636](#)
 - \includegraphics (graphicx), 636–638, [639–641](#)
 - Kapselung, 645, 646
 - skalieren, 638
 - Speicherort, 642
 - Standardschlüsselwerte, 641, 642
 - vollständige Dateinamen erzwingen, 643
 - Bounding Box-Kommentare, 633, 634
 - Entwurfsmodus, 633
 - Finalmodus, 633
 - gedrehtes Material verbergen, 634
 - Gerätetreiber, 632, 633
 - skaliertes Material verbergen, 634
- graphische Objekte, *siehe auch die einzelnen Graphikarten*
 - drehen, [648–652](#)
 - skalieren, [646–648](#)
- graphisches Frontend, Bibliographien, 813–817
- graphpap Paket, 658
- \graphpaper (graphpap), [658–660](#)
- \grave, 545
- Gravis-Akzent (`), Kurzzeichen, [571](#), [572](#)
- greek Option (babel), 559, [566](#), [575](#), [578](#), [579](#), 584, [585](#), [591](#)
 - greek.ldf Datei (babel), 602
- \greekencoding (babel), 584
- \Greeknatural (babel), [578](#)
- \greeknumeral (babel), [578](#)
- \greektext (babel), 584
- grey Option (changebar), 200
- \grid (epic), 624, [625](#)
- Griechisch, [543](#), 570, 575, 577, 590, 591, [592](#), [593](#)
- grmath Paket, 580
- Groß-/Kleinschreibung, Unterscheidung von
 - in Bibliographien, 790
 - in Indexen, 670
- Großbuchstaben
 - Initiale, [106–108](#), 406, [407](#)
 - Kapitälchen
 - Beschreibung, 344, 345
 - französische Namen, [579](#)
 - in Überschriften, 352
 - Überschriften, [28](#)
- group_skip Schlüsselwort (makeindex), 682
- grouping levels Fehler, 958, 959
- \Grtoday (babel), [575](#)
- Grundschrift, 206, 349, 350, 357
 - in diesem Buch, 1135
 - serifenlose Schrift wählen, 385

- `\gtrapprox` (amssymb), 548
- `\gtrdot` (amssymb), 546
- `\gtreqless` (amssymb), 548
- `\gtreqqlless` (amssymb), 548
- `\gtrless` (amssymb), 548
- `\gtrsim` (amssymb), 548
- Guillemets, 569, [574](#)
- `\guillemotleft`, 471
- `\guillemotright`, 471
- `\guilsinglleft`, 471
- `\guilsinglright`, 471
- gunzip Programm, 644
- `\gvertneqq` (amssymb), 548
- H**
- H Syntax
 - (fancyhdr), 234, 235
 - (float), [302](#), [303](#), [304](#)
- `\H`, 469
- Hände, Textsymbole, [411](#)
- Höhe, *siehe* Längenparameter
- Höhe einer Box, 650
- hang Schlüssel (titlesec), 40, 41, 42, 44
- hang Schlüsselwert
 - (caption), [318](#)
 - (subfig), [326](#), [328](#)
- `\hangindent`, [702](#), [703](#)
- hartnäckige Fehler, 927
- harvard Bib_{La}T_EX-Stil (harvard), 842
- harvard Paket, 72, [724](#), [725](#), [729](#), [821](#), [832](#)
- Harvard Zitierschema, 708, 713, *siehe auch* Autor-Jahr-Zitierschema
- `\harvarditem` (harvard), 725
- hash size Fehler, 959
- `\hat`, [507](#), [527](#), [545](#)
 - (eulervm), [410](#)
- Hauptteil (Paketdateien), 916
- `\hbadness`, 967, [973](#), [974](#)
- `\hbar`
 - (amssymb), 544
 - (eulervm), [409](#)
 - (euler), [409](#)
- `\hbox`, 875, 893, 903, [982](#)
 - in T_EX-Warnungen, 967, 969, 972, 973
 - Probleme mit, [903](#)
- Hcaption Font, 593
- Hclassic Font, 593
- `\hdashline` (arydshln), 275, [276](#)
- `\hdots` (amsmath), 552
- `\hdotsfor` (amsmath), [499](#)
- `\headcmd`, 34
 - (lb2), [34](#)
- `\headcmd*`, 34
- headexclude Option (typearea), 213
- `\headheight` feste Länge, 203, [205](#), [206](#), [905](#)
 - (fancyhdr), [233](#)
- headheight Schlüssel/Option (geometry), [214](#), [218](#)
- headinclude Option (typearea), [213](#), 215
- heading_prefix Schlüsselwort (makeindex), 682, [684](#)
- heading_suffix Schlüsselwort (makeindex), 682, [684](#)
- `\headingdesign` (layouts), 210
- `\headingdiagram` (layouts), 210
- headings Kolummentitel-Layout, 230, 231, [243](#), [244](#), [616](#)
- headings_flag Schlüsselwort (makeindex), 682, [684](#)
- `\headrule` (fancyhdr), [233](#), [235](#)
- `\headrulewidth` (fancyhdr), [232](#), [233](#), [234](#), [236](#), [238](#), [239](#)
- heads Option (endfloat), 299
- `\headsep` feste Länge, 203, [205](#), [206](#), 208, [905](#), 981
- headsep Schlüssel/Option (geometry), 218
- `\headtoname` (babel), 561
- `\headwidth` feste Länge (fancyhdr), [235](#), [241](#)
- `\heartsuit`, 544
- heavycircles Option, 547
 - (stmaryrd), 547
- `\heavyrulewidth` feste Länge (booktabs), 278
- hebcap Paket, 575
- `\hebdate` (babel), [575](#)
- `\hebday` (babel), [575](#)
 - hebfont Paket, 594
- Hebräisch, 591–593, [594](#), [595](#), 608
- `\hebrew` (babel), [575](#)
 - hebrew Option (babel), 559, [585](#)
- `\hebrewencoding` (babel), 583, 584
- `\Hebrewtoday` (hebcap), [575](#)
- `\height`, 894, [895](#), [899](#)
 - (graphics), [648](#)
- height Option (crop), 221
- height Schlüssel (graphicx), [116](#), 637, 639, [640](#), [641](#)
 - Fehler bei, 933
- height Schlüssel/Option (geometry), 216, 219
- height Syntax, [235](#), [900](#), [901](#)
- heightrounded Schlüssel/Option (geometry), 216
- `\help` (nfssfont.tex), 381
- helvet Paket, 14, [382](#), 385, 436
- helvetica Schlüsselwert (fancyvrb), 163, [164](#)
- Helvetica Font, 382, 386, [538](#)
 - in Formeln und Text, [538](#)
- Herausgeberdaten, Bibliographien, 768
- here Paket, 303
- Hervorheben von Text, 352, *siehe auch* kursive Schriftform; unterstreichen
- hetarom Paket, 631
- `\hfil`, [156](#), [232](#), [883](#)
- `\hfill`, [158](#), [889](#), [894](#), [896](#)
- `\hfuzz` feste Länge, 985
- hfuzz Schlüssel (fancyvrb), 165
- `\hhlne` (hhlne), [274](#), [275](#)
- hhlne Paket, 274, 275
- hiderotate Option
 - (graphics), 634
 - (graphicx), 634
- hidescale Option
 - (graphics), 634
 - (graphicx), 634
- Hilfsdateien, 8
- Hilfsquellen
 - CTAN
 - CD/DVD, 997
 - FTP-Befehle, 1000–1002
 - FTP-Server, Liste der, 996

- Hilfsquellen (*Forts.*)
- Inhalt, 996
 - Internetzugang, 998
 - DANTE FAQ, 995
 - FAQs, 995
 - FTP-Server
 - Download-Befehle, 1000–1002
 - Liste der, 996
 - Newsgroups, 996
 - Pakete
 - Beschreibungen, Online-Katalog, 998
 - Dokumentation suchen, 1002–1004
 - Programmdateien, Quellen
 - CD/DVD, 996, 997
 - FTP, 996, 1000–1002
 - Internetzugriff, 998
 - texdoc-Programm, 1002
 - texdoctk-Programm, 1003, 1004
 - TUG Homepage, 996
 - UK-TUG FAQ, 995
 - Usergroups, 1004–1006
- Hintergrund einfärben, 165, [166](#)
- Hinweise, *siehe* Fehlersuche
- hiresbb Option
 - (graphics), 633
 - (graphicx), 633
- hiresbb Schlüssel (graphicx), 637
- \hl (soul), [95](#), [99](#)
- \hline, 251, [274](#), 275, [276](#), [280–282](#), [284](#), [290](#)
 - Ausrichtungsprobleme bei Verwendung, 288
 - farbig, [273](#)
 - Fehler bei, 942
 - (array), [252–255](#), [257](#), [258](#), 289
 - (booktabs), 278
 - (hhline), 274
 - (supertabular), 265
 - (tabs), [277](#)
- hmargin Schlüssel/Option (geometry), 219
- hmarginratio Schlüssel/Option (geometry), 216, [217](#), 219
- \Hmjd (lb2), 521
- hmode boolesche Var., 908
- Hochstellungen
 - über Relationssymbolen (Mathematiksatz), 507
 - Fußnotenmarkierungen, [120](#), [121](#)
 - Kurztitel-Zitierschema, [761](#), 769
 - numerisches Zitierschema, 720, [721](#), 722
 - Positionieren von Grenzen, [503](#), [504](#)
- \hodiau (babel), [575](#)
- \hodiaun (babel), [575](#)
- \hoffset feste Länge, 204, 211, 218
- hoffset Schlüssel/Option (geometry), 218
- \hom, 513
- \hookleftarrow, 550, [551](#)
- \hookrightarrow, 550
- horizontal erweiterbare, mathematische Symbole, 509, [510](#), [512](#)
- howcited Schlüssel/Option (jurabib), [747](#), [748](#), 768
- howcited $\LaTeX$ -Feld (jurabib), 748, 768
- \howcitedprefix (jurabib), [748](#)
- \howcitedsuffix (jurabib), [748](#)
- howpublished $\LaTeX$ -Feld, [714](#), 791, 792
- \hphantom, 518
- \HR (lb2), [618](#), [634](#), [896–898](#)
- \HRule (lb2), [890](#)
- \hrule, 119, [235](#), 900, [901](#)
 - in Überschriften, [33](#)
 - (hhline), 275
- \hrulefill, [250](#), [889](#)
- hscale Schlüssel/Option (geometry), 216, 219
- \hsize feste Länge (tabularx), [260](#)
- \hslash
 - (amssymb), 544
 - (eulervm), [409](#)
 - (euler), [409](#)
- \hspace, [139](#), [156](#), 159, 522, 527, [719](#), [889](#), 890, 894
 - Fehler bei, 940
 - Silbentrennung ermöglichen, [89](#), [134](#), [254](#), [255](#), [257](#), [258](#)
- \hspace*, 881, [882](#), [889](#)
- \Hsub (lb2), [33](#)
- HTML-Dateien, von Bibliographien, [806](#), 819
- \Huge, 353, 354
- \huge, 154, [353](#), [354](#)
- humanbio $\LaTeX$ -Stil, 821
- humannat $\LaTeX$ -Stil, 821
- hungarian Option (babel), 559, [572](#)
- Hurenkind, 984
- \Hut (ifsym), [416](#)
- hvams Paket, 539
- hvmath Paket, 539
- Hyperlinks in Querverweisen, 82
- hyperref Paket, 82, 83, 184, 662, 725, 731
 - nicht kompatibel zu notocccite, 722
- hyphen.tex Datei (babel), 598
- hyphenate Option (truncate), 241
- \hyphenation, [101](#), [986](#), 987
 - Fehler bei, [939](#), 945, 958
- \hyphenchar, [439](#)
- \hyphenpenalty, 988
- hyphenrules Umgebung (babel), [562](#)
- hyphens Option (url), [101](#)
- ## I
- I Syntax (paralist), 140
- \i, 471
 - (tipa), [417](#)
- i Syntax (paralist), [140](#), [142](#), [145](#)
- ibidem Schlüssel/Option (jurabib), 752, [753](#), [754–756](#), [760](#)
- ibidem Schlüsselwert (jurabib), 761, [765](#), [766](#), [827](#)
- ibidem-Verweise, 753, [754–757](#), 765, [766](#)
- ibidemalt Schlüsselwert (jurabib), [766](#)
- \ibidemmidname (jurabib), [760](#)
- \ibidemname (jurabib), [760](#)
- icelandic Option (babel), 559, [579](#), [583](#)
- idem Schlüssel/Option (jurabib), [756](#), [757](#), [761](#)
- \idemPfname (jurabib), 760
- \idemPfname (jurabib), 760
- \idemPmname (jurabib), 760
- \idemPmname (jurabib), 760
- \idemPnname (jurabib), 760

- `\idemppname` (jurabib), 760
- `\idemSfname` (jurabib), **760**, [761](#)
- `\idemsfname` (jurabib), **760**, [761](#)
- `\idemSmname` (jurabib), 760
- `\idemsmmname` (jurabib), 760
- `\idemSnnname` (jurabib), 760
- `\idemsnnname` (jurabib), 760
- identifizierstyle Schlüssel (listings), 179
- Identifikationsteil (Paketdateien), 910–913
- `\idotsint` (amsmath), **504**, [505](#)
- `.idx` Dateierweiterung, **8**, **9**, **668**, 670, 675, 907
 - Fehler beim Lesen, 679, 681
 - (doc), 869
 - (index), 704, 705
 - (xindy), 695
- `\idxitem` (lb2), [240](#)
- `\ie` (lb2), [87](#)
- `\IeC` (inputenc), 457
- `ieeetr` $\text{\LaTeX}$ -Stil, 821
- `\IF` (algorithmic), [177](#)
 - `if$` interne $\text{\LaTeX}$ -Funktion, 839, [840–843](#)
- `\ifbottomfloat` (fancyhdr), 239
- `\ifcase`, 934
- `\ifdim`, 943
- `\iffalse`, 847
- `\IfFileExists`, 911, 917
- `\iffloatpage` (fancyhdr), [239](#)
- `\iffootnote` (fancyhdr), 239
- `\iflanguage` (babel), 562, [563](#)
- `\ifmmode`, [458](#)
- `\ifnum`, 943
 - `ifsym` Paket, 414–416
 - `ifsym.ps` Datei (ifsym), 415
 - `ifthen` Paket, 905–910
- `\ifthenelse` (ifthen), [76](#), [77](#), [158](#), [165](#), [207](#), [240](#), [316](#), [703](#), [716](#), [906](#), [907–910](#), [920](#), [927](#), 934
 - Fehler bei, 943
 - Zahlen vergleichen, 885, [906](#)
- `\iftopfloat` (fancyhdr), [239](#)
- `\ifToplevel` (docstrip), **860**, 861
- `\ifx`, [860](#)
 - `ignorehead` Schlüssel/Option (geometry), 218
 - `ignoremp` Schlüssel/Option (geometry), [219](#)
- `\ignorespaces`, [155](#)
- ignorierte Felder, Bibliographiedatenbank, 792
- `\iiiint` (amsmath), **504**, [505](#)
- `\iiint` (amsmath), **504**, [505](#)
- `\iint` (amsmath), **504**, [505](#)
- `\ij`, [607](#)
- `.ilg` Dateierweiterung, **8**, **9**
 - (makeindex), **668**, 675, **679**
- `\Im`, 544
- `\imath`, 544, 545
- `\in`, [487](#), [514](#), **549**
 - (euro), [105](#)
 - in Schlüsselwert (jurabib), [743](#), 749
 - inbook $\text{\LaTeX}$ -Publikationstyp, **791**, 793, [801](#) (jurabib), 769
- `\include`, [21](#), 22, 52, 868, 961, 963
 - Fehler bei, 939
 - Probleme mit TOC-Einträgen, [52](#)
- `\include` (Forts.)
 - Warnung bei, 968
 - (chapterbib), 773, 774
 - (excludeonly), 22
 - (index), 704
 - `includeall` Schlüssel/Option (geometry), **215**, [219](#)
 - `includefoot` Schlüssel/Option (geometry), 215
- `\includegraphics`
 - (graphics), [634](#), [635](#), [636](#), 642–644, 646
 - Fehler bei, 931
 - Graphik nicht gefunden, 935
 - (graphicx), [116](#), [221–223](#), [312](#), 632, 634, **636**, 638, [639–641](#), 642–645
 - Fehler bei, 931, 933
- `\includegraphics*`
 - (graphics), [634](#), [635](#)
 - (graphicx), 636
 - `includehead` Schlüssel/Option (geometry), [214](#), **215**
 - `includeheadfoot` Schlüssel/Option (geometry), 215
 - `includemp` Schlüssel/Option (geometry), [214](#), **215**, 218
- `\includeonly`, [21](#), 22, 704
 - (excludeonly), [22](#)
 - `incollection` $\text{\LaTeX}$ -Publikationstyp, **791**, 792, 793 (jurabib), 769
- `.ind` Dateierweiterung, **8**, **9**, 668
 - (index), 705
 - (makeindex), **668**, 675, 679, 691
 - Fehler beim Schreiben, 680
 - (xindy), **668**, 695
- `\indent` (picinpar), 115
 - `indent_length` Schlüsselwort (makeindex), 682
 - `indent_space` Schlüsselwort (makeindex), 682
 - `indentafter` Option (titlesec), 43
 - `indentfirst` Paket, **34**, 581
 - `indention` Schlüssel/Option (caption), **318**, [322](#)
- `\Index` (lb2), [674](#)
- `\index`, [350](#), 668, 669, [670](#), [671](#), [672](#), [673](#), [674](#), 675, 686–688, [907](#)
 - verursacht zusätzlichen Leerraum, 892
 - (index), 704, **705**
 - index Paket, 687, **704**, **705**, 725
- `\index*` (index), 704
- Indexbefehle (Dokumentation), Liste der, 853–857
- Indexdateien, 8
- `indexed` Schlüsselwert (jurabib), 746
- `\indexentry`, 669, **673**, **680**, [907](#) (natbib), [734](#)
- Indexerstellung
 - angehängte Leerzeichen, **670**, 678, 688, 691
 - Anwenderbefehle definieren, 673, 674
 - Auflistung im Inhaltsverzeichnis, 703
 - ausführlicher Modus, 697
 - Ausgabe-Layoutparameter, 682
 - Ausgabedateien angeben, 678, 690, 697
 - automatische Indexierung deaktivieren, 849
 - Autorenverzeichnisse, 703
 - Befehle automatisch indexieren, 849, 869
 - Buchstaben, Sortierreihenfolge nach, 678, 690
 - Buchstabengruppen, 684, 699

Indexerstellung (*Forts.*)

deutsche Wörter, Sortierreihenfolge, 678, 690, 692

Eingabe-Stilparameter, 680

Eingabedateien angeben, 678, 690

Einträge automatisch erzeugen, 849, 869

Einträge im Rand anzeigen, 703

führende Leerzeichen, **670**, 678, 688, 691

Fehlermeldungen

Liste von (*MakeIndex*), 679–681

unterdrücken, 678, 690, 697

Fehlersuche, 687, 688, 697

formatieren

mit *MakeIndex*, 675–688

mit $\LaTeX$, 702, 703, 705

mit xindy, 688–691, 702

Seitenzahlen, 672

französische Wörter, Sortierreihenfolge, 692

Glossareinträge verarbeiten, 673

Groß-/Kleinschreibung, Unterscheidung von, 670

in Inhaltsverzeichnissen aufführen, 51

indxcite, 703

Konsistenz, 688

kyrillisches Alphabet, 589

$\LaTeX$ -Befehle indexieren, 674, 691

Layoutparameter, 680, 682

Leerräume, 688, 691

Leerzeichen, **670**, 678

komprimieren, **670**, 678, 688, 691

Sortierreihenfolge, 687

Leitpunkte, 684

Makros automatisch indexieren, 849, 869

Mehrfachindexe, 704, 705

Meldungen unterdrücken, 678, 690, 697

nicht englische Wörter, 688, **691–693**

Nummer der ersten Seite festlegen, 678, 683

Ortsangaben formatieren, 700

Protokolldatei angeben, 679, 690

Quellenverweise automatisch indexieren, 734, 746

Querverweise

überprüfen, 688

erstellen, 671

römische Zahlen

Seitenbereiche unterdrücken, 700

Sortierreihenfolge, 687

Referenzarten, 700

repeatindex, 703

Rohindex erzeugen, 669

Seitenbereiche

deaktivieren, 678, 690, 694, 700

Längen begrenzen, 700

Seitenumbrüche, Eintrag wiederholen, 703

Seitenzahlen

Duplikate, 670

formatieren, 672

kapseln, 672, 695

MakeIndex, 685–687

römische Zahlen, 687, 700

Sortierreihenfolge, 678, 686, 700–702

Indexerstellung (*Forts.*)

xindy, 700–702

zusammengesetzte (kapitelweise

Nummerierung), 686

showidx, 703

Sonderzeichen, **673**, 674, 675, 683, 684

Sortierreihenfolge

deutsche Wörter, 678, 690, 692

Fehlersuche, 687, 688

französische Wörter, 692

Leerzeichen, 687

nach Buchstaben, 678, 690

römische Zahlen, 687

Seitenzahlen, 678, 686, 700–702

Sonderfälle, 689

spanische Wörter, 692, 696

Symbole, 687, 689

xindy-Regeln, 696, 699

Spaltenumbrüche, Eintrag wiederholen, 703

spanische Wörter, Sortierreihenfolge, 692, 696

Standard-Ein-/Ausgabedateien, 678, 690

Stildateien

angeben, 679

MakeIndex, 679, 681–687

xindy, 695–702

stiller Modus, 678, 690, 697

Symbole, Sortierreihenfolge, 687, 689

technischer Index, 688

tocbibind, 703

Trennzeichen für Indexebenen, 671

unabhängiger Index, 683

Untereinträge, 671

Verarbeitungsablauf, 668, 695

Vereinheitlichen von Indexeinträgen, 698

Vereinheitlichungsregeln, 695, 698

Verlaufs meldungen unterdrücken, 678, 690, 697

`\IndexInput (doc)`, **850**, 851, **853**

`\IndexMin feste Länge (doc)`, 854

`\indexname`, **37**, **702**, **703**

(babel), **561**, **566**

`\IndexParms (doc)`, 855

`\IndexPrologue (doc)`, 855

`\indexproofstyle (index)`, 704

`\Indextt (lb2)`, **674**

Indisch, 608

indxcite Paket, 703

`\inf`, 504, **513**

Info Math Font, in Formeln und Text, **539**

infomath Paket, 539

informelle Meldungen, 961–977, *siehe auch* Fehlersuche

infoshow Option

(lb2), 914

(multicol), **197**

(tracefnt), 380

`\infy`, **503**, **504**, **512**, **514**, **544**

Inhaltsverzeichnisse, 48–69, *siehe auch* minitoc-Paket; titlesec-Paket

Abbildungs- und Tabellenverzeichnisse hinzufügen, 51

Absatzformat, 65, **66**, **67**

Inhaltsverzeichnisse (*Forts.*)

- Abstände, 51
- Ausrichtung des Textes, 63, [64](#), [65](#)
- Beschreibung, 48
- Bibliographien hinzufügen, 51
- Breite der Nummerierung, 54
- Daten eintragen, 49, [50](#), 51, 52
- Einzug, 54, 62
- für Teile oder Abschnitte, [61](#), 62
- formatieren, 52, [53](#), 54, 55, 63–67
- generieren, 49
- Gliederungsebenen, 54
- Index hinzufügen, 51, 703
- kombinieren, [56](#), [57](#)
- kompakte, 58, 59
- Leitpunkte, 63
- mehrfache, [58](#), 59–62
- nicht nummerierte, 63
- optionale Code-Ausführung, 62, 63
- partielle, 59, 60, [61](#), 62, 68, [69](#)
- ungewöhnliche Nummerierung, 55
- `\init` (nfssfont.tex), [381](#)
- `\initfamily` (yfonts), 407
- Initiale, [106–108](#), 406, [407](#)
- Initialisierungsteil (Paketdateien), 913
- `\initiate@active@char` (babel), [605](#), [606](#)
- `\injl` (amsmath), 513
- inner Schlüssel/Option (geometry), 216
- innerbars Option (changebar), 200
- innerbody Option (sidecap), 333
- innercaption Option (sidecap), 333
- inparadesc Umgebung (paralist), [143](#), [144](#), 146
- inparaenum Umgebung (paralist), [139](#)
- inparaitem Umgebung (paralist), 143
- `\inplus` (stmaryd), 549
- inproceedings BibTeX-Publikationstyp, [714](#), [791](#), 792
- `\input`, 444, 868, [917](#), 961
 - Fehler bei, 935
 - (docstrip), [858](#), 861
 - (optional), 23
- inputenc Paket, 7, 14, 184, 340, 369–372, 456–460, 587, 594
 - Einschränkungen bei Schlüssel, 70
 - erforderlich für icelandic, 583
 - Fehler bei, 923, 940
 - in diesem Buch, 15, 1137
 - kombiniert mit listings, 184
- `\inputencoding` (inputenc), [371](#), [372](#), [427](#), 587
- inputencoding Schlüssel (listings), 184
- `\InputIfFileExists`, [911](#), [914](#), [917](#)
- .ins Dateierweiterung, [858](#)
 - (docstrip), 858
 - (doc), 846
- `\insert`, 959
- install-pkg.sh Programm, 1008
- install-tl.sh Programm, 1008
- Installationsunterstützung hinzufügen, 863–866
- institution BibTeX-Feld, 791, 792
- `\int`, 552
 - Platzierung von Hoch-/Tiefstellungen, [503](#), [504](#)
 - (resize), Verwenden größerer Symbole, [91](#)
- `int.to.chr$` interne BibTeX-Funktion, 839
- `int.to.str$` interne BibTeX-Funktion, 839
- INTEGERS BibTeX-Befehl, 836, [838](#)
- Integrale, Mehrfach-, 504, [505](#)
- `\intercal` (amssymb), 546
- `\interleave` (stmaryd), 546
- interlingua Option (babel), 559
- `\internallinenumbers` (lineno), [186](#)
- `\internallinenumbers*` (lineno), [186](#)
- internationale Dokumente, *siehe* fremdsprachige Dokumente
- Internationale Standardbuchnummer (ISBN), 735
- Internationale Standardseriennummer (ISSN), 735
- Internationales Phonetisches Alphabet (IPA), 416, [417](#), [418](#)
- interne Dateien, 8
- interne Tabellen, Überlauf, 958–961
- Internet-Ressourcen
 - Bibliographien, 802, 803
 - Index (xindy), 690
 - LaTeX, 995–1002
- Interpunktion
 - Bibliographien
 - Kurztitel-Zitierschema, 763, [764](#)
 - numerisches Zitierschema, 719, [721](#), [722](#)
 - fremdsprachige Dokumente
 - Abstände nach, 580
 - Sonderfälle, [607](#)
 - Kurztitel-Quellenverweise, 763, [764](#)
 - mathematische Symbole, [552](#)
 - numerische Quellenverweise, 719, [721](#), [722](#)
 - theoremähnliche Strukturen, 149
- `\intertext` (amsmath), [491](#)
- `\Interval` (ifsym), [415](#)
- `\intertextsep` elastische Länge, 293
 - (wrapfig), 309
- intlimits Option (amsmath), 504
- invert Option (crop), 222
- `\iota`, 543
- IPA Umgebung (tipa), [417](#)
- IPA (Internationales Phonetisches Alphabet), 416, [417](#), [418](#)
- irish Option (babel), 559
- `\IroningII` (marvosym), [413](#)
- is-abbrev BibTeX-Stil, 822
- is-alpha BibTeX-Stil, [808](#), [822](#)
- is-plain BibTeX-Stil, 822
- is-unstr BibTeX-Stil, 822
- isbn BibTeX-Feld, [714](#), 793, [801](#), [808](#), [809](#)
 - (BibTexMng), 820
 - (custom-bib), 833
 - (natbib), 735
- ISBN (Internationale Standardbuchnummer), 735
- `\iscurrentchapter` (lb2), [76](#)
- `\iso` (euro), 105
- iso88595 Option (inputenc), 587
- `\isodd` (ifthen), [165](#), [909](#)
- issn BibTeX-Feld, [714](#)
 - (BibTexMng), 820
 - (custom-bib), 833
 - (natbib), 735

- ISSN (Internationale Standardseriennummer), 735
- .ist Dateierweiterung, 8, 9
(makeindex), 668, 681
- \it, 358, 476
in Formeln, 361, 476
- it Option (titlesec), 39, 40
- it Schlüsselwert
(caption), 319, 321, 322, 334
(subfig), 326, 328
- italian Option (babel), 559, 560, 872
- italic Schlüsselwert (jurabib), 743, 744, 745, 759, 763
- Italic (Schriftform), 344
- ItalicNums Option (parallel), 192
- ITC Bookman Font, 385
- \itdefault, 357
- \item, 136–139, 152–154, 155–159, 175, 881, 891, 908
Fehler bei, 941, 949
in theindex, 702, 703
(fancybox), 618
(threeparttable), 286
- item_0 Schlüsselwort (makeindex), 682
- item_01 Schlüsselwort (makeindex), 682
- item_1 Schlüsselwort (makeindex), 682
- item_12 Schlüsselwort (makeindex), 682
- item_2 Schlüsselwort (makeindex), 682
- item_x1 Schlüsselwort (makeindex), 682
- item_x2 Schlüsselwort (makeindex), 682
- \itemindent feste Länge, 153, 155, 156, 159
- itemize Umgebung, 85, 135, 136, 143, 376, 618
Fehler bei, 951
Stilparameter, 136
(babel), 581
(paralist), 143
- \itemsep elastische Länge, 153, 732
- ITERATE $\LaTeX$ -Befehl, 837, 838
- \itshape, 351, 352, 355, 357, 419, 476
in Formeln, 360, 361
(ulem), 93
- ## J
- J Syntax (tabulary), 262
- \j, 464, 471
Probleme in T1, 428
- JabRef Programm, 819
- \JackStar (bbling), 414
- \JackStarBold (bbling), 414
- Jahresdaten fehlen, Bibliographien, 733, 734
- Japanisch, 608
- jas99 $\LaTeX$ -Stil (chicago), 724
- Java Datenbankmanager, 817–819
- \jbactualauthorfont (jurabib), 743, 762
- \jbactualauthorfontifannotator (jurabib), 743
- \jbannotatorfont (jurabib), 762
- \jbannotefont (jurabib), 766
- \jbauthorfont (jurabib), 762
- \jbauthorfontifannotator (jurabib), 762
- \jbauthorindexfont (jurabib), 746
- \jbbfsasep (jurabib), 762
- \jbbibhang feste Länge (jurabib), 764, 765
- \jbbstasep (jurabib), 762
- \jbbtasep (jurabib), 761, 762
- \jbcitationyearformat (jurabib), 759
- \JBibtexManager Programm, 817, 818
- \jbignorevarioref (jurabib), 752
- \jbindexbib (jurabib), 746
- \jbindextype (jurabib), 746
- \jbttitlefont (jurabib), 762
- \jbuseidemhrule (jurabib), 765, 766, 827
- \jbyearaftertitle (jurabib), 759
- \jmath, 544, 545
(mathptmx), nicht verfügbar, 388
- jmb $\LaTeX$ -Stil
(apalike), 822
(jmb), 820, 821, 822
- jmb Paket, 822
- \jobname, 781
- \Joch (ifsym), 416
- \Join
(amssymb), 551
(latsym), 476
- \joinrel, 551
- \jot feste Länge (multirow), 281
- journal $\LaTeX$ -Feld, 791, 792, 799
- joX $\LaTeX$ -Stil (jurabib), 767, 822
- .jpeg Dateierweiterung, 660, 661
- .jpg Dateierweiterung, 9, 931
- \jput (epic), 622, 623, 624
- jtb $\LaTeX$ -Stil, 822
- jura.bib Datei (lb2), 742, 806
- jurabib $\LaTeX$ -Stil
(bibtopic), 781
(jurabib), 742–766, 767, 794, 815, 822, 827
- jurabib Paket, xxviii, 741–769, 771, 822
Installationsmöglichkeiten, 863
Kompatibilitätsmatrix, 772
- jurabib.cfg Datei (jurabib), 767
für dieses Buch verwendet, 741
- \jurabibsetup (jurabib), 741, 743–751, 753–761, 766, 767
- jureco $\LaTeX$ -Stil (jurabib), 767, 822
- jurunrst $\LaTeX$ -Stil (jurabib), 765, 767, 822
- justification Schlüssel/Option
(caption), 310, 320, 321, 322, 333
(subfig), 326, 328
- justified Schlüsselwert (caption), 320
- justify Umgebung (ragged2e), 113
- \justifying (ragged2e), 113
- \JustifyingParfillskip elastische Länge (ragged2e), 113
- \JustifyingParindent feste Länge (ragged2e), 113
- ## K
- \k, 465, 471, 583
- Kästen, siehe Boxen
- Kapitalchen
Beschreibung, 344, 345
französische Namen, 579
in Überschriften, 352
- kapitelweise Nummerierung, Index, 686
- \kappa, 543
- Kapselung, Bilddateien, 645, 646

- keepaspectratio Schlüssel (graphicx), [637](#), [640](#), [641](#)
- KeepShorthandsActive Option (babel), 598
- \keepsilent (docstrip), 861
- \Ker (lb2), 479
- \ker, 513
- \kern, 182
- Kernel-Fehler, *siehe* Fehlersuche
- Kettenbrüche, mathematische Symbole, [503](#)
- key BibTeX-Feld, [793](#), [794](#), [808](#), [809](#), 825
- \Keyboard (marvosym), [413](#)
- keyval Paket, 214, 317, 641
- keyword Schlüsselwort (makeindex), [673](#), [680](#)
- keyword BibTeX-Feld (printbib), 805
- keywords BibTeX-Feld (BibTexMng), 820
- keywordstyle Schlüssel (listings), [179](#), [180](#), [181](#)
- \kill, [249](#), [250](#)
 - (longtable), 269
- Klammern
 - Bibliographien
 - Kurztitel-Zitierschema, 761
 - numerisches Zitierschema, [720](#)
 - geschweifte, weglassen, 876
- Klassen
 - Befehle, 879, 911, 917–921
 - Dateistruktur, 910–922
 - Mindestanforderungen, 922
 - Optionen, 18, *siehe auch* Optionen
- Klassendateien, 7
- Kleinerzeichen (<), Kurzzeichen, [574](#)
- Kleinschreibung in Bibliographien deaktivieren, 840, 841
- kluwer BibTeX-Stil (harvard), 725, 822
- Kodierung
 - Akzentbuchstaben, 369, [370](#), 371, 372
 - Ausgabe, [341](#), 372, 373, 460–475
 - auswählen, 372, 373
 - Definitionsdateien, 7
 - Eingabe, [339–341](#), 369, [370](#), 371, 372, 456–460
 - Fontbefehle, Low-Level, [427](#), [428](#)
 - LaTeX, 339–341, 347, [453](#), [454](#)
 - LICR-Objekte, 454–456
 - Liste der, 467–475
 - mathematische Eingaben, 458, 459
 - OT1, 348
 - Pi-Fonts, [390–393](#)
 - PostScript, [399–401](#)
 - Schemata deklarieren, 442
 - Single-Byte-Zeichen, 370, 371
 - Sprachoptionen und Fonts, [583](#), [594](#)
 - Beschreibung, 347, 348
 - griechisches Alphabet, 590, 591, 593
 - hebräisches Alphabet, 591–593, [594](#), [595](#)
 - kyrillisches Alphabet, 585–589
 - Optionen (Sprachauswahl), 582–585
 - OT1-Erweiterungen, 583
 - T1-Erweiterungen, 583
 - T2A-Kodierung, 589
 - T2B-Kodierung, 589
 - T2C-Kodierung, 589
 - T1 (Cork), [348](#), 365
 - TeX, 365
- Kodierung (*Forts.*)
 - Texteingaben, 458, 459
 - Textsymbole
 - Pi-Fonts, [390–393](#)
 - PostScript, [399–401](#)
 - TS1, 373, [374–379](#)
 - Zapf Dingbats, 390–392
 - TS1, 373, [374–379](#)
 - UTF-8-Unterstützung, [372](#), 454, 459, 460
 - Zapf Dingbats, 390–392
- koi8-r Option (inputenc), [370](#), [427](#), [586](#), [587](#)
- Kolumnentitel
 - anpassen
 - linke/rechte Seiten, 234, 235, [236–238](#)
 - Fußzeilenhöhe, 209
 - Gleitobjektseiten, Kolumnentitel-Layout, [239](#)
 - lebende, [202](#), 205, 212, 223, 240
 - formatieren, *siehe* Kolumnentitel-Layouts
 - Seitenlayout, 215, 218
 - mehrseitige Tabellen, 265, 269
 - Probleme, 205, 240
- Kolumnentitel-Layouts, 230, 231
 - anpassen
 - abhängig von Gleitobjekten, 239
 - angepasstes Layout speichern, [238](#)
 - global, [232](#), [233](#)
 - benannt, 238
 - für doppelseitigen Ausdruck, 231, 232, [234](#)
 - für Gleitobjektseiten, [239](#)
 - für Wörterbücher, 239, [240](#)
 - Linien, 233
 - mehrzeiliger Text, [233](#)
 - Text abschneiden, 240, [241](#)
 - Textmarkenbefehle, 226, [227](#), 228, [229](#), 230, [237](#), [238](#)
- Kolumnenziffer, 213
- KOMA-Script-Klassen, 244, 245
- Kombinieren von Inhaltsverzeichnissen, [56](#), [57](#)
- Kommandozeilenwerkzeuge, Bibliographien, 804–813, 816
- Kommentare aus dem Code entfernen
 - Anwendermeldungen erzeugen, 860, 861
 - beliebige Programmiersprachen, 866
 - Beschreibung, 857, 858
 - Ergebnisdatei angeben, 859, 860
 - Installationsunterstützung hinzufügen, 863–866
 - Kommentarzeichen ändern, 866
 - Konfigurationsdateien anlegen, 863–866
 - Master-Skripts anlegen, 861, 862
 - Meldungen erzeugen, 860, 861
 - Nachspann erzeugen, 862, 863
 - Präambel erzeugen, 862, 863
 - Programmaufruf, 858
 - Quelldatei angeben, 860
 - Sicherheitserwägungen, 865
 - Skriptbefehle, 858–863
 - Syntax, 858–863
 - TDS-konforme Installation sicherstellen, 863–866
 - Verbatim-Begrenzungssymbole kodieren, 866

Kommentarzeichen

- Bibliographien, 790
- doc-Paket, 847
- docstrip, 866

Kommentieren von Bibliographien, 766, 768

Kommentierte Quellenverweise, 746, 747

kommutative Diagramme, 479, 500, 501

kompakte Inhaltsverzeichnisse, 58, 59

Kompilierfehler, *siehe* Fehlersuche

Komprimieren von Quellenverweisen, 739

Konfigurationsdateien, *siehe auch* .cfg

- anlegen, 863–866, 868, 869
- babel, 571, 596–598, 606
- caption, 324
- color, 945
- docstrip, 864, 955
- dvips, 655
- endfloat, 299
- externe, Bibliographien, 767
- graphics, 632, 945
- jurabib, 741
- L^AT_EX, 441, 443, 861
- ltxdoc, 868
- natbib, 731
- paralist, 146
- subfig, 331
- T_EX, 956
- textcomp, 379

Konsistenz, Index, 688

kontinuierliche Steigung, Kurven, 629, 630, 631

Konventionen, typographische, in diesem Buch, 12–15

Kopfsteg (oberer Rand), 201, 202, 212, 214, 216, 219, 233

Kopfzeilen, *siehe* Kolumnentitel

Koreanisch, 608

Kreisausschnitte, *siehe* epic-Paket; eepic-Paket

Kreise

- ausfüllen, 628, 629
- mathematische Symbole, 547
- zeichnen, 628

kulturspezifische typographische Regeln, 558

kursive Schriftform, 344

Kursivkorrektur, 351, 352, 353

Kurven

- Bézier, *siehe* epic-Paket; eepic-Paket
- Chaikins, 628
- kontinuierliche Steigung, 629, 630, 631

Kurzformen

- für Umgebungen, 480
- fremdsprachige Dokumente, 563, 564, 565
- in Bibliographien, 798–800
- Optionen (Sprachauswahl), babel-Paket, 568–574
- Sprachdefinitionsdateien, 606, 607

Kurztitel-Zitierschema, 708, 741–771, *siehe auch*

- Zitierschemata
- Übersetzerdaten, 769
- Anpassen von Bibliographien, 762, 763, 764, 765, 766, 767
- Anpassen von Quellenverweisen, 761, 762
- ausführliche Verweise im Fließtext, 748, 749–751

Kurztitel-Zitierschema (*Forts.*)

- Autoren-Informationsfeld, 768
- Beschreibung, 741
- Definition, 708
- Dissertationsjahr, 768
- Editionsdaten, 768
- einleitender Text, 746, 747
- Einzug, 764, 765
- Endnotenverweise, 752, 753
- Fonts, 762, 763
- Fußnotenverweise, 752, 753
- Geschlecht der Autoren, 760, 761, 768
- Gründerdaten, 768
- Herausgeberdaten, 768
- hochgestellte Zeichen, 761, 769
- ibidem-Verweise, 753, 754–757, 765, 766
- Interpunktion, 763, 764
- Klammern, 761
- Kommentare, 746, 747, 766, 768
- Konfigurationsdateien, externe, 767
- mehrsprachige Dokumente, 759, 760, 761
- mit Autor-Jahr-Zitierschema kombinieren, 758, 759
- Querverweise, 757, 758
- Rückverweise, 768
- Sammlungen, 768
- Seitenanzahl-Feld, 769
- Seitengrenzen ignorieren, 755
- Sortierreihenfolge, 769
- Spaltenlayout, 765
- Stildateien, 767–769
- Titel eines Bandes, 769
- Titel Kurztiteln zuordnen, 747, 748
- Titel-Informationsfeld, 768, 769
- Titelformat, 744–746
- Trennzeichen und -wörter bei mehreren Autoren, 761, 762, 764
- übersetzte Werke, 768
- Unterstützung für Gesetzestexte, 769, 770, 771
- URLs, 768, 769
- Verweise automatisch indexieren, 746
- vollständige Verweise im Fließtext, 750
- zuletzt geändert-Feld, 769

kuvio Paket, 500, 1030

Kyrillisch, 585–587, 588, 589, 590

L

L Syntax

- (fancyhdr), 234, 235–238
- (lb2), 256
- (tabulary), 262

\L, 469

- (babel), 584
- für Doppel L, 569
- (pxfonts), Probleme mit, 402
- (txfonts), Probleme mit, 400

\l, 471, 583

- (babel), für Doppel l, 569
- (pxfonts), Probleme mit, 402
- (txfonts), Probleme mit, 400

- 1 Syntax, 251, [252](#), [253](#)
 - (array), 258
- L... Fontkodierung, 428, 442
- `\l@{sprache}` (babel), 563, 596, [597](#)
- `\l@chapter`, 53
- `\l@english` (babel), 597
- `\l@example` (lb2), [58](#)
- `\l@figure`, [53](#), [57](#)
- `\l@paragraph`, [53](#)
- `\l@part`, 53
- `\l@section`, [53](#)
- `\l@subfigure` (subfig), 330
- `\l@subparagraph`, [53](#)
- `\l@subsection`, [53](#), [55](#)
- `\l@subsubsection`, [53](#), 55
- `\l@subtable` (subfig), 330
- `\l@table`, [53](#), [57](#)
 - Längen, *siehe* Längenparameter
 - Löcher in Absätzen, [115](#), [116](#)
 - Lösen von Problemen, *siehe* Fehlersuche
- `\label`, [29](#), [70](#), 71, 73, 75–77, [118](#), 128, [138](#), 187, [224](#), [316](#), 886, [909](#), 959, 971
 - Einschränkungen für Schlüssel, 70, 874
 - Fehler bei, 928, 944
 - Probleme mit, [28](#), 70, [71](#), 92, [885](#)
 - verursacht seltsame Ergebnisse, [28](#)
 - verursacht zusätzlichen Leerraum, 892
 - Warnung bei, 966, 967, 972
 - (amsmath), 485, [494](#), 495, [497](#)
 - (babel), 70
 - (fancyvrb), [170](#)
 - (longtable), 270
 - (paralist), 139, [141](#)
 - (prettyref), [80](#)
 - (showkeys), [71](#), [72](#)
 - (subfig), [325](#), [328](#), [329](#)
 - (subfloat), [332](#)
 - (textcase), 92
 - Probleme mit, 92
 - (titleref), 81, 82
 - (varioref), [75](#)
 - (wrapfig), [309](#)
 - (xr), 82, 83
- label Schlüssel
 - (fancyvrb), [166](#), [167](#)
 - (listings), 183
- `\labelenumi`, [137](#), [138](#), [886](#)
- `\labelenumii`, [137](#), [886](#)
- `\labelenumiii`, [137](#), [886](#)
- `\labelenumiv`, [137](#), [886](#)
- labelfont Schlüssel/Option
 - (caption), [310](#), [315](#), [319](#), [320–322](#), [334](#)
 - (subfig), [326](#)
- `\labelformat` (varioref), 73, 75, [76](#), 80, 137, [138](#), [752](#)
- labelformat Schlüssel/Option
 - (caption), [319](#), [320](#), [321](#), 322, [323](#)
 - (subfig), [326](#)
- `\labelitemi`, [136](#), [376](#)
- `\labelitemii`, 136
- `\labelitemiii`, 136
- `\labelitemiv`, 136
- labelposition Schlüssel (fancyvrb), [166](#), [167](#)
- Labels
 - Format, Überschriften, 41
 - formatieren, bei Querverweisen, 75, [76](#), 77–79
 - Formeln, *siehe* Tags
 - Legenden von Gleitobjekten, [319](#), [320](#), 322–324
- `\labelsep` feste Länge, [139](#), 146, [153](#), [156](#), [159](#), 249, [883](#)
- labelsep Schlüssel/Option
 - (caption), [320](#), [321](#), [322](#), 323, [334](#)
 - (subfig), [326](#)
- labelstoglobalaux Option (bibunits), 779
- `\labelwidth` feste Länge, [153](#), [155–159](#), [883](#)
- Längenparameter
 - Beschreibung, 887
 - für horizontale Abstände, [889](#)
 - für vertikale Abstände, [890](#), 891–893
 - konfigurieren, [887](#), [888](#)
 - neue definieren, 887
- `\Lambda`, 543
- `\lambda`, [491](#), [512](#), 543
- `\land`, 546
- landscape Option, 921
 - (crop), 221
 - (typearea), [212](#), [213](#), [214](#)
- landscape Schlüssel/Option (geometry), [214](#), 215, [219](#)
- landscape Umgebung (lscape), 220
- `\langle`, 511, 553
- `\language`, 561
- language Schlüssel (listings), [179](#), [180](#), [181](#), [183](#), [184](#)
- language BibTeX-Feld
 - (BibTexMng), 820
 - (custom-bib), [832](#)
 - (jurabib), [742](#), 760
- language.dat Datei (babel), 8, 562, [597](#), 598, [601](#), 960
- language.skeleton Datei (babel), 596, 598
- `\language0` (babel), 601
- `\languageattribute` (babel), 565, 566
 - Warnung bei, 976
- `\languagename` (babel), 562, [563](#)
- `\languageshorthands` (babel), 564, [605](#)
- `\LARGE`, 353
 - Large Schlüsselwert (caption), 319
- `\Large`, 353
- `\large`, 353, [354](#), [888](#)
 - large Schlüsselwert (caption), 319
- `\largepencil` (dingbat), [411](#)
- `\larger` (relsize), 90
- largestsep Option (titlesec), 43
- last Schlüsselwert
 - (fancyvrb), [168](#)
 - (listings), 181
- `\LastDeclaredEncoding`, 443
- `\lasthdashline` (arydshn), 275, 276
- `\lasthline` (array), 276, 288, [289](#)
- `\lastleftmark` (extramarks), 228
- `\lastleftxmark` (extramarks), 229

- lastline Schlüssel
 - (fancyvrb), 170, [172](#)
 - (listings), 181
- LastPage Zähler (lastpage), [224](#), [234](#)
- lastpage Paket, xxix, [224](#), [225](#)
- \lastrightmark (extramarks), [228](#), [237](#), [239](#), [240](#)
- \lastrightxmark (extramarks), [229](#)
- L^AT_EX
 - aktuelles System, Übersicht, 6–10
 - Formatdateien, 7
 - Geschichte, 1–6
 - Verarbeitungsablauf, 10
 - verwendete Dateien, 6–10
- L^AT_EX 2.09
 - Fonts, 358
 - High-Level-Fontbefehle, 353, 358, 361
 - mathematische Symbole, 476
- L^AT_EX Project Public License (LPPL), 5, 1009
- L^AT_EX-Begleiter (dieses Buch)
 - Autoren, 1129–1133
 - Beispiele in, 15, 16
 - Entstehungsgeschichte, 1135–1137
 - Errata für, xxvi
 - typographische Konventionen, 12–15
- L^AT_EX-Dateien, Quellen
 - CD/DVD, 996, 997
 - FTP, 996, 1000–1002
 - Internetzugriff, 998
- latex.fmt Datei, 7
- latex.ltx Datei, 376, 861, 886, 887
- latex2html Programm, 872
- latexsym Paket, 476
- latin Option (babel), [559](#), [573](#)
- Latin Modern Fonts, [368](#)
- latin1 Option (inputenc), [370](#), [372](#), [583](#)
- latin2 Option (inputenc), 370
- latin3 Option (inputenc), 370
- latin4 Option (inputenc), 370
- latin5 Option (inputenc), 371
- latin9 Option (inputenc), 371
- \latinencoding (babel), 583
- \latintext (babel), [584](#), [605](#)
- Laufweite, *siehe auch* sperren
 - von Frakturfonts, [407](#)
 - von Schreibmaschinenfonts, [398](#), 399
- \layout (layout), 208
- layout Paket, 208
- Layout einer Seite, *siehe* Seitenlayout
- Layoutdateien, *siehe* Stildateien;
 - Konfigurationsdateien
- Layoutparameter
 - Absätze, 113
 - Aufzählungen, 136
 - für Dokumentation, 853–857
 - Fußnoten, 120
 - Gleitobjekte, 292–294
 - Indexerstellung, 680, 682
 - Listen, allgemeine Parameter, 153
 - nummerierte Listen, 137
 - Seitenlayout, 203
 - Tabellen, 251
- Layoutparameter (*Forts.*)
 - Verzeichnisse, 54
- layouts Paket, xxix, 31, 153, 202, [208–210](#)
- \LB (lb2), [877](#)
- LB2, *siehe* dieses Buch
- lb2 Paket, 1033
- lb2.err Datei (lb2), xxvi
- lb2exa Bib_TE_X-Stil (lb2), 727
- lb2exa Dokumentenklasse, 1008
- \Lbag (stmaryrd), 554
- \lbag (stmaryrd), 546
- \lbrace, [484](#), [485](#), [511](#), [523](#), [526](#), [553](#)
- \lbrack, 511, 553
- \lccn Bib_TE_X-Feld (BibTeXMng), 820
- \lceil, 511, 553
- \Lcs (lb2), [350](#)
- .ldf Dateierweiterung, 9
 - (babel), 7, [559](#), [596](#), 598–605
 - (jurabib), 759
- \ldf@finish (babel), 605
- \LdfInit (babel), 600
- \ldots, [508](#), [552](#), [876](#), [877](#), [907](#), [977](#)
- \le, [512](#), [548](#)
- \leadsto (latexsym), [476](#)
- lebende Kolumnentitel, 202, 205, 212, 215, 218, 223, 240
- ledmac Paket, 124, 1032
- „leere“ Größenfunktion, 435
- „LEERE SEITE“ auf generierten Seiten, [244](#)
- Leerräume
 - Bib_TE_X-Programmiersprache, 836
 - Index, 688, 691
 - nach Makronamen, 86, [87](#)
- Leerzeichen
 - Bibliographiedatenbanken, 790
 - darstellen, 168, [169](#)
 - doc-Paket, 847
 - führende, in wortwörtlicher Ausgabe entfernen, [165](#)
 - in Indexen, [670](#), 678, 687
 - in/vor Quellenverweisen, 720
 - komprimieren, Index, [670](#), 678, 688, 691
- Leerzeilen
 - Bibliographiestile, 836
 - Formeln, [492](#), [493](#)
 - nummerieren, 168
 - Probleme beim Erzeugen, 951
- \left, [490](#), [495](#), [500](#), [510](#), 511, 517, 540, [542](#), 553, 934
- Fehler bei, 942, 944
- left Option
 - (eurosym), 420
 - (lineno), 190
- left Schlüssel/Option (geometry), [216](#), [217](#)
- left Schlüsselwert
 - (fancyvrb), 167, [168](#), [172](#), [174](#)
 - (listings), 180, [181](#)
- \Leftarrow, 550
- \leftarrow, 550
- \leftarrowtail (amssymb), 550
- \leftarrowtriangle (stmaryrd), 550
- leftbars Option (changebar), 200

- leftbody Option (sidecap), 333
- leftcaption Option (sidecap), 333
- leftFloats Option (fltpage), 335
- \leftharpoondown, 550
- \leftharpoonup, 550
- \lefthyphenmin, 602, 991
- leftlabels Option (titletoc), 63
- \leftleftarrows (amssymb), 550
- leftline Schlüsselwert (fancyvrb), 166
- \leftmargin feste Länge, 153, 155–157, 159, 883
- leftmargin Schlüssel (titlesec), 40, 46
- \leftmark, 226, 234–236, 237, 240, 241 (extramarks), 228
- \leftpointright (dingbat), 411
- \Leftrightarrow, 550
- \leftrightharpoonup, 550
- \leftrightharrows (amssymb), 550
- \leftrightharrows (stmaryrd), 548
- \leftrightharrows (amssymb), 550
- \leftrightharrows (stmaryrd), 550
- \leftrightsquigarrow (amssymb), 550
- \leftroot (amsmath), 517, 518
- \leftskip elastische Länge, 191, 192
- \leftslices (stmaryrd), 546
- \leftthreetimes (amssymb), 546
- \leftthumbedown (dingbat), 411
- \leftthumbsup (dingbat), 411
- legal Option (crop), 221
- legalpaper Option, 204 (typearea), 212
- legalpaper Schlüssel/Option (geometry), 215
- Legenden
 - Gleitobjekte, *siehe* Gleitobjekte, Legenden
 - mehrseitige Tabellen, 265, 270
 - Quellenverweise in, 722
 - wortwörtliche Ausgabe, 183
- Leitpunkte
 - in Überschriften, 44–1138
 - in Indexen, 684
 - in Inhaltsverzeichnissen, 63
 - in Überschriften, 45
- lem Umgebung (lb2), 147
- \lengthtest (ifthen), 158, 316, 908, 909
- Lenny Option (fncychap), 36, 37
- \leq, 548 (eulervm), 410
- leqno Option (amsmath), 478, 481, 483, 484, 496
- \leqq (amssymb), 548
- \leqslant (amssymb), 548
- Lesen von Verbatim-Daten, 172
- \lessapprox (amssymb), 548
- \lessdot (amssymb), 546
- \lesseqgtr (amssymb), 548
- \lesseqgtr (amssymb), 548
- \lessgtr (amssymb), 548
- \lesssim (amssymb), 548
- \let, 171, 257, 317, 514, 603, 703
- \Letter (ifsym), 416
- letter Option (crop), 221
- letter Dokumentenklasse, 7, 25, 561
- letterpaper Option, 204, 205, 915 (typearea), 212
- letterpaper Schlüssel/Option (geometry), 215
- \lettrine (lettrine), 106, 107, 108
- lettrine Paket, 106–108
- lettrine.cfg Datei (lettrine), 108
- \LettrineFontHook (lettrine), 106, 107
- \LettrineTextFont (lettrine), 107
- level Schlüsselwort (makeindex), 680, 681, 683
- \levelchar (doc), 855
- lexikalische Analyse, Bibliographien, 806
- \lfloor, 511, 553
- \lfoot (fancyhdr), 232, 233
- \lg, 513
- \Lgem (babel), 569
- \lgem (babel), 569
- LGR Fontkodierung, 584, 591, 592, 593
- \lgroup, 502, 511, 553
- lhang Schlüssel (lettrine), 107, 108
- \lhd (latexsym), 476
- LHE Fontkodierung, 591, 595
- \thead (fancyhdr), 229, 232, 233, 616
- \lhook, 551
- LICENSE.TL Datei, 1009
- LICR (L^AT_EX Internal Character Representation, interne Zeichendarstellung), 454–456
- Liste der Objekte, 467–475
- \Lightning (ifsym), 416
- \lightning (stmaryrd), 544
- \lightrulewidth feste Länge (booktabs), 278
- \lim, 504, 513
- Platzierung von Hoch-/Tiefstellungen, 503, 504
- \liminf, 513
- \limits, 504, 505
- Fehler bei, 941
- \limsup, 513
- \Line (pspicture), 659, 660
- \line, 619, 625, 626
- Fehler bei, 930
- Warnung bei, 969
- (eepic), 626, 627, 629
- (epic), 626
- (pspicture), 658, 659, 660
- line_max Schlüsselwort (makeindex), 682
- \linebreak, 109, 877, 989
- (soul), 96, 97
- \linelabel (lineno), 187, 188
- \LineLoad (marvosym), 413
- lineno Paket, xxix, 185–190, 191
- in diesem Buch, 1136
- linenomath Umgebung (lineno), 187
- linenomath* Umgebung (lineno), 187
- \LineNumber (lineno), 190
- \linenumberfont (lineno), 188, 189
- \linenumbers (lineno), 185, 186, 187–191
- linenumbers Umgebung (lineno), 186
- \linenumbersep feste Länge (lineno), 188, 189, 191
- \linenumberwidth feste Länge (lineno), 188
- lineonmath Umgebung (lineno), 187
- \lineref (lineno), 188
- lines Schlüssel (lettrine), 107, 108

- lines Schlüssel/Option (geometry), 215
- lines Schlüsselwert
 - (fancyvrb), 166, [167](#)
 - (listings), 182
- \lineskip elastische Länge, [982](#)
- \linespread, 114, 212, [383](#)
- \linethickness, 629, [630](#)
 - (epic), 620
 - (picins), [314](#)
 - (pspicture), 657–659, [660](#)
- \linewidth feste Länge, [166](#), [203](#), [250](#), [258–260](#), [336](#), [642](#), [890](#), [900](#), [902–904](#)
 - (multicol), 195
- Linien, *siehe auch* Boxen; Rahmen
 - doppelte, 278
 - Farben, [273](#)
 - formale, 277, [278](#), 279, [280](#)
 - Fußnoten, 119, [126](#), 127
 - gepunktete, [620](#), 889
 - Gleitobjekte, 293, 294
 - in Tabellen
 - doppelte, 278
 - farbige, [273](#)
 - formale, 277, [278](#), 279, [280](#)
 - gestrichelte, 275, [276](#)
 - horizontale und vertikale kombinieren, 274, [275](#)
 - variable Stärke, [274](#)
 - vertikale, [274](#), 275, 277
 - in Überschriften, 44, [45](#)
 - Kolumnentitel-Layouts, 233
 - Linienstärke, [622](#)
 - Rahmen, Farben, 166
 - um Programmlistings, 182, 183
 - zeichnen, 621, [622](#), 628, *siehe auch* epic-Paket; eepic-Paket
- Linienboxen, 893, [899–901](#)
- \lining (fourier), 404
- Links in Querverweisen, 82
- linksbündige Absätze, 110, 111, [113](#), 114
- \lips
 - (lb2), [88](#)
 - (lips), [88](#), [89](#)
- lips Paket, 88, 89, *siehe auch* ellipsis-Paket
- list Umgebung, 152, 154, [155](#), 159, [883](#), 891
 - Fehler bei, 951
 - Stilparameter, 153
- List-Stack (T_EX Programm), anzeigen, 990
- \listdesign (layouts), 210
- \listdiagram (layouts), 210
- Listen
 - Arten, 135
 - Aufzählungen
 - Erweiterungen, [143](#)
 - Layoutparameter, 136
 - Standard, 135, [136](#)
 - Voreinstellungen, [144](#), 145, 146
 - benutzerdefinierte, 152–154
 - beschreibende Listen, 155, [156–158](#), 159
 - nummerierte Listen, [159](#)
 - Zitate, 154, [155](#)
- Listen (*Forts.*)
 - beschreibende
 - benutzerdefinierte, 155, [156–159](#)
 - Erweiterungen, 143, [144](#)
 - Standard, [139](#)
 - Voreinstellungen, 145, 146
 - eingerahmte, [618](#)
 - fremdsprachige Dokumente, [581](#), 582
 - Layoutparameter, allgemeine, 153
 - mit Überschriften, 146, [147](#)
 - Abstände, 149
 - anpassen, 148–150, [151](#)
 - benennen, 148, 149, [150](#), [151](#)
 - Beweisführungen, 151, 152
 - Einzug, 149
 - Fonts, 149
 - Interpunktion, 149
 - Layout definieren, [148](#)
 - Layoutnamen, 149
 - QED (□)-Zeichen, 151, 152
 - mit Aufzählungszeichen, *siehe* Listen, Aufzählungen
 - nicht nummerierte, *siehe* Listen, Aufzählungen
 - nummerierte, *siehe auch* theoremmähnliche Strukturen
 - benutzerdefinierte, [159](#)
 - Einzug, [145](#)
 - Erweiterungen, [139–143](#)
 - Layoutparameter, 137
 - Standard, 136, [137](#), [138](#)
 - Voreinstellungen, 144, [145](#), 146
 - schematische Darstellung, 153
- \listfigurename, 37, [56](#)
 - (babel), 561
 - in diesem Buch, [14](#)
- \listfiles, 23, 917
- listing Umgebung (moreverb), [161](#)
- listing* Umgebung (moreverb), [161](#)
- listingcont Umgebung (moreverb), [161](#)
- listingcont* Umgebung (moreverb), [161](#)
- listings Paket, xxviii, 162, [177–184](#), 317
 - kombiniert mit color, 180
 - kombiniert mit inputenc, 184
- \listof (float), 58, [302](#)
- \listofexamples (lb2), [58](#)
- \listoffigures, 25, [49](#), 56, 58, 231, 302, [316](#)
 - im Inhaltsverzeichnis, 51
 - (caption), [324](#)
 - (subfig), [331](#)
- listofformat Schlüssel/Option (subfig), 328, [330](#)
- listofindent Schlüssel/Option (subfig), 330, [331](#)
- listofnumwidth Schlüssel/Option (subfig), 330
- \listoftables, 25, [49](#), 56, 58, 231, 267, 302
 - im Inhaltsverzeichnis, 51
- \listparindent feste Länge, 153
- lists Option (endfloat), 298
- \listtablename, 37
 - (babel), 561
- literate programming, 845–857
- Literaturdatenbankdateien, 8
- Literaturverzeichnisse, *siehe* Bibliographien

- Lizenzdaten
 - beigelegte CD, 1009
 - LaTeX Project Public License (LPPL), 5, 1009
 - multicol-Paket, 193
 - Sprachdefinitionsdateien, 598
- \ll, 548
- \llap, 189, [190](#)
- \llbracket
 - (fourier), [403](#)
 - (stmaryrd), 511, 553
- \llceil (stmaryrd), 554
- \llcorner (amssymb), 554
- \Lleftarrow (amssymb), 550
- \llfloor (stmaryrd), 554
- \lll (amssymb), 548
- \lless (amssymb), 548
- \llparenthesis (stmaryrd), 554
 - lmargin Schlüssel/Option (geometry), [214](#), [216](#)
 - lmodern Paket, 368
- \lmoustache, 511, 553
- \ln, 513
 - ln Option (graphics), 633
- \lnapprox (amssymb), 548
- \lneq (amssymb), 548
- \lneqq (amssymb), 548
- \lnot, 544
- \lnsim (amssymb), 548
- \LoadClass, [911](#), [919](#), [920](#), [921](#)
 - Fehler bei, 941, 947, 952
 - Warnung bei, 976
- \LoadClassWithOptions, 916, [920](#), [921](#)
 - loading Option (tracefmt), [380](#), 993
 - loadonly Option (titlesec), 47, [48](#)
 - location BibTeX-Feld (BibTeXMng), 820
- .lot Dateierweiterung, [8](#), [9](#), [49](#), 51, 56
 - (chapterbib), 775
 - (subfig), 330
 - (titletoc), 62
- lotdepth Zähler (subfig), 330
- \log, [505](#), 513
 - .log Dateierweiterung, [8](#), [9](#), 679
 - logonly Option (trace), 993
- \long, [878](#), [979](#)
 - long Option (rcsinfo), 872
 - long Schlüsselwert (jurabib), [757](#), [758](#)
- \Longarrownot (stmaryrd), [551](#)
- \longarrownot (stmaryrd), 551
- \longgather Umgebung (lb2), [480](#)
- \Longleftarrow, 550
- \longleftarrow, 550
- \Longleftrightharrow, 550, [895](#)
- \longleftrightharrow, 550, [551](#)
- \longmapsfrom (stmaryrd), 550
- \longmapsfrom (stmaryrd), 550
- \Longmapsto (stmaryrd), 550
- \longmapsto, 550
 - longnamesfirst Option (natbib), [729](#), [730](#)
 - Probleme mit, [730](#)
- \longpage (lb2), [242](#)
- \Longrightarrow, 550
- \Longrightarrow, [501](#), 550
- longtable Umgebung (longtable), 267, [268](#), [269–272](#), 278, 285
 - „gleitend“, 298
- longtable Paket, [267–271](#), 317
 - kombiniert mit booktabs, 278
 - kombiniert mit caption, 270
- lookat Schlüssel/Option (jurabib), [752](#), [753](#), 754, [767](#)
- \lookatprefix (jurabib), 753
- \lookatsuffix (jurabib), 753
- lookforgender Schlüssel/Option (jurabib), [761](#)
- looktex Programm, 805
- \looparrowleft (amssymb), 550
- \looparrowright (amssymb), 550
- loose Option
 - (minitoc), 59
 - (shorttoc), 59
- \looseness, 989
 - erzeugte Ausgabe, [989](#)
 - in diesem Buch, 1137
- \lor, 546
- .lot Dateierweiterung, [8](#), [9](#), [49](#), 51
 - (chapterbib), 775
 - (subfig), 330
 - (titletoc), 62
- lotdepth Zähler (subfig), 330
- loversize Schlüssel (lettrine), [107](#), [108](#)
- \lowercase, 352
 - Probleme mit, 587
- lowersorbian Option (babel), 559
- \lozenge (amssymb), 544
- \LPNobreakList (lips), 89
 - LPPL (LaTeX Project Public License), 5, 1009
- \lproject (lb2), [101](#)
- \LR (lb2), [191](#)
 - LR-Boxen, 893, [894](#), [895](#)
 - lraise Schlüssel (lettrine), [107](#), [108](#)
 - lrbox Umgebung, [902](#), [903](#)
- \lrcorner (amssymb), 554
- lscape Paket, 220
- \Lsh (amssymb), 550
 - lsorbian Option (babel), [575](#)
- \lsthline (listings), [180](#)
- \lstinputlisting (listings), [180](#), [181](#), [183](#), [184](#)
 - lstlisting Umgebung (listings), [179](#), [181](#), [182](#), [184](#)
 - lstlisting Zähler (listings), 183
- \lstlistingname (listings), 183
- \lstlistlistingname (listings), 183
- \lstlistoflistings (listings), 183
- \lstloadlanguages (listings), [178](#), [179](#), [180](#)
- \lstset (listings), [178](#), [179](#), [180–184](#)
- \Lsub (lb2), [33](#)
- \LTcapwidth feste Länge (longtable), 270
- LTchunksize Zähler (longtable), 271
- \ltimes (amssymb), 546
- \Lleft elastische Länge (longtable), [268](#), [269](#)
 - ltoutenc.dtx Datei, 379
- \LTpost elastische Länge (longtable), 269
- \LTpre elastische Länge (longtable), 269
- \LTright elastische Länge (longtable), [268](#), [269](#)
- .ltx Dateierweiterung, 9
 - (lb2), 15, 1008

- .ltx2 Dateierweiterung (lb2), 15, 1008
- ltxdoc Dokumentenklasse, 850, 857, **867**, 868
- ltxdoc.cfg Datei (ltxdoc), 868
- Lucida Bright Font, in Formeln und Text, **537**
- lucidabr Paket, 350, 537
- luximono Paket, 398, 399
- LV1 Fontkodierung, 428
- \lvec (lb2), **877**, 878, **879**, **978**, **979**
- \lvert (amsmath), **511**, **514**, **553**
- \lvert (amsmath), **511**, **512**, **514**, **553**
- \lvertneqq (amssymb), 548
- LY1 Fontkodierung, 428
 - Liste der LICR-Objekte, 467–475
 - (pxfonts), 401
 - (txfonts), 400
- M**
- m Syntax, 253
 - (array), 252, **253**, 258
 - (tabularx), **260**
- M-xcompile Funktion (emacs), 817
- \m@ne, 875
- macce Option (inputenc), 371
- maccyr Option (inputenc), 587
- MACRO B_UL_AT_EX-Befehl, 836, **838**, **843**
- macro Umgebung (doc), **848**, 850, 854, 856, 857
- macrocode Umgebung (doc), **847**, **848**, 849, 850, 854, 856
- macrocode* Umgebung (doc), 847, 849, 854
- \MacrocodeTopsep elastische Länge (doc), 856
- \MacroFont (doc), 856
- \MacroIndent feste Länge (doc), 856
- \MacroTopsep elastische Länge (doc), **848**, **856**
- mag Schlüssel/Option (geometry), 218
- magyar Option (babel), 559
- \main (doc), 855
- \mainmatter, 24, 224
- Majuskelziffern, *siehe* Versalziffern
- make-rules Programm, 693
- \makeatletter, **16**, 20, 27, **29**, **121**, **137**, **717**, **875**, **884**
- \makeatother, **16**, 20, 27, **29**, **121**, **137**, **717**, **875**, **884**
- makebib Programm, 805
- \makebox, **120**, **166**, **250**, 867, **893**, **894**, **895**
 - ohne horizontale Ausdehnung, **134**, **155**, **192**, **647**
 - (fancybox), **615**
 - (ltxdoc), 867
 - (pspicture), 658
- makebst Programm, 709, 729, 733, 736
- makebst.tex Datei (custom-bib), **828**, **829–835**
- \makeenmark (endnotes), **133**
- \makeglossary, 673
- makeidx Paket, 669, 672, 676
- \makeindex, 669, 675
 - MakeIndex Programm, 8, 9, 589, 590, 668, 670, 672, 673, **675–688**, 860, 1137, *siehe auch* Indexerstellung; xindy-Programm
 - fremdsprachige Dokumente, 589
 - kyrillisches Alphabet, 589
- \makelabel, 153, **155**, **156**, **157**, 158, 883
- \makeLineNumber (lineno), **189**, **190**
- \makeLineNumberRight (lineno), 188
- \MakeLowercase, **40**, **67**, **92**, **352**, 587
 - (fontenc), 373
- \MakePercentComment (doc), 856
- \MakePercentIgnore (doc), 856
- \MakePerPage (perpage), **127**, **128**, **132**
- \MakePrivateLetters (doc), 856
- \MakeShortVerb
 - (doc), 848, 854
 - (shortverb), **160**, 918
- \MakeShortVerb*
 - (doc), 848, 854
 - (shortverb), **160**
- \MakeTextLowercase (textcase), 92
- \MakeTextUppercase (textcase), **92**, 93
- \maketitle, 24
 - erzeugt unerwünschte Seitenzahl, 231, 238
 - Fehler bei, 945
 - Warnung bei, 968
- \MakeUppercase, **92**, **93**, 237, 587, **796**
 - in Überschriften, **98**, **702**, **703**
 - in Überschriften, **33**
 - (fontenc), 373
 - (textcase), **93**
- Makro-Stack (T_EX Programm), anzeigen, 926
- Makros
 - benennen, 874
 - Beschreibungen erstellen, 847, 848
 - dokumentieren, *siehe* Dokumentationswerkzeuge
 - Leerraum nach Makronamen, **86**, **87**
 - Querverweise, 849, 850
- \maltese (amssymb), 544
- Mandschu (Mongolisch), 608
- manjutex Paket, 608
- manual B_UL_AT_EX-Publikationstyp, **714**, **791**, 793, **808**, **809**
- manyfoot Paket, xxix, **129–132**
- .map Dateierweiterung, 432
- \Mapsfrom (stmaryrd), 550
- \mapsfrom (stmaryrd), 550
- \Mapsfromchar (stmaryrd), 551
- \mapsfromchar (stmaryrd), 551
- \Mapsto (stmaryrd), 550
- \mapsto, 550
- \Mapstochar (stmaryrd), 551
- \mapstochar, 551
- \marg (ltxdoc), 867
- margin Schlüssel/Option
 - (caption), **318**, 327
 - (geometry), **219**, **221–223**
 - (subfig), **326**, 327
- marginal Option (footmisc), **125**, **132**, **754–756**
- Marginalien, **134**, 135, 218, *siehe auch* Endnoten
 - Fußnoten als, **125**, **126**
- Marginalsatzspalte, 207
- margincaption Option (sidecap), 333, **335**
- \marginlabel (lb2), **134**
- \marginpar, 110, **134**, 135, 187, 230, 896
 - Fehler bei, 936, 946, 952
 - mit Randausgleich, 113

- `\marginpar` (*Forts.*)
 - Probleme mit Trennung, [134](#)
 - seitenweise nummeriert, 128
 - Warnung bei, 967
 - (lineno), 186
 - (mparhack), 135
 - (multicol), nicht unterstützt, 198
 - (perpage), seitenweise nummeriert, [128](#)
 - (titlesec), Probleme mit, 40
- `\marginparpush` feste Länge, 135, 203, 205
- `\marginparsep` feste Länge, 135, 203, 205, [311](#)
 - (fancyhdr), [235](#)
 - (sidecap), 334
- `marginparsep` Schlüssel/Option (geometry), 218
- `\marginparswitchfalse` (layouts), 209
- `\marginparwidth` feste Länge, 135, 203, 205, [207](#), [211](#), [311](#)
 - (fancyhdr), [235](#)
- `marginparwidth` Schlüssel/Option (geometry), [214](#), [217](#), 218
- `marginratio` Schlüssel/Option (geometry), [214](#), 219
- `\mark`, 226
- `\markboth`, 226, [227](#), [228](#), 229, 231, 237, [238](#)
 - Fehler bei, 927
- `markers` Option (endfloat), 299
- Markieren von ausgelassenem Text, *siehe*
 - Auslassungspunkte
- Markierungen im Text, Gleitobjektplatzierung, [299](#), 300
- `\markright`, 226, [227](#), [228](#), 231, 237, [238](#), [240](#)
 - Fehler bei, 927
 - (extramarks), 229
- `markshow` Option (multicol), 197
- `markup-location` Funktion (xindy), [701](#)
- `markup-location-list` Funktion (xindy), [697](#)
- `marvdoc.pdf` Datei (marvosym), 412
- `marvosym` Paket, xxix, [412–414](#), [422](#), [423](#)
- `MarVoSym` Font, 412, [413](#), [414](#)
- Master-Skripts anlegen, 861, 862
- `mastersthesis` B^BL^AT^EX-Publikationstyp, 791
- `math` Option (inputenc), 459
- `\mathalpha`, [410](#), [446](#), [447](#), 540
- `\mathbb`
 - (amsfonts), 480, [523](#)
 - (amssymb), 523
 - (fourier), [403](#)
 - (lb2), [448](#), [523](#)
 - (mathpazo), [390](#)
- `\mathbf`, 361, 364, [484](#), [485](#), [487](#), [505](#), [508](#), [517](#), 522, [524](#), [525](#)
 - (bm), 524
 - (mathptmx), 388
- `\mathbin`, [91](#), [447](#), [541](#), [545](#), [546](#), [547](#)
 - (bm), 526
 - (relsize), 91
- `\mathcal`, [360](#), [361](#), 362, [408](#), [496](#), [501](#), [507](#), [508](#), [514](#), [521](#), 522, [524](#)
 - (ccfonts), [396](#)
 - (eucal), 408, 480
 - (eulervm), [409](#)
 - (fourier), [403](#)
- `\mathcal` (*Forts.*)
 - (mathpazo), [389](#)
 - (mathptmx), [388](#)
 - (pxfonts), [402](#)
 - (txfonts), [399](#)
- `\mathclose`, [447](#), 511, [541](#), [553](#), [554](#)
- `\mathdollar`, 544
- `\mathellipsis`, 552
- Mathematikfonts, *siehe auch* Fonts
 - Alphabetbefehle, [360–362](#)
 - AMS, 479, 480
 - automatischer Wechsel, 359, 360
 - Baskerville Math, [536](#)
 - Bitstream Charter Math, [536](#)
 - Blackboard Bold, [390](#), [523](#), 535
 - CM Bright, [538](#)
 - Computer Modern (CM), [528](#)
 - Concrete, [530](#)
 - Eingabekodierung, 458, 459
 - einstellen, 444–449
 - Euler, [407](#), [408–410](#), 480, [530](#)
 - Euler Fraktur, 480, [523](#)
 - Fettdruck, [524–527](#), 528
 - Fontbefehle, [363](#)
 - Formellayouts, [363](#), 364
 - Fourier-GUTenberg, 402, [403](#), [404](#), [531](#)
 - große Operatoren skalieren, 379
 - Helvetica Math, [538](#)
 - Info Math, [539](#)
 - Lucida Math, [537](#)
 - Palatino, [389](#), [390](#), 401, [402](#), [534](#)
 - Palatino Math, [535](#)
 - Pazo, [389](#), [390](#), [523](#), [534](#)
 - Pi, 390, 391, [392](#), 393
 - PX Fonts, [534](#)
 - Times Roman, 387, [388](#), 389, [399–401](#), [532](#)
 - TM Math, [533](#)
 - TX Fonts, [532](#)
- Mathematiksatz, *siehe auch* $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\mathcal{L}\mathcal{A}\mathcal{T}\mathcal{E}\mathcal{X}$; *die einzelnen mathematischen Elemente*
- Feinabstimmung
 - Abstände, 514, 515, [516](#), [518–520](#), 521
 - Ausrichtung, [518–520](#), 521
 - big-g (große) Begrenzungssymbole, [517](#)
 - horizontale Abstände, 521, 522
 - skalieren, 514, 515, [516](#)
 - stauchen, [519](#), 520, 521
 - Unterformeln, 516, 517
 - Wurzelexponenten verschieben, 517, [518](#)
- Operatornamen, [512](#), 513, [514](#)
- Symbolklassen, 540
- Text im, [512](#), 513, [514](#)
- mathematische Alphabetbefehle, *siehe*
 - Alphabetbefehle
- mathematische Symbole, *siehe auch* Sonderzeichen; Textsymbole
 - Abstände zwischen, [541](#), [542](#), [545](#)
 - Akzente, [545](#)
 - als hochgestellte Zeichen, [507](#)
 - Arten, 540
 - Auslassungspunkte (...), 508, [509](#)

mathematische Symbole (*Forts.*)

Begrenzungsymbole, 502, 510, [511](#), 512, [517](#)
 beschriftete Pfeile, [502](#)
 Beschriftungen, 507, [508](#)
 binäre Operatorzeichen, 547
 Brüche, [505](#), [506](#)
 Buchstaben, 542–545
 Formeln, gerahmte, [503](#), [618](#)
 Hochstellungen
 über Relationssymbolen, 507
 Positionieren von Grenzen, [503](#), [504](#)
 horizontal erweiterbare, 509, [510](#), [512](#)
 Integrale, Mehrfach-, 504, [505](#)
 Interpunktion, 552
 Kettenbrüche, [503](#)
 konfigurieren, 445–449
 $\LaTeX$ 2.09, 476
 $\backslash\mathbin$
 Boxen, [546](#)
 Kreise, [547](#)
 sonstige, [546](#)
 $\backslash\mathclose$ (öffnen/schließen), [511](#), [553](#), [554](#)
 mathematischer Typ, 540
 $\mathinner$ (Interpunktion), [552](#)
 $\mathop$ (große Operatoren), [552](#)
 $\mathopen$ (öffnen/schließen), [511](#), [553](#), [554](#)
 $\mathord$
 buchstabenförmige, [544](#)
 griechische, [543](#)
 Interpunktion, [552](#)
 sonstige, [544](#)
 $\mathpunct$ (Interpunktion), [552](#)
 $\mathrel$
 Gleichheits- und Relationszeichen, [548](#)
 Gleichheits- und Relationszeichen, negierte, [548](#)
 Mengenoperatoren, [549](#)
 Mengenoperatoren, negierte, [549](#)
 Negationen und Pfeilerweiterungen, [551](#)
 Pfeile, [550](#)
 Pfeile, negierte, [551](#)
 sonstige, [551](#)
 Modulo-Operationen, [505](#)
 neue hinzufügen, 445–449
 öffnende/schließende Symbole, 553
 Operatoren, [502–505](#), 506, 507, [508](#)
 in fremdsprachigen Dokumenten, [580](#)
 Operatorsymbole, 553
 Positionieren von Hoch-/Tiefstellungen, [503](#), [504](#)
 Punkte als Akzente, [507](#)
 reguläre Zeichen, 542–544, [545](#)
 Relationssymbole, [547](#), 548, [549](#), 550, 551
 Symbolklassen, 540, [541](#), [542](#), [545](#)
 Tiefstellungen, Positionieren von Grenzen, [503](#), [504](#)
 Typen, 540
 variable Formen, 508, [509–512](#)
 Verallgemeinerungen, [505](#), [506](#)
 vertikal erweiterbare, 510, [511](#), [512](#)
 Wurzeln, 517, [518](#)

mathematische Symbole (*Forts.*)

Ziffern, 542–545
 zusammengesetzte, [502–507](#), 508
 $\backslash\mathfrak$
 (amsfonts), [480](#), [523](#)
 (amssymb), 523
 (eufrak), [408](#), [410](#), [480](#)
 (eulervm), [409](#)
 $\backslash\mathindent$ elastische Länge (amsmath), [482](#), 483, [512](#)
 $\backslash\mathindent$ feste Länge, 483
 $\backslash\mathinner$, 511, [540](#), 552
 $\backslash\mathit$, [360](#), [361](#), 476
 $\backslash\mathlarger$ (relsize), [91](#)
 mathlines Option (lineno), [187](#)
 $\backslash\mathnormal$, 360, [361](#), 362
 (eulervm), 409
 $\backslash\mathop$, 91, [447](#), [541](#), [552](#)
 (amsmath), 504
 (bm), 526
 $\backslash\mathopen$, [447](#), 511, [541](#), [553](#), [554](#)
 $\backslash\mathord$, [447](#), 486, 511, [541](#), [543–545](#), [550](#), [551](#), [552](#)
 $\backslash\mathparagraph$, 544
 mathpazo Paket, [382](#), [383](#), [389](#), [390](#), 535
 mathpple Paket, [382](#), [383](#), 389
 mathptm Paket, [382](#), [383](#), [388](#), 389
 mathptmx Paket, [382](#), [383](#), [387–389](#), 399, 401, 533
 kombiniert mit tipa, 417
 $\backslash\mathpunct$, [447](#), [541](#), 552
 $\backslash\mathrel$, 91, [447](#), [486](#), 511, [541](#), [545](#), [548](#), [549](#), 550, 551
 (amsmath), 516
 (bm), 526
 $\backslash\mathring$, 545
 $\backslash\mathrm$, [361](#), [362](#), [501](#), 512
 $\backslash\mathscr$
 (eucal), [408](#)
 (eulervm), nicht vorhanden, [409](#)
 (lb2), 523, [524](#)
 mathscr Option (eucal), 408
 $\backslash\mathsection$, 544
 $\backslash\mathsf$, [360](#), [361](#), [362](#), [364](#), 476
 (eulervm), 410
 $\backslash\mathsmaller$ (relsize), [91](#)
 $\backslash\mathsterling$, 544
 $\backslash\mathstrut$, [518](#)
 mathtime Paket, 363
 $\backslash\mathhtt$, 361
 $\backslash\mathversion$, [363](#), 364
 Fehler bei, 941
 matrix Umgebung (amsmath), 498, [499](#)
 Fehler bei, 941, 946
 matrixähnliche Umgebungen
 äußere Begrenzungszeichen, [502](#)
 cases-Umgebung, 498
 Einzelgleichungen, wenige Varianten, [498](#)
 Matrixumgebungen, 498, [499](#), [500](#)
 mehrzeilige Hochstellungen, [500](#)
 mehrzeilige Tiefstellungen, [500](#)
 $\backslash\matrixput$ (epic), [625](#)
 $\backslash\max$, 504, [513](#), [542](#)

- `\maxdimen` feste Länge, 94
 - (tabulary), [261](#)
- `\maxfiles` (docstrip), 865
- `MaxMatrixCols` Zähler (amsmath), [499](#)
- `\maxoutfiles` (docstrip), 865, 866
- `\maxovaldiam` feste Länge (eepic), [627](#)
- `\mbox`, [156](#), 512, 526, [876](#), [893](#), 903
 - Material in `\discretionary` verstecken, 182
 - zum Unterdrücken der Silbentrennung, [719](#)
 - (bm), 526
 - (soul), [96](#)
 - (ulem), 93
- `md` Option (titlesec), 39
- `md` Schlüsselwert (caption), [319](#)
- `\mddefault`, 357
- `\mdqoff` (babel), [565](#)
- `\mdqon` (babel), [565](#), 910
- `\mdseries`, 351, 355, 357
- `\meaning`, [980](#)
- `\meas` (lb2), [514](#)
- `\measuredangle` (amssymb), 544
- Mediävalziffern, 67, [378](#)
 - erzeugen, 378, 394
 - in Computer Modern, 394, [395](#)
 - in Palatino, 390
 - nicht verfügbar, 969
- medieval Attribut (babel), 565
- medium Option (titlesec), 39
- `\medmuskip` elastische Länge, [521](#), 541, [542](#)
- `\medskip`, 891
 - in Überschriften, [33](#)
- `\medskipamount` elastische Länge, 891
- `\medspace` (amsmath), 521, 522
- mehrere Bibliographien, 771–783
- mehrfache Inhaltsverzeichnisse, [58](#), 59–62
- mehrfache Quellenverweise, [728](#), [729](#)
- Mehrfachindexe, 704, 705
- Mehrfachintegrale, 504, [505](#)
- mehrseitige Tabellen
 - Abstände um, 269
 - Anzahl der Läufe reduzieren, 271
 - Breite, 266, 267, [268](#), 269, 270, [271](#), 272
 - erstellen mit `longtable`, 267, [268](#), 269, [270–272](#)
 - erstellen mit `supertabular`, 264, [265](#), [266](#), 267
 - Fußnoten in, 270
 - horizontale Ausrichtung, 268
 - Kolumnentitel, 265, 269
 - Legenden, 265, [270](#)
 - Probleme mit, [271](#), [272](#)
 - Seitenumbrüche in, 265
 - und Gleitobjekte, [270–272](#)
 - Zeilenbefehle, 269
- mehrsprachig, *siehe* fremdsprachig
- Meldungen, *siehe auch* Fehlersuche
 - Anwendermeldungen erzeugen, 860, 861
 - informelle, 961–977
 - memory exceeded, 956–961
 - Warnungen, 961–977
- Meldungen, Fehler-
 - farbige, im Bibliographie-Frontend, 815
 - Meldungen, Fehler- (*Forts.*)
 - Index
 - Liste von (*MakeIndex*), 679–681
 - unterdrücken, 678, 690, 697
 - Liste der, 928–956
 - * (Stern), 929
 - Quellcodezeile finden, 924–928
 - Syntax, 924
 - Meldungen, Indexerstellung
 - Fehlermeldungen
 - Liste von (*MakeIndex*), 679–681
 - unterdrücken, 678, 690, 697
 - Fehlersuche, 697
 - memo Kolumnentitel-Layout (lb2), [238](#)
 - memoir Dokumentenklasse, 210, [245](#), 725
 - memory exceeded Meldung, 956–961
 - Mengenoperatoren – negierte, mathematische Symbole, [549](#)
 - Mengenoperatoren, mathematische Symbole, [549](#)
- `\merge` (stmaryrd), 546
- `merge-rule` Funktion (xindy), [699](#)
- `merge-to` Funktion (xindy), [701](#)
- `merlin.mbs` Datei (custom-bib), 829, 834
- `\MessageBreak`, [918](#), 919
- `\meta` (doc), [847](#), 854
- METAFONT, 345
- `\MetaPrefix` (docstrip), 862, [866](#)
- MFpic Programm, 1019
- `\mho`
 - (amssymb), 544
 - (latexsym), [476](#)
- `\mid`, [523](#), [551](#)
- `\middle`, verfügbar unter eTeX, 510, 511, 517, 553
- `\midrule` (booktabs), [278](#), 279, [280](#)
- `\midrulesep` feste Länge (booktabs), 279
- `\midwordellipsis` (ellipsis), 88
- `\min`, 513
- `\minalignsep` (amsmath), [488](#), [489](#), 490, [491](#)
- `\minilof` (minitoc), 60
- `\minilot` (minitoc), 60
- minipage Umgebung, 117, [896](#), [897](#), [898](#), [899](#), [902](#), [903](#)
 - Fußnoten in, [117](#), [118](#), 120, [285](#)
 - mit Randausgleich, 111, 113
 - verschachtelt, 897
 - (`supertabular`), 264
- `\minitoc` (minitoc), 60, 62
- minitoc Paket, xxix, 59–62, *siehe auch* titletoc-Paket
- minitocdepth Zähler (minitoc), 60
- `\minus` (euro), [105](#), [106](#)
- minus Syntax, [67](#), [97](#), 426, 719, [887](#)
- Minuskelziffern, *siehe* Mediävalziffern
- `\minuso` (stmaryrd), 546
- `mirror` Option (crop), [222](#), [223](#)
- `misc` Option (ifsym), [416](#)
- `misc` B^BTeX-Publikationstyp, 791
- `missing$` interne B^BTeX-Funktion, 839, [841](#)
- Mittellänge, 440
- `mktexlsr` Programm, 935
- ML Umgebung (lb2), [147](#)

- mla Option
 - (ellipsis), 88
 - (lips), [89](#)
 - MLBTeX Programm, 789
 - .mlf(*n*) Dateierweiterung (minitoc), 60
 - mlt Umgebung (lb2), [480](#)
 - .mlt(*n*) Dateierweiterung (minitoc), 60
 - mmode boolesche Var., 908
 - mnk Option (inputenc), 587
 - mnote Zähler (lb2), [128](#)
 - \Mobilefone (marvosym), [413](#)
 - \mod (amsmath), [505](#)
 - \models, 551
 - \Module (doc), 856
 - modulo Option (lineno), 188
 - Modulo-Operationen, mathematische Symbole, [505](#)
 - \modulolinenumbers (lineno), [188](#), [191](#)
 - Mongolisch (Mandschu), 608
 - Monospace-Schriften, *siehe* dicktengleiche Schriften
 - monotoniko Attribut (babel), 566, 590, 591
 - month BibTeX-Feld, [714](#), [791](#), [793](#), [799](#)
 - \moo (stmaryrd), 546
 - \morecmidrules (booktabs), [279](#), [280](#)
 - moredefs Paket, 88, 89
 - morefloats Paket, 952
 - moreverb Paket, 161
 - mos Option (inputenc), 587
 - Mottos (Zitate) für Kapitelüberschriften, 37, [38](#)
 - \Mountain (ifsym), [416](#)
 - \movetoevenpage (nextpage), 244
 - \movetooddpag (nextpage), 244
 - \mp, 546
 - mparhack Paket, 135
 - mpexclude Option (typearea), 213
 - mpfootnote Zähler, 117, 883
 - (footmisc), [118](#)
 - \mpfootnotemark (footmisc), [118](#)
 - \mpfootnoterule (footmisc), 126
 - mpinclude Option (typearea), [213](#)
 - mpsupertabular Umgebung (supertabular), 264, 285
 - mpsupertabular* Umgebung (supertabular), 264, 285
 - \mrm (lb2), [361](#)
 - mrnumber BibTeX-Feld (BibTeXMng), 820
 - \msfs1 (lb2), [362](#), 364
 - \Msg (docstrip), 860
 - .msp Dateierweiterung, 644
 - \mspace (amsmath), 522
 - .mtc(*n*) Dateierweiterung (minitoc), 60
 - \mtcfont (minitoc), 60
 - \mtcindent feste Länge (minitoc), 60
 - mtcuff Paket, 62
 - \mtcpagenumbers (minitoc), 60, 61
 - \mtcPfont (minitoc), 60
 - \mtcrule (minitoc), 60, 61
 - \mtcSfont (minitoc), 60
 - \mtcSPfont (minitoc), 60
 - \mtcSSfont (minitoc), 60
 - \mtcSSSfont (minitoc), 60
 - \mtctitle (minitoc), 60
 - \mu, [505](#), 543
 - \multfootsep (footmisc), 127
 - multibib Paket, xxix, 772, 781, **782**, **783**
 - Kompatibilitätsmatrix, 772
 - multicol Paket, 185, **193–198**, 240, 308
 - in diesem Buch, 1137
 - Lizenzdaten, 193
 - \multicolpretolerance (multicol), 195, 196
 - multicols Umgebung (multicol), 193, [194](#), [195–198](#), [703](#), 896, 908
 - Stilparameter, 194–196
 - multicols* Umgebung (multicol), 195, 917
 - \multicolsep elastische Länge (multicol), [194](#), 195
 - \multicoltolerance (multicol), 195, [196](#)
 - \multicolumn, [280](#), 281, [282](#), [284](#), [285](#), [287](#), [288](#)
 - Fehler bei, 938, 942, 943
 - (longtable), [268](#)
 - (supertabular), [265](#), [266](#)
 - (tabularx), [290](#)
 - Einschränkungen, 260
 - \multimap (amssymb), 550
 - multiple Option (footmisc), [127](#), [130](#), [132](#), [754–757](#)
 - multiple Schlüsselwert (jurabib), 747, 748, 761
 - \multiply, 904
 - \multiput, 619, 624, 625
 - (pspicture), 658
 - \multiputlist (epic), [624](#), [625](#)
 - \multirow (multirow), [281](#), [282](#), [290](#)
 - multirow Paket, 281, 282
 - \multirowsetup (multirow), [281](#), [282](#)
 - multiline Umgebung (amsmath), 481, 483, [484](#), [485](#)
 - Fehler bei, 930
 - multiline* Umgebung (amsmath), 481, 484
 - \multilinegap feste Länge (amsmath), [484](#), [485](#)
 - \myclearpage (lb2), 244
 - myheadings Kolumnentitel-Layout, 230, 231
 - MYitemize Umgebung (lb2), [136](#)
 - \MyRot (lb2), [649](#)
- ## N
- \n
 - (docstrip), [860](#)
 - (lb2), [89](#)
 - \nabla, 544
 - Nachspann erzeugen, Codeerzeugung, 862, 863
 - \name (lb2), [352](#)
 - name Schlüssel
 - (listings), [181](#)
 - (titlesec), [46](#), [47](#)
 - name Schlüsselwert (jurabib), [755](#), [756](#), [760](#)
 - name&title Schlüsselwert (jurabib), 755
 - name&title&auto Schlüsselwert (jurabib), 755, [756](#)
 - named BibTeX-Stil
 - (chicago), 724
 - (named), 820, 822
 - named Paket, 822
 - namelimits Option (amsmath), 504
 - Namen, Bibliographiedatenbank, 795–797
 - Namenskonventionen, 874, 875
 - namunrst BibTeX-Stil, 822
 - nar BibTeX-Stil, 822
 - nar Paket, 822
 - \NAT@close (natbib), [734](#)

- `\NAT@date` (natbib), [734](#)
- `\NAT@idxtxt` (natbib), [734](#)
- `\NAT@name` (natbib), [734](#)
- `\NAT@open` (natbib), [734](#)
 - natbib Paket, xxix, [72](#), [725–740](#), [832](#)
 - Kompatibilitätsmatrix, [772](#)
 - nicht kompatibel zu cite, [739](#)
 - natbib.cfg Datei (natbib), [731](#), [735](#)
 - natheight Schlüssel (graphicx), [637](#)
- `\natural`, [544](#)
- nature $\LaTeX$ -Stil (nature), [822](#)
- nature Paket, [822](#)
- natwidth Schlüssel (graphicx), [637](#)
- naustrian Option (babel), [559](#)
- ncc Option (inputenc), [587](#)
- `\ncong` (amssymb), [548](#)
- `\Ndash` (lb2), [89](#)
- ndkeywordstyle Schlüssel (listings), [179](#)
- `\ne`, [548](#)
- `\narrow`, [550](#)
- nearskip Schlüssel/Option (subfig), [327](#), [329](#), [331](#)
- nederlands $\LaTeX$ -Stil (harvard), [725](#), [842](#)
- `\NeedsTeXFormat`, [911](#), [912](#), [920](#), [922](#), [976](#)
 - Versionsdaten, [912](#)
- `\neg`, [544](#), [545](#)
 - negierte mathematische Symbole
 - Gleichheits- und Relationszeichen, [548](#)
 - Mengenoperatoren, [549](#)
 - Pfeile, [551](#)
 - Pfeilerweiterungen, [551](#)
- `\negmedspace` (amsmath), [521](#), [522](#)
- `\negthickspace` (amsmath), [521](#), [522](#)
- `\negthinspace`, [521](#), [522](#)
- `\neg`, [519](#), [545](#), [548](#)
 - neue deutsche Rechtschreibung, *siehe* Deutsch
- neveradjust Option (paralist), [143](#)
- neverdecrease Option (paralist), [142](#), [143](#)
- New Century Schoolbook Font, [386](#)
- New Font Selection Scheme (NFSS), [338](#), [339](#)
- newapa $\LaTeX$ -Stil
 - (chicago), [724](#)
 - (newapa), [822](#)
- newapa Paket, [822](#)
- `\newblock`, [711](#), [717](#), [723](#)
 - ($\LaTeX$), [840](#)
- `\newboolean` (ifthen), [907](#), [920](#)
- newcent Paket, [382](#)
- `\newcites` (multibib), [782](#), [783](#)
- `\newcolumntype`
 - (array), [256](#), [257](#), [274](#), [576](#), [579](#)
 - (colortbl), [273](#)
 - (dcolumn), [283](#), [284](#)
 - (tabularx), [259](#)
 - (tabulary), [261](#)
- `\newcommand`, [875](#), [876](#), [877](#), [878](#), [879](#), [880](#), [916](#)
 - Fehler bei, [932](#), [938](#), [942](#), [948](#), [954](#), [977](#)
 - in .bb1-Dateien, [775](#), [800](#)
- `\newcommand*`, [878](#), [947](#), [977](#)
 - newcommands Option (ragged2e), [112](#), [764](#), [765](#)
- `\newcounter`, [159](#), [206](#), [882](#), [884](#), [885](#), [886](#), [904](#)
 - Fehler bei, [932](#), [945](#)
- `\newdatelsorbian` (babel), [575](#)
- `\newdateusorbian` (babel), [575](#)
- `\newenvironment`, [880](#), [881](#), [882](#)
 - Fehler bei, [876](#), [932](#), [938](#), [942](#), [948](#), [954](#)
- `\newfloat`
 - (float), [300](#), [301](#), [302](#), [303](#), [322](#), [329](#), [330](#)
 - (rotfloat), [307](#)
- `\newfont`, [339](#)
- `\newif`, [908](#)
- `\newindex` (index), [704](#), [705](#), [735](#), [746](#)
- `\newlength`, [887](#), [908](#), [909](#), [916](#)
 - Fehler bei, [932](#)
- newfont Paket, [476](#)
- `\newline`, [893](#)
 - Fehler bei, [951](#)
 - (amsthm), [149](#), [150](#)
 - newline Schlüsselwert (caption), [320](#)
 - newline\$ interne $\LaTeX$ -Funktion, [839](#), [841](#)
- `\newpage`, [33](#), [242](#), [298](#)
 - (longtable), [269](#)
 - (multicol), [195](#)
 - (nfssfont.tex), [381](#)
- `\newreformat` (prettyref), [80](#)
- `\newsavebox`, [96](#), [882](#), [901](#), [902](#), [903](#), [990](#)
 - Fehler bei, [932](#)
- NewsGroups, [996](#)
- `\newstylenums` (eco), [395](#)
- `\newsbfloat` (subfig), [329](#)
- `\newtheorem`
 - Fehler bei, [945](#)
 - (amsthm), [146](#), [147](#), [148](#), [150](#), [883](#)
- `\newtheorem*` (amsthm), [147](#), [148](#), [151](#)
- `\newtheoremstyle` (amsthm), [148](#), [149](#), [150](#), [151](#)
- `\newtie` (textcomp), [374](#), [378](#)
- newzealand Option (babel), [559](#)
- `\nexists` (amssymb), [544](#)
- next Option (inputenc), [371](#)
- `\nextcitefull` (jurabib), [750](#)
- `\nextcitenotitle` (jurabib), [750](#)
- `\nextcitereset` (jurabib), [750](#)
- `\nextciteshort` (jurabib), [750](#)
- nextpage Paket, [243](#), [244](#)
- NFSS (New Font Selection Scheme), [338](#), [339](#)
- nfssfont.tex Paket, [357](#), [380](#), [381](#), [447](#), [524](#)
- `\NG`, [469](#)
 - Probleme in T1, [428](#)
- `\ng`, [471](#)
 - Probleme in T1, [428](#)
- `\ngeq` (amssymb), [548](#)
- `\ngeqq` (amssymb), [548](#)
- `\ngeqslant` (amssymb), [548](#)
- ngerman Option (babel), [18](#), [37](#), [407](#), [559](#), [560](#), [562](#), [563](#), [566](#), [567](#), [569–571](#), [678](#), [684](#), [694](#), [759](#), [760](#)
- `\ngtr` (amssymb), [548](#)
- `\ni`, [549](#)
 - nicht bündige Absätze, [110–114](#)
 - nicht englische Dokumente, *siehe* fremdsprachige Dokumente

- nicht lateinische Alphabete
 - Äthiopisch, 608
 - Arabisch, 608
 - Armenisch, 608
 - Chinesisch, 608
 - Griechisch, 590, 591, [592](#), [593](#)
 - Hebräisch, [591](#)–[593](#), [594](#), [595](#), 608
 - Indisch, 608
 - Japanisch, 608
 - Koreanisch, 608
 - Kyrillisch, 585–587, [588](#), 589, 590
 - Mandschu (Mongolisch), 608
- nicht numerische Querverweise, 80, [81](#), 82
- nicht nummerierte Inhaltsverzeichnisse, 63
- nicht proportionale Schriften, *siehe* dicktengleiche Schriften
- Nicht-ASCII-Symbole, 874
- nindent Schlüssel (lettrine), [108](#)
- \nintt (lb2), [476](#)
- nintt Option (rawfonts), 476
- \niplus (stmaryrd), 549
- \nLeftarrow (amssymb), 551
- \nleftarrow (amssymb), 551
- \nLeftrightarrow (amssymb), 551
- \nleftrightarrow (amssymb), 551
- \nleq (amssymb), 548
- \nleqq (amssymb), 548
- \nleqslant (amssymb), 548
- \nless (amssymb), 548
- \nmid (amssymb), 551
- \nn (lb2), [672](#)
- \nnearrow (stmaryrd), 550
- \nnwarrow (stmaryrd), 550
- \No (babel), [579](#)
- \no (babel), 579
- noadjust Option (cite), 720
- \noalign, [274](#)
- Fehler bei, 942
- noBBppl Option (mathpazo), 390
- \nobibliography
 - (bibentry), [736](#)
 - (jurabib), [751](#), [752](#)
- \nobibliography* (bibentry), 737
- nobottomtitles Option (titlesec), 43
- nobottomtitles* Option (titlesec), 43
- \nobreak, 242
- \nobreakdash (amsmath), [89](#)
- \nobreakspace, [322](#), [323](#)
- \NoCaseChange (textcase), [92](#), 93
- nocfg Option (paralist), 146
- \nochangebars (changebar), 199
- \nocite, [716](#), [717](#), [751](#), [801](#), [808](#), 823
- Fehler bei, 931
- Probleme mit, 716
- Warnung bei, 962
- (biblist), [804](#)
- (bibtopic), 779
- (bibunits), 777
- (jurabib), [748](#), [751](#), [763](#)–[766](#)
- \nocite* (bibunits), 777
- \nocite(typ) (multibib), 782
- nocompress Option (cite), 718, 720
- \nocorr, 356
- \nocorrlist, [356](#)
- \noextras(sprache) (babel), 596, 604
- \noextrarussian (babel), [605](#)
- nofancy Option (rcsinfo), 872
- nofighead Option (endfloat), 298
- nofiglist Option (endfloat), 298
- \nofiles
 - Warnung bei, 968
 - (longtable), 267
- nofoot Schlüssel/Option (geometry), 218
- nographics Option (crop), [222](#)
- nogrey Option (quotchap), 38
- nohang Schlüsselwert (jurabib), 764
- nohead Schlüssel/Option (geometry), 218
- noheadfoot Schlüssel/Option (geometry), 218
- noheads Option (endfloat), 299
- nohyphenation Option (babel), 562
- \noibidem (jurabib), 755
- \noindent, [120](#), [121](#), [134](#), [258](#), [890](#), [895](#), [900](#), [902](#) (picins), [312](#)
- noindentafter Option (titlesec), 43, [45](#)
- noinfo Option (crop), 221
- \nointerlineskip, [900](#)
- nointlimits Option (amsmath), 504
- \nolimits, [504](#)
- Fehler bei, 941
- \nolinebreak, 716, [882](#), 989
- (cite), [718](#), [719](#)
- \nolinenumbers (lineno), 185
- nolists Option (endfloat), 298
- no!l Schlüssel (listings), 183
- nomarkers Option (endfloat), 299
- \nombre (babel), [576](#), 579
- nomove Option (cite), 722
- \nomtpagenumbers (minitoc), 60, 61
- \nomtcrule (minitoc), 60, 61
- \non (lb2), [501](#)
- \nonaheterov (hetarom), [631](#)
- nonamebreak Option (natbib), 731
- nonamelimits Option (amsmath), 504
- none Schlüsselwert
- (fancyvrb), 166, 167, 170, [174](#)
- (listings), 181
- \nonfrenchspacing, 440
- \nonstopmode, [927](#)
- \nonumber (amsmath), 494
- \nopcode, 242
- \nopostamble (docstrip), 863
- \nopreamble (docstrip), 862, 863
- \noptcrules (minitoc), 61
- \norm (lb2), [514](#)
- normal Option (threeparttable), 287
- normal Schlüsselwert (jurabib), [743](#), [747](#), [748](#), [757](#), [758](#), 768
- \normal@char(zeichen) (babel), 606
- \normalcolor, 903
- \normalem (ulem), 94
- normalem Option (ulem), 94

- `\normalfont`, 32, 120, 149, 156, 232, 350, 352, 355, 356, 476, 881, 904
- Normalisieren von Bibliographien, 809–811, 816
- `\normalmarginpar`, 134
- `\normalsize`, 32, 152, 154, 206, 353, 354, 383, 491, 492, 922, 950
- `normalsize` Schlüsselwert (caption), 319
- `norsk` Option (babel), 559, 601
- `norsk.ldf` Datei (babel), 601
- `norule` Option (footmisc), 126
- `noSeparatorLine` Option (fltpage), 336
- `nosort` Option (cite), 718, 720
- `nospace` Option (cite), 720
- `nostar` Option (titleref), 82
- `nostrict` Schlüsselwert (jurabib), 755, 756
- `nosumlimits` Option (amsmath), 504
- `\not`, 547, 549, 551
 - (ifthen), 910
- `notabhead` Option (endfloat), 298
- `notablist` Option (endfloat), 298
- `\notag` (amsmath), 484, 485, 487, 494, 496, 512
- `notbib` Option (tocbibind), 51
- `notcite` Option (showkeys), 72
- `note` BibTeX-Feld, 714, 791, 793, 794, 802
- Notes Umgebung (lb2), 159
- `notes` Zähler (lb2), 159
- `\notesname` (endnotes), 133
- `notext` Option (crop), 222
- `\notin`, 549
- `notindex` Option (tocbibind), 51
- `notlof` Option (tocbibind), 51
- `notlot` Option (tocbibind), 51
- `notoccite` Paket, 722
 - nicht kompatibel zu hyperref, 722
- `notoday` Option (rcsinfo), 872
- `notref` Option (showkeys), 72
- `nottoc` Option (tocbibind), 51
- `\nparallel` (amssymb), 551
- `\nparallelslant` (fourier), 403
- `\nplus` (stmaryrd), 546
- `\nprec` (amssymb), 548
- `\npreceq` (amssymb), 548
- `\nRrightarrow` (amssymb), 551
- `\nrightarrow` (amssymb), 551
- `\nshortmid` (amssymb), 551
- `\nshortparallel` (amssymb), 551
- `\nsim` (amssymb), 548, 549
- `\nsubseteq` (amssymb), 549
- `\nsubseteqq` (amssymb), 549
- `\nsucc` (amssymb), 548
- `\nsucceq` (amssymb), 548, 549
- `\nsupseteq` (amssymb), 549
- `\nsupseteqq` (amssymb), 549
- `\ntriangleleft` (amssymb), 549
- `\ntrianglelefteq` (amssymb), 549
- `\ntrianglelefteqslant` (stmaryrd), 549
- `\ntriangleright` (amssymb), 549
- `\ntrianglerighteq` (amssymb), 549
- `\ntrianglerighteqslant` (stmaryrd), 549
- `\nu`, 543
- `null.tex` Datei, 937
- `num.names$` interne BibTeX-Funktion, 839, 842
- `numarrows` Umgebung (lb2), 190
- `number` BibTeX-Feld, 791, 793
- `number of strings` Fehler, 959
- `numberblanklines` Schlüssel
 - (fancyvrb), 168
 - (listings), 181
- `numbered` Schlüsselwert (jurabib), 765
- `numberless` Schlüssel (titlesec), 46, 47
- `\numberline`, 35, 50, 51, 52, 53, 54, 55, 57
 - (titletoc), 65, 66
- `numbers` Option (natbib), 737, 738, 739, 740
- `numbers` Schlüssel
 - (fancyvrb), 167, 168, 172, 174
 - (listings), 180, 181
- `numbersep` Schlüssel
 - (fancyvrb), 168
 - (listings), 181
- `numberstyle` Schlüssel (listings), 181
- `\numberwithin` (amsmath), 497, 884
- `numbib` Option (tocbibind), 51
- numerische Quellenverweise, 715–722, *siehe auch*
 - Ziterschemata
 - Anpassen von Quellenverweisen, 716, 717, 718, 719, 720
 - ausführliche Warnungen (verbose-Modus), 720
 - Definition, 710
 - Farben, 719
 - hochgestellte Zeichen, 720, 721, 722
 - in Überschriften, 722
 - in Legenden, 722
 - Interpunktion, 719, 721, 722
 - Klammern, 720
 - Layouts für unsortierte Verweise, 722
 - Leerzeichen in/vor, 720
 - natbib-Paket, 737–740
 - Quellenverweise komprimieren, 739
 - Seitenbereiche deaktivieren, 720
 - Sortierreihenfolge, 718, 720, 739
 - Zeilenumbrüche, 718
- `numindex` Option (tocbibind), 51
- Nummer der ersten Seite festlegen, Index, 678, 683
- nummerierte Listen, *siehe auch* Aufzählungen
 - benutzerdefinierte, 159
 - Einzug, 145
 - Erweiterungen, 139–143
 - Layoutparameter, 137
 - Standard, 136, 137, 138
 - Voreinstellungen, 144, 145, 146
- Nummerierung
 - Codezeilen, 180, 181
 - Formeln
 - Untersequenzen, 497
 - Zähler zurücksetzen, 497
 - Fußnoten, 119, 122, 123, 129, 130–132
 - in fremdsprachigen Dokumenten, 576–579, 580
 - Legenden gruppenweise nummerieren, 331, 332
 - Seiten, *siehe* Seitenzahlen
 - Überschriften, *siehe* Überschriften,
 - Nummerierung
 - unterdrücken bei Überschriften, 25, 26

- Nummerierung (*Forts.*)
 Zeilen, 185, [186](#), [187](#), 188, [189](#), [190](#)
 in wortwörtlicher Ausgabe, 167, [168](#)
 numquotation Umgebung (lineno), 186, [189](#)
 numquotation* Umgebung (lineno), 189
 numquote Umgebung (lineno), 186, [189](#)
 numquote* Umgebung (lineno), 189
 \nVDash (amssymb), 551
 \nVdash (amssymb), 551
 \nvDash (amssymb), 551
 \nvdash (amssymb), 551
 \nwarrow, 550
 nynorsk Option (babel), 559, [601](#)
- O**
- O Syntax (fancyhdr), [234](#), [235–238](#)
 \O, 469
 \o, 471
 oaddress BibTeX-Feld (jurabib), 768
 \oarg (ltxdoc), 867
 \oast (stmaryrd), 547
 \obar (stmaryrd), 547
 obeyspaces Option (url), 102
 obeytabs Schlüssel (fancyvrb), [169](#)
 oblique Fonts, 344
 \oblong (stmaryrd), 546
 \obslash (stmaryrd), 547
 \ocircle (stmaryrd), 547
 \oday (babel), [575](#)
 odd Schlüsselwert (titlesec), [46](#)
 odd Schlüsselwort (makeindex), 679
 \oddpagelayoutfalse (layouts), [208](#), [209](#)
 \oddsidemargin feste Länge, [203–205](#), [207](#), 920
 \odot, 547
 (stmaryrd), 547
 \OE, 469
 \oe, 464, [471](#)
 öffnende/schließende mathematische Symbole, [511](#),
[553](#), [554](#)
 \officialeuro (eurosym), 420
 \og (babel), [569](#), [571](#)
 \ogreaterthan (stmaryrd), 547
 \oiint (fourier), [403](#)
 \oint (fourier), [403](#)
 \oint, 552
 \olddatelsorbian (babel), [575](#)
 \olddateusorbian (babel), [575](#)
 oldfont Paket, 361, [476](#)
 oldstyle Option (fourier), 404
 OldStyleNums Option (parallel), [192](#)
 \oldstylenums, [16](#), [394](#), [759](#)
 (textcomp), [41](#), [378](#), [379](#)
 Warnung bei, 969
 \olessthan (stmaryrd), 547
 \olips (lips), [89](#)
 \Omega, [491](#), [543](#)
 (fourier), [403](#), [404](#)
 \omega, 543
 Omega, TeX-Erweiterung, 556, 586, 608, 609, 655
 \ominus, 547
 \omit, Fehler bei, 942
 OML Fontkodierung, 394, [428](#), [449](#), [466](#)
 (cmbright), 397
 (eulervm), 409
 (txfonts), 400
 OMS Fontkodierung, [376](#), 394, [428](#), [449](#)
 (cmbright), 397
 (eulervm), 409
 (txfonts), 400
 OMX Fontkodierung, [428](#), [449](#)
 (eulervm), 409
 \onecolumn, 193, [702](#), [703](#)
 onehalfspace Umgebung (setspace), 114
 \onehalfspacing (setspace), 114
 online Option (threeparttable), [286](#), [287](#)
 Online-Ressourcen, Bibliographien, 802, 803
 Online-Zugang zum CTAN, 998
 only Option
 (excludeonly), 22
 (rawfonts), 476
 \OnlyDescription (doc), [849](#), 850, [853](#), [868](#)
 \ontoday (babel), 575
 \opcit (jurabib), [757](#), [767](#)
 opcit Schlüssel/Option (jurabib), [757](#), [767](#)
 openbib Option, [717](#)
 \openin, 444
 Opening, Symbolklasse, 540
 openout_any Systemvariable (latex (web2c)), [865](#), [938](#)
 Operator, Symbolklasse, 540
 Operatoren, mathematische Symbole, [502–505](#), 506,
 507, [508](#), [553](#)
 fremdsprachige Dokumente, [580](#)
 \operatorname (amsmath), [487](#), [513](#)
 \operatorname* (amsmath), 513
 Operatornamen, Mathematiksatze, [512](#), 513, [514](#)
 \oplus, 547
 \opt (optional), [23](#)
 optional Paket, 23, 24
 optionale Argumente, [877](#), 882, [883](#)
 optionale Code-Ausführung, Inhaltsverzeichnisse, 62,
 63
 optionale Felder, Bibliographiedatenbank, 791, 792
 Optionen, *siehe auch* Kürzel
 ausführen, 914–916
 deklarieren, 913, 914
 globale, 19, 914–916, 919
 Klasse, 18
 verwenden, 19, 20
 Optionen (Sprachauswahl), babel-Paket
 Erwägungen zum Layout, 580–582
 Kurzformen (Befehle), 568–574
 sprachspezifische Befehle, 574–580
 Übersetzungen, 566, [567](#)
 \OptionNotUsed, [911](#), [914](#), [920](#)
 optischer Randausgleich, 1135
 opublisher BibTeX-Feld (jurabib), 768
 \or
 in TeX-Fehlermeldungen, 934
 (ifthen), [910](#), 934
 Ordinary, Symbolklasse, 540

- organization BibTeX-Feld, [714](#), [791](#), [793](#), [794](#), [808](#), [809](#)
 - origin Schlüssel (graphicx), [637](#), [642](#), [650](#), [651](#)
 - originalparameters Option (ragged2e), [114](#)
 - \OrnamentDiamondSolid (bbding), [414](#)
 - Ornamente, *siehe* Boxen, Kreise, Pfeile,...; *siehe auch* Sonderzeichen; Textsymbole
 - osf Option (mathpazo), [390](#)
 - \oslash, [547](#)
 - OT1 Fontkodierung, [348](#), [357](#), [365](#), [394](#), [428](#), [432](#), [442](#), [452](#), [455](#)
 - Erweiterungen, [583](#)
 - für Mathematikfonts, [449](#)
 - im Vergleich zu T1, [358](#), [366](#)
 - Liste der LICR-Objekte, [467](#)–[475](#)
 - Silbentrennung in, [439](#), [939](#)
 - (avant), [384](#)
 - (babel), [583](#), [584](#), [607](#)
 - (bookman), [384](#)
 - (chancery), [384](#)
 - (charter), [384](#)
 - (cmbright), [397](#)
 - (courier), [384](#)
 - (fourier), nicht unterstützt, [402](#)
 - (helvet), [384](#)
 - (infomath), [539](#)
 - (newcent), [384](#)
 - (palatino), [384](#)
 - (pxfonts), [401](#), [402](#)
 - (times), [384](#)
 - (txfonts), [400](#)
 - (utopia), [384](#)
 - OT2 Fontkodierung, [428](#)
 - (babel), [586](#)
 - OT3 Fontkodierung, [428](#)
 - OT4 Fontkodierung, [428](#)
 - OT6 Fontkodierung, [428](#)
 - \otherbeta (fourier), [403](#), [404](#)
 - otherlanguage Umgebung (babel), [561](#), [562](#), [563](#)
 - otherlanguage* Umgebung (babel), [561](#), [562](#)
 - \otherOmega (fourier), [403](#), [404](#)
 - \otimes, [501](#), [547](#)
 - \out (euro), [105](#), [106](#)
 - \outer, [867](#)
 - outer Schlüssel/Option (geometry), [216](#)
 - outerbars Option (changebar), [200](#)
 - outerbody Option (sidecap), [333](#)
 - outercaption Option (sidecap), [333](#)
 - \outlfamily (babel), [584](#)
 - Outline-Font, [345](#)
 - \output, in Warnungstext, [974](#), [975](#)
 - \oval, [615](#), [626](#), [629](#)
 - Warnung bei, [969](#)
 - (eepic), [627](#)
 - (epic), [626](#)
 - (pspicture), [657](#), [658](#), [659](#), [660](#)
 - (texpicture), [659](#)
 - \Ovalbox (fancybox), [614](#)
 - \ovalbox (fancybox), [614](#), [615](#), [616](#)
 - ovale Boxen, [614](#), [615](#)
 - \ovee (stmaryrd), [547](#)
 - \over, [506](#)
 - overcite Paket, [720](#)
 - Overfull box Meldung, [55](#), [109](#), [969](#), [970](#), [985](#)
 - \overfullrule feste Länge, [985](#)
 - erzeugte Ausgabe, [985](#)
 - \overleftarrow (amsmath), [510](#)
 - \overleftrightharpoon (amsmath), [510](#)
 - overload Option (textcase), [93](#)
 - \overrightarrow (amsmath), [510](#)
 - \overset (amsmath), [495](#), [507](#), [508](#)
 - \overwithdelims, [506](#)
 - \owedge (stmaryrd), [547](#)
 - \owns, [549](#)
 - oyear BibTeX-Feld (jurabib), [768](#)
 - OzTeX Programm, [633](#)
 - oxtex Option (graphics), [633](#)
- ## P
- \P, [66](#), [544](#)
 - (textcomp), [469](#)
 - \p (lb2), [89](#)
 - p Syntax, [251](#), [253](#)–[255](#), [258](#), [260](#), [271](#), [272](#)
 - Fehler bei, [943](#)
 - (tabulary), [261](#), [262](#)
 - \p@enumi, [136](#), [137](#)
 - \p@enumii, [136](#), [137](#)
 - \p@enumiii, [137](#)
 - \p@enumiv, [137](#)
 - \PackageError, [918](#)
 - erzeugte Ausgabe, [918](#), [919](#)
 - \PackageInfo, [917](#), [918](#)
 - erzeugte Ausgabe, [918](#)
 - \PackageInfoNoLine, [918](#)
 - \PackageWarning, [914](#), [917](#), [918](#)
 - erzeugte Ausgabe, [918](#)
 - \PackageWarningNoLine, [917](#), [918](#)
 - page Schlüssel (titlesec), [46](#), [47](#)
 - page Zähler, [223](#), [224](#), [883](#)
 - page_compositor Schlüsselwort (makeindex), [680](#), [687](#)
 - page_precedence Schlüsselwort (makeindex), [682](#), [686](#)
 - \pagebreak, [109](#), [135](#), [242](#), [243](#), [493](#), [617](#), [975](#)
 - (multicol), [198](#)
 - \pagedesign (layouts), [208](#), [209](#), [210](#), [211](#)
 - \pagediagram (layouts), [208](#), [210](#)
 - \pagefootnoterule (footmisc), [126](#)
 - \PageIndex (doc), [850](#), [853](#)
 - \pagename (babel), [561](#)
 - \pagenumbering, [224](#), [225](#), [922](#)
 - (chappg), [225](#)
 - (varioref), [73](#)
 - \pageref, [70](#), [72](#), [73](#), [78](#), [118](#), [224](#), [909](#)
 - kombiniert mit \ref, *siehe* varioref-Paket
 - Warnung bei, [971](#)
 - (lineno), [187](#)
 - (prettyref), [80](#)
 - (showkeys), [72](#)
 - (xr), [82](#)
 - pages BibTeX-Feld, [714](#), [791](#), [793](#), [801](#)

`\pagestyle`, 229, 230, 232–241, 616, 617, 703, 920
 leere Seite erzwingen, 231, 243
 (fancyhdr), 238
 (nextpage), leere Seite erzwingen, 244
 (rcs), 871
`\pagevalues` (layouts), 210
`pagewise` Option (lineno), 190
 Pagina, 213
 Paketdateien, 7
 Pakete
 Befehle, 879, 911, 917–919
 Beschreibungen, Online-Katalog, 998
 Dateistruktur, 910–919
 Definition, 18
 Dokumentation suchen, 1002–1004
 dokumentieren, *siehe*
 Dokumentationswerkzeuge
 Fehler
 allgemein, *siehe* Fehlersuche
 Basispakete, 923–956
 in einer Datei kombinieren, 22, 23
 laden in Paketdateien, 916
 lokale, verteilen, 22, 23
 mehrere, mit gleichen Optionen, 20
 Optionen, 18, 913–916
 verändern, 20
 verwenden, 19, 20
 a4, 207, 210
 a4dutch, 211
 a4wide, 211
 a5, 211
 a5comb, 211
 accents, 507, 1013
 ae, 368
 afterpage, 297
 alg, 177
 algorithmic, 177
 alltt, 160
 amscd, 479, 500, 501
 amsfonts, 395–397, 480, 523
 amsmath, 89, 90, 146, 477–500, 502–522, 540, 553, 1012
 amsopn, 479
 amsrefs, 1016
 amssymb, 395–397, 403, 480, 523, 525, 540–554
 amstext, 479
 amsthm, 146–152, 479, 1012
 amsxport, 1016
 amsxtra, 479, 507
 apalike, 717, 821
 array, 251–259, 288–1135
 arydshln, 275, 276
 askinclud, 22
 authordate1-4, 724, 821
 avant, 382, 385
 ba, 537
 babel, 14, 555, 557, 558, 559, 560–607, 725, 759, 775, 956
 bar, 631
 bbding, 414
 bengali, 608

Pakete (*Forts.*)
 beton, 395, 408
 bibentry, 736, 737
 biblist, 803, 804
 bibtopic, 772, 779–781, 782
 bibunits, xxix, 772, 776–779
 bigfoot, 124, 129
 bm, 524–528
 bookman, 214, 382
 booktabs, 277–280
 boxedminipage, 613
 breqn, 483, 1016
 calc, 904, 905
 camel, xxix, 704, 769–771, 1013
 captcont, 324
 caption, xxviii, 304, 305, 306, 317–324, 325, 326, 333
 caption2, 317, 325
 ccfonts, 395, 396, 410, 531
 chams, 537
 chancery, 382
 changebar, 198–200
 chappg, 225, 226
 chapterbib, 725, 732, 772, 773–775, 800
 charter, 382
 chicago, 717, 723, 724
 chmath, 537
 cite, xxix, 718–722
 citehack, 590
 cjk, 608
 cmbright, 396, 397, 539
 color, 222, 1018
 colortbl, 273
 courier, 382
 crop, 220–223
 curves, 629
 custom-bib, xxix, 801, 820, 821, 828–835
 dcolumn, 282–284
 delarray, 502
 devnag, 608
 diagram, 500, 1014
 dingbat, 411
 doc, 160, 599, 845–857, 867
 docstrip, 24, 857–866, 1024, 1025
 dvipsnames, 200
 ecltree, 630
 eco, 67, 395
 eepic, 622, 625–629, 655, 657
 eepicemu, 629
 ellipsis, xxix, 88
 endfloat, xxix, 298–300
 endnotes, xxix, 132–134
 enumerate, 141
 epic, 619–625, 627, 629, 630
 etex, 945
 ethiop, 608
 eucal, 408, 480
 euftrak, 408, 409, 480
 euler, 408, 409
 eulervm, 408–410, 447, 531
 euro, xxviii, 103–106

Pakete (*Forts.*)

europs, 422
 eurosans, 104, 106, **421**, **422**
 eurosym, 419, 420
 excludeonly, 22
 exscale, 91, **379**
 extramarks, xxix, 227, **228–230**
 fancybox, 614–618
 fancyhdr, xxix, 228, 229, **232–240**
 fancyheadings, 232
 fancyref, 80
 fancyvrb, 160, 161, **163–177**, 178, 180, 182, 183
 fix-cm, xxix, 366, **367**, 368
 fixltx2e, **240**, 353
 flafter, 74, **295**
 float, **300–304**, 966
 floatfig, 308
 floatflt, 308
 fltpage, 335, 336
 fncychap, **36**, **37**, 38
 fncylab, 75
 fnpara, 125
 fontenc, 7, 14, 164, **372**, **373**, 921
 fontinst, 94, 388, 431, 432, 449, 450, 1020
 footmisc, xxix, **121–127**, 129, 130
 footnpag, 122
 fourier, xxix, 382, **402–404**, 531
 fp, 103
 french, **608**, 1019
 ftnright, **121**, 185
 fvr-ex, 172
 geometry, xxix, 209, **214–220**
 graphics, 305, **631–636**, 638, **642–649**, 1001, 1018
 graphicx, 631–634, 636–642, 649–651
 graphpap, 658
 grmath, 580
 harvard, 72, **724**, 725, 729, **821**, 832
 hebcals, 575
 hebfont, 594
 helvet, 14, **382**, 385, 436
 here, 303
 hetarom, 631
 hhline, 274, 275
 hvams, 539
 hvmath, 539
 hyperref, **82**, 83, 184, 662, 725, 731
 ifsym, 414–416
 ifthen, 905–910
 indentfirst, **34**, 581
 index, 687, **704**, **705**, 725
 indxcite, 703
 infomath, 539
 inputenc, 7, 14, 184, 340, **369–372**, **456–460**, 587, 594
 jmb, 822
 jurabib, xxviii, **741–769**, 771, 822
 keyval, 214, 317, 641
 kuvio, 500, 1030
 lastpage, xxix, **224**, **225**
 latexsym, 476

Pakete (*Forts.*)

layout, 208
 layouts, xxix, 31, 153, 202, **208–210**
 lb2, 1033
 ledmac, 124, 1032
 lettrine, 106–108
 lineno, xxix, **185–190**, 191
 lips, 88, 89
 listings, xxviii, 162, **177–184**, 317
 lmodern, 368
 longtable, **267–271**, 317
 lscape, 220
 lucidabr, 350, 537
 luximono, 398, 399
 makeidx, **669**, 672, 676
 manjutex, 608
 manyfoot, xxix, **129–132**
 marvosym, xxix, **412–414**, **422**, **423**
 mathpazo, **382**, 383, **389**, **390**, 535
 mathpple, **382**, 383, 389
 mathptm, **382**, 383, 388, 389
 mathptmx, **382**, 383, **387–389**, 399, 401, 533
 mathtime, 363
 minitoc, xxix, **59–62**
 moredefs, 88, 89
 morefloats, 952
 moreverb, 161
 mparhack, 135
 mtcoff, 62
 multibib, xxix, 772, 781, **782**, **783**
 multicol, 185, **193–198**, 240, 308
 multirow, 281, 282
 named, 822
 nar, 822
 natbib, xxix, 72, **725–740**, 832
 nature, 822
 newapa, 822
 newcent, 382
 newlfont, 476
 nextpage, 243, 244
 nfssfont.tex, 357, **380**, **381**, 447, 524
 notoccite, 722
 oldlfont, 361, **476**
 optional, 23, 24
 overcite, 720
 palatino, 108, **382**, 410
 pamath, 535
 paralist, xxix, **139–146**
 parallel, xxix, **190–1135**
 pdfcpot, 1135
 perpage, xxix, **127**, **128**
 picinpar, 115, 116
 picins, 308, **311–315**
 pict2e, xxix, 657
 pifont, **390–393**, 412, 414
 placeins, 297
 prettyref, 79, 80
 pspicture, 657–660
 pstricks, **612**, 662, 1018, 1019
 pxfonts, **401**, **402**, 526, 535
 quotchap, 37, 38

Pakete (*Forts.*)

ragged2e, xxix, **112–114**
 rawfonts, 476
 rcs, **870, 871**, 872
 rcsinfo, 872
 relsize, xxix, **90, 91**, 164
 remreset, 884
 repeatindex, 703
 rotating, 220, **305, 306, 651, 652**
 rotfloat, 306, 307
 rplain, 232
 scrpage2, 244
 seminar, 614
 setspace, **114, 115**, 212
 shadow, 613, 614
 shorttoc, 58, 59
 shortvrb, **160, 161**, 848, 918
 showidx, 676, **703, 704**
 showkeys, **71, 72**, 725
 showtags, 807
 sidecap, xxix, **333–335**
 soul, xxviii, **94–99**
 stmaryrd, 511, **540–554**
 subfig, xxviii, 318, **325–331**
 subfigure, 325
 subfloat, xxix, **331, 332**
 supertabular, **264–267**, 269, 317
 Tabbing, 250
 tabs, 277
 tabularx, 259–261
 tabulary, 256, 259, **261–263**
 texpicture, 657, 658
 textcase, 92, 93
 textcomp, 95, **373–379**, 399, 466–468
 theorem, 148
 threeparttable, xxix, **285–287**
 times, 14, **382**
 tipa, xxix, **416–418**, 428
 titleref, 80–82
 titlesec, xxix, **38–48**, 68, 232
 titletoc, xxix, 59, **62–69**
 tocbibind, 51, 703
 trace, **992, 993**, 1027
 tracefmt, 379, 380
 truncate, 240, 241
 txfonts, **399–401**, 525, 526, 533
 typearea, xxix, **211–214**, 215, 244
 ucs, 372
 ulem, 93, 94
 upquote, xxix, **161–163**
 upref, 479
 url, xxix, **99–103**, 833
 utopia, 382
 varioref, **72–79**, 80, 560
 verbatim, **161, 163**
 vmargin, 211
 wasysym, 411, 412
 wrapfig, 185, **308–311**
 xdoc, 846
 xdoc2, 846
 xr, 82, 83

Pakete (*Forts.*)

xr-hyper, 83
 xspace, 86, 87
 xypic, 612, 1018
 yfonts, 405–407
 palatino Paket, 108, **382, 410**
 Palatino Font
 alternative Unterstützung, 401, 402
 Beschreibung, 387
 in Formeln und Text, **389, 390**, 401, **402, 534, 535**
 Palatino Math Font, **535**
 pamath Paket, 535
 paper Schlüssel/Option (geometry), **214, 215**, 219
 paper Dokumentenklasse, 22
 \paperheight feste Länge, **203, 204, 913**
 (crop), 221
 paperheight Schlüssel/Option (geometry), 215, 216,
 221–223
 papersize Schlüssel/Option (geometry), 219
 \paperwidth feste Länge, **202–204, 913**
 (crop), 221
 paperwidth Schlüssel/Option (geometry), 215, 216,
 221–223
 Papierformat
 angeben, 215
 Optionen, 204
 und Seitenlayout, 211, **212–214**
 \par, 187, **258, 878, 881**, 947
 im Argument unzulässig, 878
 (lineno), 186, 187
 para Option
 (footmisc), **125, 126, 127**, 129, **754**
 (manyfoot), 129, **130, 131**
 (threeparttable), **286, 287**
 para* Option (manyfoot), 129, **130**
 \paradescriptionlabel (paralist), **146**
 \paragraph, 25, **29**
 (minitoc), 60
 paragraph Zähler, **27, 883**
 \paragraph*, 25
 \paragraphdesign (layouts), 210
 \paragraphdiagram (layouts), 210
 paralist Paket, xxix, **139–146**
 paralist.cfg Datei (paralist), 146
 Parallel Umgebung (parallel), **190, 191, 192**, 193
 Probleme mit großen Objekten, **192**
 \parallel, 551
 parallel Paket, xxix, **190–1137**
 \ParallelAtEnd (parallel), **192**
 \ParallelDot (parallel), **192**
 \ParallelLText (parallel), **191, 192**
 \ParallelPar (parallel), **191**
 \ParallelRText (parallel), **191, 192**
 \ParallelUserMidSkip feste Länge (parallel), 191
 Parameter, *siehe* Layoutparameter; Längenparameter
 parameter stack size Fehler, 960
 \parbox, 111, **647, 649, 896**, 898, **899**, 903
 mit Randausgleich, 111, 113
 Probleme mit optionalem s-Argument, **975**

- parens Schlüsselwert
 - (caption), [319](#), [321](#)
 - (lb2), [322](#), [323](#)
 - (subfig), 326, [330](#)
- parensfirst Schlüsselwert (lb2), [323](#)
- parentequation Zähler (amsmath), 497
- `\parfillskip` elastische Länge, [272](#), 320
- `\parg` (ltxdoc), 867
- `\parindent` feste Länge, 140, [191](#), 192, 253, [254](#), [702](#), [703](#), [900](#)
- `\parpic` (picins), [311](#), [312–315](#)
- `\parsep` elastische Länge, 153
- `\parskip` elastische Länge, [31](#), 32, [702](#), [703](#), [979](#), [980](#), [982](#), 983
- `\parskip` Schlüssel/Option (caption), [320](#), [321](#)
- `\part`, 25, 27, 30, 34, 52
 - erzeugt unerwünschte Seitenzahl, 230
 - (minitoc), partielle Inhaltsverzeichnisse, 61
 - (titlesec), 40
 - (titletoc), partielle Inhaltsverzeichnisse, 68
- `part` Zähler, [27](#), [883](#), [886](#)
- `\part*`, 25
- `\partial`, [403](#), [502](#), [544](#)
- `\partial.toc` Datei (lb2), 64
- partielle Inhaltsverzeichnisse, 59, 60, [61](#), 62, 68, [69](#)
- `\partname`, 37
 - (babel), 561
- `\partopsep` elastische Länge, 153
- `\parttoc` (minitoc), 61
- Pascal Schlüsselwert (listings), [180](#), [181](#), [183](#)
- pass Schlüssel/Option (geometry), 220
- Passermarken, 220, [221–223](#)
- `\PassOptionsToClass`, [868](#), [911](#), [919](#), [920](#), [921](#)
- `\PassOptionsToPackage`, [911](#), [913](#), [914](#), 916
- `\path`
 - (eepicemu), 629
 - (eepic), 627, [628](#)
 - (url), [100](#), [101](#), [102](#)
- pattern memory Fehler, 960
- `\pausing`, 992
- `\pausing` Option (tracefnt), 380
- Pazo Font, [389](#), [390](#), [523](#), [534](#)
- `\pcharpath` (pst-char), 425
- `\pcheck` (lb2), [909](#)
- `pctex32` Option (graphics), 633
- `pctex32` Programm, 633
- `pctexhp` Option (graphics), 633
- `pctexhp` Programm, 633
- `pctexps` Option (graphics), 633
- `pctexps` Programm, 633
- `pctexwin` Option (graphics), 633
- `pctexwin` Programm, 633
- `.pcx` Dateierweiterung, 644
- `.pdf` Dateierweiterung, 8, 9, 10, 368
- PDF (Portable Document Format), *siehe auch*
 - PostScript; SVG
 - Definition, 660
 - durchsuchen, 367, [368](#)
 - Links, 662
 - mit T_EX erzeugen, 661, 662
 - Navigation, 662
 - PDF (Portable Document Format) (*Forts.*)
 - Testdateien, 662
 - vs. PostScript, 660, 661
- `pdfcpot` Paket, 1135
- `pdflatex` Option (crop), 222
- `pdftex` Option (graphics), 633
- `pdftex` Schlüssel/Option (geometry), 218
- `pdftex` Programm, 8, 218, 633, 657, 661, 1135
- `\Peace` (bbding), [414](#)
- `\penalty`, [982](#), [984](#)
- period Schlüsselwert
 - (caption), [320](#), [322](#), [334](#)
 - (subfig), [326](#)
- periodical B_BT_EX-Publikationstyp (jurabib), 744, [768](#)
- perl Programm, 788, 804, 806, 1003
- `\perp`, 551
- `\perpage` Option
 - (footmisc), 122, [123](#), 132, [754](#)
 - (manyfoot), 132
- `\perpage` Paket, xxix, [127](#), [128](#)
- Pfade
 - formatieren, 99, [100](#), [101](#), 102, 103
 - zeichnen, 628
- Pfeile (mathematische Symbole)
 - Erweiterungen, [551](#)
 - negierte, [551](#)
 - Standardpfeile, [550](#)
- Pfeilerweiterungen, mathematische Symbole, [551](#)
- `\Pfund` (marvosym), [423](#)
- `\phaip` B_BT_EX-Stil, [822](#), 825, [826](#)
- `\phantom`, [486](#), 518
- `\phapalik` B_BT_EX-Stil (apalike), 822
- `\phcpc` B_BT_EX-Stil, 822
- `\phdthesis` B_BT_EX-Publikationstyp, [791](#), 793
- `\Phi`, 543
- `\phi`, [491](#), 543
- `\phiaea` B_BT_EX-Stil, 822
- `\phjcp` B_BT_EX-Stil, 822
- `\phnf` B_BT_EX-Stil, 822
- `\phnflet` B_BT_EX-Stil, 822
- `\phone` (wasysym), [412](#)
- `\PhoneHandset` (bbding), [414](#)
- `\phpf` B_BT_EX-Stil, 822
- `\phppcf` B_BT_EX-Stil (apalike), 822
- `\phreport` B_BT_EX-Stil, 822
- `\phrmp` B_BT_EX-Stil, 822
- `\Pi`, 543
- `\pi`, 527, [543](#)
- Pi Font, [390–393](#)
- Piautolist Umgebung (pifont), [393](#)
- `pic` Programm, 655
- Pica-Punkte, 346
- `\piccaption` (picins), [314](#), [315](#)
- `\piccaptioninside` (picins), 314
- `\piccaptionoutside` (picins), [314](#), [315](#)
- `\piccaptionside` (picins), 314
- `\piccaptiontopside` (picins), 314
- `\pichskip` (picins), [312](#), [313](#), 314
- `picinpar` Paket, 115, 116
- `picins` Paket, 308, [311–315](#)
 - kombiniert mit caption, 315

- \pickskip (picins), [313](#)
- \picsquare (epic), [620](#), [623](#)
- .pict Dateierweiterung, [644](#)
- pic2e Paket, xxix, [657](#)
- picture Umgebung, [119](#), [500](#), [612](#), [619](#), [652](#)
 - (pspicture), [657](#), [658](#), [659](#), [660](#)
 - (texpicture), [657](#), [659](#)
- \Pifill (pifont), [393](#)
- \Pifont (pifont), [393](#)
 - pifont Paket, [390–393](#), [412](#), [414](#)
- \Piline (pifont), [393](#)
 - Pilist Umgebung (pifont), [393](#)
- \Pisymbol (pifont), [393](#), [412](#), [414](#)
- \pitchfork (amssymb), [551](#)
- .pk Dateierweiterung, [337](#), [341](#), [612](#)
- placeins Option (minitoc), [61](#)
- placeins Paket, [297](#)
- plain Kolumnentitel-Layout, [35](#), [230](#), [231](#), [232](#), [238](#), [702](#), [703](#)
 - (fancyhdr), [238](#)
 - (rplain), [232](#)
- plain Schlüssel (float), [301](#)
- plain $\text{\LaTeX}$ -Stil, [715](#), [716](#), [718](#), [734](#), [821](#), [822](#), [823](#), [837](#), [838](#)
 - (bibtopic), [780](#), [781](#)
 - (bibunits), [776](#)
 - (cite), [718–722](#)
 - (multibib), [782](#), [783](#)
 - (natbib), [734](#), [739](#)
- plainnat $\text{\LaTeX}$ -Stil (natbib), [734](#), [735](#), [762](#), [822](#)
- plaintop Schlüssel (float), [301](#)
- plainyr $\text{\LaTeX}$ -Stil, [822](#)
- \plitemsep elastische Länge (paralist), [140](#)
- Plotten wissenschaftlicher Daten, *siehe* Graphen
- \plparsep elastische Länge (paralist), [140](#)
- \plpartopsep elastische Länge (paralist), [140](#)
- \pltopsep elastische Länge (paralist), [140](#)
- \Plus (bbding), [414](#)
- \plus (euro), [106](#)
 - plus Syntax, [67](#), [97](#), [426](#), [719](#), [887](#), [973](#)
- \PlusOutline (bbding), [414](#)
- \pm, [546](#)
 - pmatrix Umgebung (amsmath), [498](#), [499](#)
 - Fehler bei, [946](#)
- \pmb (bm), [524](#)
- \pmod, [505](#)
- .png Dateierweiterung, [9](#), [660](#), [661](#), [663](#), [931](#)
- .pntg Dateierweiterung, [644](#)
- \pod (amsmath), [505](#)
- \pointedenum (paralist), [141](#)
 - pointedenum Option (paralist), [141](#)
- \pointlessenum (paralist), [141](#)
 - pointlessenum Option (paralist), [141](#)
- polish Option (babel), [559](#), [583](#)
- \polishrz (babel), [583](#)
- \polishzx (babel), [583](#)
 - Polnisch, [583](#)
- polutoniko Attribut (babel), [566](#), [590](#), [591](#), [602](#)
- polutonikogreek Option (babel), [559](#), [602](#)
- pool size Fehler, [960](#)
- pop\$ interne $\text{\LaTeX}$ -Funktion, [839](#)
- \poptabs, Fehler bei, [947](#)
- Portabilität, Befehle, [874](#)
- portable Dateien, Bibliographien, [804](#)
- Portable Document Format (PDF), *siehe* PDF
- portrait Schlüssel/Option (geometry), [215](#)
- portuges Option (babel), [559](#)
- portuguese Option (babel), [559](#)
- position Schlüssel/Option
 - (caption), [321](#), [328](#)
 - (subfig), [326](#), [327](#)
- Positionieren von Grenzen (Hoch-/Tiefstellungen), [503](#), [504](#)
- \possessivecite
 - (harvard), [725](#), [728](#)
 - (lb2), [728](#)
- \postamble (docstrip), [862](#)
 - postamble Schlüsselwort (makeindex), [673](#), [682](#), [683](#)
 - postbreak Schlüssel (listings), [182](#)
- \postdisplaypenalty, [493](#)
- \postmulticols feste Länge (multicol), [194](#), [195](#)
- PostScript, [653](#), [654](#), *siehe auch* PDF; SVG
 - Definition, [653](#)
 - dvips-Treiber, [655](#), [656](#)
 - erweiterte picture-Umgebung, [657](#), [658](#), [659](#)
 - erweiterte oder geänderte Befehle, [657–660](#)
 - fehlende Fonts erzeugen, [655](#)
 - Kodierung, [399–401](#)
 - Kreise, [657](#)
 - Kurven, [659](#)
 - Länge der Pfeilspitzen, [659](#)
 - Linien, [658](#), [659](#)
 - Linienstärke, [658](#), [660](#)
 - OpenType-Fonts, [654](#), [655](#)
 - Ovale, [657](#), [658](#)
 - PostScript-Fonts, [366](#)
 - Testdateien, [662](#)
 - Type1-Fonts, [654](#), [655](#)
 - Vektoren, [658](#), [659](#)
 - vs. PDF, [660](#), [661](#)
- PostScript New Font Selection Scheme, *siehe* PSNFSS
- \pounds, [471](#), [544](#)
- \Pr, [513](#)
 - Präambel, *siehe auch* Befehle
 - Bibliographiedatenbank, [800](#), [801](#)
 - Dokument, [18](#), [19](#)
 - Dokumentationsbefehle, Liste der, [853–857](#)
 - erzeugen, [862](#), [863](#)
 - Fonts definieren, *siehe* Fontbefehle, Low-Level von Tabellen, [251–256](#), [262](#)
- \preamble (docstrip), [862](#)
 - preamble Schlüsselwort (makeindex), [673](#), [682](#), [683](#)
 - preamble\$ interne $\text{\LaTeX}$ -Funktion, [839](#)
 - prebreak Schlüssel (listings), [182](#)
- \prec, [548](#)
- \precapprox (amssymb), [548](#)
- \preccurlyeq (amssymb), [548](#)
- \preceq, [548](#)
- \precnapprox (amssymb), [548](#)
- \precneqq (amssymb), [548](#)
- \precnsim (amssymb), [548](#)
- \precsim (amssymb), [548](#)

- `\predisplayspace`, 493
- `\prefacename` (babel), [561](#), [605](#)
 - `preload.cfg` Datei, 441
- `\premulticols` feste Länge (multicol), 194, 195
- `\pretolerance`, 987
 - (multicol), 196
- Pretty-Printing (formatierte Ausgabe), Bibliographien, 806, 808, 809
- `\prettyref` (prettyref), [80](#)
 - prettyref Paket, 79, 80
- `\prevdepth` feste Länge, [898](#), 991
 - Fehler bei, 898, 939, 955
- `price` BibTeX-Feld (BibTeXMng), 820
- `\prime`, 544
- `\primo` (babel), [579](#)
 - printbib Programm, 805
- `\printbibliography` (camel), [770](#)
- `\PrintChanges` (doc), 849, 851, 853
- `\printcontents` (titletoc), [68](#), [69](#)
- `\PrintDescribeEnv` (doc), 856
- `\PrintDescribeMacro` (doc), 856
- `\PrintEnvName` (doc), 856
 - printheadings Option (bibtopic), 780
- `\printheadingsfalse` (layouts), [209](#), [210](#), [211](#)
- `\PrintIndex` (doc), 849, [850](#), [851](#), [853](#)
- `\printindex`
 - (index), [705](#), [735](#), [746](#)
 - (makeidx), 670, 691
- `\PrintMacroName` (doc), 857
- `\printparametersfalse` (layouts), [210](#), [211](#)
- `\prnttime` (lb2), [904](#)
 - Probleme untersuchen, beheben, *siehe* Fehlersuche
 - proc Dokumentenklasse, 479
 - proceedings BibTeX-Publikationstyp, [714](#), [791](#)
- `\processdelayedfloats` (endfloat), 299
- `\ProcessOptions`, [911](#), [915](#), [916](#), [920](#), [921](#)
- `\ProcessOptions*`, [911](#), [915](#), [916](#)
- `\prod`, 504, [508](#), [552](#)
- `\Prog` (lb2), [674](#)
 - Programmcode formatieren, *siehe* Computercode setzen
- Programmdateien, Quellen
 - CD/DVD, 996, 997
 - FTP, 996, 1000-1002
 - Internetzugriff, 998
- Programme, Bibliographien
 - 8-Bit-Version, 787, 788
 - BibTeX++, 788, 789
 - BibTeX8, 787, 788
 - bibulus, 788
 - Java-Version, 788, 789
 - mehrsprachige Versionen, 789
 - MLBibTeX, 789
 - perl-Version, 788
 - XML-Unterstützung, 788
- Programminformationen, *siehe* Hilfsquellen
- `\projlim` (amsmath), 513
 - proof Umgebung (amsthm), [151](#), [152](#)
- `\proofmodetrue` (index), [704](#), [705](#)
- `\proofname`
 - (amsthm), 151
 - (babel), 561
- Proportionalschriften, 342, 343
- `\propto`, 551
- `\protect`, [35](#), 49, [50](#), [51](#), [76](#), 138, [175](#), 480, [927](#), [928](#), 930, 953
 - in `\index`, 674, 687
 - keine Hilfe bei `\url`, 100
 - (ifthen), 906
 - (textcase), [93](#)
- `\protected@edef`, 927
 - Protokolldateien
 - Erweiterungen, 8
 - hineinschreiben in, 990
 - Indexerstellung, 679, 690
- `\providecommand`, 775, [879](#)
 - (BibTeX), 800
- `\providecommand*`, 879
- `\providehyphenmins` (babel), 602
- `\ProvidesClass`, [911](#), [912](#), [920](#), [922](#)
 - Warnung bei, 962
- `\ProvidesFile`, 444, [449](#), [450](#), [459](#), [461](#), [911](#), [912](#)
 - Warnung bei, 964
 - (inputenc), 459
- `\ProvidesLanguage` (babel), [599](#)
- `\ProvidesPackage`, [911](#), [912](#), [916](#)
 - Warnung bei, 970
 - (babel), 599
- `\ProvideTextCommandDefault`, [459](#), [466](#)
 - (inputenc), 459
- `\PS` (lb2), [876](#)
 - .ps Dateierweiterung, 9, 10, 643
 - .ps.bb Dateierweiterung (graphics), 644
 - .ps.gz Dateierweiterung, 643, 644 (graphics), 644
- `\ps@{stil}`, 232
- `\ps@plain`, [232](#), [920](#)
- `\ps@titlepage` (doc), 857
 - psamsfonts Option
 - (amsfonts), 480
 - (amssymb), 480
 - (eucal), 480
 - (eufrak), 480
 - psfrag Programm, 612
- `\Psi`, 543
- `\psi`, [510](#), [543](#)
 - PSNFSS (PostScript New Font Selection Scheme), 381, 382, [383](#), 384, 385, *siehe auch* NFSS
 - Avant Garde Gothic, 385
 - Bitstream Charter, 386
 - Courier, 386
 - Helvetica, 382, 386
 - ITC Bookman, 385
 - Klassifizierung der Schriftfamilien, 384
 - New Century Schoolbook, 386
 - Palatino
 - alternative Unterstützung, 401, 402
 - Beschreibung, 387
 - in Formeln und Text, [389](#), [390](#), 401, [402](#), [534](#)

- PSNFSS (PostScript New Font Selection Scheme) (*Forts.*)
- Pi, 390–393
 - serifenlose Schriften, 385
 - Times Roman
 - alternative Unterstützung, 399–401
 - Alternativen für Sonderzeichen, [399–401](#)
 - Beschreibung, 387
 - in Formeln und Text, 387, [388](#), 389, [532](#)
 - Utopia, 387
 - verwendete Fonts, 382
 - Zapf Chancery, 387
 - Zeilenvorschub/-abstände, 383
- pspicture Paket, 657–660
- pspicture.ps Datei (dvips), 657
- psprint Option (graphics), 633
- psprint Programm, 633
- pstoedit Programm, 665
- pstricks Paket, [612](#), 662, 1018, 1019
- .ptc Dateierweiterung (titletoc), 68
- \ptcCfont (minitoc), 61
- Publikationstypen, Bibliographien, 789–791, 794
- publisher $\text{\LaTeX}$ -Feld, [714](#), [742](#), [791](#), 792, [793](#), [801](#)
- pubps Option (graphics), 633
- Puffergröße, *siehe* [buffer size](#) Fehler
- Punctuation, Symbolklasse, 540
- Punkt (.), Kurzzeichen, [574](#)
- Punkte
 - als Akzente, Mathematiksatze, [507](#)
 - drei aufeinander folgende (...), *siehe*
 - Auslassungspunkte
 - gepunktete Linien, [620](#), 889
 - Schriftgrad, 346
 - zwischen Binärsymbolen, [509](#)
 - zwischen Integralen, [509](#)
 - zwischen Kommas, [509](#)
 - zwischen Multiplikationszeichen, [509](#)
- purify\$ interne $\text{\LaTeX}$ -Funktion, 839
- \pushtabs, Fehler bei, 947
- \put, [624](#)
 - (epic), [622](#), [623](#), [624](#), 625
 - (pspicture), 659
- \putbib (bibunits), [776](#), [777–779](#)
- \putfile (epic), 623
- PX Fonts, [534](#)
- pxfonts Paket, [401](#), [402](#), 526, 535
 - eng stehende Zeichen, 402
- pybcheck Programm, 817
- pybconvert Programm, 816
- pybiographer Programm, 813–817
- pybiographic Programm, 813–816
- pycompact Programm, 817
- .pz Dateierweiterung, 644
- Q**
- \qauthor (quotchap), [37](#), [38](#)
- \qed (amsthm), 152
 - QED ($\square$)-Zeichen, 151, 152
- \qedhere (amsthm), [152](#)
- \qedsymbol (amsthm), 152
- \qqquad, 522, 889
- \QU (lb2), [910](#)
- \quad, 522, 881, [883](#), [889](#)
 - quad Schlüsselwert (caption), [321](#)
- \quarto (babel), 579
- Quellcodeverwaltung, 869, [870](#), 871, [872](#)
- Quellcodezeile finden, 924–928
- Quelldateien, *siehe auch* Dokumente
 - aufteilen, [21](#), 22
 - Codeerzeugung, 859, 860
- Quellenverweise, *siehe auch* Bibliographien
 - Absatzalgorithmus, 882
 - ausführliche, im Fließtext
 - Autor-Jahr-Zitierschema, [736](#), 737
 - Kurztitel-Zitierschema, 748, [749–751](#)
 - Beschreibung, 707, 708
 - DOI, 735
 - EID, 735
 - Eingabedatei anlegen, 711–1138
 - einleitender Text, 746, [747](#)
 - Einschränkungen für Schlüssel, 874
 - exportieren, 805
 - hochgestellte Zeichen
 - Kurztitel-Zitierschema, [761](#), 769
 - numerisches Zitierschema, 720, [721](#), 722
 - ibidem-Verweise, 753, [754–757](#), 765, [766](#)
 - Interpunktion, Bibliographien
 - Kurztitel-Zitierschema, 763, [764](#)
 - numerisches Zitierschema, 719, [721](#), 722
 - ISBN, 735
 - ISSN, 735
 - Klammern, Bibliographien
 - Kurztitel-Zitierschema, 761
 - numerisches Zitierschema, [720](#)
 - kommentierte, 746, [747](#)
 - komprimieren, [739](#)
 - Leerzeichen in/vor, 720
 - Markup-Aufbau, 710, 711
 - mehrere Autoren, [709](#)
 - mehrere Bibliographien, 771–783
 - numerisch, nach erstem Vorkommen im Text
 - sortiert, 710
 - Schlüsselwörter mit Datenbankeinträgen
 - verknüpfen, 713
 - Seitenbereiche deaktivieren, 720
 - Sortierreihenfolge, Bibliographien
 - Autor-Nummer-Zitierschema, [739](#)
 - Kurztitel-Zitierschema, 769
 - numerisches Zitierschema, [718](#), 720, 739
 - Standardeinstellungen, 715
 - Stildateien, Kurztitel-Zitierschema, 767–769
 - Stile, Autor-Jahr-Zitierschema, 735, 736
 - ungarische Dokumente, 580
 - URL, 735
 - Verarbeitungsablauf, 711–715
 - vergleichen, 708–710
 - Zeilenumbrüche, 882
 - Zitiersystem auswählen, 831, 832
- Querformat, 220
- Querverweise, *siehe auch* varioref-Paket
 - als dynamische Links, 82
 - anpassen, [77](#), 78, 79, [80](#)

Querverweise (*Forts.*)

- auf die aktuelle Seite, 73
- auf einen Objektbereich, 74, [75](#)
- auf externe Dokumente, 82, 83
- auf Unterabschnitte, Überschriften, [28](#), [29](#)
- bedingt zulässige Zeichen, 70
- Bibliographien, 757, [758](#), 801, 802
- Definition, 69
- doc-Paket, 849, 850
- falsche Verweise auf Gleitobjekte, 70, [71](#)
- generieren Fehler, 928
- Index

- überprüfen, 688
 - erstellen, 671

- Labelformate, 75, [76](#), 77–79
- nicht numerische, 80, [81](#), 82
- Nummern in aufrechter Serifenschrift erzwingen, 479
- nur auf eine Seitenzahl, 73
- Querverweisschlüssel anzeigen, 71, [72](#)
- Seitenzahlen, [224](#)
- ungarische Dokumente, 580
- Verweis auf die aktuelle Seite, [224](#)
- Zeilennummern, 187, [188](#)

- quotation Umgebung, 154, 842
- (lineno), 189

- quotchap Paket, 37, 38

- Quote Umgebung (lb2), 154, [155](#)

- quote Schlüsselwort (makeindex), 680, [683](#)

- quote Umgebung, 154, [881](#)
- (lineno), [189](#)

- quote\$ interne L^AT_EX-Funktion, 839

- \quotechar (doc), 855

- \quotedblbase, 471

- \quotesinglbase, 471

R

R Syntax

- (fancyhdr), [234](#), [235–238](#)
- (lb2), [256](#)
- (tabulary), 262

- \R (babel), 584, [585](#)

- \r, 471

- (pxfonts), Probleme mit, 402
 - (tipa), [417](#)
 - (txfonts), Probleme mit, [400](#)

r Syntax, 251

- (array), 258

Ränder

- Bundsteg (innerer Rand), 202
- Fußsteg (unterer Rand), 201, 202, 212, 216, 219
- Kopfsteg (oberer Rand), 201, 202, 212, 214, 216, 219, 233
- optischer Randausgleich, 1135
- Seitenlayout, 202, 216, 217, 219
- Seitensteg (äußerer Rand), 202
- von Ausgabetreibern erzeugte, 204

römische Seitenzahlen

- Index
- Seitenbereiche unterdrücken, 700

römische Seitenzahlen (*Forts.*)

- Sortierreihenfolge, 687

- Verweise auf, 73

- Vorspann, 224

- Rückverweise, Bibliographien, 768

- Rückwärtskompatibilität, 475, 476

- \Radiation (ifsym), [416](#)

- ragged Option

- (footmisc), [126](#), [130](#), [752](#), [755](#), [756](#)

- (sidecap), 333, [334](#), [335](#)

- ragged2e Paket, xxix, [112–114](#)

- Probleme mit amsthm, 150

- \raggedbottom, 126

- \raggedcolumns (multicol), 196

- RaggedLeft Schlüsselwert (caption), 320

- \RaggedLeft (ragged2e), 112, 113

- \raggedleft, 111, [128](#)

- in Überschriften, [33](#)

- (array), in Tabellen, [255](#), [257](#)

- (ragged2e), 112

- (titlesec), nicht zu verwenden mit

- \titleformat, 42

- raggedleft Option

- (sidecap), 333

- (titlesec), 39, [40](#)

- raggedleft Schlüsselwert (caption), 320

- \RaggedLeftLeftskip elastische Länge (ragged2e), 113

- \RaggedLeftParfillskip elastische Länge (ragged2e), 113

- \RaggedLeftParindent feste Länge (ragged2e), 113

- \RaggedLeftRightskip elastische Länge (ragged2e), 113

- RaggedRight Schlüsselwert (caption), 320

- \RaggedRight (ragged2e), 112, [113](#), [195](#), [196](#), 765

- \raggedright, 111, [128](#), [134](#), [150](#), [191](#), 192, [352](#), [356](#), 764, [980](#)

- in Überschriften, [33](#), [34](#)

- in Tabellen, 254, 255, [259](#), 269, [284](#)

- (array), in Tabellen, [255](#), [257](#)

- (multirow), in Tabellen, 281

- (natbib), 732

- (ragged2e), 112

- (titlesec), nicht zu verwenden mit

- \titleformat, 42

- raggedright Option

- (sidecap), 333

- (titlesec), 39

- raggedright Schlüsselwert

- (caption), 320, [322](#)

- (jurabib), [764](#), [765](#)

- (subfig), [326](#), [328](#)

- raggedrightboxes Option (ragged2e), 113

- \RaggedRightLeftskip elastische Länge (ragged2e), 113

- \RaggedRightParfillskip elastische Länge (ragged2e), 113

- \RaggedRightParindent feste Länge (ragged2e), [113](#)

- \RaggedRightRightskip elastische Länge (ragged2e), [113](#)

- Rahmen, *siehe auch* Boxen; Linien
 Boxen, [613](#)
 Seiten, [615](#), [616](#), [617](#)
 um Formeln, [503](#), [618](#)
 um Listen; Absätze, [618](#)
 um Nummern in Überschriften, [28](#)
 um Programmlistings, [182](#), [183](#)
 um wortwörtliche Ausgabe, [165](#), [166](#)
- Rahmenlinien, *siehe* Boxen; Rahmen
- `\Rain` (ifsym), [416](#)
- `\raisebox`, [158](#), [280](#), [282](#), [895](#)
- RaiseNums Option (parallel), [192](#)
- `\raisetag` (amsmath), [496](#)
- Randausgleich
 Absätze, [108–110](#), [111](#), [112](#), [113](#), [114](#)
 Legenden von Gleitobjekten, [320](#), [321](#)
 Überschriften, [33](#)
- range_close Schlüsselwort (makeindex), [680](#)
- range_open Schlüsselwort (makeindex), [680](#)
- `\rangle`, [511](#), [526](#), [553](#)
- `\ratio` (calc), [905](#), [909](#)
- rawfonts Paket, [476](#)
- `\Rbag` (stmaryrd), [554](#)
- `\rbag` (stmaryrd), [546](#)
- `\rbrace`, [484](#), [485](#), [511](#), [523](#), [553](#)
- `\rbrack`, [511](#), [553](#)
- `\rceil`, [511](#), [553](#)
- `\RCS` (rcs), [870](#), [871](#)
- rcs Paket, [870](#), [871](#), [872](#)
- RCS (Revision Control System), [869](#)
- rcs-user.tex Datei (rcs), [871](#)
- `\RCSauthor` (rcs), [870](#)
- `\RCSdate` (rcs), [870](#), [871](#)
- `\RCSdate` (rcs), [871](#)
- `\RCSdef` (rcs), [870](#)
- `\RCSID` (rcs), [871](#)
- `\rcsinfo` (rcsinfo), [872](#)
- rcsinfo Paket, [872](#)
- rcsinfo.perl Datei (rcsinfo), [872](#)
- `\rcsinfoDate` (rcsinfo), [872](#)
- `\rcsinfoDay` (rcsinfo), [872](#)
- `\rcsinfoFile` (rcsinfo), [872](#)
- `\rcsinfoLocker` (rcsinfo), [872](#)
- `\rcsinfoLongDate` (rcsinfo), [872](#)
- `\rcsinfoMonth` (rcsinfo), [872](#)
- `\rcsinfoOwner` (rcsinfo), [872](#)
- `\rcsinfoRevision` (rcsinfo), [872](#)
- `\rcsinfoStatus` (rcsinfo), [872](#)
- `\rcsinfoTime` (rcsinfo), [872](#)
- `\rcsinfoYear` (rcsinfo), [872](#)
- `\RCSrawDate` (rcs), [871](#)
- `\RCSRCSfile` (rcs), [870](#)
- `\RCSRevision` (rcs), [870](#)
- `\RCSTime` (rcs), [870](#)
- `\Re`, [544](#)
- READ BIBTeX-Befehl, [836–838](#), [840](#)
- read Schlüssel (graphicx), [638](#), [643](#)
- readme-lb2.html Datei, [1007](#)
- `\real` (calc), [905](#)
- Rechts-nach-links-Schreibweise, fremdsprachige Dokumente, [582](#), [594](#)
- rechtsbündiger Flattersatz, [111](#), [112](#)
- `\RecordChanges` (doc), [849](#), [851](#), [853](#)
- Recto-Verso-Layout, [46](#), [202](#), [207](#), [217](#), *siehe auch* doppelseitiger Ausdruck
- `\RecustomVerbatimCommand` (fancyvrb), [173](#), [174](#)
- `\RecustomVerbatimEnvironment` (fancyvrb), [173](#), [174](#)
- reducedifibidem Schlüsselwort (jurabib), [755](#)
- `\Ref` (varioref), [76](#)
- `\ref`, [28](#), [29](#), [70](#), [72](#), [73](#), [75–77](#), [79](#), [118](#), [138](#), [316](#)
 kombiniert mit `\pageref`, *siehe* varioref-Paket
 Probleme mit, [28](#), [70](#), [71](#), [885](#)
 verursacht seltsame Ergebnisse, [28](#)
 Warnung bei, [971](#)
 (amsmath), [494](#), [495](#), [497](#)
 (ftpage), [336](#)
 (lineno), [187](#), [188](#)
 (paralist), [139](#), [141](#)
 (prettyref), [80](#)
 (showkeys), [72](#)
 (subfig), [325](#), [328](#)
 (subfloat), [332](#)
 (textcase), [92](#)
 Probleme mit, [92](#)
 (titleref), [81](#)
 (upref), [479](#)
 (varioref), [75](#), [76](#)
 (wrapfig), [309](#)
 (xr), [82](#), [83](#)
- ref Option (cite), [722](#)
- Referenzen, Index
 Arten, [700](#)
 formatieren, [700](#)
- Referenzpunkt einer Box, [650](#)
- `\reflectbox`
 (graphics), [647](#)
 (graphicx), [647](#)
- `\refname`, [37](#), [751](#), [775](#)
 (babel), [561](#), [562](#)
 (bibunits), [777](#)
 (multibib), [782](#), [783](#)
- `\refstepcounter`, [35](#), [128](#), [884](#), [885](#), [886](#)
 Probleme mit, [885](#)
- `\reftextafter` (varioref), [77](#), [78](#), [79](#), [369](#)
- `\reftextbefore` (varioref), [77](#), [78](#), [79](#)
- `\reftextcurrent` (varioref), [73](#), [75](#), [78](#)
- `\reftextfaceafter` (varioref), [77](#), [78](#), [79](#)
- `\reftextfacebefore` (varioref), [77](#), [78](#), [79](#)
- `\reftextfaraway` (varioref), [77](#), [78](#)
- `\reftextlabelrange` (varioref), [78](#)
- `\reftextpagerange` (varioref), [78](#)
- `\reftextvario` (varioref), [78](#), [79](#)
- Regeln zur Großschreibung, Bibliographien, [816](#)
- Register, *siehe* Verzeichnisse; *die einzelnen Verzeichnisarten*; *siehe auch* Längenparameter
- Registerwerte anzeigen, [979](#), [980](#)
- reguläre mathematische Symbole, [542–544](#), [545](#)
- Reihe Umgebung (lb2), [302](#)
- Rejne Option (fncychap), [37](#)
- Relation, Symbolklasse, [540](#)

- Relationsmengen, mathematische Symbole, [547](#), [548](#),
 [549](#), [550](#), [551](#)
 \relax, [171](#), [458](#), [514](#), [900](#), [901](#)
 \relphantom (lb2), [486](#)
 \relscale (relsize), [91](#)
 \relsize (relsize), [90](#), [164](#)
 relsize Paket, xxix, [90](#), [91](#), [164](#)
 rem Umgebung (lb2), [148](#)
 remreset Paket, [884](#)
 \renewcommand, [876](#), [877](#), [879](#)
 Fehler bei, [929](#), [932](#), [942](#)
 \renewcommand*, [947](#)
 \renewenvironment, [880](#)
 Fehler bei, [934](#), [938](#)
 \renewindex (index), [704](#)
 repeatindex Paket, [703](#)
 report Dokumentenklasse, [7](#), [14](#), [122](#), [126](#), [155](#), [204](#),
 [231](#), [702](#)
 Alternativen zu, [244](#)
 Fußnotennummerierung, [119](#)
 Gliederungsbefehle, [24](#)–[27](#), [55](#)
 TOC-Einträge, [53](#), [55](#)
 Versionsdaten, [912](#)
 reqno Option (amsmath), [478](#), [482](#), [484](#)
 require Funktion (xindy), [697](#)
 \RequirePackage, [15](#), [367](#), [451](#), [705](#), [884](#), [911](#),
 [913](#)–[915](#), [916](#), [919](#), [920](#), [954](#)
 Fehler bei, [947](#)
 vorgezogenes Laden, [947](#)
 Warnung bei, [976](#)
 \RequirePackageWithOptions, [916](#)
 reset Schlüssel/Option (geometry), [220](#)
 reseteqn.sty Datei (lb2), [16](#)
 resetlabels Option (multibib), [783](#)
 resetmargin Schlüssel (listings), [181](#)
 resetmargins Schlüssel (fancyvrb), [165](#)
 \resetul (soul), [99](#)
 \resizebox
 (graphics), [635](#), [636](#), [647](#), [648](#)
 Fehler bei, [949](#)
 (graphicx), [647](#), [648](#)
 Fehler bei, [949](#)
 \resizebox*
 (graphics), [647](#), [648](#)
 (graphicx), [648](#)
 Restore-Werte anzeigen, [990](#)
 \restriction (amssymb), [550](#)
 \restylefloat
 (float), [303](#), [318](#)–[320](#)
 (rofloat), [307](#)
 \resumecontents (titletoc), [69](#)
 REVERSE BibTEX-Befehl, [838](#)
 \reversemarginpar, [134](#), [209](#)
 \reversemarginpartrue (layouts), [209](#)
 reversemp Schlüssel/Option (geometry), [217](#), [218](#)
 Revision Control System, *siehe* RCS
 Revisionsbalken, [198](#), [199](#), [200](#)
 \RewindToStart (marvosym), [413](#)
 rewrite.rule Funktion (bibtool), [809](#), [810](#)
 \rfloor, [511](#), [553](#)
 \rfoot (fancyhdr), [229](#), [232](#), [233](#)
 \rhd (latexsym), [476](#)
 \rhead (fancyhdr), [232](#), [233](#), [616](#)
 \rho, [543](#)
 \rhook, [551](#)
 \right, [490](#), [495](#), [500](#), [510](#), [511](#), [517](#), [540](#), [542](#), [553](#),
 [944](#)
 Fehler bei, [934](#), [942](#)
 right Option
 (eurosym), [420](#)
 (lineno), [189](#), [190](#)
 right Schlüssel/Option (geometry), [216](#), [219](#)
 right Schlüsselwert
 (fancyvrb), [167](#)
 (listings), [181](#)
 \Rightarrow, [550](#)
 \rightarrow, [182](#), [512](#), [550](#)
 \rightarrowtail (amssymb), [550](#)
 \rightarrowtriangle (stmaryd), [550](#)
 rightbars Option (changebar), [200](#)
 rightbody Option (sidecap), [333](#)
 rightcaption Option (sidecap), [333](#)
 \RightDiamond (ifsym), [416](#)
 rightFloats Option (fpage), [335](#)
 \rightharpoondown, [550](#)
 \rightharpoonup, [550](#)
 \righthyphenmin, [602](#), [991](#)
 rightlabels Option (titletoc), [63](#)
 \rightleftarrows (amssymb), [550](#)
 \rightleftharpoons (amssymb), [550](#)
 \rightmargin feste Länge, [153](#), [155](#)
 \rightmargin Schlüssel (titlesec), [40](#), [46](#)
 \rightmark, [226](#), [235](#), [236](#), [237](#)
 (extramarks), [228](#)
 \rightpointleft (dingbat), [411](#)
 \rightrightarrows (amssymb), [550](#)
 \rightskip elastische Länge, [111](#), [112](#), [982](#)
 (ragged2e), [112](#)
 \rightsline (stmaryd), [546](#)
 \rightsquigarrow (amssymb), [550](#)
 \rightthreetimes (amssymb), [546](#)
 \rightthumbsdown (dingbat), [411](#)
 \rightthumbsup (dingbat), [411](#)
 \risingdotseq (amssymb), [548](#)
 \rlap, [189](#), [190](#), [192](#), [501](#)
 \rm, [358](#), [361](#)
 in Formeln, [361](#), [476](#)
 (custom-bib), [833](#)
 rm Option (titlesec), [39](#), [40](#)
 rm Schlüsselwert
 (caption), [319](#), [321](#), [322](#)
 (subfig), [326](#)
 rmargin Schlüssel/Option (geometry), [216](#)
 \rmdefault, [357](#), [358](#), [451](#)
 \rmfamily, [350](#), [355](#), [357](#), [363](#), [420](#), [476](#)
 in Formeln, [360](#), [361](#)
 \rmoustache, [511](#), [553](#)
 robuste Befehle, [674](#), [879](#)
 Rohindex erzeugen, [669](#)
 Roman Folio-Stil, [224](#)
 \Roman, [137](#), [140](#), [885](#), [886](#)

- \roman, 137, 140, **885**
- roman Folio-Stil, 224
- Roman (Schriftform), 344
- romanian Option (babel), 559
- rootbib Option (chapterbib), 773
- rotate Option (crop), 222
- rotate Umgebung (rotating), 306, **651**, **652**
- \rotatebox
 - (graphics), **636**, 646, 648, **649**
 - Fehler bei, 948
 - (graphicx), 632, 642, **649**, **650**, 651
 - Fehler bei, 948
- rotating Paket, 220, **305**, **306**, **651**, **652**
- kombiniert mit endfloat, 300
- \rotcaption (rotating), **306**, 317
- rotfloat Paket, 306, 307
- \round (euro), **105**
- round Option (natbib), 731, 737, **740**
- round Schlüssel/Option (jurabib), **747**, **761**
- \rowcolor (colortbl), **273**
- rplain Paket, 232
- \rrbracket
 - (fourier), **403**
 - (stmaryrd), 511, 553
- \rrceil (stmaryrd), 554
- \rrffloor (stmaryrd), 554
- \Rrightarrow (amssymb), 550
- \rparenthesis (stmaryrd), 554
- .rsc Dateierweiterung (bibtool), 810
- \Rsh (amssymb), 550
- \RSPercentTolerance (relsize), 90
- \rsquare (lb2), **545**
- \Rsub (lb2), **33**
- \rtimes (amssymb), 546
- rubibtex Programm, 590
- rubibtex.bat Programm, 590
- \rule, **43**, 119, **250**, 274, **336**, **890**, **896–898**, **899**, **900**
- rulecolor Schlüssel (fancyvrb), **166**
- ruled Option (manyfoot), **131**, **132**
- ruled Schlüssel (float), **301**, **302**, **303**
- ruled Schlüsselwert (float), **319**
- rulesep Schlüssel (listings), **183**, **184**
- rumakeindex Programm, 590
- rumkidxd.bat Programm, 590
- rumkidxw.bat Programm, 590
- runin Schlüssel (titlesec), **40**, 41, 42
- russian Option (babel), **370**, 559, 584, **585–587**
- russianb Option (babel), 1024
- russianb.ldf Datei (babel), 605
- \rVert (amsmath), 511, **514**, 553
- \rvert (amsmath), 511, **512**, **514**, 553
- S**
- \S, **41**, **67**, **138**, 544
- (textcomp), 469
- s Größenfunktion, 436
- s: Syntax (yfonts), **406**, **407**
- safe Option
 - (textcomp), **374**, **376**, **377**, 378, 399
 - wirkt nicht, **378**
 - (tipa), 417
- \samepage, 242
- samepage Schlüssel (fancyvrb), 167
- samin Option (babel), 559
- Sammlung von Bibliographien der
 - Computerwissenschaften, 803
- Sammlungen, bibliographische Daten, 768
- \sample (lb2), **14**, **229**, **232**, **302**, **312**, **616**
- \sAppendix (lb2), **34**, **35**
- Satzspiegel, 201, 215
- save size Fehler, 961
- \savebox, **901**, **902**, 942, 990
- Fehler bei, 930
- savequote Umgebung (quotchap), **37**, **38**
- \SaveVerb (fancyvrb), **174**, **175**, **176**
- SaveVerbatim Umgebung (fancyvrb), 176
- \sbox, **316**, **882**, **901**, **902**, 903, 942, **990**
- Fehler bei, 930
- \sboxrule feste Länge (shadow), 613
- \sboxsep feste Länge (shadow), 613
- \sc, 358
- in Formeln, 476
- sc Option
 - (mathpazo), 390
 - (titlesec), 39
- sc Schlüsselwert (caption), 319
- Scalable Vector Graphics (SVG), *siehe* SVG
- scale Schlüssel (graphicx), **637**, **639**
- scale Schlüssel/Option (geometry), 219
- \scalebox
 - (graphics), **635**, **646**, **647**
 - (graphicx), **646**, **647**
- scaled Option
 - (eurosans), **421**, **422**
 - (helvet), 382
 - (luximono), **162**, **398**, **399**
- \scdefault, 357
- SCfigure Umgebung (sidecap), 333, **334**
- Schafe stehlen, *siehe* sperren
- schattierte Boxen, 613, **614**, **615**
- schattierte Fonts, 345
- Schlüssel, *siehe auch* Argumente
 - \$Id\$-Schlüsselwort parsen, **872**
 - anzeigen, 71, **72**
- Bibliographien
 - Definition, 789
 - Duplikate entfernen, 809, 817
 - erzeugen, 812, 813
 - extrahieren, 807
 - Groß-/Kleinschreibung, Unterscheidung von, 790
 - nach Zeichenfolgen durchsuchen, 805
 - zur Bibliographie-Auflistung hinzufügen, 807, **808**
- Einschränkungen, 874, 928
- RCS-Daten extrahieren, **870**, **871**
- Schlüsselwörter mit Datenbankeinträgen verknüpfen, 713
- school B¹TeX-Feld, 791, 793
- schräg gestellte Fonts, **344**, 351

- Schreibmaschinenschrift, 350, [398](#), [399](#), *siehe auch*
 verbatim-Umgebung; wortwörtliche
 Ausgabe
 Laufweite, [398](#)
- Schreibrichtung, fremdsprachige Dokumente, 582,
[594](#)
- Schriften, *siehe* Fonts
 in diesem Buch, 1135
- Schriftgröße, *siehe* Schriftgrade
- Schriftkegel, 346
- Schriftstärke, 345, 346
- Schusterjunge, 984
- Schwabacher Font, 405–407
- Scientific Word Programm, 633
- \ScissorLeft (bbding), [414](#)
- scottish Option (babel), 559
- scrartcl Dokumentenklasse, 244
- screen Schlüssel/Option (geometry), 215
- \scriptscriptstyle, 444, 506, [515](#)
- \scriptsize, 353
 scriptsize Schlüsselwert
 (caption), 319
 (subfig), [329](#)
- \scriptstyle, 444, [501](#), 506, [515](#)
- \scrollmode, 990
- scrpage2 Paket, 244
- \scshape, [32](#), 67, [351](#), [352](#), [353](#), 355, 357, [885](#)
 in Formeln, 360, 361
 (lettrine), 106
 (soul), 98
- SCTable Umgebung (sidecap), 333, [334](#), [335](#)
- \sdim feste Länge (shadow), [614](#)
- \searrow, [182](#), [550](#)
- \sec, 513
- \secdef, 29, [34](#)
- \secformat (lb2), [43](#), [44](#)
- secnumdepth Zähler, 25, 26, 30, [32](#), 33, 36
- \sectfont (quotchap), [38](#)
- \section, 24, 25, 27, [28](#), [29](#), [32](#), [34–36](#), 41, [50](#), 52, 225,
 227, 231, 983
 Fehler bei, 927
 Gleitobjekte unterdrücken, [295](#)
 mit Gleitobjektbarriere, [297](#)
 Querverweis auf, [70](#)
 (bibunits), [778](#)
 (minitoc), 60, [61](#)
 partielle Inhaltsverzeichnisse, 61
 (soul), mit gesperrtem Text, [98](#)
 (titleref), textueller Verweis auf, [81](#)
 (titlesec), [39–43](#), [45](#), [47](#)
 (titletoc), partielle Inhaltsverzeichnisse, 68, [69](#)
 section Option (placeins), 61, [297](#)
 section Schlüsselwert (jurabib), 749, 757
 section Zähler, [27](#), [28](#), 35, 226, 228, [883](#), [886](#)
- \section*, 25, [50](#), 732, 774
 im Inhaltsverzeichnis, [50](#)
 (titlesec), [47](#)
- sectionbib Option
 (bibunits), 778
 (chapterbib), [774](#), [775](#)
 (natbib), 732, 774
- \sectionbreak (titlesec), [45](#)
- \sectionmark, [35](#), 227, [228](#), [238](#)
 (fancyhdr), 237, 238
- \sectlof (minitoc), 61
- \sectlot (minitoc), 61
- \secttoc (minitoc), 61
 secttocdepth Zähler (minitoc), [61](#)
- \secundo (babel), 579
- sed Programm, 590, 805, 807
- \see (makeidx), 672
 see Schlüssel/Option (jurabib), [746](#), [747](#)
- \seename
 (babel), 561, 671
 (makeidx), 671
- seitenübergreifende Tabellen, *siehe* mehrseitige
 Tabellen
- Seitenanzahl-Feld, Bibliographien, 769
- Seitenbereiche
 in Bibliographien deaktivieren, 720
 Index
 deaktivieren, 678, 690, 694, 700
 Längen begrenzen, 700
- Seitenbeschreibungssprachen, 652, *siehe auch* PDF;
 PostScript; SVG
- Seitengrenzen, in Bibliographien ignorieren, 755
- Seiteninhalte symbolisch darstellen, 980–983
- Seitenlayout, 201
 ändern, 205, [206](#), [207](#)
 asymmetrisch, 217
 Auto-Vervollständigung, [214](#), 215, 216, [217](#),
 218, [219](#), 220
 Beschnittlinien/-marken, 220, [221–223](#)
 Bindung und Bundsteg, 215
 Bundsteg (innerer Rand), 202
 darstellen, [208](#), [209](#), 210
 doppelseitiger Ausdruck, 207, *siehe auch*
 Recto-Verso-Layout
 für die Anzeige am Computer, 215
 Fußnoten, 215
 Fußzeilenhöhe, 209
 geometrische Dimensionen, 202–205
 gerade/ungerade Seiten, 46
 im Verhältnis zum Papierformat, 211, [212–214](#)
 KOMA-Script-Klassen, 244, 245
 Layoutparameter, 203
 lebende Kolumnentitel, 215, 218
 „LEERE SEITE“ auf generierten Seiten, [244](#)
 Marginalien, 218
 Pakete für, 210, [211](#)
 Papierformat angeben, 215
 Papierformat-Optionen, 204
 Parametervoreinstellungen, 205
 Passermarken, 220, [221–223](#)
 Querformat, 220
 Ränder, 202, 216, 217, 219
 Recto-Verso-Layout, 202, 207, [217](#)
 Satzspiegel, 215
 schematische Seitendarstellung, 203
 Seitensteg (äußerer Rand), 202
 Seitenumbrüche, 242, 243
 symmetrisch, 217

- Seitenlayout (*Forts.*)
 - Überschriften unterdrücken, 210
 - Vergrößerung, 218
 - visuelle Formatierung, 242–244
 - von Ausgabetreibern erzeugte Ränder, 204
 - Weißraum, 207
 - Zeilen je Seite, 207
- Seitenlayout (Kolumnentitel), 230, 231
 - anpassen
 - abhängig von Gleitobjekten, 239
 - angepasstes Layout speichern, [238](#)
 - global, [232](#), [233](#)
 - linke/rechte Seiten, 234, 235, [236–238](#)
 - benannt, 238
 - für doppelseitigen Ausdruck, 231, 232, [234](#)
 - für Gleitobjektseiten, [239](#)
 - für Wörterbücher, 239, [240](#)
 - Linien, 233
 - mehrzeiliger Text, [233](#)
 - Text abschneiden, 240, [241](#)
 - Textmarkenbefehle, 226, [227](#), 228, [229](#), 230, [237](#), [238](#)
- Seitensteg (äußerer Rand), 201, 202, 212, 218, 219
- Seitenumbrüche, *siehe auch* Längenparameter
 - Badness-Grad, 892
 - Fehlersuche, 980–985
 - in mehrseitigen Tabellen, 265
 - in/um Formeln, 491–494
 - Index, Eintrag wiederholen, 703
 - Seitenlayout, 242, 243
- Seitenzahlen, 223, 224
 - aktuelle Seite, Verweise auf, [224](#)
 - Index
 - Duplikate, 670
 - formatieren, 672
 - kapseln, 672, 695
 - MakeIndex*-Optionen, 685–687
 - römische Zahlen, 687, 700
 - Sortierreihenfolge, 678, 686, 700–702
 - xindy-Optionen, 700–702
 - zusammengesetzte (kapitelweise Nummerierung), 686
 - letzte Seite, Verweis auf, 224, [234](#)
 - nach Kapiteln, [225](#), 226
 - Querverweise, 73
 - ungerade, erzwingen, 243
 - unterdrücken, 230
 - Verweise auf, [224](#)
 - Zähler zurücksetzen, 224
- `select` Funktion (bibtool), [812](#)
- `select.non` Funktion (bibtool), [812](#)
- `\selectfont`, [357](#), [366](#), [422](#), 423, [424](#), [426–428](#), 429, [430](#), [467](#)
- `\selectlanguage` (babel), [561](#), [562](#), [563](#), 587
- `semantic nest size` Fehler, 961
- Semikolon (;), Kurzzeichen, [571](#)
- seminar Paket, 614
- `SeparatedFootnotes` Option (parallel), [192](#)
- `serbian` Option (babel), 559
- Serien, Schrift-, *siehe* Fonts, Serien
- `series` $\LaTeX$ -Feld, [714](#), [791](#), [793](#)
- `\seriesdefault`, 357, 358, [429](#)
 - serifenlose Schriften, 343, 350
 - als Grundschrift, 385
 - Serifenschriften, 343, 350
- `\setboolean` (ifthen), [703](#), [907](#), [908](#), [920](#)
- `\setbox`, 903
 - Probleme mit, [903](#)
- `\setcounter`, [138](#), [884](#), [885](#), [909](#)
 - Fehler bei, [876](#), 945, 946
 - (calc), [904](#), [906](#)
 - Fehler bei, 929
- `\setdefaultenum` (paralist), [145](#)
- `\setdefaultitem` (paralist), [144](#)
- `\setdefaultleftmargin` (paralist), [145](#)
- `\setdepth` (bar), [631](#)
- `\setfnsymbol` (footmisc), 123, [124](#)
- `\setfootbox` (layouts), [209](#)
- `\sethebrew` (babel), [584](#), [585](#)
- `\sethlcolor` (soul), [99](#)
- `\setkeys`
 - (graphicx), [642](#)
 - (keyval), 641
- `\setlabelfont` (layouts), [209](#), [210](#), [211](#)
- `\setlayoutscales` (layouts), [208](#), [209](#), [211](#)
- `\setlength`, [887](#), [905](#)
 - Fehler bei, 946
 - Probleme mit, 521
 - (calc), [904](#), [905](#), [909](#)
 - Fehler bei, 929
- `\setmarginrb` (vmargin), [211](#)
- `\SetMathAlphabet`, [364](#), [448](#), 452, 941
 - Fehler bei, 932
- `\setminus`, 546
- `\setnumberpos` (bar), [631](#)
 - `setpage_prefix` Schlüsselwort (makeindex), 682
 - `setpage_suffix` Schlüsselwort (makeindex), 682
- `\setpapersize` (vmargin), [211](#)
- `\setparametertextfont` (layouts), [208](#), [210](#)
 - `setspace` Paket, [114](#), [115](#), 212
- `\setstcolor` (soul), [99](#)
- `\setstretch`
 - (bar), [631](#)
 - (setspace), 114
- `\SetSymbolFont`, [445](#), [446](#), [447](#), [448](#), [449](#), 451
 - Warnung bei, 970
- `\settodepth`, [888](#)
- `\settoheight`, [888](#)
- `\settowidth`, [288](#), [290](#), [883](#), [888](#)
- `\setul` (soul), [99](#)
- `\setulcolor` (soul), [99](#)
- `\setuldepth` (soul), [99](#)
- `\setxaxis` (bar), [631](#)
- `\setxname` (bar), [631](#)
- `\setxvaluetyp` (bar), [631](#)
- `\setyaxis` (bar), [631](#)
- `\setyname` (bar), [631](#)
- `\sf`, 338, 358, 476
 - in Formeln, 361, 476
- `sf` Option (titlesec), [39](#)

- sf Schlüsselwert
 - (caption), [310](#), [315](#), [319](#), [321](#), [322](#), [326](#), [334](#)
 - (subfig), [326](#)
- \sfdefault, [357](#), [385](#)
- \sffamily, [350](#), [352](#), [354](#), [355](#), [357](#), [363](#), [420](#), [476](#)
 - in Formeln, [360](#), [361](#)
 - Probleme mit EC Fonts, [367](#)
 - (lucidabr), [421](#), [422](#)
- sfixed Größenfunktion, [438](#)
- sgen Größenfunktion, [436](#)
- sgenb Größenfunktion, [437](#)
- \sh (babel), [580](#)
- \shabox (shadow), [613](#), [614](#)
- shadow Paket, [613](#), [614](#)
- \shadowbox (fancybox), [614](#), [615](#), [616](#)
- \shadowsize feste Länge (fancybox), [614](#), [616](#)
- \shadowthickness (picins), [314](#)
- Shalom-Fonts, [593](#)
- \shapedefault, [357](#), [358](#), [429](#)
- \sharp, [544](#)
- \Shilling (marvosym), [423](#)
- short Option (rcsinfo), [872](#)
- short Schlüsselwert (jurabib), [757](#)
- shortauthor BibTeX-Feld (jurabib), [742](#), [744](#), [757](#), [768](#)
- \shortcite
 - (authdate1-4), [724](#)
 - (chicago), [709](#), [724](#)
- \shortciteA (chicago), [724](#)
- \shortciteN (chicago), [724](#)
- \shortcites (natbib), [730](#)
- \shortdownarrow (stmaryrd), [550](#)
 - shorttext Option (minitoc), [60](#)
- \shorthandoff (babel), [565](#), [571](#), [573](#)
- \shorthandon (babel), [565](#)
 - Fehler bei, [950](#)
- \shortindexingoff (index), [704](#)
- \shortindexingon (index), [704](#)
- \shortleftarrow (stmaryrd), [550](#)
- \shortmid (amssymb), [551](#)
- \shortpage (lb2), [242](#)
- \shortparallel (amssymb), [551](#)
- \shortrightarrow (stmaryrd), [550](#)
- \shortstack, [115](#), [614](#), [619](#)
 - (pspicture), [658](#)
- \shorttableofcontents (shorttoc), [59](#)
- shorttitle BibTeX-Feld (jurabib), [714](#), [742](#), [744](#), [747](#), [757](#), [769](#)
- \shorttoc (shorttoc), [59](#)
 - shorttoc Paket, [58](#), [59](#)
- \shortuparrow (stmaryrd), [550](#)
- shortvrb Paket, [160](#), [161](#), [848](#), [918](#)
- shortvrb.sty Datei (shortvrb), [859](#)
- \shoveleft (amsmath), [484](#)
- \shoveright (amsmath), [484](#)
- \show, [946](#), [977](#), [978](#), [979](#), [980](#)
 - erzeugte Ausgabe, [977-979](#)
- \showbox, [946](#), [990](#)
 - erzeugte Ausgabe, [990](#)
- \showboxbreadth, [990](#), [991](#)
- \showboxdepth, [990](#), [991](#)
- \showclock (ifsym), [415](#)
- \showcols (array), [257](#)
- showframe Schlüssel/Option (geometry), [218](#)
- \showgroups, verfügbar unter eTeX, [944](#), [959](#)
 - erzeugte Ausgabe, [959](#)
- \showhyphens, [986](#)
 - erzeugte Ausgabe, [986](#)
- showidx Paket, [676](#), [703](#), [704](#)
- showkeys Paket, [71](#), [72](#), [725](#)
- \showlists, [946](#), [961](#), [990](#), [991](#)
 - erzeugte Ausgabe, [991](#)
- \showoutput, [981](#), [983](#), [990](#), [991](#)
 - erzeugte Ausgabe, [982](#)
- \showpage (lb2), [211](#)
- \showprogress (docstrip), [861](#)
- showspaces Schlüssel
 - (fancyvrb), [169](#), [173](#), [174](#)
 - (listings), [180](#), [183](#)
- showstringspaces Schlüssel (listings), [180](#)
- showtabs Schlüssel
 - (fancyvrb), [169](#)
 - (listings), [180](#)
- showtags Paket, [807](#)
- \showthe, [946](#), [980](#)
 - erzeugte Ausgabe, [980](#)
 - Fehler bei, [939](#)
- \shrinkheight (supertabular), [265](#)
- si960 Option (inputenc), [594](#)
- siam BibTeX-Stil, [823](#)
- Sicherheit, docstrip, [865](#)
- side Option (footmisc), [125](#), [126](#), [130](#)
- sidecap Paket, xxix, [333-335](#)
 - kombiniert mit caption, [333](#)
- \sidecaptionrelwidth (sidecap), [334](#)
- \sidecaptionsep (sidecap), [334](#)
- \sidecaptionvpos (sidecap), [334](#)
- \sideset (amsmath), [508](#)
- sideways Umgebung (rotating), [306](#)
- sidewaysfigure Umgebung (rotating), [300](#), [305](#), [306](#), [317](#)
- sidewaysfigure* Umgebung (rotating), [305](#)
- sidewaystable Umgebung
 - (rotating), [300](#), [305](#), [306](#), [307](#), [317](#)
 - (rotfloat), [307](#)
- sidewaystable* Umgebung
 - (rotating), [305](#)
 - (rotfloat), [307](#)
- sidewaysXMLexa Umgebung (lb2), [307](#)
- sidewaysXMLexa* Umgebung (lb2), [307](#)
- \Sigma, [543](#)
- \sigma, [543](#)
- Silbentrennung
 - Deutsch, [19](#), [560](#), [597](#)
 - dynamisch festlegen, [558](#)
 - ermöglichen, [89](#), [134](#)
 - Fehlersuche, [985](#), [986](#)
 - in mehrsprachigen Texten, [562](#), [596-598](#)
 - in Tabellen, [254](#)
 - in Überschriften, [33](#)
 - Italienisch, [579](#)
 - kulturelle Gesichtspunkte, [558](#)
 - nach Bindestrich, [570](#), [719](#)

- Silbentrennung (*Forts.*)
 - Niederländisch, [568](#)
 - Sonderregelungen, [569](#), [570](#)
 - sprachliche Gesichtspunkte, 557
 - Trennmuster anwenden, [562](#)
 - Trennmuster einstellen, 602, 603
 - Trennzeichen festlegen, 439
 - verhindern, [562](#)
- `\sim`, [548](#), [549](#)
- `\simeq`, 548
- simple Schlüsselwert
 - (caption), 319
 - (subfig), 330
- `\sin`, [513](#), [519](#)
- single Schlüsselwert
 - (fancyvrb), [165–167](#)
 - (listings), 182, [183](#)
- Single-Byte-Zeichen kodieren, 370, 371
- singlelinecheck Schlüssel/Option
 - (caption), [318](#), [320](#)
 - (subfig), [328](#)
- singlespace Umgebung (setspace), 114
- singlespacing (setspace), 114
- `\sinh`, 513
- `\SixFlowerOpenCenter` (bbding), [414](#)
- size $\text{\LaTeX}$ -Feld (BibTeX-Mng), 820
- size11.clo Datei, 18, 152
- skalieren
 - Bilddateien, 638
 - Fonts, relativ zum Original, [90](#), [91](#)
 - graphische Objekte, [646–648](#)
 - große Operatoren, 379
- skaliertes Material verbergen, 634
- `\skip`, [979](#), 980
- `\skip23` elastische Länge, [979](#)
- `\skip$` interne $\text{\LaTeX}$ -Funktion, 839, [841](#)
- `\skip\footins` elastische Länge, [119](#), 120
 - (footmisc), 126, 127
 - (manyfoot), 131
- `\skip\footins(suffix)` elastische Länge (manyfoot), [131](#), [132](#)
- Skriptbefehle, docstrip, 858–863
- `\sl`, 358
 - in Formeln, 476
- `sl` Option (titlesec), 39
- `sl` Schlüsselwert
 - (caption), 319
 - (fancyvrb), [164](#)
 - (subfig), [329](#)
- slantedGreek Option
 - (ccfonts), 396
 - (cmbright), 397
 - (mathpazo), 389
 - (mathptmx), 388
- `\slash` (soul), [97](#)
- `\slashint` (fourier), [403](#)
- `\sldefault`, 357
- slides Dokumentenklasse, 7
- slope Schlüssel (lettrine), [108](#)
- sloped Option (fourier), [403](#), [404](#)
- sloppy, 110
- slovak Option (babel), 559
- slovene Option (babel), 559
- `\slshape`, [351](#), [352](#), 353, 355, 357
 - in Formeln, 360, 361
- `\small`, 152, 154, 353, 354, 492
- small Option
 - (eulervm), 409
 - (titlesec), 39
- small Schlüsselwert
 - (caption), [319](#), [320](#)
 - (subfig), [328](#)
- smallcaps Schlüsselwert (jurabib), [743](#), [744](#)
- `\smaller` (relsize), 90
- `\smallfrown` (amssymb), 551
- `\smallint`, 552
- smallmatrix Umgebung (amsmath), 499, [500](#)
- `\smallpencil` (dingbat), [411](#)
- `\smallsetminus` (amssymb), 546
- `\smallskip`, 891
- `\smallskipamount` elastische Länge, 891
- `\smallsmile` (amssymb), 551
- `\smash` (amsmath), [518](#), [519](#), 520
 - unerwartete Ergebnisse, [519](#), [520](#)
- `\smile`, 551
- `\Smiley` (marvosym), [413](#)
- `\Snow` (ifsym), [416](#)
- `\Snowflake` (bbding), [414](#)
- `\so` (soul), [95–97](#)
- `\sobf` (lb2), [97](#)
- `\sodef` (soul), [97](#), 98
- Softwareversionskontrolle, 869, [870](#), 871, [872](#)
- Sonderzeichen, 356, [357](#), *siehe auch die Einträge für die einzelnen Zeichen*; mathematische Symbole; Textsymbole
 - Einschränkungen bei Querverweisen, 70
 - fremdsprachige Dokumente, [569](#)
 - in Bibliographiedatenbanken, 797, 798
 - in URLs, E-Mail-Adressen usw., 99
 - Index, [673](#), 674, 675, 683, 684
 - Index-Sortierreihenfolge, 687, 689
 - wortwörtliche Ausgabe, [160](#), [161](#)
- Sonny Option (fncychap), 36
- `\SORT` $\text{\LaTeX}$ -Befehl, 838
- sort Option
 - (cite), 720
 - (natbib), [729](#), [739](#)
- sort-rule Funktion (xindy), [697](#)
- sort.format Funktion (bibtool), [809](#)
- sort.key\$ interne $\text{\LaTeX}$ -Funktion, 838
- sort&compress Option (natbib), [739](#)
- Sortierreihenfolge
 - Bibliographien, 794, 808, 809, 837
 - Index
 - deutsche Wörter, 678, 690, 692
 - Fehlersuche, 687, 688
 - französische Wörter, 692
 - Leerzeichen, 687
 - nach Buchstaben, 678, 690
 - nicht englische Wörter, 692
 - römische Zahlen, 687
 - Seitenzahlen, 678, 686, 700–702

- Sortierreihenfolge (*Forts.*)
 - Sonderfälle, 689
 - spanische Wörter, 692, 696
 - Symbole, 687, 689
 - xindy-Regeln, 696, 699
- Quellenverweise in Bibliographien
 - Autor-Nummer-Zitierschema, [739](#)
 - Kurztitel-Zitierschema, 769
 - numerisches Zitierschema, [718](#), 720, 739
- `\SortIndex` (doc), 855
- `sortkey` `BiTeX`-Feld, 801
 - (jurabib), [769](#), 794
- `\SortNoop`, 798, **801**
 - (lb2), [800](#), [801](#)
- soul Paket, xxviii, **94–99**
 - Befehle verschachteln, 96
 - Fehler bei, 939
 - kombiniert mit `color`, 95, 99
- `\soulaccent` (soul), [95](#)
- `\soulomit` (soul), [96](#)
- `\soulregister` (soul), **95**, [96](#)
- `\source` (camel), [770](#)
 - `source2e.tex` Datei, 867–869
- `\sout` (ulem), [94](#)
- `\space`
 - in `\DeclareFontEncoding`, 442
 - in `\DeclareFontShape`, 434
 - in `.fd`-Dateien, 444
- space Option (cite), 720
- space Schlüsselwert (caption), 320
- `\spacefactor`, 991
 - Fehler bei, 939, 954
- spaces Option (url), 102
- `\spaceskip` elastische Länge, **112**, 441
 - (ragged2e), 112
- spacing Umgebung (setspace), [114](#), 115
- `\spadesuit`, 544
- Spalten, Tabellen
 - Abstände, [255](#), [256](#)
 - anlegen, 248–251
 - eine anders formatierte, 256, [257](#)
 - Layout verändern, 256, [257](#)
 - schmale, 254, 255
- Spalten, Text
 - ausbalancieren, [196](#)
 - ausgleichen, 196, 197
 - formatieren, 196
 - Fußnoten, **121**, [122](#), [192](#), 198
 - Gleitobjekte, 198
 - Material sammeln, 197
 - mehrere, 193, [194–196](#), 197, 198
 - parallele Synchronisation, 190, [191](#), [192](#), 193
 - Umbrüche
 - Index, Eintrag wiederholen, 703
 - manuell erzeugt, [198](#)
 - vertikale Abstände, 119
- Spaltenformatkürzel definieren, 256, 257
- Spaltenlayout, Bibliographien, [765](#)
- Spaltensteg, 215
- Spanisch
 - Anführungszeichen, [574](#)
 - Index-Sortierreihenfolge, 692, 696
 - spanish Option (babel), **559**, [571](#), [574](#)
- `\Sparkle` (bbling), [414](#)
- Spationierung, *siehe* Abstände
- `\spbreve` (amsxtra), [507](#)
- `\spcheck` (amsxtra), [507](#)
- `\spdddot` (amsxtra), [507](#)
- `\spddot` (amsxtra), [507](#)
- `\spdot` (amsxtra), [507](#)
- `\special`, 9, 10, 611, 612, 626, 644, 657, **1028**, 1030
 - (eepic), 626
 - (hyperref), 82
 - (listings), 182
- `\SpecialEnvIndex` (doc), 855
- `\SpecialEscapechar` (doc), 854
- `\SpecialIndex` (doc), 855
- `\SpecialMainEnvIndex` (doc), 855
- `\SpecialMainIndex` (doc), 855
- `\specialrule` (booktabs), 279, [280](#)
- `\specials`, 656
- `\SpecialUsageIndex` (doc), 855
- Sperren, 94–99
- `\sphat` (amsxtra), [507](#)
- `\sphericalangle` (amssymb), 544
- `\SpinDown` (ifsym), [416](#)
- `\SpinUp` (ifsym), [416](#)
- `\spline`
 - (eepicemu), 629
 - (eepic), 628, 629
- split Umgebung (amsmath), **481**, [485](#), [486](#), 489
 - Fehler bei, 930, 931, 933, 934
- `\splitfootnoterule` (footmisc), 126
- `\SplitNote` (manyfoot), **130**, [131](#)
- `splitrule` Option (footmisc), 126
- Sprachattribute, 565, [566](#)
- Sprachdefinitionsdateien
 - Aufbau, 598–607
 - Beschreibung, 596
 - Copyright-Vermerk, 598
 - Definitionen hinzufügen, [605](#)
 - Dokumentation, 600
 - Dokumentationstreiber, 599
 - Identifikation der Sprache, 599
 - Initialisierung, 600
 - Interpunktion, Sonderfälle, [607](#)
 - Kurzformen, 606, 607
 - Lizenzdaten, 598
 - Sprachen und Dialekte definieren, 600–602
 - sprachspezifische Zeichenfolgen übersetzen, 603
 - Trennmuster anpassen, 602, 603
 - Versionsdaten, 599
- Sprache
 - aktuelle, einstellen/abfragen, 560, 561, [562](#), [563](#)
 - definieren, 600–602
 - identifizieren, 599
 - und Formatierung, 557, 558
- Sprachoptionen, *siehe* Optionen (Sprachauswahl)

- sprachspezifische Zeichenfolgen
 - anpassen, 565, [566](#), [567](#), 596
 - babel-Paket, [558](#), [561](#), 596
 - Trennmuster, 603
 - Übersetzungen, 566, [567](#)
- `\sptilde` (amsmath), [507](#)
- `\sqcap`, 546
- `\sqcup`, 546
- `\sqr`, [505](#), 511, [518](#), [519](#)
 - (amsmath), [488](#), [489](#), 517
- `\sqrt`, 510, 511, [512](#)
 - (bm), [526](#), 527
- `\sqsubset`
 - (amssymb), 549
 - (latsym), [476](#)
- `\sqsubseteq`, 549
- `\sqsupset`
 - (amssymb), 549
 - (latsym), [476](#)
- `\sqsupseteq`, 549
- `\square` (amssymb), [544](#), [545](#)
 - square Option (natbib), 731, 737
 - square Schlüssel/Option (jurabib), [761](#)
- `\SquareShadowC` (ifsym), [416](#)
- squiggle Programm, 664
- `\SS`, 469
- `\ss`, 356, [471](#)
 - Form in EC Fonts, [366](#)
 - (yfonts), 406
- `\ssearrow` (stmaryrd), 550
- `\ssediton` BibTeX-Feld (jurabib), 761, [769](#)
- `\sslash` (stmaryrd), 546
- `\ssub` Größenfunktion, 437
- `\ssubf` Größenfunktion, 438
- `\sswarrow` (stmaryrd), 550
- `\st` (soul), [95](#), [99](#)
 - stable Option (footmisc), 127
- `stack$` interne BibTeX-Funktion, 839
- `\stackrel`, [501](#), [507](#)
- Stacks (TeX Programm)
 - List-Stack, anzeigen, 990
 - Makro-Stack, anzeigen, 926
 - parameter stack size Fehler, 960
- standard-baselineskips Option
 - (ccfonts), 396
 - (cmbright), 397
- Standard-Ein-/Ausgabedateien, Index, 678, 690
- `\StandardLayout` (babel), [581](#), 582
- `\StandardModuleDepth` Zähler (doc), 857
- Stapel, *siehe* Stacks
- `\star`, [144](#), [507](#), 546
- `\startcontents` (titletoc), 68, [69](#)
- `\StartFinalBibs` (chapterbib), [774](#), [775](#)
- `\StartShownPreambleCommands` (lb2), 171
- `\STATE` (algorithmic), [177](#)
 - stauchen, Formeln formatieren, [519](#), 520, 521
- `\stcfont` (minitoc), [61](#)
- `\stcindent` (minitoc), [61](#)
- `\stctitle` (minitoc), [61](#)
- stealing sheep, *siehe* sperren
- `\stepcounter`, [774](#), [882](#), 884, 885, [909](#)
- stepnumber Schlüssel
 - (fancyvrb), [168](#)
 - (listings), [181](#)
- Stern (*) Fehlermeldungen, 929
- Steuerfunktionen
 - arithmetische Funktionen, [904](#), 905
 - bedingte Formatierung, 905, [906-910](#)
- Stichwortverzeichnisse, *siehe* Index
- Stildateien, *siehe auch* Konfigurationsdateien
- Bibliographien
 - bearbeiten, 835-844
 - Beschreibung, 820
 - BibTeX-Programmiersprache, 835-844
 - erstellen, 828-835
 - Formatierung angeben, 834, 835
 - Initialisieren des Systems, 829, 830
 - Kurztitel-Zitierschema, 767-769
 - Liste von, 820-823
 - neue Felder hinzufügen, 841, 842
 - unterstützte Erweiterungen festlegen, 832-834
 - Unterstützung für Fremdsprachen hinzufügen, 842-844
 - verändern, 835-844
 - Zitierschema auswählen, 831, 832
- Index
 - angeben, 679
 - MakeIndex*, 679, 681-687
 - xindy, 695-702
- Stile, Autor-Jahr-Zitierschema, 735, 736
- stiller Modus, Indexerstellung, 678, 690, 697
- Stilparameter, *siehe* Layoutparameter
- stmaryrd Paket, 511, [540-554](#)
- `\stockdesign` (layouts), 210
- `\stockdiagram` (layouts), 210
- `\stop`, 929, 955, 963
 - (nfssfont.tex), 381
- `\stopcontents` (titletoc), [69](#)
- `\StopEventually` (doc), [849](#), [854](#), 868
- `\StopShownPreambleCommands` (lb2), 171
- `\StopWatchEnd` (ifsym), [415](#)
- `\StopWatchStart` (ifsym), [415](#)
 - straight Schlüssel (titlesec), [47](#), [48](#)
- `\stretch`, [888](#), [889](#), [890](#)
- strict Schlüsselwert (jurabib), [754](#), [756](#), [757](#), [761](#)
- strictdoublepage Schlüsselwert (jurabib), [754](#), 756
- `\string`, [607](#), [866](#), [978](#), [979](#)
 - (docstrip), 862
- STRINGS BibTeX-Befehl, 836, [838](#)
- Strings, Bibliographien, *siehe* Zeichenfolgen
- stringstyle Schlüssel (listings), 179
- `\StrokeFive` (ifsym), [416](#)
- Strukturformeln, chemische, [631](#)
- strukturierte Abbildungen, [325](#), 326-328, [329](#), 330, 331
- strukturierte Tabellen, [325](#), [326](#), 327-331
- `\strut`, 281, 519, [520](#)
 - (amsmath), 520
 - (sidecap), [335](#)
- .sty Dateierweiterung, 7, 9, 18
- style Schlüssel/Option (caption), [322](#)

- sub Größenfunktion, 437
- sub(*klasse*) Zähler (subfig), 327
- subarray Umgebung (amsmath), 500
- \subchapter (lb2), 47, 48
- subequations Umgebung (amsmath), 497
- subf Größenfunktion, 438
- subfig Paket, xxviii, 318, 325–331
- subfigure Zähler (subfig), 327
- subfigure Paket, 325
- subfigures Umgebung (subfloat), 331, 332
- \subfiguresbegin (subfloat), 331
- \subfiguresend (subfloat), 331
- \subfloat (subfig), 325, 326, 327, 328, 329
- subfloat Paket, xxix, 331, 332
- subfloatfigure Zähler (subfloat), 332
- subfloatfiguremax Zähler (subfloat), 332
- subfloattable Zähler (subfloat), 332
- subfloattablemax Zähler (subfloat), 332
- \subitem, 702, 703
- \subparagraph, 25, 26, 27
 - (minitoc), 60
 - subparagraph Zähler, 27, 883
- \subparagraph*, 25
- subparens Schlüsselwert (subfig), 330
- \subref (subfig), 328, 329, 330
- \subref* (subfig), 328
- \subsection, 24, 25, 26, 27, 28, 31, 32, 50, 231
 - (minitoc), 60, 61
 - (titleref), textueller Verweis auf, 81
 - (titlesec), 39, 40
- subsection Zähler, 26, 27, 28, 883, 886
- \subsection*, 25
- subsectionbib Option (bibunits), 779
- \subsectionmark, 238
 - (fancyhdr), 237, 238
- \Subset (amssymb), 549
- \subset, 549
 - subset BibTeX-Stil (aux2bib), 804
 - subset.bib Datei (makebib), 805
- \subseteq, 503, 549
- \subseteqq (amssymb), 549
- \subsetneq (amssymb), 549
- \subsetneqq (amssymb), 549
- \subsetplus (stmaryrd), 549
- \subsetpluseq (stmaryrd), 549
- subsimple Schlüsselwert (subfig), 330
- \substack (amsmath), 500
- substring\$ interne BibTeX-Funktion, 839, 844
- \subsubitem, 702
- \subsubsection, 25
 - (minitoc), 60, 61
 - (titlesec), 39, 40
- subsubsection Zähler, 27, 883, 886
- \subsubsection*, 25
- subtables Umgebung (subfloat), 331, 332
- \subtablesbegin (subfloat), 331
- \subtablesend (subfloat), 331
- \succ, 548
- \succapprox (amssymb), 548
- \succcurlyeq (amssymb), 548
- \succeq, 548, 549
- \succnapprox (amssymb), 548
- \succneqq (amssymb), 548
- \succnsim (amssymb), 548
- \succsim (amssymb), 548
- Suchen
 - in Bibliographien, 805, 807, 814, 817
 - in PDF-Dokumenten, 367, 368
- \sum, 409, 508, 552
 - Platzierung von Hoch-/Tiefstellungen, 503, 504
 - (relsize), Verwenden größerer Symbole, 91
- sumlimits Option (amsmath), 504
- \Summit (ifsym), 416
- \Sun (ifsym), 416
- \SunCloud (ifsym), 416
- \sup, 513
 - super Option
 - (cite), 720, 722, 783
 - Probleme mit, 721
 - (natbib), 738, 739, 740
 - super Schlüssel/Option (jurabib), 752, 753–757, 760, 761
 - superscript Option (cite), 720, 721, 722
 - superscriptedition Schlüssel/Option (jurabib), 761, 769
 - supertabular Umgebung (supertabular), 264, 265, 266, 267, 269, 271
 - supertabular Paket, 264–267, 269, 317
 - kombiniert mit caption, 265, 270
 - supertabular* Umgebung (supertabular), 264, 266, 269
- \supminus (lb2), 513, 514
- \suppressfloats, 35, 295, 296
- \Supset (amssymb), 549
- \supset, 549
- \supseteq, 549
- \supseteqq (amssymb), 549
- \supsetneq (amssymb), 549
- \supsetneqq (amssymb), 549
- \supsetplus (stmaryrd), 549
- \supsetpluseq (stmaryrd), 549
- \surd, 544
 - SVG (Scalable Vector Graphics), 665, *siehe auch* PDF;
 - PostScript
 - LaTeX-Dokumente umwandeln in, 664
 - portable Webgraphiken, 662–664
 - svgview Programm, 664
- \swabfamily (yfonts), 405, 406
- swap\$ interne BibTeX-Funktion, 839
- \swapnumbers (amsthm), 148
- \swarrow, 550
- swedish Option (babel), 559, 576
- switch Option (lineno), 190
- switch Schlüsselwert (jurabib), 761, 769
- switch* Option (lineno), 190
- \sym (euro), 105
- \symbol, 356, 357, 419, 674
 - Warnung bei, 968, 991
- symbol Option (footmisc), 123, 124, 752
- symbol* Option (footmisc), 123, 128
- Symbole, *siehe* mathematische Symbole; Sonderzeichen; Textsymbole

- Symbolklassen, Mathematiksatz, 540, [541](#), [542](#), [545](#)
- symmetrisches Seitenlayout, 217
- Syntax, Fehlermeldungen, 924
- Syntaxdiagramme erzeugen, 867
- systemimmanente Funktionen, Bibliographien, 836, 838, 839
- sz Syntax (yfonts), [406](#)
- T**
- \t, 375, [471](#)
 - Problem mit textcomp, [375](#)
 - (textcomp), 374
- t Syntax
 - (delarray), [502](#)
 - (hline), [274](#), [275](#)
- T1 Fontkodierung, 348, [357](#), 365–368, [377](#), 394, [427](#), [428](#), 432, 433, 442, 455, [461](#), [462](#), [463–465](#), [939](#)
 - Erweiterungen, 583
 - Form des f, [366](#)
 - im Vergleich zu OT1, 358
 - Liste der LICR-Objekte, 467–475
 - Probleme mit EC Fonts, [367](#)
 - Silbentrennung in, [439](#), 939
 - (avant), 384
 - (babel), 569, 574, 583, 584, 607
 - (bookman), 384
 - (ccfonts), 395, 396
 - (chancery), 384
 - (charter), 384
 - (cmbright), [397](#)
 - (courier), 384
 - (fontenc), [373](#)
 - (fourier), 402
 - (helvet), 384
 - (luximono), [398](#), [399](#)
 - (newcent), 384
 - (nfssfont.tex), 380
 - (palatino), 384
 - (pxfonts), 401
 - (textcomp), 373, [376](#), [377](#)
 - (times), 384
 - (txfonts), 400
 - (url), 102
 - (utopia), 384
- T1 Option (fontenc), [373](#), [376](#), [377](#), [397–399](#), [427](#), [428](#), [451](#), [583](#), [939](#)
- \T1/cmr/m/it/10, 936
- \T1/cmr/m/n/10, 981
- t1enc.def Datei, [461](#), [463–465](#)
- t1put.fd Datei, 432
- T2A Fontkodierung, 366, [377](#), [427](#), [428](#), 585, 587, [588](#), 589, 945
 - (fontenc), [373](#)
- T2A Option (fontenc), [373](#), [427](#), [586](#)
- T2B Fontkodierung, 366, [428](#), 586, [589](#), 945
- T2C Fontkodierung, 366, [428](#), 586, [589](#)
- T3 Fontkodierung, 428
 - (tipa), 417
- T4 Fontkodierung, 428
- T5 Fontkodierung, 428
- T7 Fontkodierung, [428](#), 591
- \TAB' (Tabbing), 250
- \TAB= (Tabbing), [250](#)
- \TAB> (Tabbing), [250](#)
- Tabbing Umgebung (Tabbing), [250](#)
- Tabbing Paket, 250
- tabbing Umgebung, [248](#), [249](#), [250](#), [457](#), [458](#)
 - Fehler bei, 929, 947, 950, 953
 - (inputenc), 457
- \tabbingsep feste Länge, 249
- \tabcolsep feste Länge, [251](#), 255, 256, [258](#), [288](#), [290](#)
- Tabellen
 - Absoptionen, [253](#), [254](#)
 - Akzente, [249](#), [250](#)
 - am Ende eines Dokumentes platzieren, 298–300
 - Ausrichtung
 - horizontal, 268
 - vertikal, 254, [281](#), [282](#)
 - Breite
 - automatisch berechnen, [259–262](#), 263, [290](#)
 - bei mehrseitigen, 266, 267, [268](#), 269, 270, [271](#), 272
 - dehnen, 254
 - explizit berechnen, 257, [258](#), 259
 - Weißraum gleichmäßig verteilen, [287](#), [288](#)
 - Dezimalwerte ausrichten, [280](#), 282, [283](#), [284](#)
 - färben, [272](#), [273](#)
 - Fonts einstellen, [252](#), 253
 - Fußnoten, 270, [285](#), [286](#), [287](#)
 - Gleitobjekte, 325–331
 - in Tabellen, [288](#), [289](#)
 - Layoutparameter, 251
 - Linien
 - doppelte, 278
 - farbige, [273](#)
 - formale, 277, [278](#), 279, [280](#)
 - gestrichelte, 275, [276](#)
 - horizontale und vertikale kombinieren, 274, [275](#)
 - variable Stärke, [274](#)
 - vertikale, 274, 275, 277
 - mehrseitige
 - Abstände um, 269
 - Anzahl der Läufe reduzieren, 271
 - Breite, 266, 267, [268](#), 269, 270, [271](#), 272
 - erstellen mit longtable, 267, [268](#), 269, [270–272](#)
 - erstellen mit supertabular, 264, [265](#), [266](#), 267
 - Fußnoten, 270
 - horizontale Ausrichtung, 268
 - Kolumnentitel, 265, 269
 - Legenden, 265, [270](#)
 - Probleme mit, [271](#), [272](#)
 - Seitenumbrüche, 265
 - und Gleitobjekte, [270–272](#)
 - Zeilenbefehle, 269
 - Präambelbefehle/-kürzel, 251–256, 262
 - seitenübergreifende, 263, 264, [265](#), [266](#), 267, [268](#), 269, 270, [271](#), [272](#)
 - Silbentrennung, [254](#)

- Tabellen (*Forts.*)
- Spalten
 - Abstände, [255](#), [256](#)
 - Breite automatisch berechnen, [259–262](#), [263](#), [290](#)
 - Breite explizit berechnen, [257](#), [258](#), [259](#)
 - eine anders formatierte, [256](#), [257](#)
 - gestalten, [248–251](#)
 - globale Änderungen, [253](#), [256](#), [257](#)
 - Layout verändern, [256](#), [257](#)
 - schmale, [254](#), [255](#)
 - Spaltenformatkürzel definieren, [256](#), [257](#)
 - Standardumgebungen, [248–251](#)
 - `\verb`-Unterstützung, [263](#)
 - verschachtelte, [288](#), [289](#)
 - visuelles Erscheinungsbild, [251](#)
 - Weißraum gleichmäßig verteilen, [287](#), [288](#)
 - Zeilen
 - Abstände, [252](#), [253](#), [277](#), [279](#)
 - gestalten, [250](#), [251](#)
 - Zellen verbinden, [280](#), [281](#), [282](#), [290](#)
 - Zeilenumbrüche, [255](#)
 - Tabellenverzeichnisse, [48–69](#)
 - in Inhaltsverzeichnissen, [51](#)
 - `thead` Option (`endfloat`), [298](#)
 - `\table` (`nfssfont.tex`), [380](#), [381](#)
 - `table` Option (`euro`), [104](#)
 - `table` Umgebung, [116](#), [270](#), [300](#), [315](#), [317](#)
 - Fehler bei, [936](#), [939](#), [946](#)
 - gleitet zu weit, [297](#)
 - Label in, [71](#)
 - Layoutparameter, [292](#), [294](#), [295](#)
 - Querverweis auf, [70](#)
 - Warnung bei, [969](#)
 - (`float`), [303](#)
 - (`multicol`), nicht unterstützt, [198](#)
 - (`rotfloat`), [307](#)
 - (`subfig`), [328](#), [329](#)
 - `table` Zähler, [883](#)
 - (`longtable`), [267](#)
 - `table*` Umgebung (`multicol`), [198](#)
 - `\tablecaption` (`supertabular`), [265](#), [266](#)
 - `\tablefirsthead` (`supertabular`), [265](#), [266](#)
 - `\tablehead` (`supertabular`), [265](#), [266](#)
 - `\tablelasttail` (`supertabular`), [265](#), [266](#)
 - `\tablename` (`babel`), [561](#), [567](#)
 - `tablenotes` Umgebung (`threeparttable`), [286](#), [287](#)
 - `\tableofcontents`, [25](#), [49](#), [50](#), [56](#), [58](#), [59](#), [175](#), [231](#)
 - (`minitoc`), [59](#)
 - (`shorttoc`), [59](#)
 - (`titletoc`), [64](#)
 - `\tableplace` (`endfloat`), [299](#)
 - `\tablesection` (`endfloat`), [298](#)
 - `tablesfirst` Option (`endfloat`), [299](#)
 - `\tabletail` (`supertabular`), [265](#), [266](#)
 - `\tablinesep` feste Länge (`tabs`), [277](#)
 - `tablist` Option (`endfloat`), [298](#)
 - `tabs` Paket, [277](#)
 - nicht kompatibel zu `array`, [277](#)
 - `tabsize` Schlüssel (`fancyvrb`), [169](#)
 - `tabular` Schlüsselwert (`jurabib`), [765](#)
 - `tabular` Umgebung, [110](#), [111](#), [112](#), [113](#), [248](#), [249](#), [250](#), [251–259](#), [272–647](#), [648](#), [649–1137](#)
 - Fehler bei, [927](#), [934](#), [938](#), [941–944](#)
 - Fußnoten in, [285](#)
 - Layoutparameter, [251](#)
 - (`array`), [252–257](#), [274](#), [282](#), [288](#), [289](#)
 - farbig, [272](#)
 - (`arydshln`), [276](#)
 - (`booktabs`), [278](#), [280](#)
 - (`colortbl`), [273](#)
 - (`dcolumn`), [283](#), [284](#)
 - (`hhline`), [275](#)
 - (`multirow`), [281](#), [282](#)
 - (`sidecap`), [335](#)
 - (`tabs`), [277](#)
 - (`threeparttable`), [286](#)
 - `tabular*` Umgebung, [250](#), [251](#), [263](#), [281](#), [286](#)
 - (`array`), [254](#), [258](#), [288](#)
 - `tabularc` Umgebung (`lb2`), [258](#)
 - `\tabularnewline`, [112](#), [254](#), [255](#), [257](#), [258](#), [260](#), [269](#)
 - `tabularx` Umgebung (`tabularx`), [259](#), [260](#), [261](#), [263](#), [285](#), [286](#), [290](#)
 - `tabularx` Paket, [259–261](#)
 - `\tabularxcolumn` (`tabularx`), [260](#)
 - `tabulary` Umgebung (`tabulary`), [261](#), [262](#), [263](#)
 - `tabulary` Paket, [256](#), [259](#), [261–263](#)
 - Tabulatoren darstellen, [168](#), [169](#)
 - `tabwindow` Umgebung (`picinpar`), [116](#)
 - `\tag` (`amsmath`), [484](#), [485](#), [494](#)
 - Fehler bei, [944](#), [950](#)
 - `\tag*` (`amsmath`), [494](#)
 - `\tagcurve` (`curves`), [630](#)
 - Tags (Formeln), [481](#), [482](#)
 - Definition, [481](#)
 - Formeln nummerieren, [494](#), [495](#)
 - Platzierung, [495](#), [496](#), [497](#)
 - `\tala` (`babel`), [579](#)
 - `\talloblong` (`stmaryrd`), [546](#)
 - `\tan`, [513](#)
 - `\tanh`, [513](#)
 - `\Tape` (`bbding`), [414](#)
 - `\Taschenuhr` (`ifsym`), [415](#)
 - `\tau`, [543](#)
 - `\tbinom` (`amsmath`), [506](#)
 - `tbtags` Option (`amsmath`), [485](#), [486](#)
 - `tcidvi` Option (`graphics`), [633](#)
 - TDS (T_EX Directory Structure), [863](#)
 - TDS-konforme Installation sicherstellen, [863–866](#)
 - technischer Index, [688](#)
 - `techreport` B_ET_EX-Publikationstyp, [791](#)
 - `\Telephone` (`ifsym`), [416](#)
 - `\Tent` (`ifsym`), [416](#)
 - `\tertio` (`babel`), [579](#)
 - `testpage.tex` Datei, [204](#)
 - `.tex` Dateierweiterung, [7](#), [9](#)
 - TeX capacity exceeded Fehler, [956–961](#)
 - T_EX Directory Structure (TDS), [863](#)
 - T_EX und L^AT_EX, Dateiübersicht, [9](#)
 - T_EX, Kodierung, [365](#)

- TeX-Dateien, Quellen
 - CD/DVD, 996, 997
 - FTP, 996, 1000-1002
 - Internetzugriff, 998
- TeX-Fontmetrikdateien, 7
- tex.bib Datei (lb2), 714, 716, 807
- tex.define Funktion (bibtool), 813
- tex_def.rsc Datei (bibtool), 813
- texdoc Programm, 1002, 1003
- texdoctk Programm, 1002, 1003
- texindy Programm, 690-695
- texmf.cnf Datei, 956
- texpicture Paket, 657, 658
- Text, *siehe auch* Fonts
 - Ausrichtung in Inhaltsverzeichnissen, 63, 64, 65
 - Balancieren von Spalten, 196, 197
 - Codefragmente in normalem Text, 180
 - getippter, *siehe* wortwörtliche Ausgabe
 - Groß-/Kleinschreibung verändern, 92, 93
 - hervorheben, *siehe* kursive Schriftform; unterstreichen
 - Layout in Überschriften, 31, 33, 39, 40
 - Mathematiksatze, 512, 513, 514
 - um eine Abbildung fließen lassen, 115, 116
 - unterstreichen, 93, 94, 99, 353
- \text
 - (amsmath), 479, 484, 488-490, 496, 498, 512, 522
 - (amstext), 363, 545
 - (nfssfont.tex), 380, 381
- text input levels Fehler, 961
- text.length\$ interne BibTeX-Funktion, 839
- text.prefix\$ interne BibTeX-Funktion, 839
- \textacutetdbl (textcomp), 375, 471
- \textascendercompwordmark (textcomp), 377, 471
- \textasciicircum (textcomp), 375, 471
- \textasciibreve (textcomp), 375, 471
- \textasciicaron (textcomp), 375, 471
- \textasciicircum (textcomp), 375, 471
- \textasciicircum (textcomp), 471
- \textasciidieresis (textcomp), 375, 471
- \textasciigrave (textcomp), 162, 375, 471
- \textasciimacron (textcomp), 375, 471
- \textasciitilde, 471
- \textasteriskcentered, 136
 - (textcomp), 375, 471
- \textbackslash, 350, 471, 674
- \textbaht (textcomp), 374, 472
- \textbar, 472
- \textbardbl (textcomp), 375, 472
- \textbf, 351, 355, 357, 418, 419, 451, 907
 - in Formeln, 363
 - (cmbright), 419
 - (lucidabr), 421, 422
 - (soul), 95
 - (ulem), ersetzt durch \uwave, 94
 - (yfonts), 405
- \textbigcircle (textcomp), 375, 472
- \textblack (fourier), 404
- \textblank (textcomp), 375, 472
- \textborn (textcomp), 375, 377-379, 396, 399, 402, 472
- \textbraceleft, 472
- \textbraceright, 472
- \textbrokenbar (textcomp), 375, 472
- \textbullet, 66, 67, 136, 144, 376
 - (textcomp), 375, 376, 472
- \textcapitalcompwordmark (textcomp), 377, 472
- textcase Paket, 92, 93
- \textcelsius (textcomp), 375, 472
- \textcent, 459
 - (textcomp), 374, 472
- \textcentoldstyle (textcomp), 374, 472
- \textcircled, 375, 465
 - Problem mit textcomp, 375
 - (textcomp), 374, 472
- \textcircledP (textcomp), 374, 472
- \textcolonmonetary (textcomp), 374, 472
- \textcolor (color), 165, 272, 617, 719, 721
- textcomp Paket, 95, 373-379, 399, 466-468
 - Fehler bei, 923, 930, 949
 - nicht verwendbar mit ae, 367
 - textcomp.cfg Datei (textcomp), 379
- \textcompsubstdefault (textcomp), 378, 949
- \textcompwordmark, 376, 377, 472
 - (textcomp), 377
- \textcopyleft (textcomp), 374, 378, 472
- \textcopyright (textcomp), 374, 472
- \textcurrency (textcomp), 374, 472
- \textcyrillic (babel), 584, 585
- \textdagger (textcomp), 375, 376, 472
- \textdaggerdbl (textcomp), 375, 472
- \textdblhyphen (textcomp), 375, 472
- \textdblhyphenchar (textcomp), 375, 472
- \textdegree (textcomp), 375, 472
- \textdied (textcomp), 375, 396, 399, 402, 472
- \textdiscount (textcomp), 374, 472
- \textdiv (textcomp), 375, 472
- \textdivorced (textcomp), 375, 472
- \textdollar, 377, 467, 472
 - (pxfonts), Probleme mit, 402
 - (textcomp), 374
 - (txfonts), Probleme mit, 400
- \textdollaroldstyle (textcomp), 374, 377, 396, 399, 402, 472
- \textdong (textcomp), 374, 472
- \textdownarrow (textcomp), 374, 472
- \texteightoldstyle (textcomp), 374, 472
- Texteingabe, Kodierung, 458, 459
- \textellipses, 87, 472
- \textemdash, 472
- \textendash, 136, 455, 472
- \textepsilon (tipa), 417
- \textestimated (textcomp), 374, 472
- \texteuro, 419, 466
 - nachgeahmt, 964
 - (luximono), 399
 - (textcomp), 104, 374, 379, 396, 399, 402, 418, 419, 472
- \textexclamdown, 455, 472
- \textfiveoldstyle (textcomp), 374, 472
- \textfloatsep elastische Länge, 293, 294
- \textflorin (textcomp), 374, 472

- textfont Schlüssel/Option
 - (caption), [319](#), [321](#), [322](#), [334](#)
 - (subfig), [326](#)
- \textfouroldstyle (textcomp), 374, 472
- \textfraction, 293, 295
- \textfractionsolidus (textcomp), 375, 473
 - Textfragmente formatieren, 479
- \textfrak (yfonts), [96](#), [405](#)
- \textgoth (yfonts), [405](#)
- \textgravedbl (textcomp), 375, 473
- \textgreater, 473
- \textgreek (babel), [584](#), [585](#)
- \textguarani (textcomp), 374, 473
- \textheight feste Länge, [18](#), [203–205](#), [206](#), [207](#), 216, 242, 295, [336](#), 383, [905](#), [922](#), 975
 - (fancybox), 615
 - (geometry), 215, 216
 - (lscape), 220
 - (supertabular), 264
- textheight Schlüssel/Option (geometry), 215, 219
- \textifsym (ifsym), [416](#)
- \textifsymbol (ifsym), [416](#)
- \textinit (yfonts), 407
- \textinterrobang (textcomp), 375, 473
- \textinterrobangdown (textcomp), 375, 473
- \textipa (tipa), [417](#)
- \textit, 351, 355, 357, [418](#), 672
 - in Formeln, 363
 - (lucidabr), [421](#), [422](#)
 - (yfonts), 405
- Textlänge, *siehe* Längenparameter
- \textlangle (textcomp), 374, 473
- \textlarger (relsize), 90
- \textlatin (babel), 584
- \textlbrackdbl (textcomp), 374, 473
- \textleaf (textcomp), 375, 473
- \textleftarrow (textcomp), 374, 473
- \textless, 473
- \textlira (textcomp), [104](#), 374, 473
- \textlnot (textcomp), 375, 473
- \textlquill (textcomp), 374, 473
- Textmarkenbefehle, 226, [227](#), 228, [229](#), 230, [237](#), [238](#)
- \textmarried (textcomp), 375, [396](#), [399](#), [402](#), 473
- \textmd, 351, 355, 357
- \textmho (textcomp), 375, 473
- \textminus (textcomp), 375, 473
- \textmu (textcomp), 375, 473
- \textmusicalnote (textcomp), 375, 473
- \textnaira (textcomp), 374, 473
- \textnineoldstyle (textcomp), 374, 473
- \textnormal, 175, [176](#), 350, 355
- \textnumero (textcomp), 375, [378](#), 473
- \textogonekcentered, 473
- \textohm (textcomp), 375, [378](#), [379](#), 473
- \textol (babel), 584
- \textonehalf, [459](#)
 - (textcomp), 374, 473
- \textoneoldstyle (textcomp), 374, 473
- \textonequarter (textcomp), 374, 473
- \textonesuperior (textcomp), 374, 473
- \textopenbullet (textcomp), 375, 473
- \textordfeminine (textcomp), 375, 473
- \textordmasculine (textcomp), 375, 473
- \textormath (babel), [458](#), 606, [607](#)
- \textparagraph (textcomp), 375, [376](#), 473
- \textperiodcentered, [105](#), 136, [192](#)
 - (textcomp), 375, 473
- \textpertenthousand
 - Probleme in T1, [428](#)
- (textcomp), 374, 473
- \textperthousand
 - Probleme in T1, [428](#)
- (textcomp), 374, 473
- \textpeso (textcomp), 374, 473
- \textpilmcrow (textcomp), 375, [378](#), 473
- \textpm (textcomp), 375, 473
- \textprimstress (tipa), [417](#)
- \textquestiondown, 473
- \textquotedbl, 473
- \textquotedblleft, 473
- \textquotedblright, 473
- \textquoteleft, 473
- \textquoteright, 473
- \textquotesingle (textcomp), 162, 375, 474
- \textquotestraightbase (textcomp), 375, 474
- \textquotestraightdblbase (textcomp), 375, 474
- \texttriangle (textcomp), 374, 474
- \texttrbrackdbl (textcomp), 374, 474
- \textrecipe (textcomp), 375, 474
- \textreferencemark (textcomp), 374, 474
- \textregistered, [465](#)
 - (textcomp), 374, 474
- \extrightarrow (textcomp), 374, 474
- \textrm, 350, 355, 357, 363
 - in Formeln, 363
- \textroundcap (tipa), 417
- \extrquill (textcomp), 374, 474
- \textsb (fourier), 404
- \textsc, 351, [352](#), 355, 357, [890](#)
 - in Formeln, 363
 - (fourier), 404
 - (relsize), [90](#)
- \textscale (relsize), 91
- \textschwa (tipa), [417](#)
- \textsection (textcomp), 375, 474
- \textservicemark (textcomp), 374, 474
- \textsevenoldstyle (textcomp), 374, 474
- \textsf, 350, 355, 357, 382, [418](#), [883](#)
 - in Formeln, 363
- \textsfbf (lb2), [96](#)
- \textsixoldstyle (textcomp), 374, 474
- \textsl, 351, 355, 357, [419](#)
 - in Formeln, 363
 - (cmbright), [419](#)
- \textsmaller (relsize), [90](#)
- \textsterling, [104](#), [467](#), 474
 - (pxfonts), Probleme mit, 402
 - (textcomp), 374
 - (txfonts), Probleme mit, [400](#)
- \textstyle, 444, 506, 515
 - (relsize), 91
- \textsuperscript, [120](#), [133](#), [717](#), [895](#), [906](#)

- \textsurd (textcomp), 375, 474
- \textswab (yfonts), 405
- Textsymbole, *siehe auch* mathematische Symbole;
 - Sonderzeichen
 - € (Euro-Zeichen), 418–423
 - Berge, 414, 415, 416
 - Hände, 411
 - IPA, 417, 418
 - Kodierung
 - Pi-Fonts, 390–393
 - PostScript, 399–401
 - TS1, 373, 374–379
 - MarVoSym Font, 412, 413, 414
 - Rückwärtskompatibilität, 475, 476
 - TIPA, 416–418
 - Uhren, 414, 415, 416
 - Waldis Font, 411, 412
 - Wolken, 414, 415, 416
 - Zapf Dingbats, 390–392
 - eine Alternative, 414, 415
- \TextSymbolUnavailable, 459
- \textthreeoldstyle (textcomp), 374, 474
- \textthreequarters (textcomp), 374, 474
- \textthreequartersemdash (textcomp), 375, 474
- \textthreesuperior (textcomp), 374, 474
- \texttildelow (textcomp), 375, 474
- \texttimes (textcomp), 375, 474
- \texttrademark (textcomp), 374, 474
- \texttt, 350, 355, 357, 398, 399, 418, 907
 - in Formeln, 363
 - (cmbright), 419
- \texttwelveudash (textcomp), 375, 474
- \texttwooldstyle (textcomp), 374, 474
- \texttwosuperior (textcomp), 374, 474
- \textunderscore, 474
- \textup, 150, 151, 351, 355, 357
- \textuparrow (textcomp), 374, 474
- \textupsilon (tipa), 417
- Textures Programm, 632, 633
- textures Option
 - (changebar), 199
 - (graphics), 632, 633
- \textvisiblespace, 102, 474
- \textwidth feste Länge, 190, 203–205, 207, 234, 904, 905, 922
 - (fancybox), 615
 - (fancyhdr), 232
 - (longtable), 269
 - (lscap), 220
- textwidth Schlüssel/Option (geometry), 215, 219
- \textwon (textcomp), 374, 474
- \textyen (textcomp), 374, 474
- \textzerooldstyle (textcomp), 374, 474
- .tfm Dateierweiterung, 7, 9, 337, 341, 351, 354, 424, 439, 440, 936, 937
- \tfrac (amsmath), 505, 506
- \tg (babel), 580
- \TH, 469
- \th, 474
 - (babel), 580
- \the, 138, 398, 887, 888, 980
 - Fehler bei, 939
- \the⟨zähler⟩, 886
- thebibliography Umgebung, 24, 25, 231, 710, 711, 715, 717, 724, 771, 840
 - im Inhaltsverzeichnis, 50
 - Warnung bei, 963
 - (bibunits), 778, 779
 - (chapterbib), 773, 774
 - (natbib), 732, 734
- \thebtauxfile (bibtopic), 781
- \thechapter, 27, 228, 887
 - (chappg), 225
 - (chapterbib), 775
- \theCodelineNo (doc), 857
- \thecontentslabel (titletoc), 63, 64, 65, 67
- \thecontentspage (titletoc), 63, 66, 67
- \theendnote (endnotes), 133
- \theendnotes (endnotes), 133, 134, 753
- \theenmark (endnotes), 133, 134
- \theenumi, 136, 137, 138, 886
- \theenumii, 136, 137, 886
- \theenumiii, 137, 886
- \theenumiv, 137, 886
- \theequation, 16, 75, 494, 887
 - (amsmath), 497
- \theFancyVerbLine (fancyvrb), 168
- \thefigure, 50
 - (subfloat), 332
- \thefootnote, 117, 285
- theglossary Umgebung, 673
- theindex Umgebung, 25, 231, 669, 702, 703
 - im Inhaltsverzeichnis, 51
- \thelstlisting (listings), 183
- \themainfigure (subfloat), 332
- \themaintable (subfloat), 332
- \themnote (lb2), 128
- \thempfootnote, 117, 285
- theorem Paket, 148
- theoremähnliche Strukturen, 146, 147, 148–150, 151, 152, 479
 - Abstände, 149
 - anpassen, 148–150, 151
 - benennen, 148, 149, 150, 151
 - Beweisführungen, 151, 152
 - Einzug, 149
 - Fonts, 149
 - Interpunktion, 149
 - Layout definieren, 148
 - Layoutnamen, 149
 - QED (□)-Zeichen, 151, 152
- \theoremstyle (amsthm), 148, 150, 151
- \thepage, 223, 224, 226, 232, 236, 239–241, 909
 - (chappg), 225
- \theparentequation (amsmath), 497
- \thepart, 886
- \thepostfig (endfloat), 299
- \theposttbl (endfloat), 299
- \therefore (amssymb), 551
- \Thermo (ifsym), 415, 416
- \thesection, 27, 28, 228, 886

- thesis Dokumentenklasse, 22
- \thesub(*klasse*) (subfig), 327
- \thesubfloatfigure (subfloat), 332
- \thesubfloattable (subfloat), 332
- \thesubsection, 27, 28, 886
- \thesubsubsection, 886
- \thesubtable (subfig), 328
- \Theta, 509, 543
- \theta, 487, 543
- \thetable (subfig), 328
- \thetitle (titlesec), 39
- \theTitleReference (titleref), 81
- \thevpagehrefnum (varioref), 78, 79
- \thickapprox (amssymb), 548
- \Thicklines
 - (eepicemu), 629
 - (eepic), 628
- \thicklines, 625, 629
 - (eepic), 627, 628
 - (epic), 620–622
 - (fancybox), 614
 - (pspicture), 658, 659, 660
- \thickmuskip elastische Länge, 521, 541, 542
- \thicksim (amssymb), 548
- \thickspace (amsmath), 521, 522
- \thinlines, 614
 - (epic), 620, 622
 - (pspicture), 658, 659, 660
- \thinmuskip elastische Länge, 521, 541, 542
- \thinspace, 521, 522
- \thisfancypage (fancybox), 617
- \thisfancyput (fancybox), 617
- \thisfancyput* (fancybox), 617
- \thispagestyle, 35, 230, 238, 702, 703
 - (fancyhdr), 238
 - (nextpage), 244
- thm Umgebung (lb2), 147, 148
- \thmname (amsthm), 150, 151
- \thmnote (amsthm), 150, 151
- \thmnumber (amsthm), 150, 151
- threeparttable Umgebung (threeparttable), 286
- threeparttable Paket, xxix, 285–287
- Tiefe, *siehe* Längenparameter
- Tiefe einer Box, 650
- Tiefstellungen, Positionieren von Grenzen, 503, 504
- .tif Dateierweiterung, 9, 644
- tight Option
 - (minitoc), 59
 - (shorttoc), 59
- \tilde, 545
- Tilde (~)
 - geschütztes Leerzeichen, 568
 - in verschiedenen Sprachen, 570, 571
- \time, 904, 906
- \times, 403, 502, 509, 546
 - Auslassungspunkte bei, 509
- times Option (quotchap), 38
- times Paket, 14, 382
- Times Roman
 - alternative Unterstützung, 399–401, 532
 - Beschreibung, 387
 - Times Roman (*Forts.*)
 - in Formeln und Text, 387, 388, 389, 399, 401, 532, 533
- \tiny, 181, 353, 354
 - tiny Option (titlesec), 39, 40
- tipa Paket, xxix, 416–418, 428
- tipaman Datei (tipa), 418
- Titel, Bibliographiedatenbank, 797
- Titel, Bibliographien
 - Formate, 744–746
 - Informationsfeld, 768, 769
 - Kurztitel zuordnen, 747, 748
- Titelbreite, in Überschriften messen, 44
- \title, 945
 - title BibTeX-Feld, 714, 757, 769, 791, 793, 797, 801, 808, 809
 - (jurabib), 742, 744, 747
 - titleaddon BibTeX-Feld (jurabib), 769
- \titleclass (titlesec), 47, 48
- \titlecontents (titletoc), 62, 63, 64, 65, 67, 68
- \titlecontents* (titletoc), 65, 66, 67, 69
- \titleformat (titlesec), 40, 42, 43, 44, 45–48, 68, 69, 98, 99
 - titleformat Schlüssel/Option (jurabib), 741, 745, 746, 747, 760, 761, 767
- \titleformat* (titlesec), 39, 40
- \titlelabel (titlesec), 39
- \titleline (titlesec), 44
- \titleline* (titlesec), 44, 45
 - titlepage Umgebung, 890
- \titleref (titleref), 80, 81
- titleref Paket, 80–82
- \titlerule
 - (titlesec), 44, 45
 - (titletoc), 63, 64
- \titlerule*
 - (titlesec), 44, 45
 - (titletoc), 65
- titlesec Paket, xxix, 38–48, 68, 232, *siehe auch*
 - Überschriften; titletoc-Paket
- \titlespacing (titlesec), 40, 41, 42, 43, 44, 45, 46, 48
- \titlespacing* (titlesec), 42, 69, 98, 99
- titletoc Paket, xxix, 59, 62–69, *siehe auch*
 - minitoc-Paket; titlesec-Paket
 - in diesem Buch, 1137
- \titlewidth feste Länge (titlesec), 44
- Tk Programm, 1003
- TM Math Font, 533
- tmargin Schlüssel/Option (geometry), 214, 216
- \tnote (threeparttable), 286
- \to, 503, 504, 514, 550
 - TOC, *siehe* Inhaltsverzeichnisse
- .toc Dateierweiterung, 8, 9, 25, 34, 35, 49, 50, 52, 457
 - (chapterbib), 775
 - (titletoc), 62, 64
- tocbind Paket, 51, 703
- tocdepth Zähler, 30, 52, 54, 55, 59, 64, 67, 68
- \tocdesign (layouts), 210
- \tocdiagram (layouts), 210

- `\today`, [92](#), [871](#)
 - (babel), [567](#), [574–576](#), [601](#), [604](#)
 - (rcsinfo), [872](#)
 - (rcs), [871](#)
- `today` Option (rcsinfo), [872](#)
- `\todayRoman` (babel), [574](#)
- `\toEng` (lb2), [906](#)
- `\tolerance`, [109](#), [110](#), [196](#), [987–989](#)
 - (multicol), [196](#)
- `\tone` (tipa), [417](#), [418](#)
 - tone Option (tipa), [417](#), [418](#)
- `\top`, [541](#), [544](#)
 - top Schlüssel (titlesec), [47](#), [48](#)
 - top Schlüssel/Option (geometry), [216](#), [217](#)
 - top Schlüsselwert
 - (caption), [321](#), [328](#)
 - (subfig), [327](#)
 - top\$ interne $\LaTeX$ -Funktion, [839](#)
 - topadjust Schlüssel/Option (subfig), [327](#)
- `\topcaption` (supertabular), [265](#)
- `\topfigrule`, [293](#), [294](#)
- `\topfraction`, [292](#), [293](#), [294](#)
- `\topleftxmark` (extramarks), [230](#)
- `\topline` Schlüsselwert (fancyvrb), [166](#), [167](#)
- `\topmargin` feste Länge, [203–205](#), [206](#), [905](#), [979](#), [980](#), [981](#)
- `\topmark`, [226](#), [229](#), [230](#)
 - topnumber Zähler, [292](#), [293](#)
- `\toprightxmark` (extramarks), [230](#)
- `\toprule` (booktabs), [278](#), [280](#)
- `\topsep` elastische Länge, [149](#), [153](#), [979](#), [980](#)
- `\topskip` elastische Länge, [206](#), [207](#), [982](#), [984](#)
 - (geometry), [215](#)
- `total` Schlüssel/Option (geometry), [219](#)
- `\totalheight`, [894](#), [895](#), [899](#)
 - (graphics), [648](#)
- `totalheight` Schlüssel (graphicx), [637](#), [641](#), [933](#)
- `totalnumber` Zähler, [292](#)
- `totalpages` $\LaTeX$ -Feld (jurabib), [769](#)
- `trace` Paket, [992](#), [993](#), [1027](#)
- `tracefmt` Paket, [379](#), [380](#)
- `\traceoff` (trace), [992](#)
 - traceoff Option (changebar), [200](#)
- `\traceon` (trace), [992](#), [993](#)
 - traceon Option (changebar), [200](#)
- `tracestacks` Option (changebar), [200](#)
- `\tracingall`, [986](#), [990](#), [991](#), [992](#), [993](#)
- `\tracingassigns` (trace), verfügbar unter $\epsilon\TeX$, [993](#)
- `\tracingcommands`, [991](#)
- `\tracinggroups`, verfügbar unter $\epsilon\TeX$, [944](#), [959](#)
 - (trace), [993](#)
- `\tracingifs`, [964](#)
- `\tracinglostchars`, [992](#)
- `\tracingmacros`, [991](#)
 - tracingmulticol Zähler (multicol), [195](#), [197](#)
- `\tracingonline`, [946](#), [959](#), [968](#), [983](#), [986](#), [990](#)
- `\tracingoutput`, [990](#)
- `\tracingpages`, [983](#)
 - erzeugte Ausgabe, [983](#)
- `\tracingparagraphs`, [986](#)
 - erzeugte Ausgabe, [987](#), [988](#)
- `\tracingrestores`, [990](#)
- `\tracingstats`, [957](#)
 - erzeugte Ausgabe, [957](#)
- `\tracingtabularx` (tabularx), [260](#), [261](#)
- `translator` $\LaTeX$ -Feld (jurabib), [769](#)
- Treiberdateien
 - erstellen, [850](#)
 - in bedingten Code einbinden, [852](#)
- Trennstrich (–), *siehe* Bindestrich
- Trennung nach Bindestrich, [570](#), [719](#)
- Trennzeichen
 - für Indexebenen, [671](#)
 - für Silbentrennung festlegen, [439](#)
 - Fonts, [438](#)
 - in Bibliographiedatenbanken, [790](#)
- `\triangle`, [544](#)
- `\TriangleDown` (bbding), [414](#)
- `\triangledown` (amssymb), [544](#)
- `\triangleleft`, [546](#)
- `\trianglelefteq` (amssymb), [549](#)
- `\trianglelefteqslant` (stmaryd), [549](#)
- `\triangleq` (amssymb), [548](#)
- `\triangleright`, [169](#), [546](#)
- `\trianglerighteq` (amssymb), [549](#)
- `\trianglerighteqslant` (stmaryd), [549](#)
- `\TriangleUp` (ifsym), [416](#)
- `trim` Schlüssel (graphicx), [637](#), [638](#), [639](#)
- `troff` Programm, [626](#)
- `true` Schlüsselwert
 - (caption), [318](#)
 - (fancyvrb), [165](#), [167](#), [168](#), [169](#), [173](#), [174](#)
 - (geometry), [214](#)
 - (jurabib), [741](#), [761](#)
 - (listings), [180](#), [182–184](#)
 - (titlesec), [46](#), [47](#)
- `true` Syntax, [907](#), [908](#)
- `truedimen` Schlüssel/Option (geometry), [219](#)
- `TrueTeX` Programm, [633](#)
- `truetest` Option (graphics), [633](#)
- `\truncate` (truncate), [240](#), [241](#)
 - truncate Paket, [240](#), [241](#)
- `\TruncateMarker` (truncate), [240](#)
- `\try<param>` (layouts), [208](#), [210](#)
- `\trycolumnsep` (layouts), [209](#)
- `\trycolumnseprule` (layouts), [209](#)
- `\tryevensidemargin` (layouts), [209](#)
- `\tryfootskip` (layouts), [209](#)
- `\tryheadheight` (layouts), [209](#)
- `\tryheadsep` (layouts), [208](#), [209](#)
- `\trypaperheight` (layouts), [209](#)
- `\trypaperwidth` (layouts), [209](#)
- `\trytextheight` (layouts), [209](#)
- `\trytextwidth` (layouts), [209](#)
- TS1 Fontkodierung, [124](#), [365](#), [366](#), [394](#), [428](#), [432](#), [466–468](#)
 - Liste der LICR-Objekte, [467–475](#)
 - (avant), [384](#)
 - (bookman), [384](#)
 - (ccfonts), [395](#), [396](#)
 - (chancery), [384](#)
 - (charter), [384](#)

- TS1 Fontkodierung (*Forts.*)
- (cmbright), 397
 - (courier), 384
 - (fourier), 402
 - (helvet), 384
 - (luximono), 398
 - (newcent), 384
 - (palatino), 384
 - (pxfonts), 401
 - (textcomp), 373–375, [377–379](#)
 - (times), 384
 - (txfonts), 400
 - (utopia), 384
- TS3 Fontkodierung, 428
- `\tt`, 358
- in Formeln, 361, 476
- `tt` Option (titlesec), 39
- `tt` Schlüsselwert
- (caption), 319
 - (fancyvrb), 163, [164](#)
- `\ttdefault`, [162](#), 350, [357](#), [398](#), [399](#)
- `\ttfamily`, 100, [350](#), [355](#), [357](#), [420](#), [476](#), [980](#)
- in Formeln, 360, 361
- `.ttt` Dateierweiterung (endfloat), 299
- TUG (T_EX Users Group) Homepage, 996
- turkish Option (babel), 559, [573](#)
- turn Umgebung (rotating), 306, [651](#), [652](#)
- `\twlrm` (lb2), [476](#)
- `\twlrm` Option (rawfonts), 476
- `\twocolumn`, 193, [702](#), [703](#)
- Warnung bei, 969
- `twocolumn` Option, [18](#), 121, 185, 193, 240
- `twocolumn` Schlüssel/Option (geometry), 215
- `\twocolumnlayouttrue` (layouts), [209](#)
- `\twoheadleftarrow` (amssymb), 550
- `\twoheadrightarrow` (amssymb), 550
- `twoside` Option, [207](#), [217](#), 754, [915](#)
- (biblist), 804
 - (layout), 208
- `twoside` Schlüssel/Option (geometry), 215, 216, [217](#)
- `txfonts` Paket, [399–401](#), 525, 526, 533
- einander berührende Zeichen, 401
- `\tyformat` (tabulary), 262, 263
- `\tymax` feste Länge (tabulary), [261](#), [262](#)
- `\tymin` feste Länge (tabulary), [261](#), [262](#)
- `type` Schlüssel (graphicx), 638, 645
- Fehler bei, 931
- `type` B_BT_EX-Feld, 791, 793
- `type$` interne B_BT_EX-Funktion, 839
- `\typearea` (typearea), [213](#), [214](#)
- `typearea` Paket, xxix, [211–214](#), 215, 244
- `typearea.cfg` Datei (typearea), 212
- `\typein`, 860
- `\typeout`, 444, 860, [927](#)
- typographische Fonts, *siehe* Proportionalschriften
- typographische Konventionen in diesem Buch, 12–15
- U**
- U Fontkodierung, 408, [428](#), 442, 447, [467](#)
- (eurosans), 421
 - (eurosym), 420
- `\u`, [377](#), [474](#)
- ucs Paket, 372
- Überlauffehler, 958–961, *siehe* Fehlersuche
- Überschriften, *siehe auch* Gliederungsbefehle;
- titlesec-Paket
- Abstände
- Label und Titeltext, 41
 - linker Rand, 42
 - nach, 31
 - oberhalb, 30
 - oberhalb/unterhalb, 42, 45
 - rechter Rand, 42
 - Schriftgrößen und, 43
 - vertikale, 39
 - vor, 30
 - Werkzeuge für, 42
 - zwischen aufeinander folgenden Überschriften, 43
- alphabetisch nummerieren, 28
- am Fuß einer Seite, 43
- Ausrichtung, 39
- bedingte Layouts, [46](#), [47](#)
- für Listen, *siehe* theoremmähnliche Strukturen
- formatieren, 29–36
- abgesetztes Format, 29, [30](#)
 - Abstand nach, 31
 - Abstand oberhalb, 30
 - besondere Layouts, 43
 - eingebettetes Format, [29](#), [31](#), [32](#)
 - Einzug der Überschrift, 30, 41
 - Einzug nach Überschriften, 34, 43
 - Einzug unterdrücken, 34, 42
 - Gliederungsebene, 30
 - komplexe Überschriften, 34
 - Labelformat, 41
 - Leitpunkte, 44, [45](#)
 - Linien, 44, [45](#)
 - mit Formen, 40
 - Nummern, [39](#), [40](#)
 - Nummern einrahmen, [28](#)
 - originelle Überschriften, 36, [37](#)
 - Randausgleich, 33
 - Silbentrennung, 33
 - Textlayout, 31, [33](#), 39, 40
 - vordefinierte Layouts, 36, [37](#)
 - vordefinierte Texte, [36](#)
 - Zähler, 30
 - Zeilenumbrüche, 33
- Gliederungsebene, 26
- Hierarchie verändern, 47, [48](#)
- Layoutdefinitionen, 34
- Messen der Titelbreite, 44
- Mottos (Zitate) für Kapitel, [37](#), [38](#)
- Nummerierung, 26, [27](#), 28, 29
- arabische Zahlen, [27](#)
 - formatieren, [39](#), [40](#)
 - Großbuchstaben, [28](#)
 - unterdrücken, 25, 26
 - Verweise auf Unterabschnitte, [28](#), [29](#)
- Quellenverweise in, 722
- Seitenumbruch vor, 45

- Überschriften (*Forts.*)
 - umbrechen, 26
 - umdefinieren, 34–36
 - unterdrücken (layouts), 210
 - Zähler erhöhen, [35](#), [36](#)
 - Zeilenumbrüche, 33
- Übersetzen
 - sprachspezifischer Zeichenfolgen, 603
 - von Dokumenten, *siehe* fremdsprachige Dokumente
- übersetzte Werke, Bibliographien, 768, 769
- Übersetzungen, Optionen (Sprachauswahl), 566, [567](#)
- überwachen
 - Absatzalgorithmus, 986–989
 - am Bildschirm, 989, 990
 - Fontauswahl, 379
 - Seitenumbruch, 980–985
- Uhren, Textsymbole, 414, [415](#), [416](#)
- UK-TUG FAQ, 995
- UKenglish Option (babel), 559
- ukrainian Option (babel), 559, 584
- `\ul` (soul), [95](#), [96](#), [99](#)
- `\ulcorner` (amssymb), 554
- `\ULdepth` feste Länge (ulem), 94
- `ulem` Paket, 93, 94
- `\ULforem` (ulem), 94
- `\uline` (ulem), 94
- `\ULthickness` (ulem), 94
- Umbrüche
 - Absätze, Fehlersuche, 985–989
 - Absatzalgorithmus
 - konfigurieren, 882
 - überwachen, 986–989
 - vorletzte Zeile, [882](#)
 - in Überschriften, 26
 - Seitenumbrüche
 - Badness-Grad, 892
 - Fehlersuche, 980–985
 - in mehrseitigen Tabellen, 265
 - in/um Formeln, 491–494
 - Index, Eintrag wiederholen, 703
 - Seitenlayout, 242, 243
 - vor Überschriften, 45
 - Spaltenumbrüche
 - Index, Eintrag wiederholen, 703
 - manuell erzeugt, [198](#)
 - Zeilenumbrüche
 - Badness-Grad, 892
 - in Bibliographien, 718
 - in Computercode, [182](#)
 - in Programmlistings, [182](#)
 - in Quellenverweisen, 718
 - in Tabellen, [255](#)
 - in Überschriften, 26, 33
 - in URLs, 100
 - numerische Quellenverweise, 718
 - vorletzte Zeile, [882](#)
- umdefinieren
 - Befehle, 876, [877](#), 879
 - Umgebungen, 880–883
- Umgebungen
 - benennen, 874, 875
 - Beschreibungen erstellen, 847, 848
 - die Miniseiten erzeugen (Mathematiksat), 489, [490](#), [491](#)
 - dokumentieren, *siehe* Dokumentationswerkzeuge
 - existierende umdefinieren, 880–883
 - Kurzformen, 480
 - neue definieren, 880, [881–883](#)
 - Umschreiben, Bibliographien, 809–811
 - `umvs.fd` Datei (marvosym), [414](#)
 - unabhängig nummerierte Fußnoten, 129, [130–132](#)
 - unabhängiger Index, 683
 - `unbalance` Zähler (multicol), [195](#), [196](#), 198
 - `\unboldmath`, 363
 - (bm), 526
 - `\UndeclareTextCommand`, [377](#), [467](#)
 - `\UndefineShortVerb` (fancyvrb), [177](#)
 - `\underaccent` (accents), [507](#)
 - `Underfull` box Meldung, [110](#), 215, 383, 440, 973–975
 - `\underleftarrow` (amsmath), [510](#)
 - `\underleftrightharpoon` (amsmath), [510](#)
 - `\underrightarrow` (amsmath), [510](#)
 - `\underset` (amsmath), [507](#), [508](#)
 - `\undertilde` (accents), [507](#)
 - `undotted` Option (minitoc), 59
 - Uniform Resource Locators, *siehe* URLs
 - `unitctnnoreset` Option (bibtopic), 781
 - `\unitlength` feste Länge
 - (eepic), [627](#), 628
 - (epic), [620–623](#), [625](#)
 - (pspicture), 659
 - `units` Schlüssel (graphicx), 650, 651
 - `\unkern`, [88](#)
 - `\unlhd` (latexsym), [476](#)
 - `unpack.ins` Datei, 861, 862
 - `unpublished` BibTeX-Publikationstyp, [714](#), 791
 - `\unrhd` (latexsym), [476](#)
 - `\unsethebrew` (babel), [584](#), [585](#)
 - `\unskip`, [79](#), [118](#), 155, [335](#)
 - unsortierte Quellenverweise, Layouts für, 722
 - `unsrt` BibTeX-Stil, 712, [822](#), [823](#), 824, [825](#), [837](#)
 - (bibtopic), 781
 - (notoccite), 722
 - `unsrtnat` BibTeX-Stil (natbib), [733](#), 735, [823](#)
 - Unterabschnitte
 - verweisen auf, [28](#), [29](#), 81
 - Unterbrechen abgesetzter Formeln, [491](#)
 - Unterformeln, Mathematiksat, 516, 517
 - untergeordnete Legenden, [325–329](#), 330, [331](#)
 - unterschneiden, 348
 - Unterstützung für Gesetzestexte, Bibliographien, 769, [770](#), 771
 - unterstreichen, [93](#), 94, [99](#), 353
 - unterteilte Abbildungen, [325](#), [329](#), [331](#)
 - unterteilte Tabellen, [326](#), [328](#)
 - `unzip` Programm, 420
 - `up` Option (titlesec), 39
 - `up` Schlüsselwert (caption), 319
 - `\Uparrow`, 511, 550

- `\uparrow`, 511, 550
 - `\updated` BibTeX-Feld (jurabib), 769
 - `\updatenam` (jurabib), 769
 - `\updatesep` (jurabib), 769
 - `\updefault`, 357
 - `\upDelta`
 - (ccfonts), 396
 - (cmbright), 397
 - (mathpazo), 389
 - (mathptmx), 388
 - `\Updownarrow`, 511, 550
 - `\updownarrow`, 511, 550
 - `\upharpoonleft` (amssymb), 550
 - `\upharpoonright` (amssymb), 550
 - `\uplus`, 546
 - `\upOmega`
 - (ccfonts), 396
 - (cmbright), 397
 - (mathpazo), 389
 - (mathptmx), 388
 - `\uppercase`, Probleme mit, 587, 877
 - `\uppersorbian` Option (babel), 559
 - `upquote` Schlüssel (listings), 162
 - `upquote` Paket, xxix, 161–163
 - `upref` Paket, 479
 - `upright` Option (fourier), 403
 - `\uproot` (amsmath), 517, 518
 - `\upshape`, 351, 352, 355, 357
 - `\Upsilon`, 543
 - `\upsilon`, 543
 - `\upuparrows` (amssymb), 550
 - `\urcorner` (amssymb), 554
 - `\url`
 - (custom-bib), 833
 - (natbib), 736
 - (url), 100, 101, 102, 800
 - Fehler in bewegtem Argument, 100
 - Probleme mit, 100
 - `url` BibTeX-Feld
 - (BibTexMng), 820
 - (custom-bib), 830, 833
 - (jurabib), 744, 769
 - (natbib), 735
 - `url` Paket, xxix, 99–103, 833
 - `\UrlBigBreaks` (url), 102
 - `\UrlBreaks` (url), 102
 - `urldate` BibTeX-Feld (jurabib), 769
 - `\urldatecomment` (jurabib), 769
 - `\urldef` (url), 101, 102
 - `\UrlLeft` (url), 102
 - Leerzeichen ignoriert in, 102
 - `\UrlNoBreaks` (url), 103
 - `\urlprefix` (custom-bib), 833
 - `\UrlRight` (url), 102
 - Leerzeichen ignoriert in, 102
 - URLs (Uniform Resource Locators)
 - Bibliographien, 735, 768, 769
 - formatieren, 99, 100, 101, 102, 103
 - Zeilenumbrüche, 100
 - `\urlstyle` (url), 101, 102
 - URW Antiqua Font, 404, 405
 - URW Grotesk Font, 404, 405
 - `\usage` (doc), 855
 - `\usebox`, 316, 882, 901, 902, 903
 - Fehler bei, 943
 - (soul), 96
 - `\usecounter`, 159
 - `\usedir` (docstrip), 863, 864, 865
 - `\usefont`, 383, 385, 419, 429
 - USenglish Option (babel), 559
 - `\UseOption` (optional), 24
 - `\usepackage`, 15, 18, 19, 20, 912, 914–916, 961
 - Fehler bei, 935, 954
 - Versionsdaten, 912
 - Warnung bei, 976
 - `\usepostamble` (docstrip), 860, 862
 - `\usepreamble` (docstrip), 862
 - Usergroups, 1004–1006, *siehe auch* Hilfsquellen
 - `\useshorthands` (babel), 564
 - `\UseTDS` (docstrip), 865, 955
 - `\UseTextAccent`, 467
 - (textcomp), 378
 - `\UseTextSymbol`, 376–378, 467
 - `usetoc` Option (titleref), 82
 - `\UseVerb` (fancyvrb), 174, 175, 176
 - `\UseVerb*` (fancyvrb), 175
 - `\UseVerbatim` (fancyvrb), 176
 - `usorbian` Option (babel), 575
 - UTF-8 Unterstützung, Kodierung, 372, 454, 459, 460
 - `utf8` Option (inputenc), 371, 372, 457, 557, 691
 - `utf8enc.dfu` Datei (inputenc), 460
 - `utopia` Paket, 382
 - Utopia Font, 387
 - in Formeln und Text, 531
 - `\uuline` (ulem), 94
 - `\uwave` (ulem), 94
 - `UWforbf` Option (ulem), 94
- ## V
- `\v`, 474
 - `\val` (euro), 105
 - `\value`, 138, 206, 285, 885, 904, 906, 909, 927, 980
 - Fehler bei, 943
 - `\varbigcirc` (stmaryrd), 547
 - `\varbigtriangledown` (stmaryrd), 546
 - `\varbigtriangleup` (stmaryrd), 546
 - `\varcopyright` (stmaryrd), 544
 - `\varcurlyvee` (stmaryrd), 546
 - `\varcurlywedge` (stmaryrd), 546
 - `\varepsilon`, 486, 517, 543
 - `\varhat` (lb2), 410
 - Variablen, Bibliographien, 836
 - `\varinjlim` (amsmath), 512, 513
 - `varioref` Option (fltpage), 336
 - `varioref` Paket, 72–79, 80, 560, *siehe auch* Querverweise
 - `\varkappa` (amssymb), 543
 - `\varliminf` (amsmath), 512, 513
 - `\varlimsup` (amsmath), 513, 514
 - `\varnothing` (amssymb), 544
 - `\varoast` (stmaryrd), 547
 - `\varobar` (stmaryrd), 547

- `\varobslash` (stmaryrd), 547
- `\varocircle` (stmaryrd), 547
- `\varodot` (stmaryrd), 547
- `\varogreaterthan` (stmaryrd), 547
- `\varolessthan` (stmaryrd), 547
- `\varominus` (stmaryrd), 547
- `\varoplus` (stmaryrd), 547
- `\varoslash` (stmaryrd), 547
- `\varotimes` (stmaryrd), 547
- `\varovee` (stmaryrd), 547
- `\varowedge` (stmaryrd), 547
- `\varphi`, [486](#), [517](#), 543
- `\varpi`, 543
- `\varprojlim` (amsmath), 513
- `\varpropto` (amssymb), 551
- `\varrho`, 543
- `\varsigma`, 543
- `\varsubsetneq` (amssymb), 549
- `\varsubsetneqq` (amssymb), 549
- `\varsupsetneq` (amssymb), 549
- `\varsupsetneqq` (amssymb), 549
- `\vartheta`, 543
- `\varthm` Umgebung (lb2), [151](#)
- `\vartimes` (stmaryrd), 546
- `\vartriangle` (amssymb), 549
- `\vartriangleleft` (amssymb), 549
- `\vartriangleright` (amssymb), 549
- varumlaut Option (yfonts), [405](#), [407](#)
- `\vbadness`, 967, 973, [974](#)
- `\vbox`, 383, 903, 973, [982](#)
 - in TeX-Warnungen, 967, 970, 974
- `\vcenter`, [501](#)
- `\Vdash` (amssymb), 551
- `\vDash` (amssymb), 551
- `\vdash`, 551
- `\vdots`, 552
- `\vec`, 545
- `\Vector` (pspicture), [659](#), [660](#)
- `\vector`
 - Fehler bei, 930
 - (pspicture), [658](#), 659, [660](#)
 - (texpicture), [659](#)
- `\vee`, 546
- `\veebar` (amssymb), 546
- Vektorzeichnungen, *siehe* epic-Paket; eepic-Paket
- `\veqns` (lb2), [77](#)
- Verallgemeinerungen, mathematische Symbole, [505](#), [506](#)
- Verarbeitung von Fehlern, *siehe* Fehlersuche
- Verarbeitungsablauf
 - Bibliographien, 837–840
 - BiBTeX, 712
 - Indexerstellung, 668, 695
 - L^AT_EX, 10
 - Quellenverweise, 711–715
- `\Verb` (fancyvrb), [176](#)
- `\verb`, 100, [160](#), [174](#), [176](#), [177](#), 180, 878, [890](#)
 - Ausgabe drehen, [652](#)
 - Fehler bei, 954
 - (boxedminipage), [613](#)
 - (doc), 848
- `\verb` (Forts.)
 - (ltxdoc), 867
 - (parallel), [191](#)
 - (shortvrb), [160](#)
 - (tabularx), bedingt zulässig in, 263
 - (tabulary), bedingt zulässig in, 263
 - (upquote), 162
- `\verb*`
 - (shortvrb), [160](#)
 - (tabularx), bedingt zulässig in, 263
 - (tabulary), bedingt zulässig in, 263
- Verbatim Umgebung (fancyvrb), [163](#), [164–171](#), 172, [173](#)
- verbatim Umgebung, [159](#), 160, 163, 878, 929, *siehe auch* wortwörtliche Ausgabe; Schreibmaschinenschrift
 - Fehler bei, 954
 - (doc), 848, 854
 - (parallel), 191
 - (upquote), 162
 - (verbatim), 161
- verbatim Paket, [161](#), 163
- Verbatim* Umgebung (fancyvrb), 169
- `\Verbatim*` (fancyvrb), 173
- verbatim* Umgebung
 - (doc), 854
 - (verbatim), 161
- Verbatim-Begrenzungssymbole
 - docstrip, 866
- Verbatim-Text, *siehe* wortwörtliche Ausgabe
- `\verbatimchar` (doc), 855
- `\VerbatimEnvironment` (fancyvrb), [172](#)
- `\VerbatimFootnotes` (fancyvrb), [176](#)
- `\VerbatimInput` (fancyvrb), [172](#)
- `\VerbatimInput*` (fancyvrb), 173
- VerbatimOut Umgebung (fancyvrb), [171](#), [172](#)
- verbose Option
 - (cite), 720
 - (placeins), 297
 - (wrapfig), 310
- verbose Schlüssel/Option (geometry), 218
- `\verbx` (lb2), [176](#)
- Vereinheitlichen von Indexeinträgen, 698
- Vereinheitlichungsregeln, 695, 698
- Vergrößerung, Seitenlayout, 218
- Verlaufsbeefehle (Dokumentation), Liste der, 853–857
- Verlaufsmeldungen unterdrücken (Indexerstellung), 678, 690, 697
- Versalien, 344, *siehe* Großbuchstaben
- Versalziffern, 378
- Verschachtelte Befehle, 878, [879](#)
- verschundene Zeichen aufspüren, 991
- Versionen für den Ausdruck auswählen, [23](#), 24
- Versionsdaten
 - Paketdateien, 910–913
 - Sprachdefinitionsdateien, 599
- Versionsverwaltung, 23, 24, 869, [870](#), 871, [872](#)
- `\VERT` (fourier), [403](#)
- `\Vert`, 511, 544
- `\vert`, 511, 544

vertikal erweiterbare mathematische Symbole, [511](#), [512](#)

vertikale Linien, [274](#), [275](#), [277](#)

Verweise, *siehe* Querverweise; Quellenverweise

Verweisschlüssel, *siehe* Schlüssel

Verzeichnisse, 48–69

- in Inhaltsverzeichnissen, 51
- Layoutparameter, 54
- Namen formatieren, 99, [100](#), [101](#), 102, 103

Verzierungen, [106–108](#), 406, [407](#), 614–618, *siehe auch* Initiale; Sonderzeichen; Textsymbole

`\vfill`, [198](#), [890](#), [891](#), [899](#)

Viertelkreise, *siehe* epic-Paket; eepic-Paket

viewport Schlüssel (graphicx), [637](#), [639](#)

`\Village` (ifsym), [416](#)

`\Virgo` (marvosym), [413](#)

visuelle Formatierung, 242–244

`\vitem` (lb2), [176](#)

`\vline`, 251, [273](#), 274, 275

vmargin Schlüssel/Option (geometry), 219

vmargin Paket, 211

vmarginratio Schlüssel/Option (geometry), [216](#), [217](#), 219

Vmatrix Umgebung (amsmath), [499](#)

vmatrix Umgebung (amsmath), [499](#)

vmode boolesche Var., 908

`\voffset` feste Länge, [204](#), 218

- (vmargin), 211

voffset Schlüssel/Option (geometry), 218

volume $\LaTeX$ -Feld, [714](#), [791](#), [793](#), [801](#)

volumetitle $\LaTeX$ -Feld (jurabib), 769

vordefinierte Layouts für Überschriften, 36, [37](#)

vordefinierte Texte für Überschriften, [36](#)

`\vpageref` (varioref), 72, [73](#), [74](#), 75, [77](#), 78, 79

`\vpageref*` (varioref), 74

`\vpagerefrange` (varioref), 73, 75

`\vpagerefrange*` (varioref), 75

`\vphantom`, [518](#), [521](#)

`\Vref` (varioref), 76

`\vref`

- (prettyref), 80
- (varioref), 72, 73, 74, 76, 78, 79, 957
- mögliche Fehler, 79

`\vref*` (varioref), 73

`\vrefpagenum` (varioref), [77](#)

`\vrefrange` (varioref), 73, [74](#), [75](#)

`\vrule`, [274](#), [900](#), 901

vscale Schlüssel/Option (geometry), [216](#), 219

`\vspace`, [618](#), [890](#), [891](#), 892, [897](#), [898](#), [900](#), [901](#), 951

- Fehler bei, 940
- Probleme mit, [890](#), [892](#)

`\vspace*`, [45](#), [119](#), [890](#), [897](#), 898

VT ϵ X Programm, 428, 633, 661, 662

vtex Option

- (changebar), 199
- (crop), 222
- (graphics), 633

vtex Schlüssel/Option (geometry), 218

`\Vdash` (amssymb), 551

W

w. eps Datei (lb2), 634, [635](#)

Währungssymbole, *siehe* Währungen, formatieren

Währungen

- € (Euro-Zeichen), [418–423](#)
- formatieren, [103–106](#)
- Symbole, [374](#), [423](#)

Waisenkind, 984

Waldis Font, 411, [412](#)

warn Option (textcomp), [378](#), [379](#), 950

warning\$ interne $\LaTeX$ -Funktion, 839

warningshow Option (tracefnt), 380

Warnung

- Overfull box, 55, [109](#), 969, 970, [985](#)
- Underfull box, [110](#), 215, 383, 440, 973–975

Warnungen, 961–977, *siehe auch* Meldungen; Fehlersuche

Wasserzeichen, [617](#)

wasysym Paket, 411, 412

`\wd`, [316](#)

weather Option (ifsym), [415](#), [416](#)

`\wedge`, 546

Weißraum

- in Tabellen, [287](#), [288](#)
- Kursivkorrektur, 351, [352](#), [353](#)
- um Text, 207

welsh Option (babel), 559

welsh.ldf Datei (babel), 599

wget Programm, 1000

while\$ interne $\LaTeX$ -Funktion, 839

`\whiledo` (ifthen), [909](#)

`\whline` (lb2), [274](#)

wide Option (sidecap), 333, [334](#)

`\widehat`, [509](#), [527](#), 545

- (bm), 527

`\WideMargins` (a4), 207

widespace Schlüsselwert (lb2), [323](#)

`\widetilde`, [495](#), [509](#), [521](#), 545

- (fourier), [403](#)

`\widowpenalty`, [982](#), 984

`\width`, [894](#), 895

- (graphics), 648
- (wrapfig), [310](#)

width Option (crop), 221

width Schlüssel (graphicx), [637](#), [639](#), [640–642](#)

width Schlüssel/Option

- (caption), 318
- (geometry), [216](#), 219

width Syntax, [235](#), [900](#), [901](#)

width\$ interne $\LaTeX$ -Funktion, 839

window Umgebung (picinpar), [115](#)

Windows Datenbankmanager, Bibliographien, 819, 820

Windvi Programm, 1002, 1003

withprosodicmarks Attribut (babel), [565](#), [573](#)

Witwe, 984

`\wlog`, 444

Wörterbücher, Kolumnentitel für, 239, [240](#)

Wolken, Textsymbole, 414, [415](#), [416](#)

`\wordsep` (titlesec), 43

wortwörtliche Ausgabe, *siehe auch*
 Schreibmaschinenschrift;
 verbatim-Umgebung
 Anführungsstriche in Programmlistings, 161,
 162, 163
 angepasste Varianten, 173, 174
 Befehle in, ausführen, 169, 170
 Boxen, 172, 173
 Computercode setzen, 177, 178, 179, 184
 als Gleitobjekt, 183
 Eingabekodierung, 184
 Einzug, 181
 Fragmente in normalem Text, 180
 Legenden, 183
 Linien/Rahmen um Listings, 182, 183
 nummerierte Zeilen, 180, 181
 Programm-Schlüsselwörter formatieren,
 179, 180
 unterstützte Programmiersprachen, 178
 Zeilenumbrüche, 182
 dicktengleiche Schrift, 161, 162, 163
 einfärben
 Hintergrund, 166
 Rahmenlinien, 166
 Text, 164, 165
 einrahmen, 165, 166
 Einzug entfernen, 165
 Escape-Zeichen, 169, 170
 führende Leerzeichen entfernen, 165
 Fonts festlegen, 163, 164
 hervorheben, *siehe* kursive Schriftform;
 unterstreichen
 Hintergrund einfärben, 165, 166
 in Argumenten, 174, 175, 176, 177
 in Dateien, 171
 in Fußnoten, 176
 Kopf-/Fuß-Begrenzungssymbole, 167
 Leerzeichen darstellen, 168, 169
 Lesen von Verbatim-Daten, 172
 nummerierte Zeilen, 167, 168
 nur Teil der Daten setzen, 170, 171, 172
 Schreiben von Verbatim-Daten, 171
 Sonderzeichen, 160, 161
 Start-/Stop-Begrenzungssymbole, 160, 161, 176,
 177
 Tabulatoren darstellen, 168, 169
 vertikale Abstände, 167
 \wp, 544
 \wr, 546
 wrap Schlüssel (titlesec), 40, 41, 42–44
 wrapfig Paket, 185, 308–311
 wrapfigure Umgebung (wrapfig), 308, 309, 310, 311
 wrapfloat Umgebung (wrapfig), 311
 \wrapoverhang feste Länge (wrapfig), 310, 311
 wrappable Umgebung (wrapfig), 308, 309–311
 \write, 138
 write\$ interne $\LaTeX$ -Funktion, 839, 841
 Wurzeln, mathematische Symbole, 517, 518
 www $\LaTeX$ -Publikationstyp (jurabib), 768, 769

X

X Syntax (tabularx), 259, 260, 263
 x Schlüssel (graphicx), 650, 651
 x-Höhe, 440
 X2 Fontkodierung, 366, 428, 586
 xdoc Paket, 846
 xdoc2 Paket, 846
 xdvi Programm, 632, 633, 1002
 .xdy Dateierweiterung, 8
 \Xi, 543
 \xi, 543
 xindy Programm, 8, 556, 590, 667, 668, 671, 673,
 688–702, 1020, 1074, *siehe auch*
 Indexerstellung; *MakeIndex*-Programm
 \xleftarrow
 (amsmath), 502
 (fourier), 403
 xleftmargin Schlüssel
 (fancyvrb), 165
 (listings), 181
 \xmlcode (lb2), 302
 XMLexa Umgebung (lb2), 301, 302, 307
 XMLexa* Umgebung (lb2), 307
 .xmp Dateierweiterung (lb2), 58
 \xout (ulem), 94
 Xpdf Programm, 661
 \xquad (lb2), 67
 xr Paket, 82, 83
 xr-hyper Paket, 83
 \xrightarrow (amsmath), 502
 xrightmargin Schlüssel (fancyvrb), 165
 \XSolid (bbding), 414
 \XSolidBold (bbding), 414
 \XSolidBrush (bbding), 414
 \xspace (xspace), 86, 87
 xspace Paket, 86, 87
 \xspaceskip elastische Länge, 440
 \xwordsdown (fourier), 403
 \xwordsup (fourier), 403
 \xvec (lb2), 876, 877, 977, 980
 Xverbatim Umgebung (lb2), 174
 xy Umgebung (xy), 565
 xypic Paket, 612, 1018

Y

\y (docstrip), 860
 y Schlüssel (graphicx), 650, 651
 \Ydown (stmaryrd), 546
 year Schlüsselwert (jurabib), 743, 744, 759
 year $\LaTeX$ -Feld, 714, 791, 793, 797, 801, 808, 809,
 810
 (jurabib), 742, 744
 \yen (amssymb), 544
 \yes (docstrip), 860
 yfonts Paket, 405–407
 \Yingyang (marvosym), 413
 \yinipar (yfonts), 407
 \Yleft (stmaryrd), 546

\Yright (stmaryrd), 546

\Yup (stmaryrd), 546

Z

Zähler

- Beschreibung, 883, 884
- darstellen, [885](#), 886, 887
- einstellen, 884
- erhöhen, [885](#)
- für Überschriften, [30](#), [35](#), [36](#)
- Fußnoten, je Seite zurücksetzen, 127, [128](#)
- Liste der, 883
- neue definieren, 884
- verändern, [885](#)

Zapf Chancery Font, 387

Zapf Dingbats

- eine Alternative, [414](#), [415](#)
- Kodierung, 390-392

Zeichenfolgen, Bibliographien

- alle Einträge durchsuchen nach, 805, 807
- erstellen, 798, 799
- Schlüssel durchsuchen nach, 805
- Voreinstellungen, 800

Zeichensätze, fremdsprachige Dokumente, 557

Zeichnen

- Bögen, 628
- Ellipsen, 628
- Kreise, 628
- Linien, 621, [622](#), 628, *siehe auch* epic-Paket; eepic-Paket
- Pfade, 628
- Vektoren, *siehe* epic-Paket; eepic-Paket
- wissenschaftlicher Daten, *siehe* Graphen

Zeilen

- Fonts für Zeilennummern, 188, [189](#)
- Nummerierung, 185, [186](#), [187](#), 188, [189](#), [190](#)
- Verweise auf Zeilennummern, 187, [188](#)
- Zeilenanzahl pro Seite, 207

Zeilen, Tabelle

- Abstände, 252, [253](#), [277](#), 279
- Befehle, 269
- gestalten, 250, 251
- Zellen verbinden, [280](#), [281](#), [282](#), [290](#)

Zeilenabstand, 426, *siehe auch* Zeilenvorschub

Zeilenumbrüche, *siehe auch* Längenparameter

- Badness-Grad, 892
- Bibliographien, 718
- in Computercode, [182](#)
- in Programmlistings, [182](#)
- in Quellenverweisen, 718
- in Tabellenzellen, [255](#)
- in Überschriften, 26, 33
- in URLs, 100
- numerische Quellenverweise, 718
- vorletzte Zeile, [882](#)

Zeilenvorschub, 426

- einstellen, [114](#), 115, 383
- Fontwechsel, [354](#)

Zellen verbinden, Tabellenzeilen, [280](#), [281](#), [282](#), [290](#)

zentrierte Absätze, 111, 112

zerbrechliche Befehle, 674, 906, 927, 928, 946, 953, 955

$\mathcal{MS}$ - $\mathcal{E}\mathcal{X}$, 480

\zero (euro), [106](#)

zerohyph.tex Datei (babel), 562

zeros Option (euro), 104

\zetaeta, [403](#), [502](#), [543](#)

Ziffern, mathematische Symbole, 542-545

Zirkumflex (^), Kurzzeichen, [573](#)

Zitate

- für Kapitelüberschriften (Mottos), 37, [38](#)
- Umgebung für, 154, [155](#)

Zitierschemata

Autor mit Jahr, 722-737

- Anpassen von Bibliographien, [732](#)
- Anpassen von Quellenverweisen, [730](#), [731](#)
- ausführliche Verweise im Fließtext, [736](#), 737
- Autor mit Nummer, Umschalten auf, 739
- Autoren nur beim ersten Verweis auflisten, [729](#), [730](#)
- Definition, 708
- elektronische Veröffentlichungen, 735
- erzwingen, [734](#)
- fehlende Autoredaten, [733](#), [734](#)
- fehlende Jahresdaten, 733, 734
- Geschichte, [724](#), [725](#)
- kein Umbruch zwischen Autorennamen, 731
- Kurztitelformat, kombinieren mit, 758, 759
- mehrfache Verweise, [728](#), [729](#)
- numerische Verweise, Umschalten auf, 739
- unterstützte Stile, 735, 736
- Verweise automatisch indexieren, 734

Autor mit Nummer, 737

- Anpassen von Quellenverweisen, [740](#)
- Beschreibung, 737
- Definition, 710
- Sortierreihenfolge, [739](#)
- Verweise komprimieren, [739](#)

Harvard, 708, 713

Kurztitel, 741-771

- Übersetzerdaten, 769
- Anpassen von Bibliographien, 762, [763](#), 764, [765](#), [766](#), 767
- Anpassen von Quellenverweisen, [761](#), [762](#)
- ausführliche Quellenverweise im Fließtext, [748](#), [749-751](#)
- Autor mit Jahr, kombinieren mit, [758](#), [759](#)
- Autoren-Informationsfeld, 768
- Beschreibung, 741
- Definition, 708
- Dissertationsjahr, 768
- Editionsdaten, 768
- einleitender Text, 746, [747](#)
- Einzug, 764, [765](#)
- Endnotenverweise, 752, [753](#)
- Fonts, 762, 763
- Fußnotenverweise, [752](#), [753](#)
- Geschlecht der Autoren, 760, [761](#), 768
- Gründerdaten, 768
- Herausgeberdaten, 768
- hochgestellte Zeichen, [761](#), 769

Zitierschemata (*Forts.*)

- ibidem-Verweise, 753, [754-757](#), 765, [766](#)
- Interpunktion, 763, [764](#)
- Klammern, 761
- Kommentare, 746, [747](#), [766](#), 768
- Konfigurationsdateien, externe, 767
- mehrsprachige Dokumente, 759, [760](#), 761
- Querverweise, 757, [758](#)
- Rückverweise, 768
- Sammlungen, 768
- Seitenanzahl-Feld, 769
- Seitengrenzen ignorieren, 755
- Sortierreihenfolge, 769
- Spaltenlayout, [765](#)
- Stildateien, 767-769
- Titel eines Bandes, 769
- Titel Kurztiteln zuordnen, [747](#), [748](#)
- Titel-Informationsfeld, 768, 769
- Titelformat, [744-746](#)
- Trennzeichen und -wörter bei mehreren
 - Autoren, 761, [762](#), 764
- übersetzte Werke, 768
- Unterstützung für Gesetzestexte, 769, [770](#), 771
- URLs, 768, 769
- Verweise automatisch indexieren, 746
- zuletzt geändert-Feld, 769
- numerische, 715-722
 - Anpassen von Quellenverweisen, [716](#), [717](#), 718, [719](#), [720](#)
 - ausführliche Warnungen (verbose), 720
 - Definition, 710
 - Farben, [719](#)
 - hochgestellte Zeichen, 720, [721](#), 722
 - in Überschriften, 722
 - in Legenden, 722
 - Interpunktion, 719, [721](#), [722](#)
 - Klammern, [720](#)
 - Layouts für unsortierte Verweise, 722
 - Leerzeichen in/vor Verweisen, 720
 - natbib-Paket, 737-740
 - Quellenverweise komprimieren, 739
 - Seitenbereiche deaktivieren, 720
 - Sortierreihenfolge, [718](#), 720, 739
 - Zeilenumbrüche, 718
- zuletzt geändert-Feld, Bibliographien, 769
- zusammenführen, Bibliographien, 808, 809
- zusammengesetzte mathematische Symbole, [502-508](#)
- zusammengesetzte Seitenzahlen, Index, 686
- Zusammenhalten von Material auf einer Seite, 242, *siehe auch* Gleitobjekte
- Zwischensteg, 203

Personen

- Abbott, Peter, 996
Achilles, Alf-Christian, 803
Aguilar-Sierra, Alejandro, 787
André, Jacques, 1012, 1027
Arseneau, Donald, xxix, 22, 23,
80, 90, 93, 99, 126,
240, 277, 285, 297,
308, 718, 720, 722,
773
Ashton, James, 703

Böttcher, Stephan, xxix, 185
Barr, Michael, 500, 1012
Barroca, Leonor, 305
Basso, Pierre, 779
Batada, Nizar, 817
Beccari, Claudio, 590, 1012
Beebe, Nelson, 633, 802, 803,
806, 807, 820, 1013
Beeton, Barbara, xxviii, 1013
Benguia, Ed, 385
Bennett, Jr., Frank, xxix, 769,
1013
Benton, Morris, 386
Berdnikov, Alexander, 586,
1013
Berger, Jens, xxviii, 741
Berry, Karl, 383, 421, 431, 432,
1013
Beyene, Berhanu, 608

Bezos, Javier, xxix, 38, 62, 507,
1013
Bigelow, Charles, 398, 537
Bleser, Joachim, 311, 631
Borceux, Francis, 500, 1014
Bouche, Thierry, 1014
Bovani, Michel, xxix, 402, 531
Braams, Johannes, 3, 4, 199,
208, 211, 264, 558,
846, 857, 1014,
1025, 1131
Bradley, Neil, 1014
Breitenlohner, Peter, 1014
Burykin, Alexei, 1013
Butcher, Judith, 1014

Carlisle, David, 4, 34, 71, 82,
86, 92, 142, 220,
252, 259, 261, 267,
273, 274, 282, 297,
303, 502, 657, 857,
905, 1014, 1015,
1025, 1131
Carlisle, Matthew, 1131
Carnase, Tom, 385
Carter, Matthew, 386
Chen, Pehong, 1015
Clark, James, 633
Clasen, Matthias, 368, 1016
Clausen, Jörn, 422

Cochran, Steven, xxviii, 324,
325
Cohen, Tzafrir, 591
Corff, Oliver, 608
Cosell, Bernie, 90
Covington, Michael, xxix, 162

Dachian, Serguei, 608
Dahlgren, Mats, 308
Dair, Carl, 1016
Dalalyan, Arnak, 608
Daly, Patrick, xxix, 725, 736,
828, 1023
Detig, Christine, xxviii, 1133
Donin de Rosière, Emmanuel,
788, 1016
Dorj, Dorjpalam, 608
Downes, Michael, xxviii, xxix, 4,
146, 478, 483, 1016
Drucbert, Jean-Pierre, xxix, 58,
59, 82, 250
Duchier, Denys, 4, 846, 857,
1025
Duggan, Angus, 161

Eckermann, Matthias, xxix, 190
Engbreetsen, Lars, 367
Esser, Thomas, xxix, 1002,
1003
Evans, Richard, 705

- Fairbairns, Robin, xxix, 75, 121, 225, 995, 996
 Fear, Simon, 277
 Fenn, Jürgen, 308, 1016
 Fernández, José Alberto, 776
 Fine, Michael, 199
 Finston, Laurence, 1017
 Flipo, Daniel, 6, 106, 605
 Franz, Melchior, xxviii, 94, 103, 220
 Frischauf, Adrian, 664
 Frutiger, Adrian, 386
 Fujita, Shinsaku, 631, 1017
 Fukui, Rei, xxix, 416, 1017
 Fuster, Robert, 367

 Gäßlein, Hubert, xxvi, xxix, 333, 657
 Gaulle, Bernard, 608, 1017, 1027
 Gibbons, Jeremy, 540
 Gildea, Stephen, 204
 Girou, Denis, 163, 172
 Glunz, Wolfgang, 665
 Gobry, Frédéric, 813
 Goldberg, Jeffrey, xxix, 224, 298
 Goldfarb, Charles, 1017
 Goossens, Michel, 1017, 1018, 1130
 Gordon, Peter, xxviii
 Goudy, Frederic, 94
 Grätzer, George, 477, 1018
 Graham, Ronald, 1018
 Greenwade, George, 996, 1018
 Gross, Sebastian, 335

 Hàn, Thế Thành, 633, 661, 1030
 Hafner, Thomas, 1028
 Hailperin, Max, 225
 Hakobian, Vardan, 608
 Hamilton Kelly, Brian, 633
 Hansen, Thorsten, xxix, 776, 782
 Haralambous, Yannis, 405, 557, 608, 655, 1018, 1028
 Harders, Harald, xxix, 331, 397, 703
 Harrison, Michael, 1015
 Hart, Horace, 1019
 Hefferon, Jim, 996

 Heinz, Carsten, xxviii, 178
 Hellström, Lars, xxviii, 431, 846, 1020
 Helminck, Aloysius, 389
 Henderson, Doug, 411
 Henlich, Thomas, xxix, 412
 Heslin, Peter, xxix, 88
 Hoenig, Alan, 115, 1019
 Holmes, Kris, 398, 537
 Horak, Karel, 414
 Horn, Berthold, 1019
 Horn, Blenda, xxix
 Hufflen, Jean-Michel, 789, 1019

 Ion, Patrick, 1014
 Isozaki, Hideki, 630

 Jackowski, Bogusław, 368
 Janishevsky, Andrew, 586, 1013
 Jeffrey, Alan, 4, 6, 388, 431, 533, 540, 1020
 Jensen, Frank, 395, 408, 904
 Jones, David, 478, 704, 735
 Jurafsky, Dan, 59
 Jurriens, Theo, 264

 Kastrup, David, xxix, 124, 127, 129, 267
 Kehr, Roger, 688, 1020
 Keller, Arthur, 411
 Kempson, Niel, 773, 787
 Kernighan, Brian, 1020
 Keryell, Ronan, 789
 Kettler, Howard, 386
 Khodulev, Andrey, 586
 Kielhorn, Axel, 411
 Kinch, Richard, 633
 Kirsch, Sebastian, 395
 Klöckl, Ingo, 414
 Knappen, Jörg, 366, 373, 418, 1021
 Kneser, Thomas, 308
 Knuth, Donald, 1, 2, 109, 125, 184, 338, 343, 362, 365, 380, 394, 395, 410, 417, 428, 478, 531, 556, 845, 1018, 1021–1023, 1032
 Kohm, Markus, xxix, 211, 244, 1023
 Kolodin, Mikhail, 586, 1013

 Kopka, Helmut, 1023
 Kotz, David, 804
 Krysztofiak, Claudia, xxv
 Kudlek, Manfred, 608
 Kuhlmann, Volker, 211
 Kummer, Olaf, 608
 Kwok, Conrad, 626

 Lagally, Klaus, 608, 1023
 Lamport, Leslie, xxviii, 2, 123, 160, 204, 226, 657, 703, 708, 905, 1024, 1029
 Lamy, Jean-François, 211
 Lang, Edmund, 311, 631
 Lapko, Olga, 586, 1013, 1024
 Lavagnino, John, xxix, 132, 1024
 Lavva, Boris, 591, 608
 Lawrence, Steve, 803
 Leichter, Jerry, 281
 Lemberg, Werner, 585, 608, 1025
 Lesenko, Sergey, 633
 Levy, Silvio, 590, 1025
 Liang, Franklin, 1025
 Lindgren, Ulf, 36
 Lingnau, Anselm, 300
 Louarn, Philippe, 1012
 Lubalin, Herb, 385
 Luecking, Dan, 22

 MacKay, Pierre, 1021
 Maclaine-cross, Ian, 629
 Makhovaya, Irina, 1024
 Matiaske, Wenzel, 22
 Mattes, Eberhard, 633
 McCauley, James Darrell, 298
 McDonnell, Rowland, 1020
 McLean, Ruari, 1025
 McPherson, Kent, 208
 Mehlich, Michael, 103
 Metzinger, Jochen, 608
 Miedinger, Max, 386, 539
 Miner, Robert, 1014
 Mittelbach, Frank, 3, 4, 72, 121, 148, 193, 252, 295, 339, 395, 478, 846, 857, 992, 1015, 1017, 1025–1027, 1029, 1032, 1129
 Morawski, Jens-Uwe, 1023

- Morison, Stanley, 387
- Nakashima, Hiroshi, 275
- Neergaard, Peter Møller, 414
- Neugebauer, Gerd, xxix, 808, 1027
- Neukam, Frank, 211, 244
- Nicole, Olivier, 1027
- Niepraschk, Rolf, xxix, 333, 657, 1028
- Nowacki, Janusz, 368
- Oberdiek, Heiko, 82, 662
- Orlandini, Mauro, 613
- Pakin, Scott, 411, 540, 846, 1027
- Pandey, Anshuman, 608
- Patashnik, Oren, 786, 800, 835, 837, 1018, 1027
- Phemister, Alexander, 385
- Plaice, John, 608, 655, 1028
- Plass, Michael, 109, 1021
- Podar, Sunil, 619, 1028
- Popineau, Fabrice, xxix
- Poppelier, Nico, 211, 1014
- Porrat, Rama, 591, 1028
- Puga, Diego, 389, 535
- Purtill, Mark, 4, 339
- Rahtz, Sebastian, 82, 163, 305, 339, 373, 381, 388, 390, 651, 662, 1018, 1020, 1030
- Raichle, Bernd, 996, 1028
- Raymond, Eric, 996
- Reichert, Axel, 80
- Reid, Brian, 2, 1028
- Rhead, David, xxviii, 724
- Rokicki, Tom, 632, 633, 655, 1028
- Rose, Kristoffer, 500, 612
- Rowley, Chris, xxix, 4, 125, 1015, 1025, 1026, 1028, 1029, 1132
- Rozhenko, Alexander, xxix, 129
- Rubinstein, Richard, 1029
- Ruedas, Thomas, 1003
- Ruland, Kevin, 80
- Ryan, Elizabeth, xxviii
- Ryu, Young, 399, 401, 533, 535
- Samarin, Alexander, 1017
- Schöpf, Rainer, 3, 4, 6, 161, 205, 339, 478, 996, 1026
- Schandl, Bernd, xxix, 139
- Schmidt, Walter, xxviii, xxix, 367, 381, 389, 395–398, 405, 409, 410, 421, 450, 539
- Schnier, Thorsten, 724
- Schröder, Martin, xxix, 112, 333
- Schrod, Joachim, xxviii, 7, 122, 688, 804, 870, 1029, 1133
- Schwarz, Norbert, 366
- Sendoukas, Hippocrates, 633
- Sgouros, Tom, 135
- Shell, Michael, 662
- Sivunen, Vesa, 1018
- Slimbach, Robert, 387
- Smith, Ralph, 388
- Sommerfeldt, Axel, xxviii, 306, 317, 325
- Sowa, Friedhelm, 115
- Spit, Werenfried, 842
- Spivak, Michael, 389, 478, 533
- Stiels, Rebecca, xxv
- Stiff, Paul, 110, 1029
- Straub, Pablo, 22
- Svensson, Anders, 500, 1030
- Swanson, Ellen, 478
- Swift, Matt, 88, 90
- Syropoulos, Apostolos, 590, 1012
- Tanaka, Nobuya, 1017
- Theiling, Henrik, 419, 420
- Thimbleby, Harold, 1030, 1031
- Thorup, Kresten, 904
- Tinnefeld, Karsten, 37
- Tobin, Geoffrey, 114
- Trevorrow, Andrew, 633
- Ulrich, Stefan, 135, 779
- Umeki, Hideo, xxix, 214
- Unruh, Dominique, 372
- Vabishchevich, Nikolay, 819
- Vabishchevich, Petr, 819
- Valiente Feruglio, Gabriel, 1031
- van Oostrum, Piet, xxix, 228, 232
- Van Zandt, Timothy, 160, 163, 614
- Velthuis, Frans, 608
- Vieth, Ulrik, 368, 388, 395, 531, 1016, 1020
- Vogel, Martin, 412
- Vollmer, Jürgen, 872
- Volovich, Vladimir, 366, 585, 586
- Vulis, Michael, xxix, 1031
- Waldi, Roland, 411
- Ward, Nigel, 59
- Wetmore, Alan, xxix
- Wicks, Mark, 661
- Widmann, Thomas, 788
- Williams, Graham, 308, 998, 1031
- Williams, Peter, 724
- Williamson, Hugh, 1031
- Wilson, Peter, xxix, 51, 124, 208, 244, 245, 703, 1032
- Winton, Neil, 199
- Wolczko, Mario, 211, 613
- Woliński, Marcin, 846, 857, 1025
- Wong, Wai, xxix
- Wonneberger, Reinhard, 1032
- Wooding, Mark, 846, 857, 1025
- Wujastyk, Dominik, 125, 1024
- Zapf, Hermann, 387, 395, 407, 414, 450, 531, 535, 1021, 1032
- Ziegler, Justin, 368, 394, 1032
- Zierke, Reinhard, 996
- Ziv, Alon, 591

Biographien

Stand 2004

Frank Mittelbach

Frank Mittelbach studierte Mathematik und Informatik an der Johannes-Gutenberg-Universität in Mainz. 1989 begann er seine berufliche Laufbahn bei dem IT-Unternehmen EDS (Electronic Data Systems), wo er sich in einer neu gebildeten Gruppe mit $\text{\TeX}$ und anderen Werkzeugen für die Verarbeitung von Dokumenten befasste. In seiner derzeitigen Position ist er verantwortlich für Konzeption und Implementierung von Fernüberwachung und -administration verteilter Systeme und Netzwerke.



Vor TLC2...

Schon in seiner Studienzeit befasste er sich mit der automatischen Formatierung komplexer Dokumente im Allgemeinen und mit $\text{\LaTeX}$ im Besonderen. Diesen Themen widmet er noch immer so viel Zeit und Interesse, dass man dieses Aufgabengebiet guten Gewissens als seinen „zweiten Beruf“ und nicht nur als ein Hobby bezeichnen kann. Allein

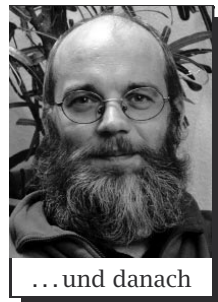
oder in Zusammenarbeit mit anderen entwickelte er viele verschiedene $\text{\LaTeX}$ -Erweiterungspakete, wie etwa $\mathcal{A}\mathcal{M}\mathcal{S}\text{-}\text{\LaTeX}$, doc, multicol und NFSS (das New Font Selection Scheme).

1989 hielt er im Rahmen der TUG-Konferenz an der Stanford University einen Vortrag über die Probleme mit $\text{\LaTeX}$ 2.09, der ihm die Verantwortung für die Pflege und weitere Entwicklung von $\text{\LaTeX}$ einbrachte. Die in diesem Zusammenhang erbrachten Leistungen sind allgemein als das $\text{\LaTeX}$ 3-Projekt bekannt. In seiner Funktion als technischer Leiter des Projektes betreute Frank das 1994 herausgegebene erste größere Release von $\text{\LaTeX}$ 2 ϵ sowie die bisher 15 weiteren Wartungsversionen dieser Software.

Durch seine vielen veröffentlichten Fachartikel über $\text{\LaTeX}$ und über allgemeine Forschungsergebnisse auf dem Gebiet der automatischen Formatierung lernte er Peter Gordon von Addison-Wesley kennen. Peter und Frank gründeten die neue Buchreihe *Tools and Techniques for Computer Typesetting* (TTCT)

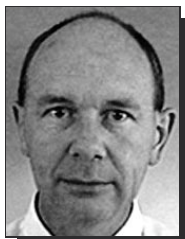
als deren Herausgeber Frank fungiert. Das englische Buch *The L^AT_EX Companion* (1994¹) eröffnete die Reihe, deren Titel L^AT_EX mittlerweile in seiner ganzen Vielfalt erfassen. Weitere geplante Bände werden diesen Schwerpunkt auf andere Werkzeuge und Konzepte für den Textsatz und die Informationsverarbeitung ausweiten.

1990 legte Frank in dem Artikel *E-T_EX: Guidelines for further T_EX extensions* [128] die wichtigsten Mängel von T_EX dar und betonte die Notwendigkeit, das System weiterzuentwickeln und die vielen offenen Fragen des automatischen Schriftsatzes weiter zu erforschen. Damit wurde das Thema der Änderung oder Erweiterung in der T_EX-Gemeinschaft erstmals offen diskutiert und, nach anfänglichem Widerstand, ergaben sich daraus mehrere wichtige Projekte, wie etwa eT_EX, Omega und NTS. Franks Interesse gilt nun der Integration dieser Früchte der für T_EX entwickelten Erweiterungen, z.B. der Projekte Omega und eT_EX, zu einem stabilen, gut gepflegten und großflächig verfügbaren Nachfolger von T_EX, auf dem ein zukünftiges L^AT_EX3 aufsetzen kann. Frank und seine Frau Christel leben gemeinsam mit ihren drei Söhnen Arno (19 Jahre) und den Zwillingen Burkhard und Holger (6 Jahre) in Mainz.



...und danach

Michel Goossens



Anfang 1979 trat Michel Goossens mit einem frisch erworbenen Dokortitel in Hochenergiephysik eine Stelle am CERN an, dem Europäischen Labor für Teilchenphysik in Genf. Dort arbeitete er einige Jahre als Physiker in der Forschung, bevor er in den Software-Support der IT-Abteilung wechselte.

Im Laufe der Jahre arbeitete er mit verschiedenen Satzsystemen: natürlich mit L^AT_EX, in letzter Zeit jedoch auch mit HTML/SGML/XML. Als Mitarbeiter eines großen internationalen Wissenschaftslabors verwendet ein Großteil der beim CERN arbeitenden Physiker und Ingenieure L^AT_EX für ihre Veröffentlichungen und Dokumentationen. Daher war Michel seit Ende der 1980er Jahre damit beschäftigt, Werkzeuge für T_EX und insbesondere für L^AT_EX zu entwickeln und zu unterstützen.

Ein Meilenstein in seinem L^AT_EX-Leben war ein Treffen mit Frank und Chris Ende 1992, als sie bei CERN einen Vortrag über L^AT_EX3 hielten. Im Anschluss an das Seminar zeigte Michel ihnen den „Local T_EX Guide“, den Alexander Samarin und er geschrieben hatten. Er schlug vor, das Material zu erweitern und als Buch herauszugeben. Das war die Geburt der ersten Auflage des Buches *The L^AT_EX Companion*, das Anfang 1994 veröffentlicht wurde. Aufgrund seiner Erfahrung mit Graphiken und Internetveröffentlichungen war Michel auch Mitautor der Werke *The L^AT_EX Graphics Companion* (1997) und *The L^AT_EX Web Companion* (1999), die beide in der TTCT-Reihe erschienen.

Michel hat bereits verschiedene Funktionen in der T_EX-Welt ausgeübt: 1995–2000 war er Präsident von GUTenberg, der französischen T_EX-

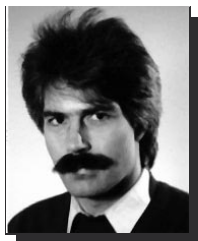
¹1995 in deutscher Sprache als *Der L^AT_EX-Begleiter* erschienen.

Benutzergruppe, und 1995–1997 zudem Präsident der TUG, der (globalen) T_EX Users Group.

Die letzten drei Jahre fungierte er bei CERN als Ansprechpartner für das EU-geförderte Projekt TIPS (Tools for Innovative Publishing in Science). Im Rahmen dieses Projektes untersuchte er, wie sich XML-Werkzeuge am besten in ein System zur effizienten Bearbeitung von elektronischen Informationen, insbesondere wissenschaftlichen Dokumenten, integrieren lassen. Er richtete sein besonderes Augenmerk auf das einander ergänzende Zusammenspiel von L^AT_EX und MathML für Formeln, SVG für Graphiken, PDF für eine hohe typographische Ausgabequalität und XHTML oder DocBook für die strukturelle Integration in Webumgebungen.

Er lebt in der Nähe von Genf und verbringt seine Zeit gerne mit Lesen, einem guten Film, Museumsbesuchen und Spaziergängen am See oder in der schönen Landschaft.

Johannes Braams



Johannes Braams studierte Elektronikingenieurwesen an der Technischen Universität in Enschede in den Niederlanden. Seine Magisterarbeit befasste sich mit dem Thema der Videokodierung nach dem Vorbild des menschlichen Sehsystems. Er begegnete L^AT_EX zum ersten Mal 1984 in den *dr. Neher Laboratories* der Niederländischen Telekom (PTT). Er war Gründungsmitglied der Niederländisch sprechenden T_EX-Benutzergruppe NTG (1988) und beteiligte sich an der Entwicklung einer Lösung zum Setzen von Dokumenten in niederländischer Sprache.

Nach der EuroT_EX-Konferenz in Karlsruhe 1989 begann Johannes mit der Entwicklung des babel-Systems und ist seit der EuroT_EX-Konferenz 1990 in Cork Mitglied des L^AT_EX3-Projektes. Außer für babel ist er derzeit für eine Reihe weiterer L^AT_EX-Erweiterungspakete zuständig, wie etwa die Dokumentenklassenfamilie ntgclass, das supertabular- und das changebar-Paket.

Johannes arbeitet weiterhin bei der Niederländischen Telekom, die heute unter dem Namen KPN bekannt ist, und zwar hauptsächlich als IT-Projektmanager. Er lebt mit seiner Frau Marion und den beiden Söhnen Tycho (11 Jahre) und Stephan (9 Jahre) in Zoetermeer.

David Carlisle

David Carlisle studierte Mathematik an der University of Manchester und arbeitete dann als Forscher in den Fachbereichen Mathematik und Computerwissenschaften in Cambridge und Manchester. Dort setzte er seit 1987 L^AT_EX ein und wurde 1992, kurz vor Beginn der Entwicklungsarbeiten an L^AT_EX 2_ε, Mitglied des L^AT_EX3-Teams.



Er arbeitet seit mittlerweile mehr als sechs Jahren für die NAG Ltd. in Oxford, in der Hauptsache an Projekten zur Entwicklung von XML-Sprachen für die Darstellung mathematischer Ausdrücke und Dokumente. Er ist einer der

Herausgeber der OpenMath-Spezifikation und wurde, als von der für MathML zuständigen W3C Math Working Group eingeladenen Experte, zum Mitherausgeber der MathML 2-Empfehlung. Zurzeit ist er an einer vorgeschlagenen Aktualisierung des ISO/IEC Technical Report 9573 für „ISO-Zeichenentitäten“ beteiligt. Dieser ermöglicht, nur mithilfe von ASCII-Zeichen eine Vielzahl verschiedenster Zeichen in XML- und SGML-Dokumente einzugeben: beispielsweise erzeugt die Syntax $\&\gamma$; das Symbol γ .

David interessiert sich auch für XSLT: Er ist in der xsl-list-Diskussionsgruppe für diese Sprache sehr aktiv und war an vielen Büchern zum Thema XSLT als Lektor oder technischer Editor beteiligt. Er lebt zusammen mit seiner Frau Joanna und Sohn Matthew (4 Monate) in Oxfordshire.

Chris Rowley

Wenn er nicht gerade seiner Reiselust frönt, lebt Chris in London zusammen mit seinem Weinkeller, seiner Keramiksammlung und seinen Erinnerungen. Zu letzteren gehören auch verschwommene Bilder der 1960er, als er nicht von der Mathematik lassen konnte und auch hin und wieder in die Welt der elektronischen Datenverarbeitung eintauchte, sowohl in die Theorie des Programmierens (damals ein noch ziemlich urwüchsiges Gebiet) als auch in die Zahlenverarbeitung (hübsche Luftschlangen aus der Lochmaschine).



Bei seiner Lieblingsbeschäftigung

Erst Anfang der 1980er entdeckte er auf einem frisch besetzten Schreibtisch ein fernsehähnliches Objekt, das an einen Computer angeschlossen war, und mit dem er kreative und nützliche Dinge tun konnte, wie zum Beispiel eine einzelne Seite mit wunderschön gesetzten Formeln zu füllen. Das geschah nicht mithilfe von $\text{\TeX}$ – daher brauchte er zwei Tage für die eine Seite. Aber es zeigte ihm, was möglich war und brachte

ihn dazu nachzudenken, wie er es einfacher erreichen konnte. Er ist sehr dankbar, dass er dann sehr bald über $\text{\TeX}$ stolperte und kurz darauf auch über $\text{\LaTeX}$. Letzteres war eine besonders glückliche Fügung, da zu seinen Kollegen auch sechs Mathematik-Schreibkräfte gehörten, die etwas benötigten, mit dem sie arbeiten konnten. Ein paar Jahre später hörte er von jemandem namens Mittlebach-and-Schöpf (sic) in Mainz und der Rest ist ... *Fortsetzung folgt.*

Heute, fünfzehn Jahre später, ist Chris Rowley ein Seniormitglied der Faculty of Mathematics and Computing an der Open University in Großbritannien. Seit der Geburt des $\text{\LaTeX}$ 3-Projektteams gehörte er ihm in leitender Funktion und als aktives Mitglied an. Damals war er noch so naiv zu glauben, alles würde in etwa zwei Jahren erledigt sein. Er war schon in zu vielen Gremien und Komitees, von denen das Editorial Board von *Tools and Techniques for Computer Typesetting* eine der angenehmsten Erfahrungen war. Zudem hat er verschiedenste Ämter in der $\text{\TeX}$ -Welt beehrt, unter anderem als Vorsitzender der UKTUG und Vizepräsident der TUG.

Als größter internationaler Anbieter industrieller Massenbildung, die durch ein Studium von zu Hause oder am Arbeitsplatz einen Universitäts-

abschluss ermöglicht, entwickelte sich die Open University zu einer großen Multimedia-Verlagsgesellschaft. Für ihre Veröffentlichungen im Bereich Mathematik verfügt sie über ein, trotz wirtschaftlichen Wettbewerbs nur mit unzureichenden Ressourcen ausgestattetes, auf $\text{\LaTeX}$ basierendes Produktionssystem. Als Mathematiker, der bereits recht viel Ahnung von der Herstellung mathematischer Texte hatte, war Chris gerade der Richtige, um beim Aufbau dieses Systems Mitte der 80er Jahre in politischer, administrativer und technischer Hinsicht eine wichtige Rolle zu spielen.

Heute widmet er sich intensiv der Forschung im Bereich der Automatisierung jeglicher Aspekte der Dokumentenverarbeitung, insbesondere der mehrsprachigen Typographie für vielseitige $\odot$ Dokumente. Im Laufe der Jahrzehnte hat er zum anderen jedoch auch an $\text{\LaTeX}$ -Systemen in Produktionsumgebungen seinen Teil an praktischer Arbeit geleistet und einigen Normierungsgremien, Firmen und Organisationen als Berater für die Digitalisierung mathematischer Texte gedient.

All diese Aktivitäten haben Chris zu der Überzeugung geführt, dass es für $\text{\TeX}$ langfristig nur zwei wichtige Verwendungen gibt: zum einen als eine Art Umgangssprache innerhalb der weniger formalen elektronischen Kommunikation unter Mathematikern und zum anderen als Schatzkiste voller wunderbarer Algorithmen, besonders zum Setzen mathematischer Formeln. Davon abgesehen glaubt er nicht, dass eine Erweiterung des monolithischen Designs und der verschlungenen Modelle des $\text{\TeX}$ -Softwaresystems ein leistungsfähiges und flexibles Satzprogramm für das 21. Jahrhundert hervorbringen wird ... aber es macht einfach mehr Spaß als Kreuzworträtsel.

Christine Detig & Joachim Schrod

1982, im Rahmen ihres Informatikstudiums, entdeckte Christine Detig $\text{\TeX}$ auf einer Bandspule und wurde so zu einem der Gründungsmitglieder von DANTE, der deutschen $\text{\TeX}$ -Benutzergruppe. Sie sammelte ihre ersten Erfahrungen mit Software am $\text{\TeX}$ -Arbeitsplatz, was zur Gründung eines kleinen Unternehmens führte, dass $\text{\TeX}$ -Distributionen vertrieb. Im Rahmen ihrer Arbeit als Forschungsassistentin an der TU Darmstadt verbreitete sie $\text{\TeX}$ nisches Wissen und schrieb ein Buch für $\text{\TeX}$ -Einsteiger: *Der $\text{\LaTeX}$ Wegweiser*. Mittlerweile hat sie auf zahlreichen internationalen Konferenzen innerhalb der bunten Menge von $\text{\TeX}$ ies viele Freundschaften geschlossen. Dort kann man sie zu einem netten Schwätzchen über die Zukunft von $\text{\TeX}$ treffen.



Im Rahmen ihrer Arbeit als Forschungsassistentin an der TU Darmstadt verbreitete sie $\text{\TeX}$ nisches Wissen und schrieb ein Buch für $\text{\TeX}$ -Einsteiger: *Der $\text{\LaTeX}$ Wegweiser*. Mittlerweile hat sie auf zahlreichen internationalen Konferenzen innerhalb der bunten Menge von $\text{\TeX}$ ies viele Freundschaften geschlossen. Dort kann man sie zu einem netten Schwätzchen über die Zukunft von $\text{\TeX}$ treffen.

Auch Joachim Schrod arbeitet seit 1982 mit $\text{\TeX}$ und er ist ein weiteres Gründungsmitglied von DANTE. Er schrieb und unterstützte die internationale Version von $\text{\LaTeX}$ bis $\text{\LaTeX} 2_{\epsilon}$ auf der Bildfläche erschien. Joachim war an Unmengen von $\text{\TeX}$ -Aktivitäten beteiligt, von denen die meisten schon so lange zurück liegen, dass man sich kaum noch daran erinnert. Zu seinen dauerhafteren Leistungen zählen das CTAN und die $\text{\TeX}$ Directory Structure. Heute ist er Geschäftsführer eines Beratungsunternehmens, in dem er versucht zwischen Geschäftsleuten und Technikern zu vermitteln.

Christine & Joachim leben in Rödermark in Deutschland.

Entstehungsgeschichte

des englischen und deutschen Buches

Das englische Original

Dieses Buch beschreibt nicht nur das Dokumentenverarbeitungssystem $\text{\LaTeX}$, es wurde auch damit gesetzt. Dabei leisteten die im Buch behandelten Pakete tatkräftige Unterstützung, ergänzt durch ein beachtliches Maß an zusätzlicher $\text{\LaTeX}$ -Programmierung.

Als Grundschrift diente Lucida Bright (Bigelow/Holmes) mit einem Schriftgrad von 8.8pt/12pt. Die zweite wichtige Schrift ist die dicktengleiche European Modern Typewriter (Y&Y) in 10.06pt/12pt. Diese Kombination wurde gewählt, um auf jeder Seite eine vernünftige Menge an Material unterzubringen und das Erscheinungsbild der „Schreibmaschinenschrift“ optisch auszugleichen: Man soll sie gut erkennen können, ohne dass der Kontrast störend wirkt.

Grundschrift

Der Text der Beispiele wurde weitgehend in Adobe Times Roman gesetzt, die serifenlosen Teile in Helvetica. Die Formeln in den Beispielen erscheinen in den mittlerweile klassischen Mathematikfonts der Computer Modern, so dass die Symbole den meisten Mathematikern vertraut vorkommen werden. Beispiele, anhand derer die Verwendung anderer Schriften erläutert wird, bilden hier natürlich die Ausnahme.

Beispielschrift

Das Buch wurde mit dem $\text{\LaTeX}$ -Basisrelease vom 1.12.2003 gesetzt. Die Engine darunter bildete das Programm $\text{\pdfTeX}$, das jedoch so eingestellt war, dass es keine PDF-Ausgabe erzeugt: Es wurde vielmehr wegen seiner Fähigkeit, einen „optischen Randausgleich“ vorzunehmen, ausgewählt. Dieses typographische Bonbon ($\text{\pdfcprot}$ -Paket) kam dem Textkorpus zugute. (Wie man dazu im Einzelnen vorgehen muss, ist in [161, 162] nachzulesen.) Zum Vergleich empfiehlt sich ein Blick auf die Seiten 987–989, da diese ohne optischen Randausgleich (und in kleinerer Schrift) gesetzt wurden.

*Optischer
Randausgleich*

Für die Produktion dieses Buchs waren angepasste Klassen- und Paketdateien erforderlich. Außerdem wurde ein komplexer Herstellungsprozess benötigt, der auf einer Reihe von durch ein „Makefile“ gesteuerten „Shell-Skripts“ basierte. Eine der wichtigsten Aufgaben dieser Skripte war es, dafür zu sorgen,

Der Produktionszyklus

dass jedes einzelne gesetzte Beispiel auch wirklich anhand des dazugehörigen Quellcodes erzeugt wurde.

Dieser Herstellungsprozess verlief folgendermaßen:

Beispiele erzeugen

- Bei der ersten Verarbeitung eines Kapitels erzeugte $\text{\LaTeX}$ für jedes Beispiel eine Datei mit einem Quelldokument. Das sind die „Beispieldateien“ auf der CD-ROM.
- Der Herstellungsprozess ließ dann jede einzelne dieser Beispieldateien von $\text{\LaTeX}$ verarbeiten (auch unter Verwendung von $\text{\BibTeX}$ oder anderen gegebenenfalls erforderlichen Erweiterungen), und zwar so oft wie nötig, um die endgültige gesetzte Form zu erhalten.

Schließlich erzeugte er mithilfe von `dvips` entweder eine oder zwei EPS-Dateien mit dem „gesetzten Beispiel“.

- Beim nächsten $\text{\LaTeX}$ -Lauf des entsprechenden Kapitels wurde dann jede dieser EPS-Dateien automatisch an der richtigen Stelle im Buch, neben (oder nahe) dem zugehörigen Quellcode, platziert. Damit war der Prozess aber noch nicht abgeschlossen: Die horizontale Position einiger Elemente, insbesondere der Beispiele, hängt nämlich davon ab, ob sie auf einer linken oder rechten Seite erscheinen. Hier wurde das in Beispiel A-3-9 auf Seite 909 angewandte Verfahren benutzt. Damit waren mindestens noch einmal ein oder zwei zusätzliche Läufe erforderlich, bevor $\text{\LaTeX}$ alle Querverweise richtig aufgelöst und die Beispiele endlich passend in den Rändern platziert hatte.

Handarbeit

Das war so ziemlich das Äußerste an möglicher Automatisierung. Die vielen großen Beispiele, die sich weder umbrechen ließen noch gleiten durften, machten das Auffinden günstiger Seitenumbrüche zu einer echten Herausforderung. Aus diesem und anderen Gründen war das abschließende Layout des Buches mit einem ziemlichen Aufwand verbunden und erforderte sogar kleinere Textänderungen (auf etwa 10% der Seiten), um unschöne Zeilen- und Seitenumbrüche zu vermeiden (wie etwa Absätze mit nur einem Wort in der letzten Zeile oder hässliche Trennungen am Seitenende). Doppelseiten durften gegebenenfalls um eine Zeile länger oder kürzer sein und in vielen Fällen wurden das Layout und der Inhalt von Beispielen manuell angepasst, um geeignete Seitenumbrüche zu erhalten.

Etwas Statistik

Hier einige statistische Näherungswerte zum Seitenlayout: 45 längere Doppelseiten, 25 kürzere Doppelseiten, 230 erzwungene Seitenumbrüche, 400 Anpassungen der vertikalen Ausrichtung, 100 sonstige manuelle Änderungen (abgesehen von Umformulierungen).

Der Index

Für den Index der „Befehle und Begriffe“ wurde zunächst eine Version des Buches mit Zeilennummern gedruckt und an einen Indexexperten übergeben. Dieser erzeugte „begriffliche Indexeinträge“, die dann in die Quelldateien des Buches eingefügt wurden. Das war ein echter Beweis für die Qualität des `lineno`-Paketes, welches, trotz der Komplexität des Buches und der Verwendung von speziellen Makros für den Seitenumbruch, auf Anhieb funktionierte. Der Index selbst wurde dann mit `MakeIndex` erzeugt, da `xindy` zu diesem Zeitpunkt noch nicht zur Verfügung stand. Aufgrund der komplexen Indexgestaltung (farbige Seitenzahlen usw.) musste die Indexdatei jedoch mit

Skripts vor- und nachbearbeitet werden, bevor sie schließlich in ihrer endgültigen Form vorlag. Danach wurde sie mithilfe einer verbesserten Version des multicol-Paketes gesetzt, das die Fortsetzungszeilen hinzufügte – auch diese Funktion lässt sich vielleicht eines Tages als eigenes Paket bereitstellen.

Die deutsche Übersetzung

Für die deutsche Übersetzung wurde das Seitenlayout an den Stil des Verlags Pearson Studium angepasst. Da der Satzspiegel, wie in Europa üblich, schmaler als der des Originals ist, stellte dies einige zusätzliche Anforderungen an den Seitenumbruch. Fast alle Beispiele mussten umgearbeitet werden und viele wuchsen aufgrund horizontalen Platzmangels in die Vertikale, was bei langen Kapiteln eine akzeptable Materialverteilung ohne größere Lücken, Witwen und Waisen nicht gerade erleichterte.

Die Grundschrift wurde mit 8.5pt/11.72pt leicht verkleinert. Die neuen Kolumnentitel wurden wie schon zuvor mit fancyhdr erstellt, während für die Erstellung des Satzspiegels geometry zum ersten Mal zum Einsatz kam. Die netten Inhaltsübersichten am Anfang eines jeden Kapitels sind „courtesy of“ titletoc, das auch benutzt wurde, um die Verzeichnisse im Vorspann zu setzen. Um die Arbeit in der Druckerei zu erleichtern, kam crop zum Einsatz. Passende L^AT_EX-Stile für die Beispiele in Kapitel 12 und die Bibliographie wurden mit custom-bib erstellt. Viele der anderen, in diesem Buch beschriebenen Pakete aus allen Kapiteln, sind schon in der englischen Originalausgabe zum Einsatz gekommen, insgesamt sind es mehr als dreißig.

*Angewandte
Ratschläge aus diesem
Buch*

Die im Abschnitt zum englischen Original beschriebene Maschinerie kam auch in dieser Ausgabe zum Einsatz, d.h. alle Beispiele wurden wie beschrieben automatisch generiert. Als einziger Unterschied wurden normalerweise zusätzlich babel und inputenc geladen, um der deutschen Sprache und ihren Umlauten Rechnung zu tragen.

Die Beispiele

Da der Index schon in die Buchquellen eingearbeitet war, musste er diesmal „nur“ übersetzt werden – nur in Anführungszeichen, weil sich in der deutschen Sprache ganz neue Wortgruppierungen ergeben und so doch viele Einträge hinzugefügt oder verändert werden mussten. So hatte etwa „leading“ im Englischen die Unterpunkte „blanks“ (führende Leerzeichen) und „vertical spacing“ (Zeilenvorschub, Durchschuss), um nur eines der vielen Beispiele aufzuzeigen. Der aus Quellen generierte Rohindex wurde weiterhin mit MakeIndex verarbeitet, da alle Mechanismen schon passend in den Makefiles vorhanden waren.

Der Index

Interessant ist, dass der manuelle Aufwand für den Satz des Buches in etwa dem des Originals entspricht: diesmal gab es 42 verlängerte und 18 verkürzte Doppelseiten und die Anzahl der erzwungenen Seitenumbrüche lag mit 228 sogar leicht niedriger. Im Gegensatz zur englischen Ausgabe wurde diesmal verstärkt mit \looseness experimentiert (insgesamt an 35 Stellen), typischerweise um ein einzelnes, oft sogar getrenntes Wort in der letzten Zeile eines Absatzes zu entfernen. Wie schon beim englischen Original wurde bei schlechtem Zeilenfall das Privileg des Autors ausgenutzt, dem es (im Gegensatz zum armen Setzer) erlaubt ist, seinen Text beliebig umzuformulieren, um typographische Monster zu vermeiden – die Anzahl solcher „Verbesserungen“ liegt auch diesmal deutlich im dreistelligen Bereich.

Mehr Statistik

CD-ROM Garantiebeschränkungen

T_EX-Live ist eine freie Software. Vertriebs- und allgemeine Information findet man unter:

<http://www.tug.org/tex-live/>

Das Software Medium wird auf einer „AS IS“ Basis ohne Garantie vertrieben. Weder die Autoren, noch die Übersetzer, noch die Softwareentwickler, noch die veröffentlichende Firma, noch Pearson Education Deutschland GmbH übernimmt irgendeine Vertretung oder Garantie (weder ausgedrückt noch impliziert) für die Softwareprogramme, ihre Qualität, Genauigkeit oder Einsetzbarkeit für bestimmte Zwecke. Deswegen übernehmen weder die Autoren, noch die Übersetzer, noch die Softwareentwickler, noch Pearson Education Deutschland GmbH irgendeine Verantwortung anderen Personen oder juristischen Personen gegenüber bezüglich Haftpflicht, Verlust oder Schaden, der angeblich oder tatsächlich direkt oder indirekt durch Programme verursacht wird, die in diesem Medium enthalten sind. Dies beinhaltet, ist aber nicht begrenzt, auf die Unterbrechung von Diensten (Programmabsturz), den Verlust von Daten, den Verlust von Unterrichtszeit, den Verlust von Beratungszeit oder im Voraus bezahlten Honoraren oder von aus dem Gebrauch dieser Programme folgenden Schäden.



Copyright

Daten, Texte, Design und Grafiken dieses eBooks, sowie die eventuell angebotenen eBook-Zusatzdaten sind urheberrechtlich geschützt. Dieses eBook stellen wir lediglich als persönliche Einzelplatz-Lizenz zur Verfügung!

Jede andere Verwendung dieses eBooks oder zugehöriger Materialien und Informationen, einschliesslich

- der Reproduktion,
- der Weitergabe,
- des Weitervertriebs,
- der Platzierung im Internet, in Intranets, in Extranets,
- der Veränderung,
- des Weiterverkaufs
- und der Veröffentlichung

bedarf der schriftlichen Genehmigung des Verlags.

Insbesondere ist die Entfernung oder Änderung des vom Verlag vergebenen Passwortschutzes ausdrücklich untersagt!

Bei Fragen zu diesem Thema wenden Sie sich bitte an: info@pearson.de

Zusatzdaten

Möglicherweise liegt dem gedruckten Buch eine CD-ROM mit Zusatzdaten bei. Die Zurverfügungstellung dieser Daten auf unseren Websites ist eine freiwillige Leistung des Verlags. Der Rechtsweg ist ausgeschlossen.

Hinweis

Dieses und viele weitere eBooks können Sie rund um die Uhr und legal auf unserer Website



herunterladen