

LaTeX

Das Praxisbuch

- > LaTeX einsetzen und beherrschen
- > Gestalten Sie Magister- und Diplomarbeiten, Dissertationen, Fachaufsätze und Bücher
- > So erstellen Sie perfekte mathematische Ausdrücke und Formeln

Alexander Schunk

LaTeX

Alexander Schunk

LaTeX

Das Praxisbuch

Mit 125 Abbildungen

FRANZIS

Bibliografische Information der Deutschen Bibliothek

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

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

© 2009 Franzis Verlag GmbH, 85586 Poing

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Fachlektor: Benjamin Fleckenstein

Satz: DTP-Satz A. Kugge, München

art & design: www.ideehoch2.de

Druck: Bercker, 47623 Kevelaer

Printed in Germany

ISBN 978-3-7723-6730-4

Vorwort

Sie kennen das Problem: Sie möchten gerne ein Dokument ansprechend gestalten und dieses soll natürlich verschiedenen Lesern und Betrachtern auf diversen Plattformen zur Verfügung stehen. Um dieser Anforderung gerecht zu werden, stehen Ihnen diverse Textverarbeitungssysteme für normale Dokumente sowie HTML für Online-Dokumente zur Verfügung. Alle diese Systeme haben jedoch meistens ihre Grenzen, was die Flexibilität und Gestaltungsfreiheit betrifft.

Hier kommt LaTeX ins Spiel. LaTeX ist ein Textsatzsystem, ja eigentlich eher eine Formatierungssprache, mit der Sie attraktive Dokumente im PDF-, PostScript- oder DVI-Format gestalten können. LaTeX hat viele Ähnlichkeiten mit der Auszeichnungssprache HTML, ist jedoch weitaus mächtiger. Außerdem ist es unabhängig von Betriebssystemen, da es praktisch für jede PC-Plattform verfügbar ist. In diesem Buch lernen Sie nicht nur die Standardbefehle kennen, sondern auch viele Ergänzungspakete, die es Ihnen erlauben, Dokumente für ganz unterschiedliche Einsatzzwecke zu gestalten. Das Buch richtet sich sowohl an Einsteiger als auch an Fortgeschrittene. Die ersten vier Kapitel vermitteln Ihnen einen Einblick in die Grundlagen von LaTeX. Die darauffolgenden Kapitel widmen sich spezielleren Themen wie Tabellen, Bildern, mathematischen Dokumenten usw.

Genauso wie andere Programmiersprachen und Softwarelösungen entwickelt sich LaTeX immer weiter. Allerdings verläuft die Entwicklung nicht immer koordiniert. Viele Einzelpersonen und Gruppen arbeiten an verschiedenen Baustellen und bauen LaTeX auf diese Weise weiter aus.

Diese neue Auflage reflektiert die aktuellen Entwicklungen und versucht, den Wert der bisherigen Ausgaben zu erhalten, die von Elke und Michael Niedermair zusammengestellt wurden. Der Autor möchte sich beim Verlag für die gute Zusammenarbeit bedanken und hofft, dass die aktuelle Auflage den bisherigen Ausgaben qualitativ in nichts nachsteht.

Inhaltsverzeichnis

1	Einführung in LaTeX	17
1.1	Was ist LaTeX?	17
1.2	LaTeX installieren	18
1.2.1	Windows	18
1.2.2	Linux	19
1.2.3	Mac OS X	19
1.3	Ergänzende LaTeX-Tools	19
1.4	LaTeX-Texte erstellen	21
1.4.1	Exkurs: Satzspiegelkonstruktion	21
1.5	LaTeX-Texte übersetzen und betrachten	21
1.5.1	Quellcode eingeben	22
1.5.2	Quellcode kompilieren	23
1.5.3	LaTeX-Dokumente erzeugen und betrachten	25
1.6	Neuigkeiten in LaTeX	26
1.6.1	Neue Distributionen	26
1.6.2	Neue Paketversionen	27
2	LaTeX-Grundlagen	29
2.1	Erste Schritte mit LaTeX	29
2.1.1	Parallelen zu HTML	29
2.1.2	LaTeX-Dateiformate	29
2.1.3	Makros und Pakete	31
2.1.4	Layout des Dokumentes	31
2.1.5	LaTeX-Modi	37
2.1.6	LaTeX-Umgebungen	37
2.1.7	LaTeX und Bilder	37
2.2	Grundlagen von Zeichensätzen	38
2.2.1	Codierung	38
2.3	LaTeX-Fehlermeldungen	39
2.4	LaTeX-Befehle und -Gruppen	40

3	Überblick über LaTeX-Befehle	41
3.1	Papiergröße festlegen	41
3.2	Ränder vorgeben	41
3.3	Rahmen, Striche und Boxen	41
3.4	Gleitende Objekte	42
3.5	Tabellen und Bilder	42
3.5.1	Tabellen	42
3.5.2	Bilder	44
3.6	Aufzählungen, Listen und Texteinbindungen	45
3.6.1	Aufzählungen und Listen	45
3.6.2	Einbinden von vorformatiertem Text	46
3.7	Mathematische Formeln	47
3.8	Bilder und Zeichnungen	49
3.8.1	Grundlagen der picture-Umgebung	49
3.8.2	Koordinatensystem der picture-Umgebung	49
3.8.3	Objekte positionieren	50
3.9	Farbe	50
3.10	Einfache Texte	51
4	Einfache Texte mit LaTeX	57
4.1	Papierformate	57
4.1.1	Das Paket typearea	58
4.2	Ränder definieren	59
4.2.1	Das Kommando \setlength	60
4.2.2	Das Paket geometry	61
4.3	Spracheigenschaften	62
4.4	Klassenparameter	64
4.5	Schriftmerkmale	64
4.5.1	Schriftfamilien	64
4.5.2	Schriftfamilien festlegen	65
4.5.3	Schriftformatierung	65
4.5.4	Schriftstärke setzen (Fettung)	66
4.5.5	Schriftgröße	67
4.5.6	Zeichensatzbefehle	68
4.5.7	Das Paket soul	69
4.5.8	Text ausrichten	70
4.5.9	Fußnoten	73

4.5.10	Gliederungsebenen.....	74
4.5.11	Umbrüche	76
4.5.12	Sonderfälle der Silbentrennung	77
4.5.13	Abstände zwischen Zeilen und Absätzen	78
4.5.14	Querverweise	82
4.5.15	Erweiterte Möglichkeiten für Fremdsprachen.....	84
4.5.16	Symbole.....	85
4.5.17	Rahmen, Striche und Boxen	85
4.5.18	Farbe	90
4.5.19	Mehrspaltiger Druck.....	92
4.5.20	Standardseiten und Zeilennummern	94
4.5.21	Einheiten besonders berücksichtigen	95
5	Tabellen.....	97
5.1	Grundlagen von Tabellen	97
5.2	Exkurs: einfache LaTeX-Tabellen	97
5.2.1	Rahmen für Tabellen	100
5.2.2	Tabellen mit fester Gesamtbreite	118
5.2.3	Tabellen über mehrere Seiten	119
5.2.4	Gleitende Tabellen und Abbildungen beeinflussen	120
6	Bilder und gleitende Objekte	125
6.1	Bildformate	125
6.1.1	Die Pakete graphics und graphicx	126
6.1.2	Farben	127
6.2	Bilder aus Dateien einbinden	127
6.2.1	Dateiendungen für Bilder	130
6.2.2	Suchpfad für Dateien	131
6.2.3	Bilder zentrieren	133
6.2.4	Hintergrundbilder einbinden	135
6.2.5	Objekte in gleitenden Umgebungen drehen	139
6.3	Bilder beschriften und nummerieren	140
6.3.1	Änderung der Beschriftung.....	142
6.3.2	Neue gleitende Umgebungen	142
6.3.3	Umgebungen drehen	145

6.3.4	Vorhandene Umgebungen anpassen.....	147
6.3.5	Nichtgleitende Umgebungen.....	147
6.4	Textumflossene Objekte.....	148
6.4.1	Bilder und Tabellen im Absatz.....	148
6.4.2	Initiale erstellen	150
7	Eigene Grafiken erstellen	153
7.1	Bilder zeichnen mit der picture-Umgebung.....	153
7.1.1	Linien zeichnen.....	153
7.1.2	Linien mit Pfeilen	154
7.1.3	Kreise	155
7.1.4	Rechtecke	156
7.1.5	Ovale und gerundete Ecken.....	158
7.1.6	Bézierkurven	160
7.1.7	Mehrere Figuren zeichnen	163
7.1.8	Text mit Bildern.....	164
7.1.9	Text vertikal zeichnen	165
7.1.10	Grafiken verschachteln	166
7.1.11	Grafiken für die Wiederverwendung speichern	167
7.1.12	Erweiterung für die picture-Umgebung	168
7.2	Das Paket PSTricks.....	168
7.2.1	Das PSTricks-Paket einbinden.....	169
7.2.2	PSTricks-Standardwerte festlegen	169
7.2.3	Die Umgebung	170
7.2.4	Farben	170
7.2.5	Linienstile	172
7.2.6	Füllstile	173
7.2.7	Koordinatensysteme	174
7.2.8	Achsen.....	174
7.2.9	Linien, Polygone und Rechtecke.....	175
7.2.10	Kreise und Ellipsen	177
7.2.11	Kurven	178
7.2.12	Punkte	180
7.2.13	Grid.....	181
7.2.14	Plots	181
7.2.15	Pfeile	184
7.2.16	Eigene Objekte und Stile	185

7.2.17	Eigene Grafiken.....	185
7.2.18	Hackertricks.....	188
8	Aufzählungen	189
8.1	Einfache Aufzählungen	189
8.1.1	Listen verschachteln	194
9	Eigene Listen	199
9.1	Die Umgebung list.....	199
9.1.1	Neue Listenumgebungen	202
9.1.2	Listen als Grundlage für andere Umgebungen	204
9.2	Ergänzungspakete für Listen	204
9.2.1	Das Paket expdlist	204
9.2.2	Das Paket paralist.....	205
9.2.3	Das Paket mdwlist	206
9.2.4	Das Paket desclist.....	208
9.2.5	Weitere Ergänzungspakete.....	208
10	Unformatierte Texte und Listings einbinden.....	211
10.1	Text einbinden mit der verbatim-Umgebung.....	211
10.1.1	Text mit LaTeX-Befehlen einbinden	213
10.1.2	Verkürzte Syntax für \verb.....	214
10.1.3	Das Paket verbatim	214
10.1.4	Originaltext als Befehl einbinden	215
10.1.5	Text mit dem Paket moreverb einbinden	216
10.1.6	Text einbinden mit dem Paket sverb.....	216
10.1.7	Text mit fancyvrb einbinden	217
10.1.8	Listings einbinden	219
11	Mathematische Formeln	221
11.1	Grundlagen	221
11.1.1	Einfache Mathematikschreibweise.....	221
11.1.2	Mathematikumgebungen	222
11.1.3	Weitere Umgebungen.....	223
11.1.4	Schriftgröße und Schriftarten	224

11.1.5	Grundlegende Formelelemente	227
11.1.6	Mathematische Symbole.....	229
11.1.7	Mathematische Operatoren.....	231
11.1.8	Mathematische Akzente.....	233
11.1.9	Exponenten und Indizes.....	235
11.1.10	Wurzel	235
11.1.11	Brüche	236
11.1.12	Summen und Integrale.....	237
11.1.13	Komplexe mathematische Strukturen.....	239
11.1.14	Pfeile über und unter mathematischen Ausdrücken	240
11.1.15	Gestapelte Symbole.....	241
11.1.16	Matrizen	241
11.1.17	Binomialkoeffizienten	242
11.1.18	Abstände in Formeln variieren.....	243
11.1.19	Mehrzeilige Formeln und Gleichungssysteme.....	244
11.1.20	Feintuning für Nummerierung und mehrzeilige Formeln.....	244
11.1.21	Gestaltung von Formeln	246
11.2	Das Paket AMS LaTeX.....	247
11.2.1	Die Pakete von AMS LaTeX	247
11.2.2	Die Parameter von AMS LaTeX.....	248
11.2.3	Darstellung einzelner Formeln mit AMS LaTeX	249
11.3	Sätze und Definitionen.....	251
12	Fehler	255
12.1	Fehler finden.....	255
12.2	Fehlermeldungen und Warnungen.....	259
12.3	Fehler mit Hilfsprogrammen auffinden	262
13	Längenmaße, Zähler und neue Befehle	265
13.1	Längen und Maßeinheiten	265
13.1.1	Zähler	267
13.1.2	Rechenmöglichkeiten.....	268
13.1.3	Vergleichen von Zahlen und Längen.....	269
13.2.1	Neue Befehle erzeugen	271
13.2.2	Neue Umgebungen erzeugen	276

14	Bücher und wissenschaftliche Arbeiten	279
14.1	Teildokumente	279
14.1.1	Einbinden von Teildokumenten	279
14.2	Buchgliederungen	284
14.2.1	Die Umgebung abstract	285
14.3	Verzeichnisse	286
14.3.1	Inhaltsverzeichnis	286
14.3.2	Tabellen- und Abbildungsverzeichnisse	296
14.3.3	Literaturverzeichnis	296
14.3.4	Literaturverzeichnisse mit BibTeX	298
14.3.5	Index	301
14.3.6	Glossar	308
14.4	Kopf- und Fußzeilen	309
14.4.1	Fußnoten	312
14.5	Juristische Texte	313
14.5.1	Das Paket jurabase	313
14.5.2	Die Klasse jurabook	313
14.5.3	Die Klasse jurautrl	314
14.5.4	Die Klasse juraovw	315
14.6	Arbeiten mit make-Dateien	316
14.7	Arbeiten im Team	317
15	Grundlagen von Zeichensätzen	319
15.1	Grundaufbau eines Zeichensatzes	319
15.2	Schriftinitialisierung	319
15.3	Fonttypen	320
15.3.1	Pixelfonts vs. Type1-Fonts	320
15.3.2	Fonttyp festlegen	321
15.3.3	Ergänzungspakete für Schriften	322
15.3.4	Zeichen eines Fonts	323
15.3.5	Symbolfonts	324
15.3.6	Der Font Zapf Dingbats	324
15.3.7	Weitere Symbolfonts	325
15.3.8	Fontdokumentationen	328

16	Praktische Anwendungen	329
16.1	Ein einfacher Artikel mit LaTeX	329
16.2	Ein einfacher Brief mit LaTeX	331
16.2.1	Hübsche Kleinigkeiten	332
17	PDFLaTeX und PostScript	337
17.1	Von PostScript nach PDF konvertieren	337
17.2	Temporäre Auslagerung von PostScript-Code	338
17.3	PostScript-Code in eine externe Datei auslagern	341
18	Fremdformate konvertieren	343
18.1	Von OpenOffice nach LaTeX	343
18.2	Installation von Writer2LaTeX unter Windows und Linux	344
18.3	HTML nach LaTeX	350
18.4	Von DocBook nach LaTeX	351
18.5	Notlösungen	351
19	LaTeX und PDF-Dokumente	353
19.1	Links in PDF-Dokumente einbinden	353
19.2	Das Paket ifpdf für pdfTeX	355
19.3	In PDF-Dokumenten suchen	356
19.4	Optischer Randausgleich und Zeichendehnung	356
19.5	Einbinden von PDF-Seiten	356
19.6	In PDF zeichnen	358
19.7	Einbetten von Multimedia-Objekten	360
19.8	Wasserzeichen	360
19.9	PDF-Präsentationen	361
10	Anhang	363
20.1	Installation von LaTeX	363
20.2	Der Aufbau von LaTeX	364
20.2.1	Pakete installieren	366
20.2.2	Paketdokumentationen anzeigen lassen	367
20.3	Paket-Updates	368
20.4	Dateiformate von LaTeX	368

20.4.1	Von LaTeX erzeugte Dateien	368
20.4.2	Ein- und Ausgabe mit LaTeX	369
20.4.3	Weitere Informationen	370
Stichwortverzeichnis		371

1 Einführung in LaTeX

1.1 Was ist LaTeX?

LaTeX ist ein Satzsystem, das von dem Mathematiker Leslie Lamport als Erweiterung zu der von Donald Knuth entwickelten Software TeX geschaffen wurde. Knuth hatte TeX in den Siebziger- und Achtzigerjahren entwickelt, weil er mit den Ergebnissen der damaligen Satztechnik der Verlage unzufrieden war. Zu dieser Zeit gab es kaum digitale Satzsysteme und Knuth sah sich mit der Herausforderung konfrontiert, etwas Neues zu erfinden. Er plante ursprünglich, innerhalb von drei Monaten mit der ersten Version von TeX fertig zu sein. Doch es sollte knappe zehn Jahre dauern, bis er 1986 das komplette System vorstellen konnte.

LaTeX stellt im Wesentlichen eine Vereinfachung von TeX dar. Das Besondere an LaTeX ist, dass es rechnerunabhängig ist und somit für eine Vielzahl von Plattformen eingesetzt werden kann. Die meisten LaTeX-Distributionen sind unter einer Open-Source-artigen Lizenz verfügbar.

Mit LaTeX lassen sich verschiedene Arten von Dokumenten wie zum Beispiel wissenschaftlich-technische Artikel, Bücher, Briefe usw. in einem ästhetisch ansprechenden Layout setzen. Dazu werden Kommandos benutzt, die von den verschiedenen LaTeX-Distributionen implementiert werden. In diesem Buch wird überwiegend die Distribution MikTeX zusammen mit dem Programm Texmaker verwendet.

Hinweis: Die Begriffe TeX und LaTeX werden in der Umgangssprache häufig synonym verwendet. Bei TeX handelt es sich allerdings um das von Donald Knuth entwickelte Kernsystem, während LaTeX im Wesentlichen eine Sammlung von Makrobefehlen darstellt, die die Verwendung von TeX deutlich vereinfacht. Sogenannte Distributionen, die für verschiedene Rechnersysteme erhältlich sind, liefern in der Regel den vollen Umfang von LaTeX oder einen Ausschnitt davon sowie noch weitere Tools wie zum Beispiel Viewer oder Ähnliches.

1.2 LaTeX installieren

Die Installation von LaTeX verläuft je nach Distribution und Betriebssystem unterschiedlich.

1.2.1 Windows

Unter Windows erfolgt die Installation über die Setup-Datei der jeweiligen Distribution. In diesem Buch wird überwiegend die Distribution MikTeX mit dem Programm Texmaker benutzt. Die Distribution MikTeX können Sie von der Internetseite <http://miktex.org/> beziehen. Gehen Sie dazu auf die genannte Website und besuchen Sie den Link »Download«.

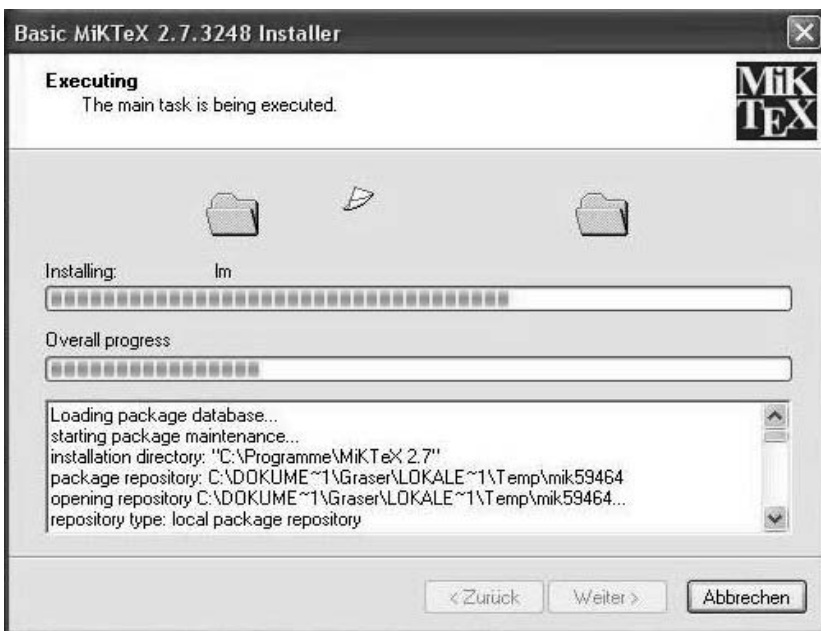


Abbildung 1.1: Die Installation von MikTeX unter Windows ist schnell erledigt.

MikTeX gibt es in zwei Varianten: Zum einen existiert eine Basic-Variante, die etwa 82 Megabyte umfasst und eine Basisinstallation mit den gängigsten Klassen und Paketen enthält. Für die Komplettinstallation laden Sie sich ein vergleichsweise kleines Installationspaket herunter. Wenn Sie es starten, lädt dieses Paket den kompletten Funktionsumfang herunter. Je nach Internetanbindung kann dieser Download ein paar Stunden dauern. Zu MikTeX gibt es außerdem eine Handvoll ergänzender Tools, die Sie ebenfalls herunterladen können.

Weitere Informationen über die Installation unter Windows finden Sie im Anhang.

1.2.2 Linux

Unter SuSE Linux können Sie die für LaTeX relevanten Pakete sehr einfach über den Paketmanager YaST installieren. Gehen Sie dazu auf das Startsymbol und rufen Sie das Programm YaST auf.

In YaST wählen Sie dann das jeweilige Paket aus, z. B. `tetex`. Nach Auswahl der erforderlichen Pakete installiert YaST dann die jeweilige LaTeX-Distribution. Da das in einigen älteren Versionen von SuSE Linux enthaltene Paket `tetex` nicht mehr weitergepflegt wird, finden Sie in TeX Live eine aktuelle Alternative, die sich im Internet unter der Adresse <http://www.tug.org/texlive/acquire.html> befindet.

Weitere Informationen über die Installation unter Linux erhalten Sie im Anhang.

1.2.3 Mac OS X

Die Installation unter Mac OS X erfolgt analog zu der unter Windows oder Linux. Unter Mac OS X gibt es z. B. die Distribution MacTeX. Gehen Sie dazu auf die Website von MacTeX (www.tug.org/mactex) und laden Sie sich die Distribution herunter.

Ein Doppelklick auf das Setup-Programm installiert das Paket unter Mac OS X.

Weitere Informationen zur Installation unter Mac OS X erhalten Sie im Anhang.

1.3 Ergänzende LaTeX-Tools

Für die Arbeit mit LaTeX gibt es für die entsprechenden Betriebssysteme eine Reihe weiterer Tools. Dazu gehören z. B. Editoren oder Kommandozeilenprogramme für Windows und Linux. Die Tabelle listet einige zusätzliche Werkzeuge auf, mit denen bestimmte Aufgaben im LaTeX-Bereich erledigt werden können. Diese Tools sind in der Regel in den gängigen LaTeX-Distributionen bereits enthalten; wenn Sie aber eine aktuellere Variante der Werkzeuge suchen, kann sich ein Besuch auf den entsprechenden Seiten durchaus lohnen.

Programm	Beschreibung	URL
BibTeX	Für die Erstellung von Literaturverzeichnissen	www.bibtex.org
pdfLaTeX	Für die Erstellung von PDF-Dokumenten	http://www.tug.org/applications/pdftex/

<i>Programm</i>	<i>Beschreibung</i>	<i>URL</i>
MakeIndex	Für die Erstellung von Stichwortverzeichnissen	http://tug.ctan.org/tex_archive/indexing/makeindex/

Tabelle 1.1: Hilfsprogramme für LaTeX

Das Programm BibTeX ist ein Programm zum Erstellen von Indexverzeichnissen. Es ist eine Alternative zu dem LaTeX-Standardprogramm MakeIndex. Makeindex ist ein weiteres Programm zum Erstellen von Indexseiten. Weitere Informationen zu dem Programm BibTeX und Makeindex finden Sie in Kapitel 14.

Mit dem Programm pdfLaTeX können Sie PDF-Dokumente erstellen. Weitere Informationen zu dem Programm pdfLaTeX finden Sie in Kapitel 17.

Die oben aufgeführten Programme sind alle kommandozeilenorientiert. Für eine komfortablere Arbeit mit LaTeX gibt es aber auch einige kostenlose bzw. kommerzielle Editoren:

<i>Programm</i>	<i>Betriebssystem</i>	<i>Version</i>	<i>URL</i>
Texmaker	Windows, Linux, Mac OS X	1.9.2	http://www.xmlmath.net/texmaker/
TeXnicCenter	Windows	RC 1	http://www.texniccenter.org
Kile	Linux, KDE	2.03	http://kile.sourceforge.net/
TeXShop	Mac OS X	2.20	http://www.uoregon.edu/~koch/texshop/

Tabelle 1.2: Editoren für LaTeX

Das Programm Texmaker ist ein Freewareprogramm, das viele LaTeX-Kommandos unterstützt und einfach zu bedienen ist, da es eine grafische Oberfläche bietet. Voraussetzung für den Betrieb von Texmaker ist, dass Sie bereits eine LaTeX-Distribution auf Ihrem Rechner installiert haben. Das Programm können Sie von der oben genannten Downloadseite beziehen. Texmaker gibt es für verschiedene Betriebssysteme.

TeXnicCenter ist in vielerlei Hinsicht mit Texmaker vergleichbar, da das Programm ebenfalls eine grafische Oberfläche zur Verfügung stellt und als freie Software erhältlich ist. Es ist allerdings nur unter Windows lauffähig.

Das Softwarepaket Kile ist ein unter Linux häufig genutzter Editor, mit dem Sie auf einfache Weise LaTeX- und TeX-Dokumente erstellen können. Der Editor Kile ist in den meisten Linux-Distributionen enthalten.

TeXShop ist ein Freeware-Editor für das Betriebssystem Mac OS X. TeXShop bietet, ähnlich wie Texmaker, viele verschiedene Funktionen und unterstützt viele LaTeX-Standardbefehle.

Alle oben genannten Programme sind frei erhältlich und stehen zumeist unter einer OpenSource-artigen Lizenz.

Tipp: Alle Beispiele in diesem Buch wurden mit Texmaker und MikTeX unter Windows getestet. Es ist möglich, dass Sie leicht abweichende Ergebnisse erzielen, wenn Sie die Beispiele unter anderen Systemumgebungen und mit anderen Tool-Kombinationen ausprobieren.

1.4 LaTeX-Texte erstellen

Wenn Sie Textverarbeitungssysteme wie z. B. Microsoft Word oder OpenOffice gewöhnt sind, gestaltet sich die Erzeugung von LaTeX-Dokumenten vordergründig etwas weniger komfortabel. Zum einen benötigen Sie für die Erstellung eines LaTeX-Dokumentes spezielle Kommandos, die bestimmte Grundeigenschaften eines Dokumentes, wie z. B. Größe, Schrift, Ränder und Ähnliches festlegen. Zum anderen müssen Sie sich bereits vor der Erstellung eines LaTeX-Dokumentes grundlegende Gedanken zum Layout machen, um z. B. die Breite des Satzspiegels usw. festzulegen.

Ein weiterer Unterschied von LaTeX zu normalen Textverarbeitungsprogrammen ist, dass LaTeX von vornherein mehrere Arten von Dokumenten unterscheidet, z. B. Artikel, Buch, Brief usw., und diese auch unterschiedlich behandelt.

1.4.1 Exkurs: Satzspiegelkonstruktion

Ein wichtiges Thema bei allen Arten von Satzsystemen, egal ob herkömmlicher Buchdruck oder digitale Textsysteme, ist die Konstruktion des Satzspiegels. Unter Satzspiegel versteht man im Allgemeinen und etwas vereinfacht ausgedrückt den Textbereich eines Dokumentes. Außerhalb des Satzspiegels, also dem eigentlichen Text, gibt es noch sogenannte Stege. Unter einem Steg wird der freie, nicht bedruckte Bereich zwischen Text und Rändern des Dokumentes verstanden. So gibt es Kopfsteg, Fußsteg, Bundsteg und Außensteg.

Im Verlauf der Geschichte der Buchdruck- und Satzsysteme sind verschiedene Möglichkeiten entstanden, einen optisch schönen und ansprechenden Satzspiegel zu konstruieren. Die Harmonie des Gesamteindrucks spielt dabei eine wichtige Rolle.

1.5 LaTeX-Texte übersetzen und betrachten

Um ein LaTeX-Dokument zu erstellen, müssen Sie drei Schritte befolgen. Der Vorgang ähnelt dem Schreiben eines Computerprogramms in einer Hochsprache wie C oder

Delphi, bei der ein Compiler dafür sorgt, dass der ursprüngliche Quelltext in eine maschinenverständliche Form gebracht wird. Die drei Schritte im Einzelnen:

1. Quellcode schreiben
2. Quelldatei kompilieren
3. Ausgabedatei betrachten

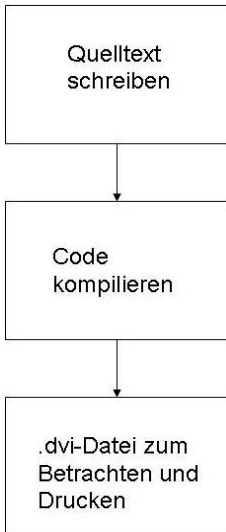


Abbildung 1.2: Vom Quellcode zur TeX-Datei

Diese drei Schritte werden im Folgenden genauer beschrieben.

1.5.1 Quellcode eingeben

Zuerst geben Sie den Quellcode des LaTeX-Dokumentes – also den eigentlichen Text mit den Formatierungsbefehlen – mit einem Editor wie z. B. Texmaker ein. Dann speichern Sie dieses Dokument als Datei mit der Endung `.tex` ab. Alle LaTeX-Dateien werden standardmäßig mit der Endung `.tex` gespeichert.

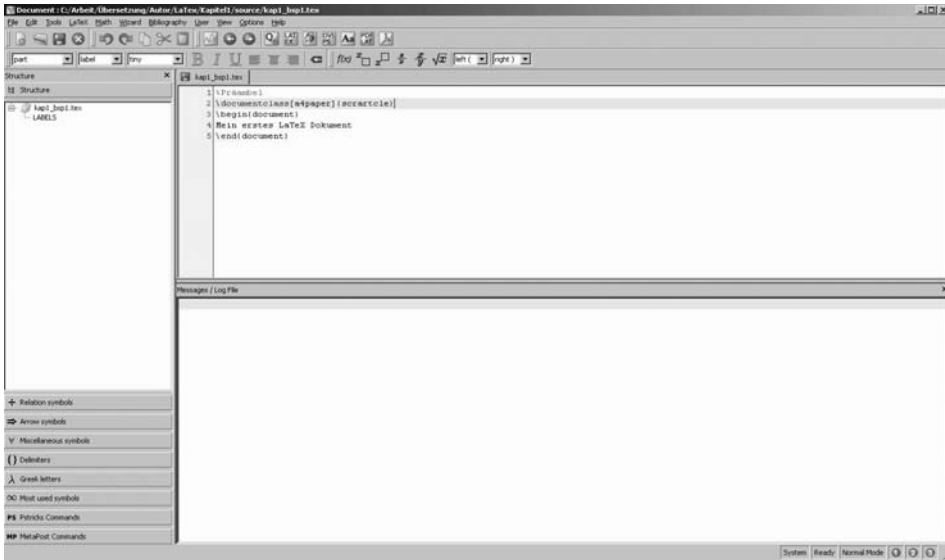


Abbildung 1.3: Quellcodeeingabe mit Texmaker

1.5.2 Quellcode kompilieren

Nachdem Sie den Quellcode eingegeben haben, können Sie ihn mit dem LaTeX-Compiler kompilieren lassen. Der Aufruf des LaTeX-Compilers erfolgt auf der Konsole über:

```
latex document.tex
```

Dabei erzeugt LaTeX standardmäßig ein Dokument mit dem Dateiformat DVI (das Kürzel steht für Device Independent File Format). Die MikTeX-Distribution enthält einen Viewer namens Yap (das Kürzel steht für Yet another Previewer), mit dem Sie das erzeugte Dokument betrachten und auch ausdrucken können.

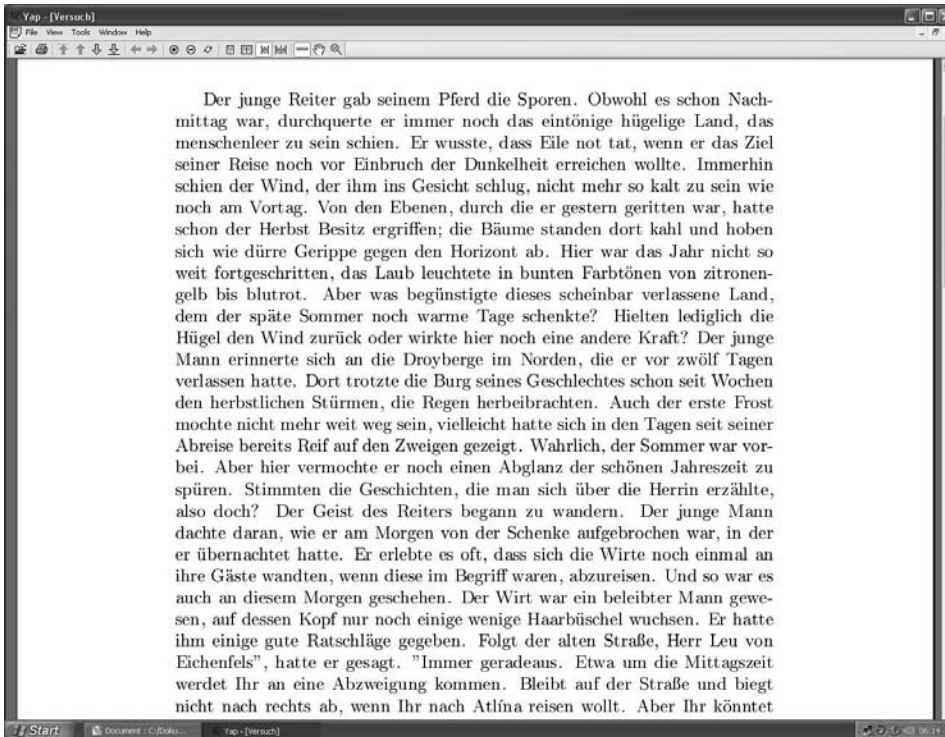


Abbildung 1.4: Kompiliertes Textdokument im DVI-Betrachter Yap

Aus einem DVI-Dokument können Sie über verschiedene Tools andere Dateiformate erstellen, wie z. B. PostScript oder PDF. Um aus einem DVI-Dokument ein PostScript-Dokument zu erzeugen, können Sie den DVI-to-PS-Converter `dvips` benutzen:

```
dvips document.dvi
```

PDF-Dateien werden z. B. über das Programm `pdfLaTeX` erstellt:

```
pdflatex document.tex
```

Tipp: Wenn Sie mit einer integrierten LaTeX-Umgebung wie z. B. Texmaker arbeiten, können Sie PDF-Dokumente auch automatisch erstellen lassen. Dazu müssen Sie Texmaker entsprechend konfigurieren. Derartige integrierte Umgebungen mit grafischer Benutzeroberfläche werden in Anlehnung an Programmierwerkzeuge auch als IDE (Integrated Development Environment) bezeichnet.

1.5.3 LaTeX-Dokumente erzeugen und betrachten

Im dritten Schritt wird dann entweder ein PDF- oder PostScript-Dokument erzeugt. Diese beiden Dokumentformate sind die wichtigsten Ausgabeformate in LaTeX. Je nachdem, welches Hilfsprogramm Sie verwenden, erzeugt die IDE bzw. der Compiler ein entsprechendes Dokument.

PDF-Dokumente

PDF-Dokumente werden mit dem Hilfsprogramm `pdfLaTeX` erzeugt. Das Programm `pdfLaTeX` ist ein Konsolenprogramm, das auch in viele LaTeX-IDEs wie Texmaker integriert ist.

Ein PDF-Dokument lässt sich auch über den Umweg eines PostScript-Dokumentes erzeugen. Dazu benötigt man den Konverter `ps2pdf`. Dieser Konverter wird im Kapitel 17 im Abschnitt über PostScript-Dokumente näher beschrieben.

Die folgende Abbildung veranschaulicht den Prozess:

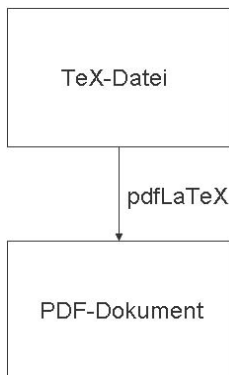


Abbildung 1.5: Erzeugung eines PDF-Dokumentes mit `pdfLaTeX`

PostScript-Dokumente

Neben PDF-Dokumenten können Sie auch PostScript-Dokumente erstellen. Die Dokumente werden als `DVI`-Datei gespeichert. Ein PostScript-Dokument wird – wie gesehen – mit dem Programm `dvips` erzeugt. Das Programm ist ein Konverter, den Sie über die Kommandozeile aufrufen können.

Die folgende Abbildung veranschaulicht den Erstellungsprozess:

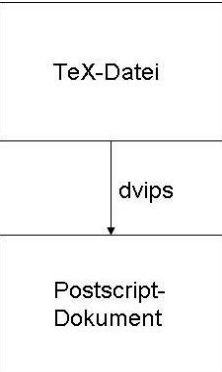


Abbildung 1.6: PostScript-Dokumente gestalten

Nach erfolgreichem Compilerlauf können Sie sich dann die von LaTeX erzeugte DVI- oder PDF-Datei mit einem entsprechenden Viewer ansehen. Ein solcher Viewer ist das bereits erwähnte Programm Yap. PDF-Dokumente können Sie mit dem bekannten Adobe Acrobat Reader betrachten.

1.6 Neuigkeiten in LaTeX

Seit der letzten Überarbeitung dieses Buches hat sich in der LaTeX-Welt einiges getan. Es sind neue Distributionen entstanden und weiterentwickelte Versionen von existierenden Paketen wurden veröffentlicht. Einige Distributionen werden aber nicht mehr weiter gepflegt, dazu gehört z. B. `tetex`.

1.6.1 Neue Distributionen

Die Tabelle zeigt neu entstandene LaTeX-Distributionen mit der aktuellen Versionsnummer, Lizenz, Autor, Version und URL:

Distribution	Lizenz	Autor	Version	URL
xetex	MIT-Lizenz	Jonathan Kew	0.999.6	http://Scripts.sil.org/xetex
luatex	GNU General Public License	Hans Hagen, Hartmut Henkel, Taco Hoekwater	Derzeit 0.40	www.luatex.org

Tabelle 1.3: Neue LaTeX-Distributionen

XeTeX ist eine Alternative zu pdfLaTeX und wird von Jonathan Kew entwickelt. Bei LuaTeX handelt es sich um eine neue TeX-Engine, die auf der Skriptsprache Lua basiert. LuaTeX soll die Funktionalität von pdfTeX, eTeX, Omega (einer Unicode-Erweiterung für TeX) sowie Aleph (einer Lösung für Satz in unterschiedlichen Richtungen) in sich aufnehmen. Zum aktuellen Zeitpunkt ist LuaTeX allerdings noch in einem recht frühen Stadium. Die Version 1.00 ist für das Jahr 2012 geplant.

1.6.2 Neue Paketversionen

Neben neuen Distributionen sind auch erweiterte Versionen von existierenden Paketen entstanden. Von der in Deutschland und auch in Europa weitverbreiteten Distribution KOMA-Script gibt es z. B. mittlerweile die Version 3.04. Zum Download verfügbar ist KOMA-Script unter der Internetadresse <http://developer.berlios.de/projects/koma-script3/>.

2 LaTeX-Grundlagen

LaTeX eilt der Ruf voraus, ein Expertensystem zu sein, mit dem sich überwiegend Wissenschaftler der technischen Disziplinen beschäftigen. Für einen absoluten Neuling mutet LaTeX auch relativ kompliziert an, doch wenn man sich mit den Grundlagen vertraut gemacht hat und die ersten Hürden genommen sind, ist man mit LaTeX nach wenigen Stunden fast so vertraut wie mit einer anderen Textverarbeitung.

2.1 Erste Schritte mit LaTeX

2.1.1 Parallelen zu HTML

LaTeX funktioniert vom Grundprinzip her so ähnlich wie die Markup-Sprache HTML, d. h., es benutzt Befehle zur Formatierung von Text, allerdings bietet LaTeX mehr Gestaltungsmöglichkeiten als HTML. Genauso wie bei der Entwicklung von HTML-Seiten benötigen Sie zur Erstellung von LaTeX-Dokumenten einen Editor für die Quellcodes und einen Viewer zum Betrachten der Seite. Der einzige Unterschied zwischen LaTeX und HTML ist, dass LaTeX-Quelldateien kompiliert werden müssen, bevor Sie das Ergebnis betrachten können. Für HTML-Dateien ist dieser Zwischenschritt nicht nötig.

2.1.2 LaTeX-Dateiformate

Bei der Arbeit mit LaTeX kommen Sie mit einer Reihe von typischen Dateiformaten in Berührung. Die Tabelle listet die wichtigsten Formate auf, die Sie kennen sollten:

<i>Dateityp</i>	<i>Beschreibung</i>
.tex	Dateiendung für LaTeX-Quelltexte
.sty	Dateiendung für LaTeX-Pakete
.dvi	Ausgabeformat für kompilierte LaTeX-Dateien (Device Independent File Format)
.pdf	Portable Document Format
.eps	Bildformat, das unter anderem für .dvi-Dokumente verwendet wird (Encapsulated PostScript)

Dateityp	Beschreibung
.png	Portable Network Graphics für PDF-Dokumente
.jpg	Bildformat für PDF-Dokumente
.inc	Dateityp für Include-Dateien

Tabelle 2.1: Dateitypen von LaTeX

Die Endung `.tex` ist die Standardendung für LaTeX-Quelltexte. LaTeX-Pakete werden standardmäßig mit der Endung `.sty` versehen. Für die Ausgabe von LaTeX-Dokumenten stehen `.dvi`, `.ps` und `.pdf` zur Verfügung.

Für die Arbeit mit Bildern unterstützt LaTeX, je nachdem, welches Ausgabeformat Sie für Ihr Dokument bevorzugen (d. h., ob Sie PDF- oder DVI-Dokumente erzeugen), die Formate `.eps`, `.png` und `.jpg`. Dabei werden die Formate nicht direkt von LaTeX selbst, sondern von den jeweiligen Formaten PDF und DVI unterstützt.

Include-Dateien

Die Endung `.inc` steht für Include-Dateien. Sie können in LaTeX Dateien definieren, in die Sie Kommandos eingeben, die Sie immer wieder brauchen. Es lässt sich z. B. die sogenannte Präambel Ihres Dokumentes als Include-Datei definieren und dann in die jeweiligen Dokumente einbinden, ohne dass Sie jedes Mal die Präambel neu schreiben müssen. Dieses Verfahren lehnt sich an die Hochsprachenprogrammierung an: Insbesondere bei der Programmiersprache C werden über das Kommando `include` die Rahmenbedingungen für die Kompilierung und die Ausführung eines Programms eingebunden.

Eine Include-Datei wird wie eine normale LaTeX-Datei erzeugt. Man schreibt den Quellcode in die Datei und speichert diese unter dem Namen `include`. Unten sehen Sie eine einfache Include-Datei für die deutsche Sprache mit einem T1-Schriftsatz:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
```

Wichtig ist hier insbesondere der Parameter `[ngerman]` in eckigen Klammern. Er besagt, dass für das nachfolgende Textdokument die neue deutsche Rechtschreibung gelten soll. `ngerman` ist Teil des LaTeX-Pakets `babel`, das in geschweiften Klammern notiert ist. Diese Include-Datei können Sie z. B. unter der Endung `deutsch.inc` abspeichern. Weitere Informationen zu Zeichensätzen und Spracheigenschaften finden Sie weiter unten.

Der Parameter `[latin1]` in der letzten Zeile des Beispiellistings kann nur dann verwendet werden, wenn die Quelltextdatei auch mit dem Zeichensatz Latin-1, also ISO 8859-1,

arbeitet. Das ist unter anderem bei Windows-Betriebssystemen der Fall. Aktuelle Linux-Distributionen verwenden den Unicode-Zeichensatz UTF-8. Dort müsste dann in der eckigen Klammer `utf8` statt `latin1` stehen.

2.1.3 Makros und Pakete

Alle LaTeX-Befehle (engl. *command*) sind als Makros (engl. *macro*) programmiert. Ein Paket (engl. *package*) enthält verschiedene LaTeX-Befehle. Ein LaTeX-Befehl wird durch einen `\` Backslash eingeleitet, gefolgt von dem Namen des Befehls sowie optionaler Parameter und Optionen, die in geschweiften bzw. eckigen Klammern gesetzt werden:

```
\Befehl{<Parameter>}[<Option>]
```

Ein Paket enthält eine Menge verschiedener Befehle für eine bestimmte Aufgabe. Bevor Sie die Befehle eines Pakets in einem Dokument nutzen können, müssen Sie das Paket erst über den Befehl `\usepackage{}` in das Dokument einbinden, wie Sie es schon in unserem ersten Beispiel einer Include-Datei gesehen haben. Der Befehl `\usepackage{}` muss dabei in der Präambel des Dokumentes stehen. Ohne hier dem nächsten Abschnitt vorgreifen zu wollen, sei dazu nur gesagt, dass in der Präambel wichtige Definitionen stehen, die für das ganze Dokument gelten.

Neue Befehle erzeugen

Neben vorhandenen LaTeX-Befehlen können Sie auch Befehle neu definieren. Dazu gibt es das Kommando `\newcommand` in verschiedenen Variationen. Das Kommando `\newcommand` wird wie ein normaler Befehl aufgerufen, gefolgt von dem Befehl, den Sie neu definieren möchten:

```
\newcommand{\Befehl}[Parameter]
```

Der neu erzeugte Befehl kann seinerseits Parameter und Optionen haben. Neben einfachen Befehlen können Sie auch Umgebungen neu definieren. Dafür gibt es z. B. den Befehl `\newenvironment`:

```
\newenvironment{<Umgebung>}[<Option>]
```

Auch neue Umgebungen – mehr dazu in Abschnitt 2.1.6 – können Parameter und Optionen haben. Weitere Informationen über die Neudefinitionen von Befehlen und Umgebungen finden Sie durchgängig im gesamten Buch. In den entsprechenden Passagen wird dann auch ausführlich auf die dazugehörigen Parameter und Optionen eingegangen.

2.1.4 Layout des Dokumentes

Jedes LaTeX-Dokument folgt einem bestimmten, vorab festgelegten Aufbau. Dieser Aufbau kann sich je nach Art des Dokumentes, das Sie schreiben wollen, unterscheiden.

Es gibt jedoch gewisse Bestandteile, die man in nahezu jedem LaTeX-Dokument findet. Hierzu gehören:

1. Präambel
2. Hauptteil
3. Kapitel
4. Abschnitte
5. Unterabschnitte

Präambel

Wenn Sie ein LaTeX-Dokument schreiben wollen, müssen Sie zuvor in der sogenannten Präambel (engl. *preamble*) das Layout des Dokumentes definieren. In der Präambel können Sie z. B. die verwendete Dokumentenklasse (engl. *document classes*), Schrift (engl. *font*), Seitengröße (engl. *page size*), Zeichensatz (engl. *character encoding*), den Namen des Autors usw. festlegen. Die in der Präambel festgelegten Eigenschaften des Dokumentes gelten global für das ganze Dokument.

Dokumentenklassen

Die Tabelle zeigt einige Standarddokumentenklassen von LaTeX:

<i>Dokumentenklasse</i>	<i>Beschreibung</i>
book	Für Bücher
article	Für Artikel
letter	Für Briefe
report	Für Reports

Tabelle 2.2: Parameter für Dokumentenklassen

Es gibt noch andere Dokumentenklassen, wie z. B. `slides` für Präsentationen usw. Da sich die Standarddokumentenklassen in Bezug auf Typografie und Layout an amerikanische Konventionen halten, wurden bald Alternativen wie KOMA-Script entwickelt. Diese Erweiterung wurde geschaffen, um sich den europäischen Besonderheiten in Sachen Typografie und Layout anzupassen. Das Paket KOMA-Script, das neben MikTeX für die Beispiele dieses Buches herangezogen wurde, stellt daher seinerseits noch weitere Dokumentenklassen zur Verfügung.

Die Tabelle 2.3 gibt eine Übersicht über die Dokumentenklassen und ihre entsprechenden Namen in KOMA-Script:

Dokumentenklasse	Beschreibung
Scrbook	Für Bücher
Scrartcle	Für Artikel
scr1ttr2	Für Briefe
Scrrreport	Für Reports

Tabelle 2.3: KOMA-Script-Parameter für Dokumentenklassen

Tipp: KOMA-Script ist ein von Markus Khom geschriebenes Paket, das in Deutschland und Europa weit verbreitet ist. Die aktuelle Version von KOMA-Script können Sie von der Website <http://developer.berlios.de/projects/koma-script3/> beziehen.

Hier ein einfaches Beispiel für eine Präambel mit der Dokumentenklasse `scrartcle`:

```
% Beispiel für eine Dokumentenklasse
\documentclass{scrartcle}
%Titel des Dokumentes
\title{Erstes Dokument}
%Autor des Dokumentes
\author{Alexander Schunk}
%Hauptteil des Dokumentes
\begin{document}
\end{document}
```

In diesem Beispiel wird über den Befehl `\documentclass{scrartcle}` die Dokumentenklasse `scrartcle` festgelegt – wir wollen also einen Artikel schreiben. Über den Befehl `\begin{document}` wird der Anfang des eigentlichen Dokumentes eingeleitet und mit dem Befehl `\end{document}` wird das Ende des eigentlichen Dokumentes angezeigt. Mit dem Prozentzeichen (%) wird ein Kommentar eingeleitet, d. h. der ganze Text, der hinter dem %-Zeichen steht, wird vom LaTeX-Compiler ignoriert und nicht ausgewertet. Mit dem Befehl `\title` wird der Titel des Dokumentes definiert und mit `\author` der Autor des Dokumentes.

Wenn Sie mit LaTeX größere Dokumente wie z. B. Bücher schreiben, können Sie Kapitel mit dem Kommando `\chapter` in dem Hauptteil des Dokumentes einfügen. Darüber hinaus gibt es für Bücher noch einen größeren Teilabschnitt, nämlich `\part`, mit dem sich Bücher in einzelne Teile gliedern lassen.

```
%Ein Buch mit einem Kapitel
\documentclass{scrbook}
%Hauptteil des Dokumentes
\begin{document}
\chapter{Erstes Kapitel}
\end{document}
```

Wenn Sie kleinere Texte wie z. B. einen Artikel schreiben, lassen sich Abschnitte und Unterabschnitte über das Kommando `\section` bzw. `\subsection` definieren:

```
\documentclass{scrartcle}
\begin{document}
Mein kleiner Artikel
\section{Erster Abschnitt}
In diesem Abschnitt möchte ich Ihnen etwas über die Grundlagen von LaTeX
erzählen.
\subsection{Erster Unterabschnitt}
Einfügen von Kapiteln in LaTeX.
\section{Zweiter Abschnitt}
Einfügen von Abschnitten in LaTeX.
\subsection{Zweiter Unterabschnitt}
Einfügen von Unterabschnitten in LaTeX.
\end{document}
```

Die in geschwungenen Klammern notierten Inhalte werden dann als Abschnittsüberschriften in das Dokument übernommen und entsprechend hervorgehoben wie das folgende Bild zeigt:

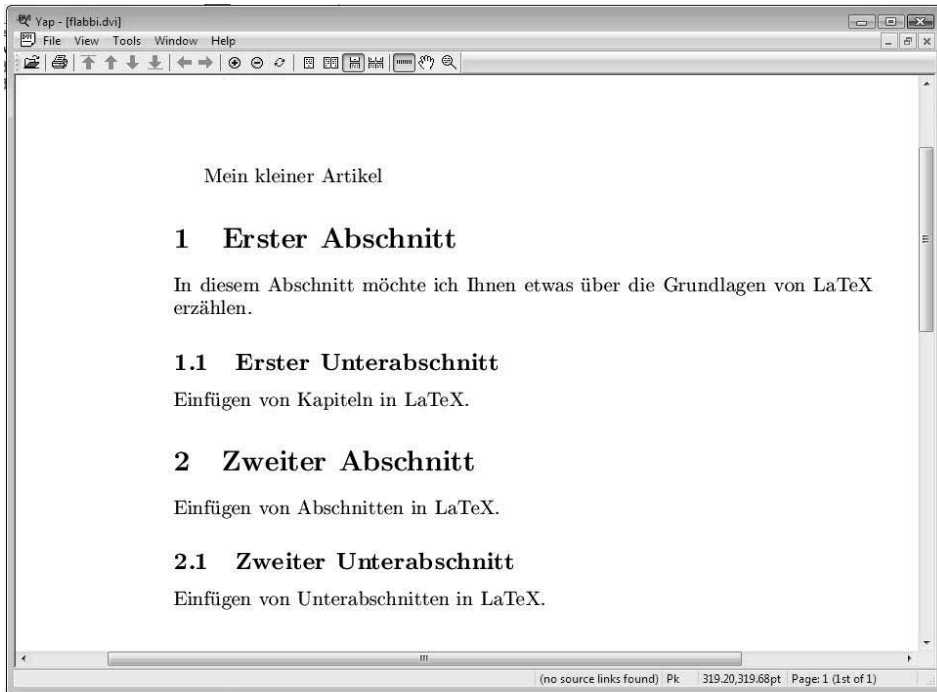


Abbildung 2.1: Artikel mit Abschnitten und Abschnittsüberschriften

Spracheigenschaften

LaTeX unterstützt standardmäßig verschiedene Sprachen. Wenn Sie hauptsächlich mit der deutschen Sprache, aber auch mit Fremdsprachen arbeiten, lässt sich das Paket `babel` einbinden. Für die deutsche Sprache gibt es die beiden Optionen `german` bzw. `ngerman`. `ngerman` benutzt dabei wie schon erwähnt die neue deutsche Rechtschreibung, `german` die althergebrachte:

```
\usepackage[ngerman]{babel}
```

Wenn Sie andere Sprachen einbinden wollen, müssen Sie lediglich den Parameter abändern, hier zum Beispiel für Englisch und Französisch:

```
\usepackage[english]{babel}
\usepackage[french]{babel}
```

LaTeX sieht die zuletzt hinzugefügte Sprache als Standardsprache des Dokumentes an. Dadurch wird u. a. die Silbentrennung sowie die Beschriftung von Abschnitten beeinflusst. So bewirkt die Verwendung von `\usepackage[ngerman][{babel}]`, dass LaTeX das Inhaltsverzeichnis (später mehr dazu) auch wirklich als Inhaltsverzeichnis bezeichnet und nicht als »Table of Contents«.

Fonts (Schriftarten)

LaTeX unterstützt eine Reihe von Schriftarten. Dazu gehören sowohl OpenType-Fonts als auch T1-Fonts. Die verschiedenen Fonts werden in Zeichensätzen codiert, wie z. B. T1, OT1, OT2, usw. T1 steht dabei für Type1 und OT für OpenType. Bei Type1-Fonts handelt es sich um PostScript-Fonts. Die grundlegende Schriftart ist *Computer Modern*, die es in mehreren Varianten gibt.

Für die Erzeugung von Schriften gibt es verschiedene Befehle, die Sie in Kapitel 15 kennenlernen werden. Eine T1-Schriftart kann wie folgt eingebunden werden:

```
\usepackage[T1]{fontenc}
```

LaTeX und Zeichensätze

LaTeX verwendet standardmäßig den amerikanischen Zeichensatz, d. h. die ASCII-Codierung ohne Umlaute und Sonderzeichen. Wenn Sie in einem deutschen Dokument deutsche Umlaute nach der alten oder neuen deutschen Rechtschreibung einfügen wollen, gibt es mehrere Möglichkeiten.

So können Sie zum einen die Umlaute oder Sonderzeichen manuell in den Text einfügen:

```
"a, "o, "u % für ä, ö und ü
```

Eine Alternative bietet das Paket `inputenc` mit dem Parameter `latin1`:

```
\usepackage[latin1]{inputenc}
```

Wenn Sie dieses Paket einbinden, werden die Umlaute automatisch richtig dargestellt. Unten sehen Sie ein Beispiel für ein Dokument für die deutsche Sprache mit deutschen Umlauten:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
\end{document}
```

Das Paket `inputenc` ist für windows- und unixbasierte Systeme vorgesehen. Für den Mac gilt folgende Variante:

```
\usepackage[applemac]{inputenc}
```

Eine Alternative zu den oben genannten Möglichkeiten stellt folgende Zeile dar. Sie bindet die Unicode-basierte Zeichencodierung ein.

```
\usepackage[utf8]{inputenc}
```

2.1.5 LaTeX-Modi

Wenn LaTeX den Quellcode verarbeitet, befindet es sich immer in einem der drei Modi:

1. LR-Modus
2. Mathematik-Modus
3. Paragrafen-Modus

Der häufigste Modus ist der Paragrafen-Modus (engl. *paragraph mode*). LaTeX befindet sich im Paragrafen-Modus, wenn es gewöhnlichen Text verarbeitet. In den Math-Modus wechselt LaTeX nur dann, wenn es eine mathematische Formel verarbeitet oder auf eine Mathematikumgebung trifft. Im LR-Modus (engl. *left-to-right mode*) verarbeitet LaTeX den Quelltext von links nach rechts, insbesondere auch Leerzeichen. Das bedeutet, dass Leerzeichen im LR-Modus nicht ignoriert werden wie es beim Paragrafen- oder Math-Modus der Fall ist, sondern mit ausgegeben werden. Dafür kennt LaTeX im LR-Modus keinen Zeilenumbruch.

2.1.6 LaTeX-Umgebungen

LaTeX bietet für bestimmte Befehle spezielle Umgebungen (engl. *environment*). So gibt es z. B. eine Umgebung für Aufzählungen (engl. *enumeration*), mathematische Formeln usw. Eine Umgebung wird in LaTeX so eingeleitet:

```
\begin{environment}  
\end{environment}
```

Umgebungen lassen sich schachteln. So können Sie z. B. innerhalb der Mathematikumgebung noch weitere mathematikspezifische Umgebungen definieren.

2.1.7 LaTeX und Bilder

Jedes gute Dokument sollte ein paar Bilder enthalten. Allerdings ist das Einbinden von Bildern in LaTeX-Dokumente vordergründig nicht ganz so einfach. Das kommt daher, dass Sie beim Einfügen von Bildern darauf achten müssen, welches Ausgabeformat Ihr Dokument haben soll. Wenn Sie PDF-Dateien erzeugen wollen, können Sie nur die Formate PNG und JPG benutzen. Wenn Sie DVI-Dateien erzeugen wollen, müssen Sie auf das EPS-Format zurückgreifen.

Tipp: Wenn Sie sowohl DVI- als auch PDF-Dateien einsetzen wollen, sollten Sie die Grafiken in allen drei Formaten erzeugen und in dem Verzeichnis ablegen, in dem sich der LaTeX-Quelltext Ihres Dokumentes befindet.

Bei der Arbeit mit Bildern gibt es noch ein paar Besonderheiten, die Sie beachten sollten. Theoretisch können Sie zwar auch andere Bildformate wie z. B. .tif, .gif oder .bmp

über bestimmte Befehle einbinden. Allerdings werden Sie dann feststellen, dass das Bild nicht in das erzeugte PDF- oder DVI-Dokument eingebunden wird.

So ist es z. B. mit dem Befehl `\DeclareGraphicsRule` möglich, weitere Grafikformate festzulegen, die Sie in Ihrem Dokument verwenden möchten:

```
\DeclareGraphicsRule{.bmp}{}{}{}
\includegraphics[scale 1.0, keepaspectratio]{kap2 bild1.bmp}
```

Sie können zwar die Bitmap-Datei (das Bild `kap2 bild1.bmp` in geschwungenen Klammern) auf diese Weise einbinden, werden aber feststellen, dass LaTeX bzw. pdfLaTeX die Bitmap nicht in die PDF-Datei eingebaut hat.

Tipp: Um Probleme beim Einbinden von Bildern zu vermeiden, sollten Sie nur die von DVI bzw. PDF unterstützten Bildformate benutzen.

Die Tabelle gibt einen Überblick, über die von DVI und PDF unterstützten Bildformate:

DVI	PDF
.eps	.png, .jpg

Tabelle 2.4: Bildformate für DVI und PDF

2.2 Grundlagen von Zeichensätzen

Da LaTeX ein Textsatzsystem ist und viele Befehle für die Formatierung und Ausrichtung von Text bietet, ist es hier und da hilfreich, etwas Grundwissen über Zeichensätze zu haben, d. h., wie sie in LaTeX dargestellt werden. Ein Zeichensatz wird in LaTeX über fünf Attribute definiert: Codierungs-, Familien-, Serien-, Form- und Größenattribut.

2.2.1 Codierung

Über das Codierungsattribut legen Sie die Codierung eines Zeichensatzes fest. Das Attribut selbst ist dabei ein Parameter, der aus drei Buchstaben besteht, wobei die letzten beiden Zeichen auch Zahlen sein können. Die Tabelle gibt einen Überblick über die verschiedenen Codierungen und ihre Beschreibung:

<i>Codierung</i>	<i>Beschreibung</i>
T1	Erweiterte TeX-Zeichensätze (z. B. ec-Schriften)
OT1	TeX-Textzeichensätze (z. B. cm-Schriften)
OT2	Kyrillische Zeichensätze
OT3	Internationale phonetische Zeichensätze
OML	Mathematische TeX-Zeichensätze
OMS	Mathematische TeX-Symbolzeichensätze
OMX	Erweiterte mathematische TeX-Zeichensätze
U	Unbekannte Zeichensätze
L<xx>	Lokale Zeichensätze

Tabelle 2.5: Parameter für Codierung

Diese Attribute können Sie über spezielle LaTeX-Befehle festlegen.

2.3 LaTeX-Fehlermeldungen

Während der Erstellung eines LaTeX-Dokumentes können Fehler auftreten. So meldet LaTeX z. B. einen Fehler, wenn die Dateiendung eines Bildes nicht stimmt, das Sie eingebunden haben, oder wenn Sie einen Befehl falsch geschrieben haben, oder wenn Sie vergessen haben, ein Paket einzubinden, dessen Befehle Sie benutzen.

Je nachdem, mit welcher LaTeX-Distribution Sie arbeiten bzw. welchen Editor Sie benutzen, werden die Fehlermeldungen unterschiedlich angezeigt. So formatiert z. B. Texmaker Fehlermeldungen standardmäßig in Rot. Bei einem erfolgreichen Compilerlauf erscheint der Name des Dokumentes bei Texmaker in Grün.

In dem folgenden Beispiel ist der Befehl `\begin{document}` falsch geschrieben (nämlich ohne einleitenden Backslash):

```
\documentclass{scrartcle}
%falscher Befehl \begin{document}
begin{document}
\end{document}
```

LaTeX antwortet darauf mit der Fehlermeldung: »!LaTeX Error: Missing \begin{document}«. Damit Sie den Fehler schnell finden können, gibt LaTeX zusätzlich den Namen des Dokumentes sowie die Zeilennummer aus, in der der Fehler aufgetreten ist.

Aus Fehlern entstehen oft Folgefehler. So meldet in diesem Beispiel LaTeX als weiteren Fehler, dass es die Datei `kap2 bsp5.aux` nicht finden kann: »LaTeX Error: can't find

file kap2 bsp5.aux". Das kommt daher, weil LaTeX wegen des Fehlers den Compilerlauf abgebrochen hat und somit keine Ausgabedatei erstellt hat.

2.4 LaTeX-Befehle und -Gruppen

LaTeX-Befehle lassen sich zu Gruppen zusammenfassen. Eine Gruppe von LaTeX-Befehlen wird durch geschweifte Klammern `{ ... }` ausgedrückt:

```
{\Befehl{\Befehl...}}
```

Wenn Sie Befehle in Gruppen organisieren wollen, müssen Sie auf die Reihenfolge der geschweiften Klammern achten. LaTeX meldet einen Fehler, wenn die Reihenfolge der Klammern nicht richtig ist:

```
\Befehl1{\Befehl2{}}
```

In diesem Beispiel wird zunächst `Befehl1` aufgerufen, gefolgt von `Befehl2`, der innerhalb von `Befehl1` aufgerufen wird. Diese Reihenfolge von Befehlen lässt sich fast beliebig fortsetzen. Sie müssen nur beachten, die Klammern richtig zu setzen.

3 Überblick über LaTeX-Befehle

Nachdem Sie im letzten Kapitel einige zentrale Grundlagen von LaTeX kennengelernt haben, erhalten Sie in diesem Kapitel einen Überblick über die verschiedenen Möglichkeiten, die LaTeX Ihnen bietet.

3.1 Papiergröße festlegen

Sie können mit LaTeX die Papiergröße für Ihr Dokument festlegen, z. B. DIN A4, DIN A3, DIN A5, US-Letter usw. Die Einstellung für die Papiergröße lässt sich am einfachsten über den Befehl `\documentclass` in der Präambel des Dokumentes vornehmen:

```
\documentclass[a4paper]{article}  
\documentclass[a3paper]{article}
```

3.2 Ränder vorgeben

Mit dem Befehl `\setlength` können Sie die Ränder eines Dokumentes fest vorgeben. Die Ränder eines Dokumentes sind einerseits fest vorgegeben, können andererseits aber auch berechnet werden. Über den Befehl `\setlength` wird bestimmt, ob ein Rand für Notizen eingerichtet oder die Größe der Kopfzeile, Fußzeile usw. geändert werden soll.

Der allgemeine Aufruf des Befehls `\setlength` lautet:

```
\setlength{Parameter}{weitere Parameter}
```

Weitere Informationen zum Festlegen von Randeinstellungen finden Sie in Kapitel 4.

3.3 Rahmen, Striche und Boxen

Das Boxenkonzept ist ein grundlegendes Konzept in LaTeX. Viele sogenannte gleitende Objekte wie Tabellen oder Bilder können mithilfe von Boxen dargestellt werden. Es gibt verschiedene Arten von Boxen. So gibt es z. B. Parboxen, LR-Boxen oder Rule-Boxen. Diese Boxen können Sie über verschiedene Befehle erzeugen.

Die einfachste Möglichkeit zur Erstellung einer Box liefern diese Befehle:

```
\makebox[<breite>][<pos>]{<text>}  
% bzw.  
\framebox[<breite>][<pos>]{<text>}
```

Die Unterschiede bestehen darin, dass Sie mit `\makebox` eine Box mit fester Breite erzeugen können und mit `\framebox` eine Box mit Linien und Strichen. Weitere Informationen zu Boxen erhalten Sie in Kapitel 4.

3.4 Gleitende Objekte

In LaTeX werden Bilder oder Tabellen unter dem Begriff gleitende Objekte gefasst. Bilder werden dabei unter anderem als Boxen dargestellt. Eine Tabelle ist ein gleitendes Objekt, das über mehrere Seiten gehen kann.

Neben der einfachen Darstellung von einzelnen Tabellen oder Bildern können Sie gleitende Objekte mit der `minipage`-Umgebung auch nebeneinander anordnen. Die `minipage` Umgebung ist eine sehr praktische Einrichtung, mit der Sie viele interessante Effekte erzielen können. So können Sie damit z. B. zweispaltigen Text setzen, in dem zwei Tabellen nebeneinander aufgeführt sind.

Im nachfolgenden Abschnitt erfahren Sie, wie Tabellen und Bilder erstellt werden.

3.5 Tabellen und Bilder

LaTeX bietet verschiedene Befehle für Tabellen und Bilder. Diese sind, wie oben beschrieben, gleitende Objekte.

3.5.1 Tabellen

Zur Erzeugung einer Tabelle ist der Befehl `\begin{tabbing}` zuständig:

```
\begin{tabbing}  
Tabellentext  
\end{tabbing}
```

Die Grundlage aller Tabellen in LaTeX ist der Tabulator (engl. *tabstop*) des Computers. Bei `tabbing` handelt es sich um eine LaTeX-Umgebung für Tabellen.

Für Spalten in Tabellen gibt es noch verschiedene Kommandos:

1. `\` zum Erzeugen von Tabstops
2. `\>` zur Ansteuerung der Tabstops
3. `\\` zum Abschluss einer Zeile
4. `\kill` für eine Musterzeile. Der Befehl `\kill` verhindert, dass die Zeile gedruckt wird. Nachfolgende Zeilen, die mit `\>` festgelegt werden, übernehmen die Abstände der Musterzeile.

Für einfache Tabellen mit zwei Spalten gehen Sie so vor:

```
\begin{tabbing}
Spalte 1 \   Spalte 2 \\
\end{tabbing}
```

Für die erste Spalte in der Tabelle benötigen Sie normalerweise nicht das Kommando `\>`. Erst für die zweite Spalte müssen Sie `\>` benutzen und die letzte Spalte mit `\\` abschließen.

Das Kommando `\\` wird außerhalb der Tabellenumgebung für das Einfügen von Absätzen in normalen Text benutzt.

Mit dem Kommando `\hspace{}` wird ein horizontaler Abstand in bestimmten Maßeinheiten gesetzt, z. B. in Zentimetern. Das unten stehende Beispiel bewirkt eine Tabelle mit drei Spalten und sechs Elementen:

```
\begin{tabbing}
Spalte 1 \hspace{1cm} \   Spalte 2 \hspace{1cm} \   Spalte 3 \\
Element 1 \> Element 2 \> Element 3 \\
Element 4 \> Element 5 \> Element 6 \\
\end{tabbing}
```

Bei dieser Tabelle wird mit dem Kommando `\hspace{1cm}` ein Abstand von 1 cm zwischen den Spalten festgelegt.

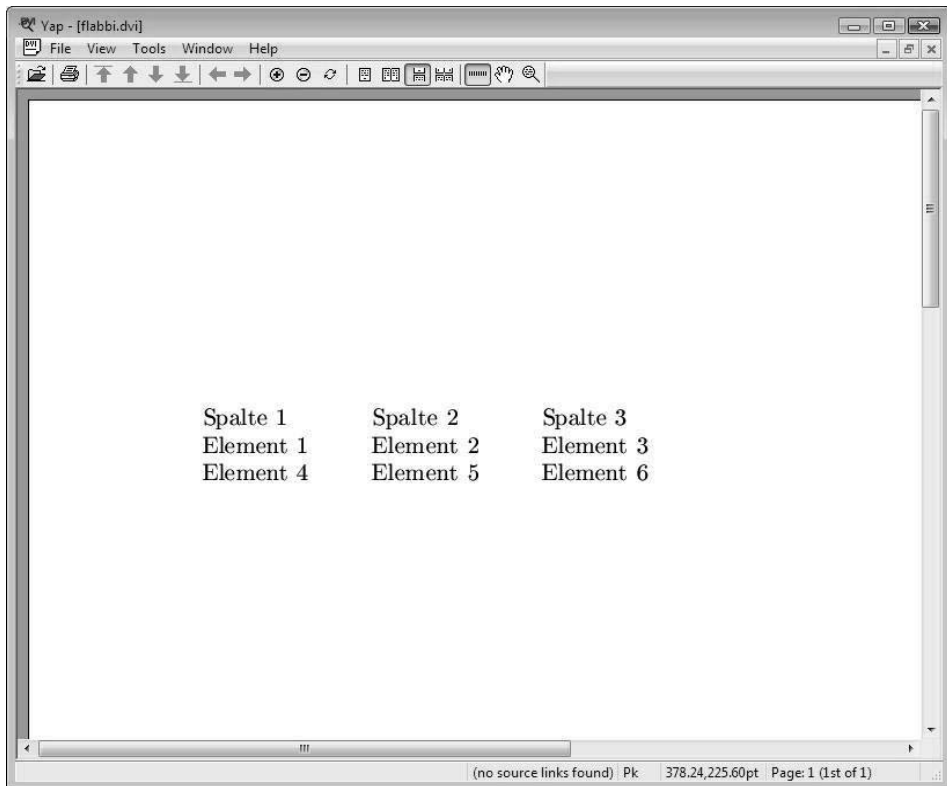


Abbildung 3.1: Die erzeugte Tabelle

Tipp: Bei der Arbeit mit Tabellen kann es passieren, dass die Elemente einer Tabelle nicht korrekt dargestellt werden. Durch die Verwendung von `\hspace` können Sie solche Probleme vermeiden.

3.5.2 Bilder

Auch für das Einbinden von Bildern bietet LaTeX verschiedene Möglichkeiten. Die einfachste Vorgehensweise erfolgt mit dem Befehl `\includegraphics` aus dem Paket `graphicx`:

```
%Paket graphicx in der Präambel einbinden
\usepackage{graphicx}
%Bild in Dokument einfügen
\begin{document}
\includegraphics[bb 0 0 1 1] {kap3 bild1.png}
\end{document}
```

In diesem Beispiel wird das Bild `kap3 bild1.png` eingebunden. Achten Sie hier auf die in Kapitel 2 hingewiesenen Besonderheiten bei der Arbeit mit Bildern. Über den Parameter `bb` des Befehls `\includegraphics` wird eine sogenannte Boundingbox definiert. Ohne die Definition einer Boundingbox meldet LaTeX einen Fehler und das Bild wird nicht eingebunden.

In diesem Beispiel wird die Bounding Box mit gerade einem Quadratmillimeter festgelegt und ist deshalb nicht zu sehen. In Bildformaten wie EPS ist die Bounding Box allerdings bereits enthalten und muss nicht mehr manuell festgelegt werden.

3.6 Aufzählungen, Listen und Texteinbindungen

LaTeX stellt für Aufzählungen und Listen verschiedene Umgebungen (engl. *environments*) bereit. Dabei unterscheidet LaTeX zwischen nummerierten, symbolischen und einfachen Aufzählungen bzw. Listen.

3.6.1 Aufzählungen und Listen

Aufzählungen können über die Umgebung `enumerate` definiert werden. Das folgende Beispiel zeigt eine einfache Aufzählung:

```
\begin{enumerate}
\item Erstes Element
\item Zweites Element
\end{enumerate}
```

Das Ergebnis ist eine einfach nummerierte Liste, bei der die Nummerierung mit 1 beginnt:

1. Erstes Element
2. Zweites Element

Mit der Umgebung `itemize` können Sie eine symbolische Liste erstellen:

```
\begin{itemize}
\item Erstes Element
\item Zweites Element
\end{itemize}
```

Das Ergebnis ist eine Liste mit Aufzählungszeichen. LaTeX setzt standardmäßig einen schwarzen Punkt:

- Erstes Element
- Zweites Element

Wenn Sie eine Liste bzw. Aufzählung ohne irgendeine Formatierung erstellen wollen, können Sie die Umgebung `list` benutzen:

```
\begin{list}{}{}  
\item Erstes Element  
\item Zweites Element  
\end{list}
```

Damit wird eine einfache Liste ohne jegliche Formatierung erzeugt:

Erstes Element

Zweites Element

Es gibt noch weitere Umgebungen für Aufzählungen und Listen. Mit der Umgebung `description` können Sie zum Beispiel Listen von Beschreibungen oder Definitionen erzeugen:

```
\begin{description}  
\item[Buch1]Erste Beschreibung  
\item[Buch2]Zweite Beschreibung  
\end{description}
```

Mit dem Label-Parameter (hier in eckigen Klammern) des `\item`-Kommandos können Sie das Thema bzw. den Text angeben, das beschrieben wird, in diesem Beispiel sind es also die Labels `Buch1` und `Buch2`. LaTeX formatiert diese Labels standardmäßig fett. Die Ausgabe des obigen Beispiels ist:

Buch1 Erste Beschreibung

Buch2 Zweite Beschreibung

Aufzählungen und Listen können wie andere Umgebungen auch geschachtelt werden. Dadurch lassen sich verschachtelte Aufzählungen und Listen erstellen, die mehrere Unterpunkte enthalten.

3.6.2 Einbinden von vorformatiertem Text

LaTeX stellt für die Einbindung von Text verschiedene Umgebungen bereit. Damit können Sie beliebig formatierten Text in das LaTeX-Dokument einbinden. Dieser Text kann z. B. Umlaute, LaTeX-Kommandos usw. enthalten. LaTeX selbst überprüft diesen Text nicht, d. h., der Text wird nicht interpretiert und auch nicht auf eventuelle syntaktische Fehler oder Ähnliches überprüft. Das ist zum Beispiel für das Einbinden von Programmlistings sinnvoll.

Die einfachste Möglichkeit, einen Text in LaTeX einzubinden, ist die Umgebung `verbatim`:

```
\begin{verbatim}  
Die einfachste Möglichkeit, einen Text in LaTeX einzubinden,
```

```
ist die verbatim Umgebung  
\end{verbatim}
```

3.7 Mathematische Formeln

Für mathematische Formeln stellt LaTeX die `math`-Umgebung bereit. Wenn Sie eine Formel in ein LaTeX-Dokument einbinden wollen, können Sie die `math`-Umgebung benutzen:

```
\begin{math}  
E m*c^2  
\end{math}
```

Als Ergebnis formatiert LaTeX die Formel $E=mc^2$ in kursiver Schrift mit hochgestellter 2.

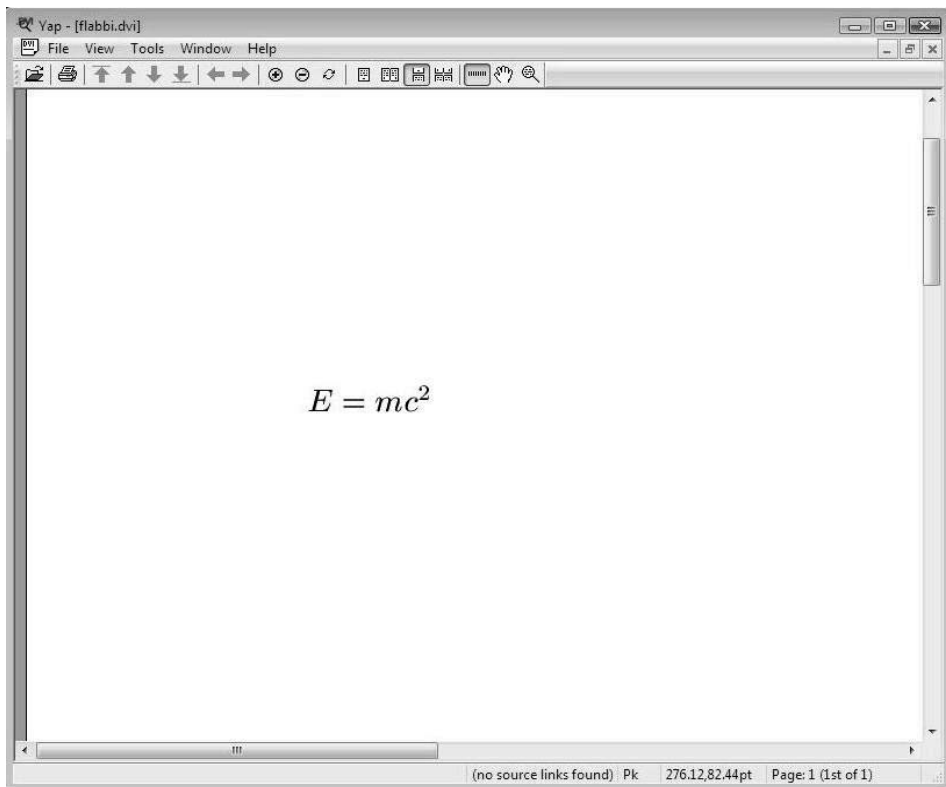


Abbildung 3.2: Einstein-Formel in LaTeX

Eine andere Möglichkeit, eine Formel in einen Text einzubinden, ist das Dollarzeichen (\$). Das \$-Zeichen ist eine Abkürzung für die Mathematikumgebung:

```
$A a*a$
```

Das Ergebnis ist in diesem Beispiel das gleiche, d. h., die Formel $A=a*a$ wird kursiv dargestellt. Es gibt noch weitere Umgebungen für mathematische Formeln. Die Umgebung `displaymath` setzt die Formel z. B. in eine einzelne Zeile:

Ein Beispiel für ein Polynom:

```
\begin{displaymath}  
x^3+2*x^2+4  
\end{displaymath}
```

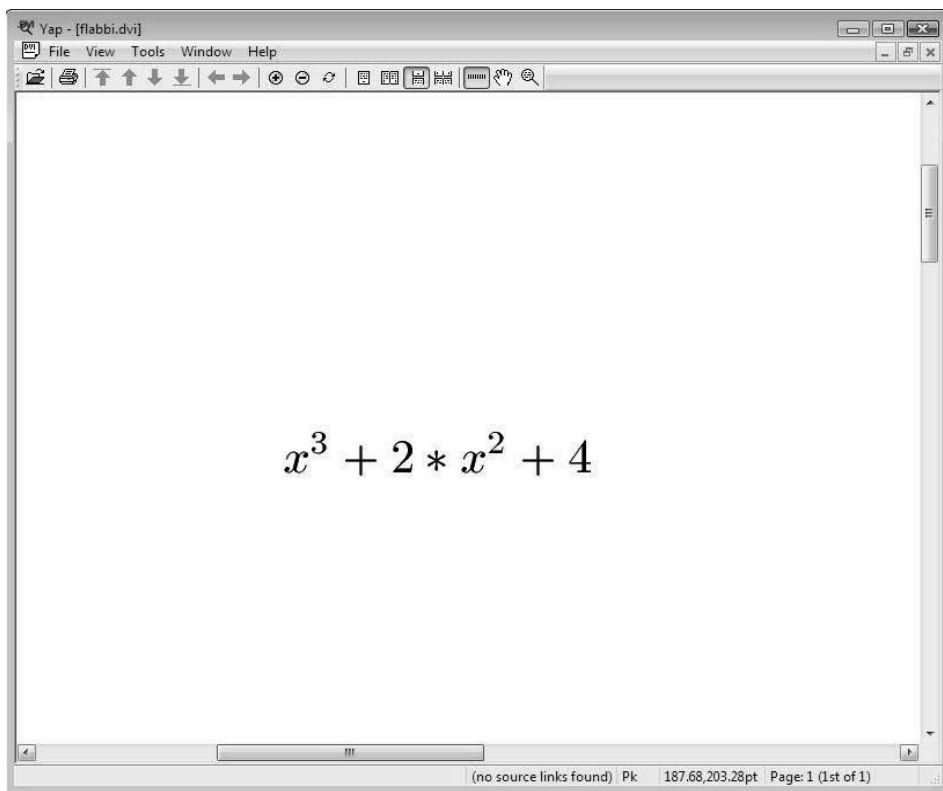


Abbildung 3.3: Polynom-Gleichung in LaTeX

In diesem Beispiel wird das Polynom hinter dem umgebenden Text eine Zeile nach unten versetzt dargestellt.

3.8 Bilder und Zeichnungen

LaTeX stellt für das Zeichnen standardmäßig die Umgebung `picture` zur Verfügung. Neben dieser Umgebung gibt es noch das Extrapaket `pstricks`, das viele Möglichkeiten zum Zeichnen von Bildern bietet. Dieser Abschnitt soll sich jedoch nur mit der `picture`-Umgebung beschäftigen.

3.8.1 Grundlagen der `picture`-Umgebung

Die `picture`-Umgebung wird über den Befehl `\begin{picture}` eingebunden:

```
\begin{picture}(x, y)
% Befehle
\end{picture}
```

Über die Parameter `x` und `y` der `picture`-Umgebung können Sie die Koordinaten der Zeichnung oder des Bildes festlegen. Der Parameter `x` legt dabei die Breite und der Parameter `y` die Höhe des Bildes oder der Zeichnung fest. Über den Befehl `\setlength` können Sie dabei die verwendete Längeneinheit festlegen:

```
\setlength{\unitlength}{Längeneinheit}
```

Der Aufruf des Kommandos `\setlength` erfolgt vor der Einbindung der `picture`-Umgebung:

```
% Längeneinheit festlegen
\setlength{\unitlength}{2cm}
\begin{picture}(5, 5)
%Befehle
\end{picture}
```

In diesem Beispiel wird die Längeneinheit auf 2 cm festgelegt.

3.8.2 Koordinatensystem der `picture`-Umgebung

Um die Zeichnungen platzieren zu können, benötigt LaTeX ein Koordinatensystem. Für die Definition des Koordinatensystems werden Bezugspunkte benötigt, d. h., die linke untere Ecke des Koordinatensystems sowie die Breite und Höhe des Koordinatensystems.

Das Koordinatensystem kann positive und negative Längeneinheiten verarbeiten. Für die Lage des Bezugspunktes des Bildes zum Nullpunkt des Koordinatensystems muss ein Offset angegeben werden. `offset` ist dabei ein Parameter der `picture`-Umgebung:

```
\begin{picture}(x, y)(x offset)(y offset)
%Befehle
\end{picture}
```

3.8.3 Objekte positionieren

Die Bilder und Zeichnungen werden über den Befehl `\put` positioniert. Die grundsätzliche Syntax lautet:

```
\put(x, y){Element}
```

Über die Parameter `x` und `y` werden die Koordinaten des Objekts bzw. Elements angegeben und mit `{Element}` das zu zeichnende Objekt. Sie können dabei verschiedene Formen von Objekten, aber auch z. B. Text zeichnen. Das folgende Beispiel zeichnet zwei Texte. Der eine Text erscheint an der Position `(1,1)` und der andere Text an der Position `(0,0)`:

```
\setlength{\unitlength}{1cm}
\begin{picture}(1,1)
\put(1,1){Ein Text an der Position (1,1) }
\put(0,0){Ein Text an der Position (0,0) }
\end{picture}
```

Es gibt noch weitere Befehle zum Zeichnen verschiedener Objekte, wie z. B. Linien, Kreise, Ovale, Rechtecke usw. Das folgende Beispiel zeichnet mit dem Befehl `\circle` einen Kreis mit dem Radius 2:

```
\setlength{\unitlength}{1cm}
\begin{picture}(1,1)
\put(1,1){\circle{2} }
\end{picture}
```

Hinweis: Achten Sie bei der Benutzung der `picture`-Umgebung auf die korrekte Syntax der Parameter. Bei falscher Syntax kann es passieren, dass LaTeX schwer nachvollziehbare Fehler meldet, z. B. wenn Sie die Parameter `x` und `y` falsch angeben.

Tipp: Neben den Standardmöglichkeiten von LaTeX gibt es noch das Paket `pict2e`, das Sie mit `\usepackage{pict2e}` einbinden können. Dieses Paket hebt einige Beschränkungen der Standardfunktionen von LaTeX auf – so etwa die Limitierung, in Graphen nur bestimmte Steigungen und Kreisradien darstellen zu können.

3.9 Farbe

Es gibt verschiedene Möglichkeiten, Text mit Farbe auszuzeichnen. Die einfachste Vorgehensweise ist, den ausgesuchten Text mit dem entsprechenden Befehl zu markieren.

Nachdem Sie in der Präambel den Befehl `\usepackage{color}` eingefügt haben, können Sie mit dem Befehl `\color` die Farbe für den nachfolgenden Text definieren. Mit dem

Befehl `\textcolor` können Sie Textabschnitte farbig formatieren. Der allgemeine Aufruf lautet:

```
\color{red}
% oder
\textcolor{blue}{Mickey Mouse}
```

LaTeX unterstützt dabei verschiedene Farbformate, unter anderem RGB, CMYK usw.

Unten sehen Sie ein einfaches Beispiel für farbigen Text:

```
Mickey Mouse trägt eine rote \textcolor{red}{Hose}
```

3.10 Einfache Texte

Mit dem oben genannten Rüstzeug und den Grundlagen aus Kapitel 2 können Sie schon einfache Texte mit LaTeX erstellen. In diesem Abschnitt lernen Sie, wie Sie mit den bisherigen Mitteln simple Dokumente schreiben können. In den folgenden Kapiteln lernen Sie weitere Möglichkeiten kennen.

Aufbau eines einfachen Dokumentes

Unten sehen Sie nochmals ein Beispiel für einen Text mit deutschen Spracheigenschaften und Zeichensatz:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
\end{document}
```

Ein einfaches Dokument könnte wie folgt aussehen:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\usepackage{color}
\begin{document}
\section{Einführung}
LaTeX ist eine \textcolor{red}{Formatierungssprache}, mit der man
verschiedene \textcolor{red}{Texte} schreiben und formatieren kann.
LaTeX wird dabei überwiegend für mathematische oder wissenschaftliche
Arbeiten verwendet.
\end{document}
```

Farbe hinzufügen

In diesem Text wird unter der Überschrift »Einführung« ein kurzer Text geschrieben, in dem zwei Wörter rot eingefärbt sind.

Um den Text etwas aufzulockern, können Sie z. B. eine Liste einfügen:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\usepackage{color}
\begin{document}
\section{Einführung}
LaTeX ist eine \textcolor{red}{Formatierungssprache}, mit der man
verschiedene \textcolor{red}{Texte} schreiben und formatieren kann.
LaTeX wird dabei überwiegend für mathematische oder wissenschaftliche
Arbeiten verwendet.
\section{Befehle}
Diese Formatierungen können mit verschiedenen Befehlen erfolgen:
\begin{itemize}
\item Einfache Befehle
\item Umgebungen
\item Spezialbefehle
\end{itemize}
\end{document}
```

So könnte z. B. ein einfacher Text mit LaTeX aussehen:

1 Einführung

LaTeX ist eine Formatierungssprache, mit der man verschiedene Texte schreiben und formatieren kann. LaTeX wird dabei überwiegend für mathematische oder wissenschaftliche Arbeiten verwendet.

2 Befehle

Diese Formatierungen können mit verschiedenen Befehlen erfolgen:

- Einfache Befehle
- Umgebungen
- Spezialbefehle

Abbildung 3.4: Einfaches Dokument mit Aufzählungspunkten und Farbe

Text zentrieren und formatieren

Eine weitere wichtige Gruppe von Befehlen beschäftigt sich mit der Formatierung von Texten. Dazu gehört z. B. die Auszeichnung von Text in Fett, Kursiv, Unterstrichen usw. oder zentriert gesetzter Text.

Die einfachste Möglichkeit, einen Text zu zentrieren, ist die Umgebung `center`:

```
\begin{center}
Ein zentrierter Text
\end{center}
```

Mit den Befehlen `\emph` oder `\itshape` können Sie Text besonders hervorheben. `\emph` (der Befehl leitet sich vom englischen Wort *emphasize* ab) setzt den Text kursiv. Wird der Befehl dagegen in einem bereits kursiven Text verwendet, so wird der hervorzuhebende Text wiederum in das Standardformat umgewandelt.

Der Befehl `\emph` bezieht sich lediglich auf den nachgestellten Text in geschweiften Klammern. Möchte man dagegen längere Textpassagen kursiv setzen, benutzt man den Befehl `\itshape`. Dieses Kommando setzt den kompletten nachfolgenden Text kursiv, bis es durch den Befehl `\normalfont` aufgehoben wird.

```
Dieser Text wird \emph{besonders} hervorgehoben.
Dieser Text wird ebenfalls \itshape kursiv dargestellt. \normalfont und
jetzt geht es normal weiter.
```

Eine weitere Möglichkeit, Text auszuzeichnen, ist der Befehl `\underline`:

```
Dieser Text wird \underline{unterstrichen} dargestellt.
```

Hier ein Beispiel für ein einfaches Dokument mit ausgezeichnetem Text:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
Viele Befehle von LaTeX dienen der \emph{Auszeichnung} von Text.
Der Text wird dabei \itshape{kursiv}, \underline{unterstrichen} oder anders
dargestellt.
\begin{center}
Manchmal möchte man Text auch so hervorheben, dass er in der Mitte steht.
Dazu können Sie die center Umgebung benutzen.
\end{center}
\end{document}
```

Neben diesen grundlegenden Befehlen gibt es noch verschiedene Ergänzungspakete, die weitere Befehle anbieten. Weitere Informationen über die Auszeichnung von Text finden Sie in Kapitel 4.

Mathematische Formeln

Wenn Sie häufig Formeln und mathematische Symbole verwenden, können Sie die `math`-Umgebung und verschiedene Ergänzungspakte einbinden. Angenommen, Sie schreiben eine mathematische Arbeit und haben in diesem Text viele Formeln unterzubringen. Dann könnte Ihr Dokument etwa so aussehen:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
\section{Formeln}
Mit LaTeX können Sie viele wissenschaftliche Dokumente erstellen, die unter
anderem Formeln enthalten. Eine Formel können Sie mit der math Umgebung
erstellen:
\begin{math}
c^2 = a^2 + b^2
\end{math}
In diesem Beispiel wird die Formel von Pythagoras dargestellt. Eine
Alternative für die math Umgebung ist das Setzen der Formel zwischen zwei
Dollarzeichen (\$ \$):
$c^2 = a^2 + b^2$
end{document}
```

In diesem Beispiel wird die Formel des Pythagoras sowohl in der `math`-Umgebung als auch zwischen zwei `$$` dargestellt. Die `math`-Umgebung ist die einfachste Umgebung, um mathematische Formeln darzustellen. Die zwei `$$` sind einfach eine Abkürzung dafür.

1 Formeln

Mit LaTeX können Sie viele wissenschaftliche Dokumente erstellen, die unter anderem Formeln enthalten. Eine Formel können Sie mit der `math`-Umgebung erstellen: $c^2 = a^2 + b^2$. In diesem Beispiel wird die Formel von Pythagoras dargestellt. Eine Alternative für die `math`-Umgebung ist das Setzen der Formel zwischen zwei Dollarzeichen (`$$`): $(c^2 = a^2 + b^2)$.

Abbildung 3.5: Dokument mit mathematischen Ausdrücken

Tip: Das Dollarzeichen wird von LaTeX als öffnendes beziehungsweise schließendes Zeichen für den Mathematikmodus interpretiert. Wenn Sie das Dollarzeichen als solches, nämlich als Symbol für die amerikanische, die kanadische oder die australische Währung, in Ihrem Text darstellen wollen, müssen Sie ihm einen Backslash voranstellen ($\backslash$ \$).

Weitere Informationen über das Setzen von Formeln finden Sie in Kapitel 11.

Größere Dokumente

Neben einfachen Artikeln können Sie auch größere Dokumente wie z. B. Bücher schreiben. Ein Buch wird durch die Dokumentenklasse `book` definiert. Ein Buch unterscheidet sich von einem Artikel dadurch, dass es unter anderem unterschiedliche Gliederungsebenen hat.

So kann ein Buch z. B. die Gliederungsebenen `\chapter`, `\section`, `\paragraph` oder `\part` haben. Ein Artikel hingegen hat dagegen nur die Gliederungsebenen `\section` und `\paragraph`.

```
\documentclass[a4paper]{book}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
\frontmatter
\section{Vorwort}
\mainmatter
\chapter{Kapitel 1: Grundlagen}
In diesem Kapitel lernen Sie die Grundlagen von LaTeX kennen.
\chapter{Kapitel 2: Einfache Befehle}
In diesem Kapitel lernen Sie einfache LaTeX Befehle kennen und anzuwenden.
\backmatter
\end{document}
```

In diesem einfachen Beispiel wird der grundlegende Aufbau eines Buches erläutert. Ein Buch setzt sich aus Vorspann, Hauptteil und Nachspann zusammen. Der Vorspann wird durch den Befehl `\frontmatter` eingeleitet und kann z. B. das Vorwort enthalten. Häufig werden im Vorspann auch Include-Dateien eingebunden. So wäre es zum Beispiel möglich, das Vorwort zu einem Text von einem anderen Autor schreiben zu lassen und es mittels `\include` einzubinden. Bei einer Korrektur im Vorwort muss man sich dann nur noch die neue Datei vom Co-Autor schicken lassen und braucht das eigene Dokument nicht mehr abzuändern.

Im Hauptteil steht dann der eigentliche Text des Dokumentes. Dieser kann entweder manuell geschrieben oder als Datei eingebunden werden. Der Hauptteil wird durch den Befehl `\mainmatter` eingeleitet.

Im Nachspann können Sie dann weitere Elemente wie den Index, ein Namens- oder Stichwortverzeichnis einfügen.

Unten sehen Sie ein Beispiel für die Verwendung von Include-Dateien, die Sie mit dem Befehl `\include` einbinden:

```
\documentclass[a4paper]{book}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\begin{document}
\frontmatter
\section{Vorwort}
\include{vorwort}
\mainmatter
\include{kapitel1}
\include{kapitel2}
\backmatter
\end{document}
```

4 Einfache Texte mit LaTeX

Nachdem Sie in den vorangegangenen Kapiteln die wichtigsten Grundlagen von LaTeX kennengelernt haben, werden in diesem Kapitel weiterführende Themen besprochen.

4.1 Papierformate

Wie Papierformate eingerichtet werden, haben Sie schon in den Kapiteln 2 und 3 erfahren. Es gibt allerdings noch weitere Möglichkeiten, Papierformate zu bestimmen. Die Tabelle zeigt eine Übersicht der von LaTeX unterstützten Formate:

<i>Parameter</i>	<i>Papierformat</i>
a4paper	DIN A4
A5paper	DIN A5
b5paper	DIN B5
letterpaper	Letter US
legalpaper	Legal Letter US
landscape	Querformat
executivepaper	US Executive Paper

Tabelle 4.1: Übersicht über Papierformate

Die oben genannten Werte sind LaTeX-Standardparameter, d. h., Sie können sie direkt an das Kommando `\documentclass` übergeben:

```
\documentclass[executivepaper]{article}
```

Tipp: Das Paket KOMA-Script unterstützt die gleichen Parameter sowie zusätzlich verschiedene Papierformate des ISO-Standards. Diese werden über den Parameter `\isopaper[<reihe>]{<Formatnr>}` angegeben, z. B. `\isopaper[a]{3}`

4.1.1 Das Paket typearea

Für weitere Parameter gibt es das Paket `typearea`. Dieses Paket wird mit `\usepackage{typearea}` in das Dokument eingebunden. Im folgenden Beispiel erhält `typearea` den Parameter `landscape`. Das hat zur Folge, dass die Seite im Querformat gestaltet wird. Alternativ können Sie auch direkt ein Papierformat als Parameter mit angeben:

```
\usepackage{typearea}
\usepackage[a4paper, landscape]{typearea}
```

Beispiel 4.1: Einstellungen für Papierformate mit `typearea`

Tipp: Bei der Verwendung von `typearea` können unter Umständen Probleme mit dem gewählten Papierformat auftreten. LaTeX meldet z. B. den Fehler »bad type area settings«, wenn die heuristisch ermittelte Zeilenbreite nicht mit der aktuell ermittelten Zeilenbreite übereinstimmt

Höhe und Breite

Über das Kommando `\setlength` können Sie die Höhe (`\paperheight`) und Breite (`\paperwidth`) von exotischen Papierformaten festlegen, z. B.:

```
\documentclass{article}
\usepackage{typearea}
\setlength{\paperwidth}{15cm}
\setlength{\paperheight}{15cm}
\typearea{1}
```

Beispiel 4.2: Festlegung von Höhe und Breite eines Dokumentes mit `setlength`

Die Angabe des Kommandos `\setlength` erfolgt dabei in der Präambel. Über das Kommando `\typearea{1}` erfolgt die Neuberechnung der Standardeinstellungen des Pakets `typearea`. Für das Paket `typearea` gibt es noch weitere Parameter:

Parameter	Beschreibung
<code>dvips</code>	Weiterleitung der Papiergröße an <code>dvips</code>
<code>pdftex</code>	Übergabe der Papiergröße an <code>pdftex</code>
<code>pagesize</code>	Dadurch kann zur Laufzeit entschieden werden, ob <code>dvips</code> oder <code>pdftex</code> verwendet wird. Die Werte werden dann entsprechend gesetzt.

Tabelle 4.2: Parameter des Pakets `typearea`

Hier ein Beispiel für die Anwendung des Parameters `pagesize`:

```
\usepackage[a4paper, landscape, pagesize]{typearea}
```

Beispiel 4.3: Anwendung für den Parameter `pagesize`

Hinweis: KOMA-Script und die darin enthaltenen Dokumentenklassen-Parameter verwenden das Paket `typearea` bereits. Wird also KOMA-Script verwendet, dann kommt es zu einer Fehlermeldung, wenn `typearea` nochmals aufgerufen wird.

Tipp: Das Paket KOMA-Script berechnet die Ränder des Dokumentes automatisch nach den Regeln der typografischen Satzspiegelkonstruktion. Weitere Informationen finden Sie in der Dokumentation von KOMA-Script. Die Dokumentation von KOMA-Script finden Sie auf dem Server der deutschsprachigen Anwendervereinigung TeX e.V. (Dante) unter der Adresse <ftp://ftp.dante.de/tex archive/macros/latex/contrib/koma-script/scrguide.pdf>

Seiten vergrößern oder verkleinern

Mit dem Kommando `\enlargethispage{<groesse>}` können Sie eine Seite nach Bedarf verkleinern oder vergrößern. Die Größenangabe kann dabei positiv oder negativ sein. Bei positiven Werten wird die Seite vergrößert, bei negativen Angaben verkleinert:

```
\enlargethispage{1cm}
```

Beispiel 4.4: Der Befehl `\enlargethispage`

Mit dem Parameter `\baselineskip` können Sie die Seite um eine Zeilenbreite vergrößern oder verkleinern. Auch hier können Sie positive oder negative Werte angeben:

```
\enlargethispage{\baselineskip}  
\enlargethispage{ \baselineskip}
```

Beispiel 4.5: Der Befehl `\enlargethispage` mit dem Parameter `baselineskip`

4.2 Ränder definieren

LaTeX bietet verschiedene Möglichkeiten, die Ränder eines Dokumentes speziell an Ihre Bedürfnisse anzupassen.

4.2.1 Das Kommando `\setlength`

Die einfachste Möglichkeit ist das Kommando `\setlength`, das Sie bereits kennengelernt haben. Es gibt über ein Dutzend verschiedene Parameter für die Einstellung von Rändern. Die Tabelle zeigt die verschiedenen Parameter:

<i>Parameter</i>	<i>Beschreibung</i>	<i>Standardwert in Punkt</i>
Ein Zoll + <code>\hoffset</code>	Breite des Randes zwischen Textbereich und Seitenrand	variabel
Ein Zoll + <code>\voffset</code>	Breite des Randes zwischen Text und oberem Rand	variabel
<code>\oddsidemargin</code> bzw. <code>\evensidemargin</code>	Linker (offside)margin aller Seiten bzw. bei zweiseitigen Dokumentenklassen wie »book« der linke Rand der Seiten mit ungerader Seitenzahl. <code>\evensidemargin</code> legt den linken Rand der geraden Seiten fest.	-9 pt bzw. -6 pt
<code>\topmargin</code>	Abstand zwischen oberem Seitenrand und Kopfzeile für gerade (engl. even) und ungerade (engl. odd) Seiten	-38 pt
<code>\headheight</code>	Höhe der Kopfzeile	15 pt
<code>\headsep</code>	Kleiner Rand zwischen oberer Textbereichsgrenze und Textbereich über die ganze Papierbreite	18 pt
<code>\textheight</code>	Höhe des Textbereichs, d. h. des Satzspiegels	558 pt
<code>\textwidth</code>	Breite des Textbereichs, d. h. des Satzspiegels	355 pt
<code>\marginparsep</code>	Abstand zwischen Textbereich und Randnotizen	8 pt
<code>\marginparwidth</code>	Breite für Randnotizen	54 pt
<code>\hoffset</code> und <code>\voffset</code>	Offset-Parameter für den Rand zwischen Textbereich und Seitenrand	0pt bzw. 0pt
<code>\marginparpush</code>	Minimaler Abstand zwischen zwei Randnotizen	5 pt
<code>\footheight</code>	Höhe des Freiraums, der für eine Fußzeile reserviert wird.	0pt
<code>\footskip</code>	Abstand zwischen Rumpf und Fusszeile	0pt
<code>\paperwidth</code> bzw. <code>\paperheight</code>	Papierbreite und Papierhöhe	483 pt bzw. 682 pt

Tabelle 4.3: Parameter des Kommandos `\setlength`

Wenn Sie z. B. eine Kopfzeile für Ihr Dokument definieren möchten, können Sie Folgendes eingeben:

```
%Präambel
\setlength{\headheight}{10pt}
%Dokument
\begin{document}
\end{document}
```

Beispiel 4.6: Anwendung des Parameters `headheight`

4.2.2 Das Paket `geometry`

Neben dem Befehl `\setlength` gibt es noch das Paket `geometry`, das ähnliche Parameter für die Definition von Rändern bietet. Das Paket `geometry` können Sie mit `\usepackage[parameter]{geometry}` einbinden. Als Alternative gibt es den Befehl `\geometry`, der die gleichen Parameter akzeptiert. Die Tabelle gibt eine Übersicht über die Parameter des Pakets `geometry`:

<i>Parameter</i>	<i>Beschreibung</i>
Papierformate	Papierformate ähnlich wie bei <code>\documentclass</code> der DIN-Reihe usw. z. B. <code>a4paper</code>
<code>screen</code>	Format für Bildschirmpräsentationen
<code>landscape</code>	Querformat
<code>portrait</code>	Hochformat
<code>paperheight</code>	Papierhöhe
<code>paperwidth</code>	Papierbreite
<code>papersize</code>	Papiergröße
<code>twoside</code>	Schaltet auf ungerade und gerade Seiten um
<code>left</code> bzw. <code>inner</code>	Linker Rand bzw. innerer Rand bei Doppelseiten
<code>right</code> bzw. <code>outer</code>	Rechter Rand bzw. äußerer Rand bei Doppelseiten
<code>width</code>	Breite des Textes, d. h. des Satzspiegels inklusive Randnotiz
<code>textwidth</code>	Breite des Textes
<code>marginparwidth</code>	Breite der Randnotiz
<code>marginparsep</code>	Abstand zwischen Text und Randnotiz
<code>height</code>	Höhe des Textes inklusive Kopf- und Fußzeile

Tabelle 4.4: Parameter des Pakets `geometry`

Die Parameter werden dann z. B. so angegeben:

```
\usepackage[a4paper, textwidth 10cm, textheight 10cm]{geometry}  
%bzw.  
\geometry{a4paper, textwidth 10cm, textheight 10cm}
```

Beispiel 4.7: Paket `geometry` mit Parametern

Um den Befehl `\geometry` nutzen zu können, müssen Sie zunächst das Paket `geometry` einbinden:

```
%Einbinden von geometry  
\usepackage{geometry}  
%Aufruf des Kommandos \geometry  
\geometry{a4paper, textwidth 10cm, textheight 10cm}
```

Beispiel 4.8: Unterschied zwischen dem Paket und dem Befehl `geometry`

Es gibt noch viele weitere Parameter, die Sie dem Paket `geometry` bzw. dem Kommando übergeben können.

Tipp: LaTeX schreibt die angegebenen Parameter von `geometry` in eine Logdatei, die Sie jederzeit einsehen können.

4.3 Spracheigenschaften

Sie können für Ihre Dokumente Eigenschaften für die deutsche Sprache festlegen. LaTeX bietet dafür – wie bereits angesprochen – das Paket `babel` mit dem Parameter `german` für die alte Rechtschreibung bzw. `ngerman` für die neue deutsche Rechtschreibung. Dieses Paket wird über das Kommando `\usepackage` eingebunden:

```
\usepackage[ngerman]{babel}  
% bzw.  
\usepackage[german]{babel}
```

Beispiel 4.9: Pakete für Spracheigenschaften `german` und `ngerman`

Durch die Angabe dieser Parameter behandelt LaTeX den Text nach der alten bzw. neuen Rechtschreibregelung. Das wirkt sich folgendermaßen aus:

- **Silbentrennung**

Die Silbentrennung richtet sich nach den Regeln der jeweiligen Rechtschreibung. So wird z. B. nach der alten Rechtschreibregelung »ck« als »k-k« getrennt und »st« wird nicht getrennt, wohingegen nach der neuen Rechtschreibung das »ck« vollständig in die neue Zeile gelangt und das »st« getrennt wird. Gleiches gilt für Wörter mit drei Konsonanten, wie z. B. Schifffahrt.

- **Spezielle Formatierungen**

Spezielle Formatierungen, wie z. B. Überschriften, die automatisch z. B. für Inhaltsverzeichnisse, Sachregister, Bibliographie usw. gesetzt werden, oder Abbildungen und Tabellenbeschriftungen erscheinen auf Deutsch. Gleiches gilt für Datumsangaben mit dem Befehl `\today`.

- **Anführungszeichen**

In LaTeX werden Anführungszeichen entweder über das Kommando `\glqq` und `\grqq` bewirkt oder über die Angabe der Zeichen " ' " `:

```
\usepackage[ngerman]{babel}
%Hauptdokument
\begin{document}
Ein Text mit "` hochgestellten Anführungszeichen "`
% hochgestellte Anführungszeichen
Ein Text mit \grqq hochgestellten Anführungszeichen\grqq
% tiefgestellten Anführungszeichen
Ein Text mit \glqq tiefgestellten Anführungszeichen\glqq
\end{document}
```

Beispiel 4.10: Erzeugung von Anführungszeichen in LaTeX

Mit `\grqq` werden hochgestellte Anführungszeichen " " (auch Abführungszeichen genannt) und mit `\glqq` tiefgestellte Anführungszeichen „ „ generiert.

Tipp: Sie können deutsche Umlaute über das Paket `inputenc` mit dem Parameter `latin1` erhalten, z. B. `\usepackage[latin1]{inputenc}`. Als Alternative können Sie die Umlaute durch die Angabe von "a, "o und "u erzeugen. Die Angabe "s generiert das ß.

Hier ein Beispiel:

```
%Präambel
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
```

Beispiel 4.11: Einbinden des Pakets `latin1` mit dem Parameter `inputenc`

Ein Text mit scharfem "s.

Beispiel 4.12: Deutsche Sonderzeichen mit "

4.4 Klassenparameter

In Abschnitt 4.1 dieses Kapitels haben Sie bereits die wesentlichen Parameter des Kommandos `\documentclass` kennengelernt. Das Paket KOMA-Script bietet gegenüber den Standardparametern von LaTeX noch weitere Parameter, die Sie hier angeben können. Die folgende Tabelle zeigt eine Übersicht der nützlichen Einstellungen:

Parameter	Beschreibung
oneside, twoside	Einseitiger oder zweiseitiger Text
onecolumn bzw. twocolumn	Einspaltiger oder zweispaltiger Text
10pt, 11pt, 12pt, xpt	Größe der Grundschrift in Punkt
smallheadings, normalheadings, bigheadings	Größe der Überschrift hinsichtlich der Grundschrift
draft, final	Entwurfsmodus oder nicht

Tabelle 4.5: Parameter des Pakets KOMA-Script

Bei der Angabe der einzelnen Parameter gilt es, noch einige Besonderheiten zu beachten. Wenn Sie z. B. den Parameter `draft` angeben, d. h. im Entwurfsmodus arbeiten, werden keine Bilder eingebunden oder überlange Zeilen mit einem Balken angezeigt.

4.5 Schriftmerkmale

LaTeX bietet, genau wie andere Textbearbeitungsprogramme auch, viele Möglichkeiten, das Schriftbild eines Dokumentes zu ändern. Dazu gehören z. B. Befehle zur Änderung der Schriftform, Schriftgröße und Schriftstärke. Daneben bietet LaTeX noch weitere spezielle Befehle, die auf den Fluss eines Textes einwirken.

4.5.1 Schriftfamilien

Es gibt zahlreiche Schriftfamilien mit speziellen Eigenschaften. So z. B. proportionale und nichtproportionale Schriften sowie serifenlose und Serifenschriften. Eine Serifenschrift zeichnet sich durch bestimmte Kennzeichnungen der Zeichen aus. Dieser Text

hat z. B. Serifen. Diese kleinen Linien schließen die Buchstabenstriche quer zu ihrer Grundrichtung ab, was Sie u. a. an dem T und dem S erkennen können. Eine serifenlose Schrift hat diese Merkmale nicht, wie z. B. die Listings in diesem Buch. Ein weiteres Beispiel für eine serifenlose Schrift ist Arial Sans Serif bzw. Arial als Serifenschrift unter Windows.

Das entscheidende Merkmal einer Proportionalchrift ist der Platz bzw. die Breite, die für ein Zeichen gebraucht wird. Bei einer nichtproportionalen Schrift ist der Platz für jedes Zeichen der Schrift immer gleich breit (schmale Buchstaben wie i oder t benötigen also gleichviel Platz wie breite Buchstaben, z. B. das O), während bei einer Proportionalchrift der Platz angepasst wird.

4.5.2 Schriftfamilien festlegen

Um eine Schriftfamilie festzulegen, gibt es in LaTeX drei Befehle:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\rmfamily</code>	Mit Serifen
<code>\ttfamily</code>	Schreibmaschine
<code>\sffamily</code>	Serifenlose Schrift

Tabelle 4.6: Befehle für Schriftfamilien

Nachfolgend sehen Sie ein paar Beispiele:

```
%Serifenschrift
\rmfamily So wird ein Text mit Serifen formatiert.
% Schreibmaschinenschrift
\ttfamily So wird ein Text in Schreibmaschinentext formatiert.
% Serifenlose Schrift
\sffamily So wird ein Text ohne Serifen formatiert.
```

Beispiel 4.13: Schriftfamilien festlegen

4.5.3 Schriftformatierung

Die Formatierung eines Textes bzw. der Schrift lässt sich auch ändern. So gibt es z. B. Befehle für kursiven, fettgedruckten und mit Kapitälchen gedruckten Text.

Die Tabelle zeigt die wichtigsten Befehle:

<i>Befehle</i>	<i>Beschreibung</i>
<code>\itshape</code>	Kursivschrift
<code>\slshape</code>	Schrägschrift
<code>\scshape</code>	Kapitälchen
<code>\upshape</code>	Aufrecht

Tabelle 4.7: Befehle für Schriftformatierung

Hier ein paar Beispiele:

```
% Text in Kursiv
\itshape So wird ein Text kursiv formatiert.
% Schräger Text
\slshape So wird ein Text in Schrägschrift formatiert.
% Text mit Kapitälchen
\scshape So wird ein Text mit Kapitälchen formatiert.
```

Beispiel 4.14: LaTeX-Standardbefehle für Schriftformatierung

Der schräge Text wird dabei wie kursiver Text in Kursiv dargestellt.

Leerzeichengröße in kursiver Schrift ausgleichen

Es gibt einen speziellen Befehl, mit dem Sie den Abstand zwischen einem kursiven und einem aufrechten Buchstaben ausgleichen können. Dieser Befehl besteht aus einem `\`, gefolgt von einem `/`, also `\/`.

```
Ein Text {\itshape in kursiver Schrift \/}, bei dem der Abstand zum
nachfolgenden Text korrigiert wird.
```

Beispiel 4.15: Abstand von Leerzeichen anpassen

4.5.4 Schriftstärke setzen (Fettung)

Die Schriftstärke können Sie mit den Befehlen `\bfseries` und `\mdseries` setzen. Unten sehen Sie ein Beispiel für diese beiden Befehle. Der Text wird jeweils fett oder normal ausgegeben:

```
\bfseries Ein fetter Text.
\mdseries Ein normaler Text.
```

4.5.5 Schriftgröße

Sie können mit LaTeX die Schriftgröße eines Textes verändern. Dazu gibt es eine Reihe von Befehlen, mit denen Sie winzige und große Schrift erzeugen können.

Die Tabelle listet die verschiedenen Befehle auf:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\tiny</code>	Winzige Schrift
<code>\scriptsize</code>	Sehr kleine Schrift
<code>\footnotesize</code>	Klein (für Fußnoten)
<code>\small</code>	Klein
<code>\normalsize</code>	Normal
<code>\large</code>	Groß
<code>\Large</code>	Noch größer
<code>\huge</code>	Riesig
<code>\Huge</code>	Gigantisch

Tabelle 4.8: Befehle für die Schriftgröße

Relative Schriftgröße

Neben den durch die oben genannten Befehle festgelegten Schriftgrößen ist es auch möglich, die Schriftgröße relativ festzulegen. Dazu gibt es das Paket `relsize`. Dieses Paket wird mit `\usepackage{relsize}` eingebunden:

```
\usepackage{relsize}
```

Neben dem Paket `relsize` existiert auch noch der Befehl `relsize`, der als Parameter die Angabe der relativen Größe enthält:

```
Dieser Text enthält einen {\relsize{1} relativ großen} Text.
```

Der betreffende Text wird dabei in geschweifte Klammern nach dem Befehl `\relsize` gesetzt.

Kleiner und großer Text

Neben dem Befehl `\relsize` finden sich noch die Befehle `\textsmaller` und `\textlarger`, mit denen die Textgröße beeinflusst wird.

```
\usepackage{relsize}
\large Ein K\textsmaller[2]{leiner} Text.
```

Der Befehl `\textsmaller` bekommt als ersten Parameter den Grad der Größe des Textes und als zweiten Parameter den Text, der verkleinert werden soll.



Abbildung 4.1: Mit dem relsize-Paket lässt sich die relative Größe der Schrift beeinflussen.

4.5.6 Zeichensatzbefehle

Sie können mit LaTeX den Zeichensatz einer Schrift beeinflussen. Die Schrifteinstellung gilt dabei jeweils nur für den jeweiligen Text, den der Befehl als Parameter (also in geschweiften Klammern) umschließt.

Befehl	Beschreibung, Schriftfamilie
<code>\textrm</code>	Roman
<code>\texttt</code>	Schreibmaschine
<code>\textsf</code>	Serifenlos
<code>\textit</code>	Kursiv
<code>\textsl</code>	Geneigt
<code>\textsc</code>	Kapitälchen
<code>\textup</code>	Aufrecht
<code>\textbf</code>	Fett
<code>\textmd</code>	Normal
<code>\textnormal</code>	Normal (hebt vorangegangene Befehle wie <code>\itshape</code> auf)

Tabelle 4.9: Zeichensatzbefehle

Ligaturen

Ligaturen sind Buchstabenfolgen, die aus zwei bis drei Buchstaben bestehen und wie folgt aussehen: `fl`, `ff`, `fi`, `ffi` und `ffl`. Diese Ligaturen lassen sich in LaTeX mit dem Befehl `\/` ausschalten. Das Ergebnis ist, dass die Ligaturen dann als einzelne Buchstaben geschrieben werden.

```
stofflich \quad stof\/f\/lich
```

Das Ergebnis ist: stofflich

Eine andere Möglichkeit, Ligaturen zu trennen, stellt der Befehl `|` aus dem Paket `babel` dar.

```
stofflich \quad stof"|flich
```

4.5.7 Das Paket soul

Das Paket `soul` dient zur Formatierung von Text. Mit diesem Paket können Sie Text ebenfalls unterschiedlich formatieren, z. B. fett, kursiv, mit Kapitälchen usw. Eigentlich ist das Paket nur eine Erweiterung zu den Standardfunktionen von LaTeX. Das Paket `soul` können Sie wie andere Pakete auch mit `\usepackage` einbinden.

```
\usepackage{soul}
```

Das Paket `soul` stellt folgende Befehle zur Verfügung:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\so</code>	Sperren von Text
<code>\caps</code>	Sperren von Text in Großbuchstaben
<code>\st</code>	Durchstreichen von Text
<code>\hl</code>	Hervorheben von Text
<code>\ul</code>	Unterstreichen von Text

Tabelle 4.10: Befehle des Pakets `soul`

Das Paket `soul` ist in der Standardinstallation mancher LaTeX-Distributionen wie z. B. der Basisversion von MikTeX nicht enthalten. Sie müssen es gegebenenfalls nachinstallieren. Nachfolgend sehen Sie einige Beispiele der Befehle des Pakets `soul`:

```
\usepackage{xcolor}
\usepackage{soul}
\begin{document}
Beispiel für das Paket soul in Kombination mit xcolor:
\\
\caps{Gesperrter Text mit Großbuchstaben}
\\
\st{Durchgestrichener Text}
\\
\hl{Hervorgehobener Text}
\\
\ul{Unterstrichener Text}
```

```

\\
\so{Gesperrter Text}
\end{document}

```

Beispiel 4.16: Das Paket `soul`

So sieht das Ergebnis aus:

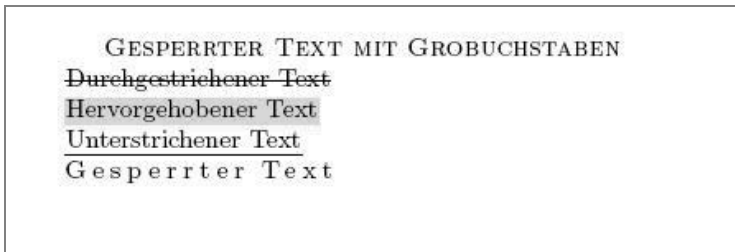


Abbildung 4.2: Das Paket `soul`

4.5.8 Text ausrichten

Der normale Text wird von LaTeX standardmäßig links oder rechts gesetzt. Es gibt jedoch verschiedene Umgebungen, mit denen sich Text zentrieren oder mit Flatterrand auszeichnen lässt.

Text zentrieren

Mit der Umgebung `center` können Sie Text zentrieren:

```

\begin{center}
\end{center}

```

Ein Beispiel:

```

\begin{center}
Ein zentrierter Text.
\end{center}

```

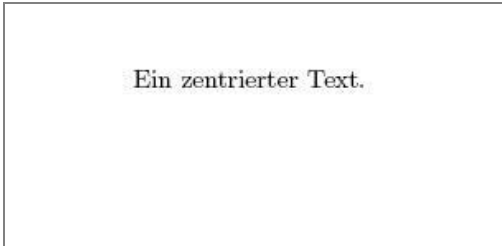


Abbildung 4.3: Zentrierter Text

Für einen Zeilenumbruch müssen Sie einen doppelten Backslash `\\` einfügen.

Text mit Flatterrand

Um Text links- oder rechtsbündig auszurichten, müssen Sie die Umgebungen `flushleft` bzw. `flushright` einbinden:

```
\begin{flushleft}
\end{flushleft}
\begin{flushright}
\end{flushright}
```

Beispiel:

```
\begin{flushleft}
Ein linksausgerichteter Text
\end{flushleft}
```

Das Ergebnis sieht so aus:

```
\begin{flushright}
Ein rechtsausgerichteter Text
\end{flushright}
```

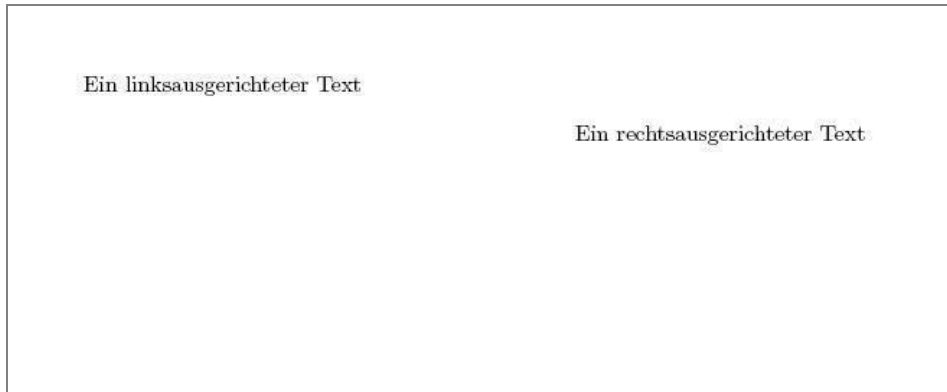


Abbildung 4.4: Links- und rechtsbündig ausgerichteter Text

Eine weitere Möglichkeit, Text auszurichten, bieten die Befehle `\raggedright` bzw. `\raggedleft`. `\raggedright` erzeugt einen linksbündigen, `\raggedleft` einen rechtsbündigen Flattersatz.

Formatierung von Textziten

Mit der `quote`-Umgebung können Sie Text aus anderen Werken zitieren:

```
\begin{quote}  
\end{quote}
```

Beispiel:

```
\documentclass[a4paper]{article}  
\begin{document}  
\begin{quote}  
Wer zuletzt lacht, lacht am längsten.  
\end{quote}  
\end{document}
```

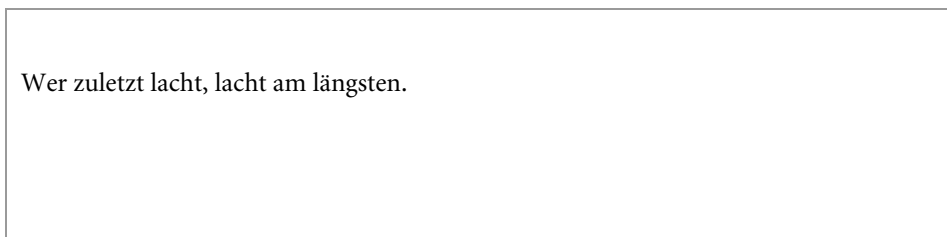


Abbildung 4.5: Zitierter Text

Die Breite des Randes wird über die Befehle `\leftmargin` und `\rightmargin` bestimmt. Innerhalb der `quote`-Umgebung beginnen alle Absätze in der ersten Zeile linksbündig. Es wird jedoch ein Zwischenraum vor und nach dem eingerückten Text eingefügt. Daher müssen Sie dort keine zusätzlichen Leerzeilen einfügen.

Wenn Sie den zitierten Text einziehen wollen, müssen Sie die `quotation`-Umgebung benutzen:

```
\begin{quotation}
\end{quotation}
```

4.5.9 Fußnoten

Fußnoten werden in LaTeX mit dem Befehl `\footnote` erzeugt:

```
\footnote{<Fußnotentext>}
```

Der Fußnotentext kann verschiedenen Text enthalten, unter anderem Tabellen, mathematische Formeln usw. Eine Fußnote kann sich über mehrere Seiten erstrecken, wie es z. B. bei Gesetzeskommentaren üblich ist. Die Fußnoten werden automatisch durchnummeriert und in einer kleinen Schrift gesetzt.

Eine Fußnote können Sie wahlweise auch an das Ende eines Dokumentes setzen, indem Sie das Paket `endnotes` einbinden. Die Nummerierung der Fußnote richtet sich nach der jeweiligen Dokumentenklasse. Bei der Dokumentenklasse `article` werden z. B. alle Fußnoten fortlaufend nummeriert.

```
Der Film Krieg der Sterne erhielt 7 Oscars \footnote{Original: Star Wars}
```

Bei Fußnoten wird standardmäßig eine horizontale Linie eingefügt. Das Aussehen der Linie lässt sich durch den Befehl `\footnoterule` verändern:

```
\renewcommand{\footnoterule}{\rule{<breite>}{<höhe>}}
```

Die Linie lässt sich auch komplett unterdrücken:

```
\renewcommand{\footnoterule}{}%
```

Fußnoten nummerieren

Die Nummerierung von Fußnoten lässt sich über eine Neudefinition des Befehls `\thefootnote` erreichen. Wenn Sie weniger als neun Fußnoten haben, können Sie sie z. B. über Symbole nummerieren (es existieren lediglich neun Fußnotensymbole):

```
\renewcommand{\thefootnote}{\fnsymbol{footnote}}
```

Zur normalen Nummerierung mit arabischen Ziffern kommt man dann wie folgt zurück:

```
\renewcommand{\thefootnote}{\arabic{footnote}}
```

Für die Nummerierung stehen verschiedene weitere Befehle zur Verfügung wie `\arabic`, `\roman` usw.

Abstand von Fußnoten verändern

Den Abstand zwischen den Fußnoten können Sie ebenfalls mit dem Befehl `\setlength` variieren:

```
\setlength{\footnotesep}{abstand}
```

Fußnoten in Boxen

Möchten Sie in Tabellen oder Boxen Fußnoten einsetzen, lässt sich der Befehl `\footnote` nicht verwenden. Die Fußnote muss hier durch zwei verschiedene Befehle erzeugt werden. Dabei steht der Befehl `\footnotemark[<nummer>]` immer innerhalb einer Tabellen- oder Box-Umgebung und generiert die hochgestellte Zahl. Der mit ihm korrespondierende Befehl `\footnotetext` steht dagegen immer außerhalb der Umgebung und erzeugt den Fußnotentext.

```
\footnotetext[nummer]{<fußnotentext>}
```

Tipp: In KOMA-Script werden Fußnoten über den Befehl `\deffootnote` erzeugt, der die Parameter `<markenbreite>`, `<Einzug>`, `<Absatzeinzug>` und `<Markendefinition>` hat:

```
\deffootnote[<Markenbreite>][<Einzug>][<Absatzeinzug>][<Markenende>]
```

Fußnoten schachteln

Die Standardbefehle erlauben es leider nicht, Fußnoten zu schachteln, d. h. mehrere Fußnoten zu erzeugen, die sich aufeinander beziehen. Abhilfe schafft hier das Paket `footmisc`, das Sie in Kapitel 14 kennenlernen werden.

4.5.10 Gliederungsebenen

Ein LaTeX-Dokument wird normalerweise in verschiedene Kapitel und Abschnitte gegliedert. Für die Gliederung in Kapitel und Abschnitte stehen folgende Befehle zur Verfügung:

<i>Befehle</i>	<i>Beschreibung</i>
<code>\part</code>	Teil
<code>\chapter</code>	Kapitel (nur bei <code>book</code>)
<code>\section</code>	Abschnitt

Befehle	Beschreibung
\subsection	Unterabschnitt
\paragraph	Text
\subparagraph	Überschrift

Tabelle 4.11: Befehle zur Gliederung

Die einzelnen Befehle gliedern das Dokument in verschiedene Segmente, z. B. Kapitel, Überschrift, Abschnitt usw. Das Gleiche gilt für die Dokumentenklasse report.

```
\documentclass[a4paper]{article}
\begin{document}
\section{Erster Abschnitt}
  Der erste Abschnitt
\subsection{Erster Unterabschnitt}
  Der erste Unterabschnitt
\section{Zweiter Abschnitt}
  Der zweite Abschnitt
\subsection{Der zweite Unterabschnitt}
  Der zweite Unterabschnitt
\end{document}
```

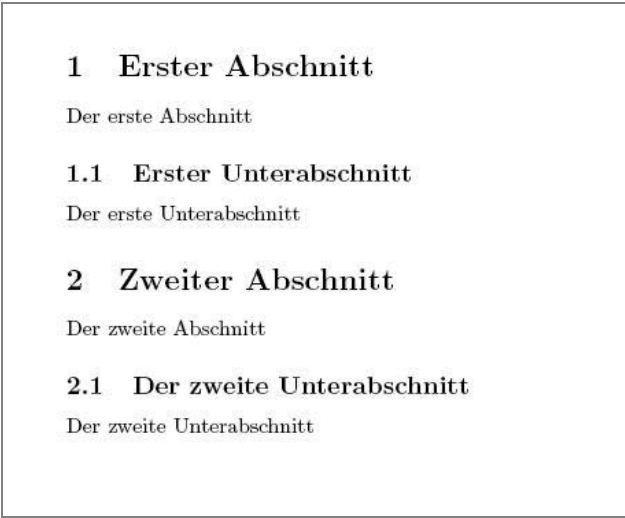


Abbildung 4.6: Gliederungsebenen

Je nachdem, welche Dokumentenklasse Sie verwenden, können Sie verschiedene Gliederungsebenen benutzen. So lässt sich z. B. die Gliederungsebene `\chapter` nur mit der Dokumentenklasse `book` verwenden.

Tipp: Das Paket KOMA-Script bietet zusätzlich die Gliederungsebene `minisec` (für `minisection`) an, die keinem Bereich zugeordnet ist. Der Abstand zu den vorangegangenen und nachfolgenden Absätzen wird dabei verkleinert. Es wird keine Nummerierung durchgeführt, und es findet auch keine Eintragung ins Inhaltsverzeichnis statt.

4.5.11 Umbrüche

LaTeX führt Zeilen- und Seitenumbrüche normalerweise selbstständig aus. Das liegt an dem von LaTeX verwendeten Boxenmodell. Sie können aber auch selbst Zeilen- und Seitenumbrüche einfügen.

LaTeX formatiert den Text normalerweise im Blocksatz, d. h., es werden Leerzeichen eingefügt, um den Text links oder rechtsbündig auszurichten.

Zeilenumbruch einfügen

Die einfachste Art, einen Zeilenumbruch einzufügen, ist der Befehl `\\`:

```
\\
```

Der Befehl `\\` hat den optionalen Parameter [`<abstand>`], mit dem Sie angeben können, wie viel Zwischenraum zwischen den einzelnen Zeilen erzeugt werden soll.

```
\\[1cm]
```

Damit wird ein Zeilenumbruch von 1cm erzeugt. Es existieren noch weitere Befehle für einen Zeilenumbruch:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\newline</code>	Erzeugt einen Zeilenumbruch
<code>\linebreak</code>	Erzeugt einen Zeilenumbruch

Tabelle 4.12: Befehle für Zeilenumbrüche

Der Befehl `\linebreak` hat einen optionalen Parameter [`<num>`], mit dem Sie angeben können, wie notwendig ein Zeilenumbruch ist. Dabei steht der niedrigste Wert 0 für die Empfehlung, einen Zeilenumbruch vorzunehmen, und der Wert 4 für einen zwingenden Zeilenumbruch.

```
\linebreak[4]
```

Der Unterschied zwischen `\newline` und `\linebreak` besteht darin, dass bei `\newline` die umbrochene Zeile nicht in Blocksatz erscheint.

Wenn Sie einen Zeilenumbruch verhindern wollen, können Sie dies über den Befehl `\nolinebreak` erreichen. Auch dieser Befehl hat einen optionalen Parameter [`<num>`]. Hier steht der Wert 0 für die Empfehlung, einen Zeilenumbruch zuzulassen, und der Wert 4 verhindert einen Zeilenumbruch.

Wenn Sie den Parameter weglassen, hat das die gleiche Bedeutung wie der Aufruf:

```
\nolinebreak[4]
```

Seitenumbrüche einfügen

Genauso wie für Zeilenumbrüche gibt es auch Befehle für Seitenumbrüche. Der Befehl für den Seitenumbruch ist `\pagebreak[<num>]`. Dieser Befehl ist das Gegenstück zu `\linebreak`.

Der optionale Parameter [`<num>`] hat die gleiche Bedeutung wie bei `\linebreak`.

Der Befehl `\pagebreak` formatiert die Absätze so, dass der Text in Kopf und Fuß bündig ist. Das bedeutet, dass die erste und die letzte Zeile der Seite auf dem gleichen Niveau wie die Vorgängerseiten sind. Wenn Sie dieses Verhalten nicht wollen, verwenden Sie den Befehl `\newpage`. Der Befehl `\newpage` bricht an der jeweiligen Stelle um, und der Rest der Seite bleibt leer.

Wenn Ihr Dokument gleitende Objekte wie z. B. Bilder oder Tabellen enthält, verwenden Sie den Befehl `\clearpage`. Der Befehl `\clearpage` bewirkt, dass alle bis dahin definierten Bilder und Tabellen, die noch nicht ausgegeben wurden, auf die nächste Seite verschoben werden.

Eine weitere Alternative ist der Befehl `\cleardoublepage`. Dieser beendet die aktuelle Seite. Danach beginnt auf jeden Fall eine ungerade Seite.

Tipp: Wenn der Text zweispaltig ausgerichtet ist, beendet der Befehl `\pagebreak` bzw. `\newpage` die aktuelle Spalte und beginnt eine neue. Der Befehl `\clearpage` hingegen beginnt jedoch zwingend eine neue Seite.

4.5.12 Sonderfälle der Silbentrennung

Für die Silbentrennung gibt es wie bereits beschrieben das LaTeX-Paket `babel` und die Optionen [`german`] und [`ngerman`] als Parameter. Das Paket `babel` führt die Silbentrennung nach den Regeln der deutschen Grammatik durch.

Bei der Trennung von Fremdwörtern muss die Silbentrennung unter Umständen von Hand vorgenommen werden. Dann müssen Sie `\` als Trennzeichen verwenden:

```
Mykor\ rhiza
```

Trennung bei Wörtern mit Bindestrich

Wenn Sie Wörter oder Wortkombinationen mit Bindestrich im Text haben, ist es nicht sinnvoll, einen weiteren Trennstrich einzufügen, wenn genau am Bindestrich getrennt wird. Um den Trennstrich zu verhindern, benutzen Sie zwei hochgesetzte `''` Anführungsstriche:

```
Ho ''Chi ''Min Stadt
```

Bei langen Internetadressen muss der Trennungstrich bei einem Zeilenumbruch entfallen, damit der Leser ihn nicht als Bestandteil der Adresse in den Browser eingibt.

Bei Wörtern mit doppelter Bedeutung, z. B. Staubecken, kommt es natürlich darauf an, wo der Trennungsstrich gesetzt wird, damit das Staubecken eben nicht mit staubigen Ecken verwechselt wird! Wenn Sie unsicher sind, ob LaTeX das Wort an der Stelle trennt, die Sie für richtig halten, können Sie sich mit dem Befehl `\showhyphens` die Standardtrennung anzeigen lassen.

Die Eingabe dafür lautet:

```
\showhyphens{Staubecken}
```

Mit dem Befehl `\hyphenation` können Sie LaTeX eine andere Trennung beibringen:

```
\hyphenation{Staub ecken}
```

Dieser Befehl lässt sich sowohl in der Präambel eines Textes als auch im Text selbst verwenden. Der Befehl bewirkt, dass das Wort nur an den von Ihnen gewünschten Stellen getrennt wird. Sie können auch alle Wörter, die LaTeX nicht sauber trennt, in einem `\hyphenation`-Befehl zusammenfassen.

Die einzelnen Wörter werden dazu lediglich durch ein Leerzeichen voneinander getrennt.

Es gibt aber auch Fälle, in denen Sie auf keinen Fall trennen sollen. In diesen Fällen müssen Sie an die entsprechende Stelle eine Tilde (`~`) einfügen.

Das führt dann dazu, dass an dieser Stelle kein Zeilenumbruch stattfindet.

4.5.13 Abstände zwischen Zeilen und Absätzen

LaTeX legt die Abstände zwischen den Zeilen und den einzelnen Absätzen automatisch fest. Die LaTeX-Standardklassen setzen Absätze normalerweise mit Absatzeinzug und ohne Abstand zwischen den einzelnen Absätzen. Wenn Sie völlig ohne Einzüge und Abstand arbeiten, könnte es unter Umständen schwierig zu erkennen sein, wo der Absatz aufhört und der nächste beginnt, wenn die letzte Zeile fast voll ist. Diese Einstellungen können Sie jedoch manuell vornehmen.

- `\baselineskip`
Dieser Befehl legt den Abstand zwischen zwei Zeilen eines Absatzes fest. Diese Einstellung können Sie nur im Text verwenden.
- `\parskip`
Damit wird der Abstand zwischen zwei Absätzen festgelegt. Das sollte ein elastisches Maß sein und in der Einheit `ex` ausgedrückt werden.
- `\parindent`
Hiermit wird der Wert angegeben, um den die jeweils erste Zeile eingerückt wird. Die Werte werden über `\setlength` festgelegt.

Wenn der Wert für `baselineskip` innerhalb eines Absatzes geändert wird, so gilt dies für den ganzen Absatz. Die Wirkung der Einstellung endet auch mit dem Umschalten der Schriftgröße, da jede Schriftgröße einen eigenen Wert für `\baselineskip` besitzt. Die beiden anderen Einstellungen können sowohl in der Präambel, dann gelten sie im ganzen Dokument, als auch im Text festgelegt werden. Wenn sie im Text verwendet werden, gelten die Einstellungen ab der Stelle, ab der sie auftreten, bis zur nächsten Änderung, wenigstens aber bis zum Ende der aktuellen Umgebung.

Absatzabstände

Für die Festlegung von Absatzabständen gibt es die Ergänzungspakete `parskip` und `nparskip`. Details zu den Möglichkeiten dieser Pakete entnehmen Sie bitte den Paketdokumentationen.

Zeilenabstand

Wenn Sie den Zeilenabstand für ein ganzes Dokument ändern wollen, sollten Sie nicht den Wert für `\baselineskip` ändern, sondern im Vorspann den Befehl

```
\linespread{<faktor>}
```

verwenden. Bei dem Befehl `lineskip` werden keine absoluten Maßangaben eingegeben, sondern der Zeilenabstand `\baselineskip` mit dem in `\lineskip` angegebenen Faktor multipliziert. Die Standardeinstellung von LaTeX bei Abständen ist 1,2 (10 pt Schriftgröße mit 2 pt Durchschuss). Mit diesem Faktor wird die Voreinstellung multipliziert. Für einen eineinhalbfachen Zeilenabstand darf somit nicht der 1,5-fache Wert verwendet werden, da sonst ein 1,8-facher Zeilenabstand entsteht.

Damit erzeugt

```
\linespread{1.25}
```

einen 1,5-fachen Zeilenabstand.

Zeilenabstand nur für einen Absatz ändern

Wenn Sie den Zeilenabstand nur für einen Absatz ändern wollen, können Sie den Befehl `\linespread` in geschweifte Klammern setzen. Zum Aktivieren dieses Befehls müssen Sie den Befehl `\selectfont` verwenden. Der jeweilige Absatz muss jeweils mit dem Befehl `\par` oder einer Leerzeile abgeschlossen werden.

```
<Erster Absatz mit normalem Zeilenabstand>
{\linespread{1.25}\selectfont<Zweiter Absatz mit 1.5 fachem Faktor> \par}
<Dritter Absatz mit normalem Zeilenabstand>
```

Absatzabstände mit `setspace` ändern

Eine andere und einfachere Möglichkeit, den Zeilenabstand zu ändern, liefert das Ergänzungspaket `setspace`, das per `\usepackage` eingebunden wird. Wenn Sie den Zeilenabstand für ein ganzes Dokument einstellen wollen, müssen Sie im Vorspann einen der Befehle `\singlespacing` (einfacher Zeilenabstand), `\onehalfspacing` (eineinhalbfacher Zeilenabstand) oder `\doublespace` (doppelter Zeilenabstand) wählen.

Um den Zeilenabstand für einzelne Textpassagen zu ändern, verwenden Sie die Umgebungen `singlespace`, `onehalfspace` und `doublespace`. Dabei müssen Sie jedoch beachten, dass die Umgebungen nur dazu gedacht sind, den Zeilenabstand zu vergrößern. Die Umgebung `onehalfspace` darf also nicht in einem Dokument verwendet werden, das ohnehin schon einen doppelten Zeilenabstand aufweist. Wenn Sie einen selbstdefinierten Zeilenabstand setzen wollen, wählen Sie die allgemeine `spacing`-Umgebung, die als Parameter den Faktor enthält, um den der Zeilenabstand zu erhöhen ist.

```
\begin{spacing}{1.25}
  Beispieltext
\end{spacing}
```

Den Abstand zwischen Absätzen bestimmen Sie durch die Befehle `\bigskip`, `\medskip` und `\smallskip`, mit denen Sie variablen Zwischenraum setzen können. Ein zu großer Absatzabstand kann allerdings dazu führen, dass der Zusammenhang verloren geht. So ist es beispielsweise schwer zu erkennen, ob nach einer gesetzten Formel der vorangegangene Absatz weitergeht.

Absatzabstände mit KOMA-Script

Wenn Sie für die Abstände KOMA-Script verwenden, so können Sie die Abstände zwischen den einzelnen Absätzen mit den Befehlen `parskip` und `halfparskip` exakter definieren. Es gibt insgesamt acht Befehle:

<code>parskip</code>	<code>halfparskip</code>
<code>parskip*</code>	<code>halfparskip*</code>
<code>parskip+</code>	<code>halfparskip+</code>
<code>parskip-</code>	<code>halfparskip-</code>

Die vier `parskip`-Einstellungen setzen jeweils einen Abstand von einer Zeile zwischen den Absätzen. Der Befehl `halfparskip` verwendet die halbe Zeilenhöhe. Die Varianten der jeweiligen Parameter dienen dazu, die Länge der letzten Zeile bei einem Zeilenumbruch zu regulieren, damit kein Absatzwechsel unerkannt erfolgt. Als Standardparameter wird mindestens ein Leerraum von 1 mm am Ende der letzten Zeile des Absatzes gelassen, bei der Plusvariante bleibt mindestens ein Drittel der Zeile frei. Bei der Sternvariante ist dies ein Viertel der Zeile. Bei der Minusvariante entfallen solche Festlegungen.

Alle acht Einstellungen haben auch Auswirkungen auf den Abstand vor, nach und innerhalb von Listenumgebungen. Dadurch wird erreicht, dass diese Umgebungen und Absätze innerhalb dieser Umgebungen den gleichen Abstand zueinander haben wie die Absätze des normalen Textes. Im Inhaltsverzeichnis sowie im Abbildungs- und Tabellenverzeichnis wird kein zusätzlicher Abstand verwendet.

Bei KOMA-Script ist der Parameter `parindent` voreingestellt, der dafür sorgt, dass jeder Absatz mit einem Erstzeileneinzug von 1cm beginnt, aber keinen Abstand zum vorangehenden Absatz besitzt.

Horizontale und vertikale Zwischenräume

Mit dem Befehl `\hspace` setzen Sie einen Zwischenraum in Textteilen. Diesem übergeben Sie als Parameter den Zwischenraum.

Anstelle einer festen Breite können Sie dem Befehl auch das Maß `\fill` übergeben, um eine Zeile links- oder rechtsbündig abzuschließen. Als Abkürzung für die Schreibweise `\hspace{\fill}` kann der Befehl `\hfill` benutzt werden.

Wenn in einer Zeile mehrere `\hfill`-Befehle hintereinander stehen, so wird dieser dadurch definierte Zwischenraum gleichmäßig aufgeteilt, sodass die Zeile wieder auf beiden Seiten bündig ist.

```
Text\hspace{1cm}Text2
Links\hfill Rechts
Links\ Mitte \hfill Rechts
```

Am Anfang einer Zeile kann der Befehl `\hfill` nicht verwendet werden, da dieser hier von LaTeX eingefügt wird. An dieser Stelle wird vielmehr der Befehl `\hspace` verwendet.

Zwischenraum mit Punkten und Linien füllen

Zwischenräume lassen sich auch mit einer horizontalen oder gepunkteten Linie füllen. Dies geschieht mit den beiden Befehlen `\dotfill` für eine gepunktete und `\hrulefill` für eine durchgezogene Linie.

Diese Befehle lassen sich beliebig miteinander und mit dem Befehl `\hfill` kombinieren. Tritt einer der Befehle mehrmals hintereinander auf, so wird die Länge dementsprechend ausgegeben.

```
Links \dotfill Rechts
Links \hrulefill{} Rechts
Links \dotfill \dotfill \dotfill
Mitte \hrulefill{} Rechts
```

Analog zu den horizontalen Zwischenräumen lassen sich auch vertikale Abstände definieren. Mit dem Befehl

```
\vspace{<abstand>} oder \vspace*{<abstand>}
```

wird zusätzlicher vertikaler Abstand eingefügt. Bei der mit dem Stern ausgezeichneten Variante wird auch dann Zwischenraum eingefügt, wenn an dieser Stelle ein Seitenumbruch stattfindet oder der Befehl am Seitenanfang steht. Wenn diese Befehle am Anfang eines Absatzes verwendet werden, so wird die aktuelle Zeile noch rechtsbündig mit Text ausgefüllt und dann der Abstand eingefügt.

Analog zum Befehl `\hfill` gibt es den Befehl `\vfill` als Abkürzung für `\vspace{\fill}`. Dieser Befehl bewirkt, dass zwischen den Textpassagen ein so großer vertikaler Abstand eingefügt wird, bis die Seite oben und unten bündig ist.

4.5.14 Querverweise

Bei größeren Dokumenten ist es oft nötig, auf eine andere Stelle im Dokument zu verweisen. Sei es, um ein angeschnitts Thema zu vertiefen oder auf bereits erläuterte Sachverhalte zurückzugreifen.

Hierfür muss zunächst die Stelle markiert werden, auf die verwiesen werden soll. Dies können Sie über den Befehl

```
\label{<markierung>}
```

vornehmen.

Den Namen der Textmarke können Sie beliebig wählen, allerdings muss er im Dokument eindeutig sein. LaTeX achtet dabei auch auf Groß- und Kleinschreibung, es ist also durchaus möglich, in einem Dokument die beiden Markierungen Grafiken und grafiken gleichzeitig zu verwenden.

Auf eine solche Textmarke kann dann mit den beiden Befehlen

```
\ref{<markierung>}
\pageref{<markierung>}
```

verwiesen werden. Dabei fügt der Befehl `\ref` die entsprechenden Kapitelnummern ein, der Befehl `\pageref` die Seitenzahl der Seite, auf der die Textmarke definiert wurde.

```
\section{<München>}
```

```
In München gibt es viele Sehenswürdigkeiten. Mehr darüber erfahren Sie in  
Abschnitt \ref{sw muenchen} auf Seite \pageref{sw muenchen}\dots
```

Ein Querverweis kann sowohl auf einer vorangegangenen als auch einer neuen Seite begonnen werden. LaTeX legt für die Querverweise eine Hilfsdatei mit der Dateiendung `.aux` an. Beim ersten LaTeX-Compilerdurchlauf werden dann die Querverweise gesammelt und in diese Datei geschrieben. Bei einem zweiten LaTeX-Lauf werden die entsprechenden Kapitelnummern und Seitenzahlen in das Dokument integriert. Daher muss ein Dokument, das Querverweise enthält, immer zweimal übersetzt werden, damit alle Verweise aktualisiert werden können.

Hier nochmals die beiden Schritte:

1. Sammeln der Querverweise
2. Integrieren der Kapitelnummern und Seitenzahlen

Beim ersten Durchlauf erscheinen statt der aktuellen Kapitelnummern und Seitenzahlen nur Fragezeichen (`??`).

Das Paket `varioref`

Das Paket `varioref` stellt eine Erweiterung für Querverweise dar. Diesem Paket können Sie für deutschsprachige Dokumente den optionalen Parameter `german` übergeben, damit Sie den richtigen deutschsprachigen Text einfügen:

```
\usepackage[german]{varioref}
```

Der Vorteil dieses Pakets ist, dass sich Kapitelnummern und Seitenzahlen von Querverweisen mit nur einem Befehl einfügen lassen.

Der Befehl `\vref` erzeugt, abhängig davon, ob sich Textmarke und Querverweis auf derselben Seite befinden, auf zwei hintereinanderliegenden Seiten oder weit voneinander entfernten Seiten unterschiedliche Texte.

- Wenn die Textmarke und der Verweisbefehl auf derselben Seite liegen, so wird lediglich die entsprechende Kapitelnummer ohne Seitenzahl angezeigt.
- Wenn die Textmarke und der Verweisbefehl auf zwei aufeinanderfolgenden Seiten liegen, so wird, je nachdem, was zuerst kommt, der Text »auf der vorherigen Seite« oder der Text »auf der nachfolgenden Seite« erzeugt.
- Wenn die Textmarke und der Verweisbefehl mehrere Seiten auseinander liegen, so wird der Text »auf der Seite ...« erzeugt.

Diese Standardtexte können Sie auch ändern, indem Sie die erzeugenden Befehle umdefinieren.

Tipp: Weitere Informationen entnehmen Sie der Paketdokumentation unter http://www.ctan.org/tex_archive/macros/latex/required/tools/varioref.pdf

Verweise auf andere Dokumente

Mit dem Paket `xr` wird auf Marken innerhalb anderer LaTeX-Dokumente verwiesen. Bei dem Verweis wird die Kennung vor die Textmarke gestellt und mit einem Bindestrich von ihm getrennt. Dadurch weiß LaTeX, welche Textmarke zu welchem Dokument gehört.

Der dazu verwendete Befehl lautet:

```
\externaldocument[<kennung>]{<externer dateiname>}
```

Im folgenden Beispiel werden drei Dokumente miteinander verknüpft:

```
\usepackage{xr}

\externaldocument[a]{doc1}
\externaldocument[b]{doc2}

\begin{document}
In diesem Dokument existiert ein Verweis auf \pageref{a} das andere Dokument}
\end{document}
```

Tipp: Weitere Informationen zu dem Paket finden Sie unter http://www.ctan.org/tex_archive/macros/latex/required/tools/xr.pdf

4.5.15 Erweiterte Möglichkeiten für Fremdsprachen

Das Paket `babel` unterstützt neben der deutschen Sprache eine Reihe von Fremdsprachen, die Sie als optionalen Parameter angeben können. Neben der Möglichkeit, Fremdsprachen über das Paket `babel` einzubinden, existieren auch einfache Kürzel, die sich in den Text integrieren lassen:

So erhalten Sie z. B. mit der Eingabe von `\'{a}` das Zeichen á und mit `\^a` das Zeichen â. Normalerweise setzen Sie ein solches Zeichen durch die Eingabe von einem `\`, gefolgt von dem jeweiligen Akzent.

Andere Symbole bzw. Zeichen werden durch entsprechende Abkürzungen gesetzt. So können Sie ein ä durch die Eingabe von `\"a` setzen.

Wenn Sie Akzente auf ein i oder j wünschen, müssen Sie beim Setzen eines Akzentes den Punkt entfernen. Durch die Eingabe von `\'i` oder `\H{j}` können Sie ein i oder j mit zwei Akzenten erzeugen.

Neben diesen Akzenten gibt es in unterschiedlichen Sprachen auch Buchstaben, die im Deutschen nicht geläufig sind. So können Sie z. B. durch die Eingabe von `\OE` den Buchstaben *œ* (wie im französischen Wort *Ceuvre*) erzeugen. Entsprechendes gilt für andere Buchstaben.

4.5.16 Symbole

In LaTeX werden einige Symbole als Sonderzeichen interpretiert. Wenn Sie diese im Text verwenden wollen, müssen Sie ihnen einen Backslash voranstellen. Es handelt sich um die Zeichen: `# $ & % {}`

Manchmal benötigt man auch Zeichen und Symbole, die auf der Tastatur nicht zu finden sind, wie z. B. das Copyright-Zeichen. Diese Symbole lassen sich jeweils mit spezifischen LaTeX-Befehlen erreichen.

So erhalten Sie z. B. durch die Eingabe von `\S` ein §-Zeichen, durch die Eingabe von `\p` ein Absatzzeichen – wie z. B. das Absatzzeichen in Word. Durch die Eingabe von `\copyright` erscheint das Copyright-Zeichen.

Andere Symbole setzen Sie durch das Einbinden der jeweiligen Zeichensätze.

4.5.17 Rahmen, Striche und Boxen

Viele Objekte in LaTeX werden durch einfache Boxen gesetzt. Das Boxenkonzept ist daher ein wichtiges Konzept. Boxen gibt es in verschiedenen Formen und Ausführungen.

So gibt es z. B. Parboxen und Rule-Boxen sowie LR-Boxen. Eine Parbox ist eine Absatz-box, d. h., es wird ein Absatz in eine Box gesetzt. Eine Rule Box ist eine Box, mit der Linien und Balken gezeichnet werden:

- Parboxen
Damit werden ganze Absätze gesetzt.
- Rule-Boxen
Damit können Sie Linien und Balken zeichnen.

LR-Boxen und ihre Eigenschaften

Um eine LR-Box ohne Rahmen zu setzen, verwenden Sie den Befehl `\mbox`. Wenn Sie dieser Box einen Rahmen verpassen wollen, benutzen Sie den Befehl `\fbox`. Der Text in geschweiften Klammern wird dann in einen Rahmen gesetzt.

Hier steht eine `\mbox{Box}` ohne Rahmen
Hier steht eine `\fbox{Box}` mit Rahmen

Was bezweckt der Befehl `\mbox`? Bei einer `\mbox` wird der gesamte Text als Einheit betrachtet und es wird kein Zeilenumbruch innerhalb der Box erzeugt. Das kann allerdings dazu führen, dass die Box über das Zeilenende hinausragt.

Wenn Sie eine Box mit einer festen Zeilenlänge setzen wollen, lässt sich dies mit dem Befehl

```
\makebox[<breite>][<pos>]{<text>} bzw.  
\framebox[<breite>][<pos>]{<text>}
```

erreichen. Über den optionalen Parameter `pos` können Sie zusätzlich zur Breite auch die Textausrichtung innerhalb der Box festlegen. Dabei steht der Wert `l` für linksbündig, der Wert `r` für rechtsbündig und der Wert `s` steht für rechts- und linksbündigen Text.

Wenn Sie keinen Parameter angeben, wird der Text zentriert dargestellt.

Die Breite einer solchen Box hängt nicht von der Breite des enthaltenen Textes ab, sodass eine `framebox` auch kleiner als der entsprechende Text sein kann.

Größe einer Box berechnen

Hier die verschiedenen Befehle, um die Größe einer Box zu berechnen:

- `\height`
Berechnet die Höhe des Textes über seiner Grundlinie.
- `\depth`
Berechnet die Tiefe eines Textes unterhalb seiner Grundlinie.
- `\totalheight`
Berechnet die Summe aus Höhe und Tiefe, d. h. die Gesamthöhe eines Textes.
- `\width`
Berechnet die Breite eines Textes.

Gestaltung des Rahmens

Das Aussehen des Rahmens einer Box wird über den Befehl `\fboxrule` bzw. `\fboxsep` verändert. Mit dem ersten Befehl lässt sich die Strichstärke festlegen, mit dem zweiten Befehl der Abstand zwischen Rahmen und eingeschlossenem Text.

Mit dem Befehl `\setlength` können Sie diese Eigenschaften setzen:

```
\setlength{\fboxrule}{0.01cm}  
\setlength{\fboxsep}{0.5cm}  
\fbox{Benjamin Blümchen}
```

Dies führt zu folgender Ausgabe:

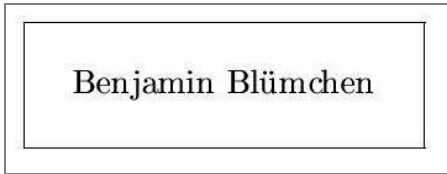


Abbildung 4.7: »Benjamin Blümchen« in einer Box

Wenn Sie den Befehl `\makebox` verwenden, richtet sich die Größe der Box nach dem enthaltenen Text. Es kann aber Fälle geben, in denen sich die Länge der Box nach einem Referenztext richten soll, zum Beispiel, wenn mehrere Boxen untereinander stehen. Dafür gibt es das Paket `makebox`, das per `\usepackage` eingebunden werden muss.

Dieses Paket liefert den Befehl `\makebox*`, mit dem Sie die Größe der Box nach einem Referenztext ausrichten. Dem Befehl `\makebox*` können Sie als Argument zwei Parameter übergeben: erstens den Text, dessen Länge die Größe der Box bestimmt, und zweitens den Text, der in der Box stehen soll. Optional lässt sich noch angeben, ob der Text in der Box links- oder rechtsbündig ausgerichtet werden soll. Das geschieht über die Parameter `l` beziehungsweise `r`, die in eckigen Klammern stehen. Ein Beispiel:

```
\usepackage{makebox}
...
\makebox*{Referenztext}[l]{links}
\makebox*{Referenztext}[r]{rechts}
\makebox*{Referenztext}[l]{Das ist nun definitiv zu lang!}
```

Das Paket `fancybox`

Mit dem Ergänzungspaket `fancybox` lassen sich spezielle Textumrandungen setzen, die über die Standardmöglichkeiten hinausgehen. So können Sie z. B. Boxen mit Schatten usw. darstellen. Das Paket `fancybox` bietet folgende Befehle:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\fbox</code>	Text mit Rahmen
<code>\framebox</code>	Text mit variabler Rahmenbreite
<code>\shadowbox</code>	Rahmen mit Schatten
<code>\doublebox</code>	Doppelter Rahmen
<code>\ovalbox</code>	Eine ovale Box
<code>\Ovalbox</code>	Eine ovale, fette Box

Tabelle 4.13: Das Paket `fancybox`

LR-Boxen mehrfach setzen

Wenn Sie eine Box mit gleichem Inhalt mehrfach setzen wollen, bietet es sich an, sie zu speichern und beim nächsten Gebrauch wieder neu zu laden. Mit dem Befehl `\newsavebox{\boxname}` lässt sich die Box speichern.

Der allgemeine Aufruf lautet:

```
\newsavebox{\boxname}
```

Legen Sie einen Namen für die Box fest. Mit dem Befehl

```
\sbox{\boxname}{text} oder  
\savebox{\boxname}[breite][pos]{text}
```

können Sie eine Box mit dem angegebenen Inhalt als Text erzeugen und abspeichern. Wenn Sie diese Box dann wieder im Dokument benutzen möchten, verwenden Sie folgenden Befehl:

```
\usebox{\boxname}
```

Benutzen Sie als Alternative die Umgebung

```
\begin{lrbbox}{\boxname}  
\end{lrbbox}
```

um die Box abzuspeichern. Das hat den Vorteil, dass Sie auch den `\verb-` oder `\verbatim-`Befehl innerhalb der Box verwenden können.

Boxen verschieben

Mit dem Befehl `\raisebox` lassen sich Boxen auch vertikal verschieben. Der allgemeine Aufruf dazu lautet:

```
\raisebox{<shift>[<oben>][<unten>]{<text>}
```

Mit dem Parameter `shift` wird festgelegt, wie weit die Box verschoben werden soll. Geben Sie positive Werte ein, wird die Box nach oben, bei negativen Werten nach unten geschoben.

Mit den beiden optionalen Parametern `oben` und `unten` wird festgelegt, wie weit die Box über die Grundlinie hinausgehen soll. Standardmäßig berechnet LaTeX diese Angaben über den angegebenen Text. Das Beispiel

```
Im ganzen Leben geht es \raisebox{1ex}{rauf} und \raisebox{1ex}{runter}
```

führt zu folgendem Ergebnis:

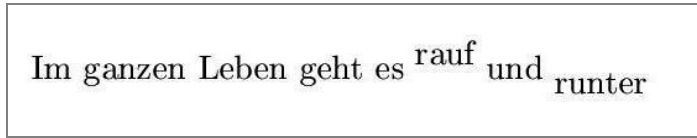


Abbildung 4.8: Mit `raisebox` lassen sich Textstücke heben und senken.

Absatzboxen

Eine Absatzbox rahmt, wie der Name schon sagt, einen Absatz ein. Sie lässt sich mit dem Befehl `\parbox` erzeugen. Im Unterschied zu LR-Boxen gehen bei Absatzboxen die Zeilen nicht über den Seitenrand hinaus. Wenn Sie also längeren Text in eine Box setzen wollen, so müssen Sie eine Absatzbox verwenden:

```
\parbox[<pos>][<höhe>][<ipos>]{<breite>}{text}
```

Eine `parbox` kann eine beliebige Breite aufweisen. Sie hat drei verschiedene Ausprägungen für den Positionierungsparameter `pos`:

- `t` (top)
Ausrichtung am Absatzanfang
- `b` (bottom)
Ausrichtung am Absatzende
- `c` (center)
Ausrichtung an der Absatzmitte

Für den Parameter `ipos` geben Sie dieselben Werte an. Er bezieht sich allerdings nicht auf die Ausrichtung der Box selbst, sondern auf die Positionierung des Textes in der Box. Daher gibt es für `ipos` noch eine vierte Variante `s`, mit der Sie den Text gleichmäßig über die komplette Höhe der Box verteilen können. Der Parameter `ipos` wirkt sich allerdings nur aus, wenn Sie über den Parameter `höhe` eine Boxenhöhe angeben, die größer als der Platzbedarf des Textes ist.

Text nebeneinandersetzen

Wenn Sie Text absatzweise nebeneinandersetzen wollen, verwenden Sie neben dem Befehl `\parbox` auch die `minipage`-Umgebung.

Der allgemeine Aufruf lautet:

```
\begin{minipage}[<Anordnung, t oder b>]{<breite>}
\end{minipage}
```

In der `minipage`-Umgebung sind Absätze, Fußnoten, Tabellen usw. erlaubt, jedoch können Sie keine Gleitobjekte wie Bilder erzeugen.

Die Fußnoten erscheinen standardmäßig am Ende der `minipage`-Umgebung und werden in Kleinbuchstaben gesetzt.

Die Höhe einer `minipage`-Umgebung richtet sich nach der Länge des Textes.

```
\begin{minipage}[t]{17mm}
  Ein Text in einer minipage Umgebung
\end{minipage}
```

Rule-Boxen

Eine Rule-Box ist nichts anderes als ein mit einer Farbe gefülltes Rechteck, also ein farbiger Balken. Damit lassen sich hervorragend Textpassagen voneinander trennen. Jedoch sollten Sie den Balken der Schrift anpassen, um nicht das Gesamterscheinungsbild der Seite unschön zu gestalten.

```
\rule[<shift>]{<breite>}{<höhe>}
```

Wenn Sie den optionalen Parameter `shift` weglassen, liegt die untere Kante auf der Grundlinie des umgebenden Textes. Wenn Sie für `shift` positive Werte eingeben, wird die Box nach oben, bei negativen Werten nach unten geschoben.

```
Nach oben verschobener Text \rule[3mm]{1cm}{1mm}
```

4.5.18 Farbe

Farbe ist ein wichtiges Gestaltungsmittel für anspruchsvolle Dokumente. Für die Gestaltung farbiger Dokumente gibt es verschiedene Befehle sowie zwei Pakete, die Sie ganz einfach mit `\usepackage` einbinden können.

Die Pakete `xcolor` und `color`

Das Paket `color` ist das grundlegende Paket für die Arbeit mit Farben. Für normale PDF-Dateien ist es ideal geeignet. Für andere Ausgabeformate oder andere Farbmodelle benutzt man das Paket `xcolor`, da dieses alle Farbmodelle unterstützt. Das Paket `xcolor` ist abwärtskompatibel zu `color`.

Befehle für Farbe

Um Text farbig darzustellen, gelten folgende Befehle:

- `\color{farbe}`
Stellt den Text bis zum Ende einer Gruppe farbig dar. Als Farbparameter lassen sich die englischen Bezeichnungen `red`, `blue`, `green`, `yellow` etc. einsetzen.

- `\textcolor{farbe}{text}`

Färbt den entsprechenden Text in einer bestimmten Farbe. Für die Farbparameter gilt dasselbe wie beim Befehl `\color`.

```
Benjamin Blümchen trägt eine \textcolor{blue}{blaue} Jacke
```

Mit dem Befehl `\color` können Sie auch eigene Farbwerte über die Angaben der RGB-Komponenten mischen:

```
\color[rgb]{0.8,0.4,0.2}
```

Zur Gestaltung der Hintergrundfarbe gilt der Befehl:

```
\pagecolor{<farbe>}
```

Diese Farbeinstellung ist nicht nur für die jeweils aktuelle Seite aktiv, sondern auch für die nachfolgenden Seiten, bis der Befehl erneut aufgerufen wird.

Farbmodelle

LaTeX unterstützt folgende Farbmodelle standardmäßig:

- `gray`
Damit können Sie einen Grauwert zwischen 0 und 1 festlegen. 0 steht für Schwarz, 1 für Weiß. Je höher die angegebene Zahl ist, umso heller ist der Grauwert.
- `rgb`
Das RGB-Modell. Hierbei werden die jeweiligen Grundfarben Rot, Grün und Blau miteinander gemischt. Auch hier können Sie wieder eine Angabe von 0 und 1 machen. Die drei Werte werden im Parameter durch Kommata voneinander getrennt.
- `Cmy` (nur mit `xcolor`)
Das CMY-Modell. Die Abkürzung CMY steht hier für Cyan, Magenta und Yellow, die in ihren jeweiligen Anteilen zwischen 0 und 1 subtraktiv gemischt werden.
- `Cmyk`
Ähnlich wie das CMY-Modell. Als weitere Farbe kommt hier Schwarz hinzu. Die Anteile werden wieder jeweils mit 0 und 1 subtraktiv gemischt.
- `HSB`
Hierbei werden die drei Komponenten Farbton, Sättigung und Helligkeit in ihren jeweiligen Anteilen zwischen 0 und 1 additiv gemischt.
- `Gray`
Hier können Sie einen Grauwert zwischen 0 und 15 angeben. 0 steht für Schwarz und 15 für Weiß. Je höher Sie also den Farbwert angeben, desto weißer wird der Farbton.

- RGB
Hier werden die Farbanteile Rot, Grün und Blau additiv miteinander gemischt. Für jede Farbe können Sie einen Wert zwischen 0 und 255 angeben. Die Werte werden jeweils im Parameter durch Komma getrennt.
- HSB
Hier werden die drei Komponenten Farbton, Sättigung und Helligkeit in ihren entsprechenden Anteilen zwischen 0 und 240 additiv gemischt.

Welches der Farbmodelle verwendet werden soll, hängt davon ab, ob es sich bei der Ausgabe um ein gedrucktes Dokument oder ein Online-Dokument handelt. Für gedruckte Dokumente können Sie jeweils die Modelle gray, Gray oder cmyk wegen der subtraktiven Farbmischung verwenden. Für die Darstellung am Bildschirm bieten sich die Modelle rgb, RGB, hsb oder HSB an.

Die Farbmodelle werden von verschiedenen Befehlen eingesetzt, zum Beispiel um selbst Farben zu definieren, wie im nächsten Abschnitt beschrieben wird.

Farben definieren

Mit dem Befehl `\definecolor` können Sie die Grundfarben neu definieren. Der Befehl `\definecolor` wird wie folgt eingesetzt:

```
\definecolor{meinefarbe}{rgb}{1,1,1}
```

Mit dem ersten Parameter `meinefarbe` legen Sie einen neuen Farbwert fest. Über den zweiten Parameter legen Sie das verwendete Farbmodell fest, in diesem Fall `rgb`. Mit dem dritten Parameter legen Sie die Werte für die Farbe fest. Somit können Sie über den Befehl `\definecolor` eigene Farben definieren und in Ihrem Dokument verwenden.

4.5.19 Mehrspaltiger Druck

Mit LaTeX lassen sich grundsätzlich Dokumente mit mehrspaltigem Text erzeugen. Hierzu dient der Klassenparameter `twocolumn` bei der Deklaration im Dokumentkopf.

```
\documentclass[10pt, twocolumn]{article}
```

Dann wird das gesamte Dokument zweispaltig gedruckt.

Wenn Sie nur einen Teil des Dokumentes zweispaltig drucken wollen, so erreichen Sie dies über die Befehle:

- `\twocolumn[<text>]`
Damit wird die laufende Seite beendet und eine neue, zweispaltige Seite begonnen. Der Text geht dabei über die gesamte Seitenbreite.
- `\onecolumn`
Damit wird die laufende zweispaltige Seite beendet und eine neue, einspaltige Seite gesetzt.

Das Beispiel zeigt den Unterschied, wenn der optionale Parameter bei

`twocolumn` verwendet wird:

```
\twocolumn
Thorr ist eine alte römische Siedlung, die an der Römerstraße liegt. Die
größte Sehenswürdigkeit ist der Römerturm.
```

Das Beispiel verwendet den Befehl `\twocolumn` mit einem Parameter:

```
\twocolumn[Hörkassette]
Thorr ist eine alte römische Siedlung, die an der Römerstraße liegt. Die
größte Sehenswürdigkeit ist der Römerturm.
```

Der Parameter des Befehls `\twocolumn` legt eine Spaltenüberschrift fest, die über beide Spalten geht. Sie können den Text so lang setzen, wie Sie wollen.

Mehrspaltige Seiten

Mit dem Paket `multicol` lassen sich ebenfalls mehrspaltige Seiten setzen. Das Paket `multicol` wird so eingebunden:

```
\usepackage{multicol}
```

Je höher die Spaltenzahl ist, d. h., je schmäler die einzelnen Spalten sind, umso schwieriger wird es für LaTeX, einen sauberen Umbruch zu erzeugen. Das kann mit der Umgebung `multicols` vermieden werden:

```
\begin{multicols}{<spaltenzahl>}[<titel>][<abstand>]
Der Löwe ist ein Wüstentier, das in Rudeln lebt. Der Leitlöwe führt das
Rudel an.
\end{multicols}
```

Abstand zwischen einzelnen Spalten

Mit dem Befehl `\columnsep` bestimmen Sie den Abstand zwischen den einzelnen Spalten.

Dazu muss vor dem Aufruf der `multicol`-Umgebung oder des `\twocolumn`-Befehls der Befehl `\columnsep` aufgerufen werden:

```
\setlength{\columnsep}{2cm}
\begin{multicols}{<spaltenzahl>}[<titel>][<abstand>]
Der Löwe ist ein Wüstentier, das in Rudeln lebt. Der Leitlöwe führt das
Rudel an.
\end{multicol}
```

Mit dem Befehl `\columnseprule` ist es möglich, das Aussehen einer mehrspaltigen Textpassage zu beeinflussen.

Dabei müssen Sie jedoch darauf achten, die Linien nicht zu breit zu wählen, um das Erscheinungsbild nicht negativ zu beeinflussen:

```
\setlength{\columnseprule}{1mm}
\begin{multicols}{<spaltenzahl>}[<titel>][<abstand>]
Der Löwe ist ein Wüstentier, das in Rudeln lebt. Der Leitlöwe führt das
Rudel an.
\end{multicol}
```

4.5.20 Standardseiten und Zeilennummern

Viele Berufsgruppen arbeiten nach Standardseiten und Zeilennummern. So arbeiten z. B. Übersetzer, Lektoren oder Schriftsetzer nach der sogenannten Normzeile oder Normseite. Solche Seiten können Sie ganz einfach mit dem Paket `stdpage` setzen. Durch die Angabe von optionalen Werten lässt sich die Zeilen- oder Zeichenzahl einstellen. Im folgenden Beispiel wird mit dem Parameter `linenumbers` die Zeilennummer eingelen-

```
\usepackage[chars 20, lines 10, noindent, linenumbers]{stdpage}

\begin{document}
Köln ist eine Stadt, die am Rhein liegt und viele Sehenswürdigkeiten hat.
Eine der berühmtesten Sehenswürdigkeiten ist der Kölner Dom.
\end{document}
```

Bei diesem Beispiel liefert der Parameter `linenumbers` nur die Zeilennummern für jede fünfte Zeile. Wenn Sie jedoch eine Nummer in jeder Zeile haben wollen, wählen Sie das Paket `lineno`, `linenox0`.

```
\usepackage{lineno}

\begin{document}
\linenumbers
Köln ist eine alte römische Stadt, die sich im Laufe der Jahrhunderte stark
verändert hat. Im Altertum war Köln die größte römische Festung in
Germanien.
\end{document}
```

4.5.21 Einheiten besonders berücksichtigen

In mathematischen und physikalischen Texten werden häufig verschiedene Einheiten für Formeln, Maße usw. benötigt. Mit den Ergänzungspaketen `units` und `nicefrac` geht so etwas sehr einfach:

```
\unit[zahl]{einheit}
\unitfrac[zahl]{zähler}{nenner}
\nicefrac[fontbefehl]{zähler}{nenner}
```

Mit `\unit` werden eine optionale Zahl und eine Einheit gesetzt. Als Zwischenraum wird hier ein Backslash `\` eingegeben. Das entspricht der Breite eines halben Leerzeichens. Außerdem wird an dieser Stelle ein Zeilenumbruch verhindert. Der Befehl wird wie folgt verwendet:

Dieser Satz enthält ein `\` Leerzeichen

Mit dem Befehl `\unitfrac` wird für die Einheit ein Bruch erzeugt, und der Befehl `\nicefrac` erlaubt es, besondere Fontbefehle zu verwenden, mit denen der Bruch formatiert wird.

Im folgenden Beispiel werden die Einheitenbefehle kurz vorgestellt. Außerdem wird das Paket `marvosym` eingebunden, mit dem weitere Symbole verwendet werden können:

```
\usepackage{units}
\usepackage{marvosym}

\begin{document}
Die Tür ist \unit[2]{m} hoch.\
Der Bus hat eine Geschwindigkeit von \unitfrac[180]{km}{h}
\end{document}
```

Das `units`-Paket erfüllt jedoch nicht alle Wünsche. So können z. B. Einheiten wie Quadratmeter oder Kubikmeter nicht so einfach dargestellt werden. Dafür existieren jedoch andere Lösungen.

Das Paket `Sistyle`

Wenn Sie in Ihren Dokumenten viele physikalische Einheiten wie z. B. die SI-Einheiten verwenden müssen, benutzen Sie das Paket `Sistyle`. Das Paket `Sistyle` wird wie folgt eingebunden:

```
\usepackage{sistyle}
```

Zu beachten ist, dass beim Einbinden des Pakets der Paketname kleingeschrieben wird.

Dieses Paket stellt die drei Befehle `\SI` für Zahlen mit Einheiten, `\num` für Zahlen und `\ang` für Winkelangaben zur Verfügung:

```
\usepackage{sistyle}

\begin{document}
Der Baum ist \SI{3}{m} hoch.

\end{document}
```

Tip: Die Winkelangaben können Sie sowohl in Grad als auch in Minuten und Sekunden angeben. Dazu müssen Sie jeweils die Komponenten durch ein Semikolon voneinander trennen.

Einheiten mit Brüchen

Ein weiteres wichtiges Einsatzgebiet von Maßeinheiten sind physikalische und mathematische Formeln. Wenn diese einen Bruch enthalten, so lassen sie sich in mathematischen Umgebungen auf verschiedene Art und Weise darstellen:

```
$v \SI{10}{m.s^{ 1}}$\\
$v \SI{10}{m/s}$\\
```

Weitere Befehle

Neben diesen herkömmlichen Befehlen gibt es noch weitere Befehle zur Darstellung von Einheiten.

So sind z. B. die Befehle `\angstrom`, `\micro`, `\ohm`, `\degC` usw. für verschiedene physikalische Maßeinheiten zuständig wie z. B. Angström, Ohm oder Grad Celsius.

5 Tabellen

In Kapitel 3 wurden bereits einige grundlegende Tabellenfunktionen von LaTeX vorgestellt. In diesem Kapitel werden Ihnen weiterführende Tabellenfunktionen von LaTeX gezeigt, z. B. Tabellen mit Rahmen oder Bildern auszustatten.

5.1 Grundlagen von Tabellen

Wie Sie schon gesehen haben, besteht eine LaTeX-Tabelle, die mit der `tabbing`-Umgebung erzeugt wird, aus normalen Tabulatorstopps. Diese können mit verschiedenen Befehlen gesteuert und angesprungen werden.

Die Breite eines Tabulators richtet sich dabei nach dem jeweiligen Betriebssystem. Sie können die Breite aber auch beliebig fest vorgeben, z. B. mit dem Befehl `\hspace`. Ein weiteres wichtiges Merkmal von Tabellen sind die Ränder, d. h. der linke und der rechte Rand. Der linke Rand wird bei LaTeX standardmäßig auf null gesetzt, d. h. auf den linken Rand der vorherigen aktiven Umgebung. In der `tabbing`-Umgebung ist es möglich, ihn mit dem Befehl `\+ tabulatorweise` zu versetzen.

Außerdem ist es in LaTeX möglich, Tabulatorstopps zu speichern. Dazu dienen die Befehle `\pushtab` und `\poptab`, die jedoch nur in der `tabbing`-Umgebung zur Verfügung stehen.

5.2 Exkurs: einfache LaTeX-Tabellen

In Kapitel 3 haben Sie bereits eine erste einfache Tabelle mit der `tabbing`-Umgebung erzeugt. Dabei wurden die wichtigsten Befehle für Spalten in Tabellen angewendet. Es gibt aber noch weitere Befehle bzw. Kommandos, die die `tabbing`-Umgebung bereitstellt. Die nachfolgende Tabelle zeigt eine Übersicht der wichtigsten Befehle der `tabbing`-Umgebung:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\</code>	Neue Spalte bzw. Tabulator erzeugen
<code>\></code>	Erzeugten Tabulator anspringen

Befehl	Beschreibung
<code>\\</code>	Ende der Zeile
<code>kill</code>	Musterzeile erzeugen
<code>\</code>	Tabulatorweise Verkleinerung des linken Einzugs
<code>\+</code>	Tabulatorweise Vergrößerung des linken Einzugs
<code>\'</code>	Versetzung des Textes links vor dem Tabulatorstopp nach links
<code>\‘</code>	Versetzung des Textes nach dem Tabulatorstopp nach rechts
<code>\pushtabs</code>	Speichern einer Tabulatorposition
<code>\poptabs</code>	Setzen einer Tabulatorposition

Tabelle 5.1: Befehle in der tabbing-Umgebung

In Kapitel 3 haben Sie einfache Beispiele zu den ersten drei Befehlen gesehen und erfahren, wie in LaTeX standardmäßig Tabellen erzeugt werden. Das folgende Beispiel zeigt, wie Sie mit dem Befehl `kill` eine Musterzeile erzeugen können:

```
Beispiel für eine Tabelle mit Musterzeile:
\begin{tabbing}
\hspace*{4cm} \ \hspace{2cm} \ \hspace{2cm} \ \kill
1. Anzahl Wörter \> Preis \> Gesamtpreis \\
4. 1000 \> 1 EUR \> 1000 EUR \\\
```

Beispiel 5.1: Tabelle mit Musterzeile

Das Ergebnis sieht so aus:

Beispiel für eine Tabelle mit Musterzeile:		
1. Anzahl Wörter	Preis	Gesamtpreis
4. 1000	1 EUR	1000 EUR

Abbildung 5.1: Tabelle mit Musterzeile

In diesem Beispiel muss die Sternvariante des Befehls `\hspace` benutzt werden, um den horizontalen Abstand am Anfang der ersten Zeile zu erzwingen, da LaTeX diese standardmäßig ignoriert.

Das nachfolgende Beispiel demonstriert den Befehl `\+`:

```
\begin{tabbing}
\hspace*{6cm} \ \hspace{2cm} \ \hspace{2cm} \ \ \+ \kill
1. Anzahl \> Preis \> Gesamtpreis \\
4. 1000 \> 1 EUR \> 1000 EUR \\
\end{tabbing}
```

Beispiel 5.2: Der Befehl `\+`

Das Ergebnis sieht so aus:

Linker Rand			
	1. Anzahl	Preis	Gesamtpreis
	4. 1000	1 EUR	1000 EUR

Abbildung 5.2: Ergebnis des Befehls `\+`

Hier wird die ganze Tabelle um eine Tabulatorposition nach rechts verschoben. Der Befehl `\-` bewirkt das Gegenteil.

Das folgende Beispiel zeigt, wie Sie mit dem Befehl `\'` den Text links vom Tabulator weiter nach links verschieben können:

```
\begin{tabbing}
Sie haben folgende Optionen: \ PC \+ \\
Internet \\
oder \'' Notebook \\
\end{tabbing}
```

Beispiel 5.3: Der Befehl `\'`

Das Ergebnis sieht dann so aus:

Sie haben folgende Optionen:	PC
	Internet
oder	Notebook

Abbildung 5.3: Beispiel für linksbündig formatierte Tabellen

Das Beispiel für rechtsbündig verschobene Tabellen:

```
\begin{document}
\begin{tabbing}
Sie haben folgende Optionen: \ PC \+ \\
Internet \\
oder \` Notebook \\
\end{tabbing}
\end{document}
```

Beispiel 5.4: Rechtsbündig formatierte Tabellen

Sie haben folgende Optionen: PC	
Internet	
oder	Notebook

Abbildung 5.4: Beispiel für rechtsbündig formatierte Tabellen

Sonstige Besonderheiten der Tabellenumgebung

Neben diesen grundlegenden Befehlen der Tabellenumgebung sind noch ein paar Besonderheiten zu beachten. So gibt es die Möglichkeit, Akzente und Zeilenumbrüche in der `tabbing`-Umgebung zu erzeugen. Dazu werden allerdings nicht die herkömmlichen LaTeX Befehle benutzt, sondern spezielle für Tabellen ausgelegte Befehle.

Um Akzente in einer Tabelle zu setzen, müssen Sie `\a`, `\a'` oder `\a`` benutzen. Mit dem Befehl `\l[10cm]` lässt sich ein manueller Seitenumbruch in einer Tabelle erzwingen. Die Standardbefehle `\newpage`, `\clearpage` oder `\pagebreak` stehen in Tabellen nicht zur Verfügung.

5.2.1 Rahmen für Tabellen

Neben der `tabbing`-Umgebung für Tabellen gibt es noch die `tabular`-Umgebung, die für Tabellenrahmen zuständig ist. Diese Umgebung wird genauso aufgerufen wie die `tabbing`-Umgebung.

```
\begin{tabular}[pos]{spaltenform}
\end{tabular}
```

Der Parameter `pos` gilt für die vertikale Position der Tabelle, der Parameter `spaltenform` für die Form der Spalten. Für `pos` existieren nur zwei Varianten:

Parameter	Beschreibung
<code>t</code>	Setzt die oberste Tabellenzeile auf die Umgebung
<code>b</code>	Setzt die unterste Tabellenzeile auf die Umgebung

Tabelle 5.2: Parameter für die Option `pos`

Für die Option `spaltenform` gibt es wesentlich mehr Parameter. Die Tabelle zeigt diese auf:

Parameter	Beschreibung
<code>l</code>	Setzt den Inhalt der Spalte linksbündig
<code>r</code>	Setzt den Inhalt der Spalte rechtsbündig
<code>c</code>	Setzt den Inhalt der Spalte mittig
<code>P{breite}</code>	Setzt den Text in Blocksatz mit einer Breite des Wertes <code>breite</code>
<code>*{num}{spaltenform}</code>	Formatiert Anzahl <code>num</code> mal <code>spaltenform</code> , z. B. <code>*{5}{3}</code> ist gleichbedeutend mit <code>rrrrr</code>
<code> </code>	Der Balken wird als Spaltenrand benutzt
<code> </code>	Eine Doppellinie zwischen den Spalten.
<code>@{text}</code>	Der Text wird je nach Anordnung rechts oder linksbündig gesetzt. Der Zwischenraum zwischen Rand und Spalteninhalt wird dabei entfernt.
<code>&</code>	Dieses Symbol wird als Spaltentrenner verwendet. Man muss jeweils einen weniger verwenden als die Anzahl der Spalten beträgt.
<code>\\</code> bzw. <code>\\[abstand]</code>	Damit wird eine Zeile beendet. Mit <code>abstand</code> können Sie zusätzlichen vertikalen Zwischenraum einfügen.
<code>\hline</code>	Fügt eine horizontale Linie unterhalb der Zeile ein. Darf nur nach dem Tabellenkopf oder einem <code>\\</code> verwendet werden.
<code>\cline{n m}</code>	Zeichnet eine horizontale Linie unterhalb der Zeile vom linken Rand der Spalte <code>n</code> bis zum rechten Rand der Spalte <code>m</code> . Darf nur nach <code>\\</code> verwendet werden.

Parameter	Beschreibung
<code>\multicolumn{num}{spaltenform}{text}</code>	Es werden in der aktuellen Zeile Spalten zu einer zusammengefasst. Die Zahl der betreffenden Spalten gibt der Parameter <code>num</code> an. Die Spaltenform der zusammengesetzten Spalte wird über den Parameter <code>spaltenform</code> gesetzt. Über den Parameter <code>text</code> wird der Inhalt der Spalte bestimmt.
<code>\vline</code>	Zeichnet einen vertikalen Strich in Höhe der Zeile an der Stelle des Auftretens.
<code>\tabularnewline[abstand]</code>	Es wird eine neue Zeile in der Spalte erzeugt. Kann als Alternative zu <code>\\</code> in der letzten Spalte eingesetzt werden. Mit dem Parameter <code>abstand</code> können Sie einen zusätzlichen vertikalen Abstand einfügen.

Tabelle 5.3: Parameter für die Option `spaltenform`

Hier ein einfaches Beispiel für eine Tabelle in der `tabular`-Umgebung:

```
%Beispiel für eine einfache Tabelle
\begin{tabular}{|c|c|}\hline
%Spalte zentriert über zwei Spalten
\multicolumn{2}{|c|}{Spalten}\\hline
Erste Spalte & Zweite Spalte\\
Erste Zeile & Zweite Zeile\\
\hline
\end{tabular}
```

Beispiel 5.5: Einfache Tabelle mit `tabular`

Spalten	
Erste Spalte	Zweite Spalte
Erste Zeile	Zweite Zeile

Abbildung 5.5: Einfache Tabelle mit der `tabular`-Umgebung

In diesem Beispiel wird eine einfache Tabelle mit zwei Spalten erzeugt, in der der Text linksbündig ausgerichtet ist. Mit `\hline` wird jeweils eine horizontale Linie gezeichnet.

Tipp: Es kann sein, dass bei der Verwendung von `tabular` Probleme bei der Erzeugung von PDF-Dateien auftreten, wenn Sie MikTeX unter Windows benutzen. Da die `tabular`-Umgebung allgemein etwas komplizierter ist als die `tabbing`-Umgebung, empfiehlt es sich hier, zunächst ein paar Optionen auszuprobieren und das Ergebnis zu begutachten.

Den Tabellenstil ändern

Neben den normalen Parametern der beiden Tabellenumgebungen gibt es die Möglichkeit, Einfluss auf den Tabellenstil außerhalb einer Tabellenumgebung zu nehmen. So können Sie z. B. in der Präambel über den Befehl `\setlength` den Stil der Tabelle verändern. Nachfolgende Tabelle gibt einen Überblick über die verschiedenen Optionen:

Parameter	Beschreibung
<code>\tabcolsep</code>	Legt die halbe Breite des Spaltenzwischenraums von benachbarten Spalten fest.
<code>\arrayrulewidth</code>	Legt die Dicke vertikaler und horizontaler Linien fest.
<code>\doublerulesep</code>	Legt den Abstand von Doppellinien fest.
<code>\arraystretch</code>	Legt den Streckungsfaktor für den Zeilenabstand fest. Wird mit dem normalen Zeilenabstand multipliziert. Standardwert ist hier 1.

Tabelle 5.4: Parameter für den Befehl `setlength`

```
%in der Präambel
\setlength{\tabcolsep}{2mm}
\setlength{\doublerulesep}{3pt}
\renewcommand{\arraystretch}{1.2}
%Hauptdokument
\begin{document}
%tabular Umgebung
\begin{tabular}
\end{tabular}
\end{document}
```

Beispiel 5.6: Ändern des Tabellenstils mit `\setlength`

Diese Befehle zeigen, wie Sie den Tabellenstil außerhalb der `tabular`-Umgebung mit dem Befehl `\setlength` ändern können. Dies wirkt sich unmittelbar auf die im Hauptdokument erzeugte `tabular`-Umgebung aus.

Erweiterte Tabellenumgebungen

Neben den Standard-Tabellenumgebungen `tabbing` und `tabular` gibt es noch das Paket `array`, mit dem Sie die Standardumgebungen erweitern können. Das Paket bietet viele nützliche Funktionen für Tabellen. Das `array`-Paket wird von einigen Ergänzungspaketen vorausgesetzt und sollte daher bei der Arbeit mit Tabellen immer eingebunden werden.

Die nachfolgende Tabelle gibt einen Überblick über die Befehle des Pakets `array`:

Parameter	Beschreibung
<code>m{breite}</code>	Die Spalten werden mittig gesetzt. Ist analog zum Parameter <code>p</code> .
<code>b{breite}</code>	Wie Parameter <code>p</code> , der Text wird allerdings unten gesetzt.
<code>>{Befehle}</code>	Kann vor den Spaltenparametern <code>l,r,c,p{}</code> , <code>m{}</code> oder <code>b{}</code> benutzt werden, um dort Befehle einzufügen, die dann für die jeweilige Spalte gelten.
<code><{Befehle}</code>	Analog zu <code>>{Befehle}</code>
<code>!{Befehle}</code>	Anstelle einer vertikalen Linie werden Befehle eingefügt. Anders als <code> </code> wird hier der normale Leerraum mit <code>@[]</code> zwischen den Spalten nicht unterdrückt.
<code>\extrarowheight</code>	Es wird ein zusätzlicher Längenwert zu der Höhe einer Tabellenzeile addiert. Kann zur optisch schöneren Gestaltung von Tabellen genutzt werden.
<code>\newcolumntype{type}{\param}{befehle}</code>	Damit wird ein neuer Spaltentyp definiert. Es können optionale Parameter angegeben werden. Damit soll Tipparbeit gespart werden.
<code>\showcols</code>	Damit werden alle Spaltentypen aufgelistet und auf der Konsole ausgegeben. Die Spaltentypen werden in die Logdatei geschrieben.
<code>\hline</code>	Zeichnet eine horizontale Linie. Die Kanten treffen dabei sauber aufeinander.
<code>\firsthline</code>	Damit wird die erste horizontale Linie in einer Tabelle besonders ausgezeichnet.
<code>\lasthline</code>	Die letzte horizontale Linie in einer Tabelle wird besonders ausgezeichnet.
<code>\extratabsurround</code>	Es wird ein zusätzlicher Rand um die Tabelle gezeichnet. Kann dann genutzt werden, wenn mehrere Tabellen geschachtelt werden.

Tabelle 5.5: Befehle des Pakets `array`

Beispiele für die Verwendung des Pakets array

Das Paket `array` können Sie wie andere Pakete auch mit `\usepackage` einbinden. Dann lassen sich die Befehle des Pakets entweder einzeln im Hauptdokument oder in der Tabellenumgebung `tabular` nutzen.

Hier ein Beispiel für den Befehl `\newcolumnntype`:

```
\documentclass[a4paper]{article}
\usepackage{array}
%neue Spaltentypen definieren
\newcolumnntype{N}{>{\small}l}
\newcolumnntype{V}[1]{>{\small\raggedright}p{#1}}
\begin{document}
%Beispiel für eine einfache Tabelle
\begin{tabular}{|l|l|}\hline
\multicolumn{2}{|l|}{Spalten}\\hline
Erste Spalte & Zweite Spalte\\
Erste Zeile & Zweite Zeile\\
\hline
\end{tabular}
\end{document}
```

Beispiel 5.7: Neue Spaltentypen definieren

Mit diesem Aufruf werden zwei neue Spaltentypen mit den Bezeichnungen `N` und `V` definiert. Der Befehl `\small` setzt den Text auf eine kleine Schrift und `\raggedright` formatiert den Befehl linksbündig. Über den Parameter `p` werden der originale Spaltentyp und die Spaltengröße als Parameter festgelegt. Sie müssen hier auf Groß- und Kleinschreibung achten. Das Ergebnis sieht so aus:

Spalten	
Erste Spalte	Zweite Spalte
Erste Zeile	Zweite Zeile

Abbildung 5.6: Tabellenstile ändern

Ergänzungstypen

Mit dem Befehl `\newcolumnntype` können Sie neue Spaltentypen mit einer bestimmten Spaltenbreite für links bzw. rechts ausgerichteten Text definieren:

```
\newcolumnntype{L}{1}{>\raggedright}p{#1}}
\newcolumnntype{R}{1}{>\raggedright}p{#1}}
```

Die Parameter `L` bzw. `R` stehen jeweils für links- bzw. rechtsausgerichtete Tabellen. Der Befehl `\raggedright` richtet den Text wie bereits gesehen linksbündig aus.

Tabellen mittig über mehrere Spalten ausrichten

Über die Angabe des Parameters `m` können Sie andere Spalten zu einer bestimmten Spalte mittig ausrichten. Dieser Parameter wird in der Definition der `tabular`-Umgebung angegeben:

```
%Beispiel für den Parameter m
\begin{tabular}{@{}m{2cm} r@{}}\hline
Diese Spalte bestimmt
die Zentrierung der
anderen Spalte & Zweite Spalte mittig ausgerichtet\\
\hline
\end{tabular}
```

Beispiel 5.8: Beispiel für mittig formatierte Spalten

Diese Spalte bestimmt die Zentrierung der anderen Spalte	Zweite Spalte mittig ausgerichtet
--	-----------------------------------

Abbildung 5.7: Beispiel für den Parameter `m`

Optisch ansprechende Tabellen mit dem Paket `booktabs`

Das Paket `booktabs` ist ein weiteres Paket, mit dem Sie Tabellen ansprechender gestalten können. Auch dieses Paket wird mit `\usepackage` eingebunden. Wenn Sie dieses Paket

einbinden, stehen Ihnen sieben Befehle zur Verfügung, die überwiegend die Gestaltung der horizontalen oder vertikalen Linien beeinflussen. Die Tabelle gibt einen Überblick über diese Befehle:

Befehl	Beschreibung
<code>\toprule</code>	Zeichnet eine horizontale Linie für den Tabellenkopf.
<code>\midrule</code>	Zeichnet eine horizontale Linie und legt einen größeren Zeilenabstand fest.
<code>\bottomrule</code>	Zeichnet eine horizontale Linie für den Tabellenfuß und legt einen größeren Zeilenabstand fest.
<code>\cmidrule{trim}{n m}</code>	Ähnlich wie <code>\cline</code> . Die Linie wird allerdings verkürzt. Über den <code>trim</code> -Parameter können Sie angeben, auf welcher Seite verkürzt werden soll.
<code>\addlinespace[länge]</code>	Fügt einen zusätzlichen Zeilenabstand hinzu. Muss nach <code>\</code> verwendet werden.
<code>\morecmidrules</code>	Damit können Doppellinien über den Befehl <code>\cmidline</code> gezeichnet werden.
<code>\specialrule{dicke}{abstand davor}{abstand danach}</code>	Zeichnet eine dicke Linie. Sie können den Abstand vor und nach der Linie angeben.

Tabelle 5.6: Befehle des Pakets `booktabs`

Hier ein Beispiel für die Befehle `\toprule` und `\bottomrule`:

```
%Beispiel für \toprule und \bottomrule
\begin{tabular}{|l|l|}\toprule\toprule
Diese Spalte bestimmt
die Zentrierung der
anderen Spalte & Zweite Spalte mittig ausgerichtet\\
\bottomrule\bottomrule
\end{tabular}
```

Beispiel 5.9: Die Befehle `\toprule` und `\bottomrule`

Diese Spalte bestimmt die Zentrierung der anderen Spalte	Zweite Spalte mittig ausgerichtet
--	-----------------------------------

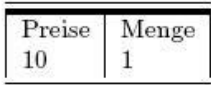
Abbildung 5.8: Beispiel für das Paket `booktabs`

In diesem Beispiel werden die Befehle `\toprule` und `\bottomrule` zweimal aufgerufen, um jeweils zwei horizontale Linien oben und unten zu zeichnen. Das Ergebnis zeigt die Abbildung oben.

Das folgende Beispiel zeigt, wie Sie den Befehl `\specialrule` einsetzen können:

```
\begin{tabular}{|l|l|}\toprule
\specialrule{2pt}{2pt}{1pt}
Preise & Menge\\
10 & 1\\
\bottomrule
\end{tabular}
```

Beispiel 5.10: Der Befehl `\specialrule`



Preise	Menge
10	1

Abbildung 5.9: Ergebnis des Befehls `\specialrule`

Das folgende Beispiel zeigt noch die Verwendung des Befehls `\morecmidrules`:

```
\begin{tabular}{|l|l|l|}\toprule
\multicolumn{2}{c}{Preisübersicht}\\
\cmidrule(r){1 2}\morecmidrules\cmidrule(r){1 2}
Preise & Menge & Gesamt\\
10 DM & 1 & 15 DM\\
\bottomrule
\end{tabular}
```

Beispiel 5.11: Der Befehl `\morecmidrules`

Um mit `\morecmidrules` weitere Linien per `\cmidrule` zeichnen zu können, müssen Sie zunächst den Befehl `\morecmidrules` aufrufen und anschließend noch mal den Befehl `\cmidrule`. Das Ergebnis zeigt Abbildung 5.10:

Preisübersicht		
Preise	Menge	Gesamt
10 DM	1	15 DM

Abbildung 5.10: Beispiel für den Befehl `\morecmidrules`

Mit dem Befehl `\cmidrule` können Sie, wie gesehen, weitere Effekte realisieren, z. B. eine Spaltenzuordnung mit gekürzter Linie. Die Spaltenzuordnung erfolgt über den zweiten Parameter, so können Sie z. B. für die Spalte 1 die Zuordnung `\cmidrule(r){1 2}` angeben. Die Linie verläuft dann von Spalte 1 bis Spalte 2.

Spalten am Dezimaltrenner ausrichten

Es ist möglich, Spalten von Tabellen nach dem Dezimaltrenner auszurichten. Dazu gibt es das Paket `dcolumn`. Dieses Paket wird, wie andere Pakete auch, über `\usepackage` eingebunden. Wenn Sie dieses Paket verwenden, können Sie beliebige Tabellenspalten nach dem jeweiligen Dezimaltrenner, also z. B. Komma oder Punkt, ausrichten. Um das Paket `dcolumn` benutzen zu können, muss das Paket `array` eingebunden werden.

Das Standardformat für den Dezimaltrenner ist:

```
D{Eingabe trenner}{Ausgabe trenner}{Kommastellen}
```

Der Eingabetrenner ist ein beliebiges Zeichen, das als Dezimaltrenner verwendet wird, etwa der Dezimalpunkt oder das Dezimalkomma. Der Ausgabetrener wird anstelle des Eingabetrenners gedruckt. Um den Ausgabetrener zu drucken, wechselt LaTeX in den Mathematikmodus.

Über das Paket `dcolumn` lässt sich ein neuer Dezimaltrenner definieren, und zwar über den Befehl `\newcolumnmtype`:

```
\newcolumnmtype{d}{D{.}}{\cdot}{#1}}
```

Hier ein Beispiel für `dcolumn`:

```
\begin{tabular}{@{}lD{,}{,}{ 1}@{}}\toprule
\multicolumn{2}{c}{Preisübersicht}\ \cmidrule(r){1 2}
Preise & Mwst & Gesamt\ \
\cmidrule(r){1 1}\cmidrule(r){2 2}\cmidrule(r){3 3}
\multicolumn{1}{l@{}}{Preise in DM}
10 DM & 19 & 11.90\ \
20 DM & 19 & 22.80\ \
\bottomrule
\end{tabular}
```

Beispiel 5.12: Dezimaltrenner mit `dcolumn`

Weitere Formatierungsmöglichkeiten für Dezimaltrenner

Neben dem Standarddezimaltrenner gibt es noch weitere Formatierungsmöglichkeiten, um Spalten nach dem Dezimaltrenner auszurichten. So definiert der Befehl `\newcolumnmtype` einen Spaltentyp, der sich erst am Dezimaltrenner und dann linksbündig ausrichtet.

Dieser Befehl wird zwischen die beiden Befehle `\makeatletter` und `\makeatother` eingebettet:

```
\makeatletter
\newcolumnmtype{d}[1]{>\DC@{,}{,}{#1}}1<\D@end}}
\makeatother
```

Zahlen runden

Mit dem Paket `rrcol` werden Zahlen, die in Spalten auftreten, automatisch auf eine bestimmte Anzahl von Stellen gerundet. Dieses Paket baut auf dem Ergänzungspaket `fltpoint` auf, das einfache Operationen für Fließkommazahlen bereitstellt.

Für das Paket `rrccol` gelten folgende optionale Parameter:

<i>Parameter</i>	<i>Beschreibung</i>
<code>rounding</code>	Aktiviert das Runden für Zahlen
<code>norounding</code>	Deaktiviert das Runden für Zahlen
<code>comma</code> bzw. <code>german</code>	Das Komma wird als Dezimaltrenner verwendet
<code>Point</code> bzw. <code>English</code> bzw. <code>USEnglish</code>	Der Punkt wird als Dezimaltrenner verwendet

Tabelle 5.7: Parameter für das Paket `rrccol`

Das Paket `rrccol` enthält folgende Befehle:

<i>Befehl</i>	<i>Beschreibung</i>
<code>\rcRoundingtrue</code>	Aktiviert das Runden für Zahlen
<code>\rcRoundingfalse</code>	Deaktiviert das Runden für Zahlen
<code>\rcDecimalSign</code>	Legt den Dezimaltrenner für die Ein- und Ausgabe fest
<code>\rcDecimalSignInput</code>	Legt den Eingabetrenner fest
<code>\rcDecimalSignOutput</code>	Legt den Ausgabetrener fest

Tabelle 5.8: Befehle des Pakets `rrccol`

Zeilen zusammenfassen mit `multirow`

Es gibt zwei Möglichkeiten, Zeilen und Spalten zusammenzufassen. Mit dem Befehl `\multicolumn` ist es möglich, mehrere Spalten zu einer zusammenzulegen und mit dem Paket `\multirow` lassen sich mehrere Zeilen zusammenfassen:

```
\multirow{nrows}[bigstruts]{width}{text}
```

Die Tabelle erklärt die einzelnen Parameter:

<i>Parameter</i>	<i>Beschreibung</i>
<code>nrows</code>	Anzahl der Zeilen, über die sich die Tabelle erstreckt.
<code>bigstruts</code>	Gibt die Anzahl der <code>bigstrut</code> -Bereiche an, die verwendet werden. Dazu wird das Paket <code>bigstrut</code> benötigt.

Parameter	Beschreibung
width	Gibt die Breite des Textes an. Mit »*« wird die Breite entsprechend dem Text gesetzt. Es wird kein Zeilenumbruch durchgeführt.
text	Solange Sie kein »*« verwenden, kann der Text beliebig mit \\ umbrochen werden.
fixup	Verschiebt den Text vertikal, ändert aber die Größe nicht.

Tabelle 5.9: Parameter des Befehls \multirow

Hier ein Beispiel für den Befehl \multirow. Darin werden eine Spaltenbreite von 2 cm sowie vier Zeilen und die Überschrift *Preise* festgelegt.

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{multirow}
\usepackage{booktabs}
\begin{document}
\begin{tabular}{|c|c|}\hline
\multirow{4}{2cm}{Preise} & Preis 1\\
& & Preis 2\\
& & Preis 3\\
& & Preis 4\\
\hline
\end{tabular}
\end{document}
```

Beispiel 5.13: Beispiel für den Befehl \multirow

Das Ergebnis sieht so aus:

Preise	Preis 1
	Preis 2
	Preis 3
	Preis 4

Abbildung 5.11: Beispiel für den Befehl \multirow

Überschriften drehen

Es ist möglich, Spaltentexte in Tabellen zu drehen. Dazu gibt es das Paket `rotating`. Um eine Tabelle zu drehen, verwenden Sie den Befehl `\newcolumnntype`:

```
\newcolumnntype{v}[1]{%
>{\begin{turn}{90}\begin{minipage}{#1}\raggedright\hspace{0pt}}%<{\end{minipage}\end{turn}}%
}
```

Der Text in der Spalte wird dabei mit der Umgebung `turn` um 90 Grad gedreht. Die Umgebung `turn` wird so aufgerufen:

```
\begin{turn}{Winkel}
\end{turn}
```

Damit lassen sich verschiedene Spalten um einen beliebigen Winkel drehen. Die Spalten erzeugen Sie dann mit `\multicolumn` und legen mit `\newcolumnntype` den Drehwinkel wie oben beschrieben fest. Hier ein Beispiel für eine um 90 Grad gedrehte Schulnoten-tabelle der Klasse 8a:

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{array}
\usepackage{rotating}
\usepackage{booktabs}
\newcolumnntype{v}[1]{
>{\begin{turn}{ 90}\begin{minipage}{#1}\raggedright\hspace{0pt}}1
<{\end{minipage}\end{turn}}%
}
\begin{document}
\begin{tabular}{@{}l*{7}c@{}}\toprule
\multicolumn{8}{@{}c@{}}{Notenspiegel: Klasse 8a}\\midrule
& \multicolumn{1}{v{6em}}{Deutsch}
& \multicolumn{1}{v{6em}}{Mathematik}
& \multicolumn{1}{v{6em}}{Englisch}
& \multicolumn{1}{v{6em}}{Französisch}
& \multicolumn{1}{v{6em}}{Sport}
& \multicolumn{1}{v{6em}}{Geschichte}
& \multicolumn{1}{v{6em}}{Erdkunde}\\midrule
Dieter, Thomas & 1 & 2 & 4 & 3 & 2 & 1\\
\dots\\
\bottomrule
\end{tabular}
\end{document}
```

Beispiel 5.14: Gedrehte Tabelle

Notenspiegel: Klasse 8a							
	Deutsch	Mathematik	Englisch	Französisch	Sport	Geschichte	Erkunde
Dieter, Thomas	1	2	4	3	2	1	
...							

Abbildung 5.12: Gedrehte Tabelle

Tipp: Mit dem Paket `rotating` lassen sich sowohl Tabelleninhalte als auch ganze Bilder usw. drehen.

Mit `\newcommand` lässt sich auch der Befehl `\multicolumn` umdefinieren bzw. es lassen sich Abkürzungen dafür festlegen:

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{array}
\usepackage{rotating}
\usepackage{booktabs}
\newcolumntype{v}[1]{
>\begin{turn}{ 90}\begin{minipage}{#1}\raggedright\hspace{0pt}}1
<\end{minipage}\end{turn}}%
}
\newcommand{\mch}[2]{\multicolumn{1}{#1}{#2}}
\begin{document}
\begin{tabular}{@{}l*{7}c@{}}\toprule
\multicolumn{8}{@{}c@{}}{Notenspiegel: Klasse 8a}\\midrule
& \mch{v{6em}}{Deutsch}
& \mch{v{6em}}{Mathematik}
& \mch{v{6em}}{Englisch}
& \mch{v{6em}}{Französisch}
& \mch{v{6em}}{Sport}
& \mch{v{6em}}{Geschichte}
& \mch{v{6em}}@{}{Erkunde}\\midrule
```

```
Dieter, Thomas & 1 & 2 & 4 & 3 & 2 & 1\\
\dots\\
\bottomrule
\end{tabular}
\end{document}
```

Beispiel 5.15: Neudefinition des Befehls `\multicolumn`

Das Ergebnis ist mit dem in Abbildung 5.12 identisch.

Wenn Sie noch mehr Freiheit haben wollen, können Sie den Winkel für die Umgebung `turn` auch noch als freien Parameter definieren. Unten sehen Sie ein Beispiel dazu:

```
\documentclass[a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{array}
\usepackage{rotating}
\usepackage{booktabs}
\newcolumnntype{v}[2]{
>{\begin{turn}{#1}\begin{minipage}{#2}\raggedright\hspace{0pt}}1<
{\end{minipage}\end{turn}}%
}
\newcommand{\mch}[2]{\multicolumn{1}{#1}{#2}}
\begin{document}
\begin{tabular}{@{}l*{7}c@{}}\toprule
\multicolumn{8}{@{}c@{}}{Notenspiegel: Klasse 8a}\\midrule
& \multicolumn{1}{v{60}{6em}}{Deutsch}
& \multicolumn{1}{v{60}{6em}}{Mathematik}
& \multicolumn{1}{v{60}{6em}}{Englisch}
& \multicolumn{1}{v{60}{6em}}{Französisch}
& \multicolumn{1}{v{60}{6em}}{Sport}
& \multicolumn{1}{v{60}{6em}}{Geschichte}
& \mch{v{60}{6em}@{}}{Erdkunde}\\midrule
Dieter, Thomas & 1 & 2 & 4 & 3 & 2 & 1\\
\dots\\
\bottomrule
\end{tabular}
\end{document}
```

Beispiel 5.16: Frei definierter Parameter für den Winkel

Das Ergebnis sieht dann so aus:

Notenspiegel: Klasse 8a						
	Deutsch	Mathematik	Englisch	Französisch	Sport	Geschichte
Dieter, Thomas	1	2	4	3	2	1
...						

Abbildung 5.13: Gedrehte Tabelle mit frei definiertem Winkel

Linien für Tabellen

Neben den oben gezeigten Linienstilen für Tabellen bietet das Paket `hhline` noch weitere Linienformen, mit denen Sie Tabellen ausschmücken können. Das `hhline`-Paket erstellt weitere Linien:

```
\begin{tabular}{|c|c|c|}
\hhline{|ttt|}\n
Erste Zeile\n
\hhline{|bbb|}\n
Zweite Zeile
\end{tabular}
```

Beispiel 5.17: Beispiel für den Befehl `\hhline`

Der Befehl `\hhline` hat verschiedene Parameter, mit dem man unterschiedlich geformte Linien zeichnen kann. Die Tabelle gibt einen Überblick über die Parameter des Befehls `\hhline`:

Parameter	Beschreibung
t	Es wird die obere Hälfte eines Doppellinien-Segmentes gezeichnet.
b	Es wird die untere Hälfte eines Doppellinien-Segmentes gezeichnet.
	Es wird eine Doppellinie gezeichnet.
	Es wird eine einfache horizontale Linie gezeichnet.

Tabelle 5.10: Parameter für den Befehl `\hhline`

Das Paket `arydshln`

Das Paket `arydshln` ist für vertikale und horizontale Linien zuständig. Das Paket `arydshln` muss nach dem Paket `array` eingebunden werden.

- `\hdashline[<strichlänge/lücke>]`
Der Befehl `\hdashline` erzeugt eine horizontale Strichlinie, bei der Sie optional Strichlänge und Abstand zwischen den Strichen eingeben können. Die Voreinstellung ist eine Länge von 4 pt für Strichlänge und Lücke.
- `\cdashline{<spalten>}[<strichlänge/lücke>]`
Der Befehl `\cdashline` erzeugt eine horizontale Strichlinie über die angegebenen Spalten, bei der ebenfalls optional eine Strichlänge und eine Lücke zwischen den Strichen angegeben werden.
- `\firstdashline[<strichlänge/lücke>]`
Damit wird eine horizontale Strichlinie zu Beginn der Tabelle erzeugt, bei der Sie ebenfalls optional eine Strichlänge und eine Lücke zwischen den Strichen eingeben können.
- `\lastdashline[<strichlänge/lücke>]`
Dieser Befehl erzeugt eine horizontale Strichlänge am Ende der Tabelle, bei der ebenfalls optional eine Strichlänge und eine Lücke zwischen den Strichen angegeben werden können.

Das Paket `arydshln` führt einen neuen Spaltentrenner `»:«` ein, den Sie in der Definition der Tabelle verwenden können.

Tipp: Weitere Informationen zu dem Paket finden Sie in der LaTeX-Dokumentation unter:

www.ctan.org/tex_archive/macros/LaTeX/contrib/arydshln/arydshln_man.pdf

Das Paket `colortbl`

Das Paket `colortbl` ist ein Ergänzungspaket, mit dem sich Tabellen farbig gestalten lassen.

Hier ein Beispiel; mit diesem Befehl können Sie eine Spalte farbig gestalten:

```
\columncolor[<farbmodell>][<farbe>][<linker rand>][<rechter rand>]
```

Der erste Parameter gibt dabei das Farbmodell an, der zweite Parameter die Farbe und die anderen beiden Parameter den jeweils linken und rechten Rand.

Mit dem Befehl `\newcolumntype` definieren Sie eigene farbige Spaltentypen:

```
\newcolumntype{g}{>{\columncolor{gray}{0.8}}
```

Tipp: Weitere Informationen zum Paket `colortbl` finden Sie in der Dokumentation unter

<http://www.ctan.org/tex-archive/macros/latex/contrib/colortbl/colortbl.pdf>

5.2.2 Tabellen mit fester Gesamtbreite

In der `tabular*`-Umgebung für Tabellen lässt sich die Breite einer Tabelle fest vorgeben. Die Breite einer Tabelle wird dabei direkt bei der Definition der Tabelle vorgegeben:

```
\begin{tabular*}{<breite>}{<pos>}{<spaltenform>}
\end{tabular*}
```

Das Ergänzungspaket `tabularx`

Mit dem Ergänzungspaket `tabularx` wird eine Tabellenumgebung wie `tabular` zur Verfügung gestellt, die einen zusätzlichen Spaltentyp namens `X` definiert, mit dem sich die Breite der Tabelle voll ausnutzen lässt.

Der allgemeine Aufruf lautet:

```
\begin{tabularx}{<breite>}{<spaltenform>}
\end{tabularx}
```

Das Paket `tabularx` enthält folgende Befehle:

- `\tracingtabularx`
Hiermit lassen sich Informationen aus der Logdatei auf die Konsole schreiben und in den entsprechenden Spaltenbreiten ausgeben.
- `\tabularxcolumn`
Der Spaltentyp `X` wird standardmäßig in `p{<breite>}` gesetzt. Wenn Sie anstelle von `p` den Spaltentyp `m` verwenden wollen, so lautet der Aufruf:

```
\renewcommand{\tabularxcolumn}[1]{m#1}
```

Als Parameter wird dabei die berechnete Breite der Spalte übergeben.

Tipp: Weitere Informationen zum Paket finden Sie unter

<http://tug.ctan.org/pkg/tabularx>

5.2.3 Tabellen über mehrere Seiten

Mit dem Ergänzungspaket `longtable` stellen Sie Text über mehrere Seiten dar. Das Ergänzungspaket enthält die Umgebung `longtable`, die standardmäßig wie folgt aufgerufen wird:

```
\begin{longtable}[<pos>]{<spaltenform>}
\end{longtable}
```

Die Umgebung `longtable` kann mit verschiedenen Befehlen kombiniert werden, um Zeilenumbrüche mit `\\`, Beschriftungen mit `\caption` oder Seitenumbrüche mit `\pagebreak` einzufügen.

Tipp: Weitere Informationen erhalten Sie in der Paketdokumentation unter <http://ftp.uni-erlangen.de/mirrors/CTAN/macros/LaTeX/required/tools/longtable.pdf>

Spaltentyp X für die `longtable`-Umgebung

Das Ergänzungspaket `LTxtable` ist eine Kombination von `longtable` und `tabularx`. Damit ist es möglich, den Spaltentyp X wie bei der `tabular`-Umgebung zu verwenden.

Der allgemeine Aufruf der Umgebung lautet:

```
\LTxtable{<breite>}{<datei>}
```

So wird die neue Tabelle definiert:

```
\LTxtable{12cm}[bsp_01.inc]
```

Tipp: Weitere Informationen finden Sie in der Paketdokumentation unter <http://www.tex.ac.uk/tex-archive/macros/LaTeX/contrib/carlisle/ltxtable.pdf>

Die `supertabular`-Umgebung

Die `supertabular`-Umgebung ist eine weitere Tabellenumgebung, die die Standardumgebung `table` erweitert. Das Ergänzungspaket `supertabular` bietet außerdem weitere Befehle, die die Umgebung beeinflussen.

Der allgemeine Aufruf lautet:

```
\begin{supertabular}{<pos>}{<spaltenform>}
\end{supertabular}
```

Es stehen folgende Befehle zur Verfügung:

- `\tablefirsthead`
Hiermit wird der Tabellenkopf für die erste Seite festgelegt.
- `\tablehead`
Damit wird der Tabellenkopf für alle übrigen Seiten festgelegt.
- `\tablelasttail`
Damit wird der Tabellenfuß für die letzte Seite festgelegt.
- `\topcaption`
Damit wird die Tabellenbeschriftung über der Tabelle festgelegt.
- `\bottomcaption`
Damit wird die Beschriftung unterhalb der Tabelle festgelegt.
- `\tablecaption`
Damit wird die Tabellenbeschriftung an der Standardstelle festgelegt, d. h. über der Tabelle.
- `\shrinkedheight`
Legt die Höhe fest, um die Tabelle kleiner bzw. größer zu setzen.

5.2.4 Gleitende Tabellen und Abbildungen beeinflussen

Tabellen und Abbildungen können in gleitende Umgebungen wie `table` (für Tabellen) und `figure` (für Bilder) eingebettet werden. Dies erlaubt es, den zur Verfügung stehenden Platz optimal zu nutzen und störende Leerräume zu vermeiden.

Eine solche Umgebung bzw. Abbildung wird mit dem Aufruf

```
\begin{table}  
\end{table}  
% bzw.  
\begin{figure}  
\end{figure}
```

definiert. Mit den folgenden Einstellungen und Befehlen können Sie diese Umgebungen beeinflussen:

- `topnumber`
Damit wird die maximale Anzahl von Gleitobjekten bestimmt, die oben auf der Seite dargestellt werden.

```
\setcounter{topnumber}{2}
```

- `bottomnumber`
Die maximale Anzahl von Gleitobjekten, die unten auf der Seite angeordnet werden können.

```
\setcounter{bottomnumber}{2}
```
- `totalnumber`
Die maximale Anzahl von Gleitobjekten, die gesamt auf der Seite platziert werden.

```
\setcounter{totalnumber}{2}
```
- `dbltopnumber`
Wie `topnumber`, jedoch für zweispaltigen Text.

```
\setcounter{dbltopnumber}{2}
```
- `\topfraction`
Damit wird der Bruchteil der Seite angegeben, bis zu dem gleitende Objekte oben angeordnet werden können.

```
\renewcommand{\topfraction}{.80}
```
- `\bottomfraction`
Damit wird der Bruchteil der Seite angegeben, bis zu dem gleitende Objekte unten angegeben werden können.

```
\renewcommand{\bottomfraction}{.80}
```
- `\textfraction`
Gibt den Bruchteil der Seite an, der mindestens für einen Text zur Verfügung steht.

```
\renewcommand{\textfraction}{.80}
```
- `\floatpagefraction`
Gibt den maximalen Bruchteil einer Seite an, der erreicht werden muss, bevor evtl. eine weitere Seite verwendet wird.

```
\renewcommand{\floatpagefraction}{.80}
```
- `\dbltopfraction`
Wie `topfraction`, jedoch für zweispaltigen Text.
- `\dblfloatfraction`
Wie `floatpagefraction`, jedoch für zweispaltigen Text.
- `\floatsep`
Gibt den vertikalen Abstand zwischen zwei Gleitobjekten an.
- `\textfloatsep`
Damit wird der vertikale Abstand zwischen den Gleitobjekten angegeben.

- `\intextsep`
Damit wird der vertikale Abstand zwischen den umgebenden Text und Gleitobjekten angegeben, die mit der Position angeordnet worden sind.
- `\dblfloatsep`
Wie `floatsep`, jedoch für zweispaltigen Text.
- `\dbltextfloatsep`
Wie `textfloatsep`, jedoch für zweispaltigen Text.
- `\topfigrule`
Damit wird ein horizontaler Balken vor dem Gleitobjekt erzeugt, um die Abgrenzung zu verdeutlichen:

```
\renewcommand{\topfigrule}{%
\vspace{ 2pt}%
\rule{\columnwidth}{0.5pt}%
\vspace{2..5pt}%
}
```

Dadurch wird eine horizontale Linie (mit Dicke 0.5 pt) über die gesamte Breite der Tabelle ausgezeichnet. Außerdem wird entsprechender Leerraum eingefügt.

- `\botfigrule`
Erzeugt einen horizontalen Balken nach dem Gleitobjekt zur horizontalen Abgrenzung. Dieser Befehl ist standardmäßig leer.
- `\dblfigrule`
Wie `topfigrule`, jedoch für die gesamte Breite des zweispaltigen Textes.

Diese Einstellungen lassen sich alle mit dem Befehl `\renewcommand` umdefinieren. Wenn Sie diese Einstellungen zu Beginn eines Dokumentes verwenden, gelten sie global für das gesamte Dokument.

Gleitobjekte auf einer Seite verhindern

Wenn Sie verhindern wollen, dass bestimmte Gleitobjekte auf einer Seite erscheinen, wählen Sie den Befehl `\suppressfloats`.

Der allgemeine Aufruf lautet:

```
\suppressfloats[<pos>]
```

Mit dem Parameter `<pos>` wird die Position – oben [t] oder unten [b] oder auf der Seite [p] – des Gleitobjektes festgelegt, das unterdrückt werden soll. Wenn Sie den Parameter nicht angeben, wird jeweils das folgende Objekt unterdrückt. Ein gleitendes Objekt, das vor dem Befehl auftritt, wird dabei nicht berücksichtigt. Das bedeutet, dass trotzdem ein gleitendes Objekt auftreten kann. In diesem Fall müssen Sie den `\suppress`-Befehl vor dem gleitenden Objekt aufrufen.

Mit dem Ergänzungspaket `flafter` wird sichergestellt, dass prinzipiell kein Gleitobjekt mehr vor der Stelle erscheint, an der es definiert worden ist. Das Paket wird ohne Parameter wie folgt angegeben:

```
\usepackage{flafter}
```

Über- und Unterschriften

Mit dem Befehl `\caption` werden Überschriften für Gleitobjekte festgelegt:

```
\caption[<kurzform>]{<Überschrift>}
```

Damit können Sie dann Überschriften z. B. für eine Tabelle oder eine Abbildung bestimmen. Der Text wird außerdem in das Tabellen- bzw. Abbildungsverzeichnis aufgenommen, welches getrennt ausgegeben werden kann. Wenn Sie `<kurzform>` angeben, wird dieser Text anstelle der Überschrift in das Tabellen- bzw. Abbildungsverzeichnis aufgenommen. Die Überschrift wird automatisch nummeriert, sodass Sie darauf verweisen können. Das Verweislabel wird mit dem Befehl `\label` erzeugt.

Tipp: Bei dem Befehl `\label` sollten Sie zur besseren Kennzeichnung vor den eigentlichen Text ein Kürzel setzen, um besser unterscheiden zu können, ob der Verweis auf eine Tabelle, ein Bild oder ein Kapitel zeigt.

Überschrift mit KOMA-Script ändern

Das Format der Über- bzw. Unterschrift lässt sich auf verschiedene Weise ändern. Mit dem Paket KOMA-Script können Sie u. a. die Befehle `\captionbelow[kurzform]{unterschrift}` unter das Gleitobjekt schreiben. Mit dem Befehl `\captionabove[kurzform]{ueberschrift}` können Sie eine Überschrift über dem Gleitobjekt erzeugen. Der Befehl `\caption` fügt hier ebenfalls eine normale Überschrift ein.

Beschriftung neben der Tabelle

Mit der Umgebung

```
\begin{captionbeside}[<kurzform>]{<beschriftung>}[<pos>][<breite>][<offset>]  
\end{captionbeside}
```

wird die Überschrift neben das Gleitobjekt gesetzt. Der Parameter `kurzform` gibt dabei den Text für das Tabellen- bzw. Abbildungsverzeichnis wieder. Der Parameter `beschriftung` gibt den Text an, der über das Gleitobjekt gesetzt wird. Mit dem Parameter `pos` können Sie die Position des Gleitobjektes angeben. Der Parameter `breite` gibt die Gesamtbreite einschließlich Beschriftung an. Mit dem Parameter `offset` setzen Sie einen zusätzlichen Rand.

Tipp: Weitere Informationen über das Paket KOMA-Script finden Sie auf der Homepage des Projektes unter <http://www.komascript.de/>

6 Bilder und gleitende Objekte

In Kapitel 3 haben Sie erste Grundlagen zum Thema Bilder kennengelernt. Sie haben dort u. a. mit dem Befehl `\includegraphics` Bilder in ein Dokument eingebunden. In diesem Kapitel werden Ihnen weitere Befehle für die Einbindung von Bildern in Dokumenten gezeigt.

6.1 Bildformate

LaTeX unterstützt standardmäßig nahezu jedes Bilddatenformat mit einem jeweils eigenen Treiber. Was für LaTeX gilt, trifft allerdings nicht auf die von LaTeX erzeugten Dokumenttypen PDF und DVI zu. So können Sie bei PDF zwar JPG-Bilder einbinden, nicht jedoch GIF-Dateien.

Grundsätzlich ist ein Bild in LaTeX eine Box, die verschiedene Eigenschaften bzw. Parameter hat, unter anderem Breite, Höhe sowie eine Bounding-Box. Eine Bounding-Box ist eine Art Rahmen, in den das Bild eingebettet wird.

Pakete und Treiber für Bilder

Für die Arbeit mit Bildern unter LaTeX gibt es die Pakete `color`, `graphics` und `graphicx`. Diese Pakete arbeiten standardmäßig mit verschiedenen Treibern und Programmen zusammen, z. B. `dvips` und `pdftex`. Die Informationen zu diesen Treibern stehen in den Konfigurationsdateien von LaTeX. Für die Arbeit mit Farben gibt es die Datei `color.cfg`. Zusätzliche Treiber können dieser Datei über den Befehl `\ExecuteOptions` hinzugefügt werden.

Weitere Pakete für Grafiken

Neben den oben genannten drei Grundpaketen existieren noch weitere Grafikpakete, unter anderem Pakete für 3-D-Grafiken und weitere Tools.

Tipp: Weiterführende Informationen über die Arbeit mit Bildern unter LaTeX finden Sie in der englischsprachigen Anleitung zum `graphics`-Paket:

http://tug.ctan.org/tex_archive/macros/latex/required/graphics/grfguide.pdf

6.1.1 Die Pakete graphics und graphicx

Die einfachste Möglichkeit, Bilder einzubinden, sind die Pakete `graphics` und `graphicx`. Die Pakete sind im Hinblick auf die Funktionen nahezu identisch: Viele Befehle von `graphics` sind in erweiterter Form auch im Paket `graphicx` enthalten. Das Paket `graphicx` hat verschiedene Parameter. Die Tabelle gibt einen Überblick über die verschiedenen Parameter:

Parameter	Beschreibung
Treiber	Legt den Grafiktreiber fest, der eingebunden werden soll.
draft	Wechselt in den Entwurfsmodus.
final	Wechselt in den Arbeitsmodus. <code>final</code> ist das Gegenteil des Entwurfsmodus.

Tabelle 6.1: Parameter des Pakets `graphicx`

Diese Parameter können Sie dem Befehl `\usepackage` übergeben:

```
\usepackage[Parameter]{graphicx}
\usepackage[Parameter]{graphics}
```

Für den Parameter `Treiber` können Sie verschiedene Hilfsprogramme von LaTeX angeben, z. B. `dvips`, `pdftex`, `dvipdf`, `dvipdfm` und Ähnliches. Bei `pdftex` lädt LaTeX dann automatisch den passenden Treiber. Bei DVI muss zunächst der passende Konvertierer geladen werden.

```
\usepackage[dvips]{graphics}
\usepackage[pdftex]{graphicx}
```

In dem obigen Beispiel werden die Pakete `graphics` und `graphicx` mit den Parametern `dvips` und `pdftex` aufgerufen.

Tipp: Wenn Sie keinen Parameter angeben, sucht sich LaTeX standardmäßig den richtigen Treiber selbst aus. Dieses Vorgehen sollten Sie bevorzugen.

6.1.2 Farben

Das `color`-Paket bietet auch verschiedene Funktionen für Farben. Die Tabelle gibt eine Übersicht über die verschiedenen Befehle:

<i>Befehle</i>	<i>Beschreibung</i>
<code>\DefineColor</code>	Definiert eine neue Farbe.
<code>\Color</code>	Legt eine existierende Farbe fest.
<code>\textcolor</code>	Legt Farben für Text fest.

Tabelle 6.2: Farbbefehle

6.2 Bilder aus Dateien einbinden

Mit dem Befehl `\includegraphics`, den Sie schon in Kapitel 3 kennengelernt haben, lassen sich auf ganz einfache Weise Bilder einbinden. Auch der Befehl `\includegraphics` hat verschiedene Parameter. Einige davon haben Sie zwar bereits in Kapitel 3 gesehen. Die folgende Tabelle gibt aber noch mal einen Überblick über die wichtigsten Parameter des Befehls `\includegraphics`:

<i>Parameter</i>	<i>Beschreibung</i>
<code>bb x0,y0,x1,y1</code>	Bounding Box-Parameter, gibt die Werte für die Bounding Box an.
<code>width</code>	Die Breite des Bildes
<code>height</code>	Die Höhe des Bildes
<code>totalheight</code>	Die absolute Höhe
<code>scale</code>	Ein Skalierungsparameter, d. h., das Bild wird entsprechend skaliert.
<code>angle</code>	Der Winkel
<code>origin</code>	Der Ursprungspunkt des Bildes. Gilt als Referenzwert.
<code>natheight</code>	Zwei Werte für die rechte obere Ecke des Bildes. Die linke untere Ecke wird mit (o,o) angegeben.
<code>viewport</code>	Der Viewport ist ein rechteckiger Ausschnitt, den Sie anzeigen können. Als Bezugspunkt gilt die linkere obere Ecke.
<code>trim x0,y0,x1,y1</code>	Damit können Sie die Boundingbox verkleinern.
<code>clip</code>	Ein clipping-Parameter. Damit können Sie überstehene Bereiche des Bildes abschneiden.

Parameter	Beschreibung
draft	Wechselt in den Entwurfsmodus.
Final	Endmodus bzw. Bearbeitungsmodus. Ist das Gegenteil von draft.
keepaspectratio	Das Verhältnis von Breite zu Höhe des Bildes, d. h. das Seitenverhältnis. Wenn Sie diesen Parameter verwenden, wird das Seitenverhältnis beibehalten, auch wenn das Bild vergrößert oder verkleinert wird.
ext, type, read, command	Legt das Verhalten von Fremdformaten fest.

Tabelle 6.3: Parameter für den Befehl `includegraphics`

Die Dateierweiterung für das Bild brauchen Sie dabei normalerweise nicht anzugeben. LaTeX erkennt die Dateierweiterung eines Bildes in der Regel automatisch.

Beispiele

Nachfolgend finden Sie ein paar Beispiele zum Einbinden von Bildern:

```
\includegraphics[width 1cm, height 1cm]{bild1}
```

Das Ergebnis zeigt Abbildung 6.1:



Abbildung 6.1: Bild mit 1 cm Breite und 1 cm Höhe

Mit dem Parameter `angle` können Sie Bilder um einen bestimmten Winkel drehen:

```
\includegraphics[width 1cm, height 1cm, angle 30]{bild1}
```



Abbildung 6.2: Das Bild wird um einen Winkel von 30 Grad gedreht.

Der Befehl `keepaspectratio` behält das Seitenverhältnis des Bildes bei, wenn es vergrößert oder verkleinert wird:

```
\includegraphics[keepaspectratio, width 2cm, height 2cm]{bild1}
```

Das Ergebnis zeigt Abbildung 6.3:



Abbildung 6.3: Seitenverhältnis beibehalten

Abbildung 6.4 zeigt ein Beispiel für den Parameter `viewport`:

```
\includegraphics[width 2cm, height 2cm, viewport 0 0 50 50]{bild1}
```

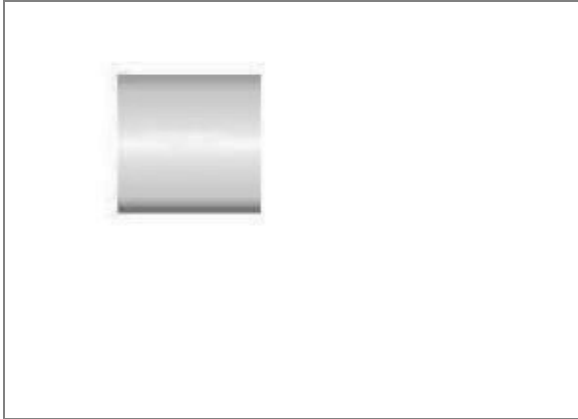


Abbildung 6.4: Viewport ändern

Mit dem Befehl `trim` können Sie Teile des Bildes verkürzen bzw. abschneiden:

```
\includegraphics[width 1cm, height 1cm, trim 10 10 10 10]{bild1}
```



Abbildung 6.5: Beispiel für den Parameter `trim`

Tipp: Mit dem Befehl `\includegraphics` können Sie auch Verzeichniseinträge setzen. Dann muss aber bei jeder Änderung der Verzeichnisstruktur das Dokument angepasst werden.

6.2.1 Dateiendungen für Bilder

Normalerweise brauchen Sie beim Einbinden von Bildern keine Dateiendung fest anzugeben. Auf Wunsch können aber die Dateiendungen von Bildern fest vorgegeben werden. Dazu dient der Befehl `\DeclareGraphicsExtensions`:

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz,eps.Z}  
\DeclareGraphicsExtensions{.jpg,.png,.mps,.tif,.pdf}
```

Wenn Sie `\DeclareGraphicsExtensions` so aufrufen, sucht LaTeX nach folgenden Dateiendungen bzw. Formaten für die Programme `dvips` und `pdftex`. Die erste Tabelle bezieht sich dabei auf `dvips`:

<i>Dvips</i>	<i>Format</i>
Bild.eps	EPS-Format (Encapsulated PostScript)
Bild.ps	PS-Format (PostScript)
Bild.eps.gz	Eps.gz Format (komprimiertes EPS)
Bild.ps.gz	Ps.gz. Format (komprimiertes PS)
Bild.eps.Z	Eps.Z Format

Tabelle 6.4: Formate für Bilder bei `dvips`

Die folgende Tabelle gilt für `pdftex`:

<i>Pdftex</i>	<i>Format</i>
Bild.jpg	JPEG-Format
Bild.png	PNG-Format
Bild.mps	MPS-Format
Bild.pdf	PDF-Format

Tabelle 6.5: Formate für Bilder bei `pdftex`

LaTeX handelt sich dann dabei durch alle Dateiformate, vom ersten bis zum letzten. Wenn LaTeX keines der angegebenen Formate findet, gibt es eine Fehlermeldung aus.

Wenn Sie keine Endung angeben wollen, müssen Sie eine leere Definition angeben, also ein Paar geschweiffter Klammern anhängen. Dieses kann dann als letzter Eintrag in der Parameterliste erscheinen:

```
\DeclareGraphicsExtensions{.eps,.ps,.eps.gz,.ps.gz, eps.Z, {} }
```

Tipp: Es können auch selbstdefinierte Regeln für den Umgang mit Grafikformaten erstellt werden, die standardmäßig nicht unterstützt werden.

6.2.2 Suchpfad für Dateien

LaTeX sucht normalerweise in den folgenden Verzeichnissen:

- Im Verzeichnis, in dem das aktuelle TeX-Dokument liegt.

- In den `texmf`-Verzeichnisbäumen unter `tex/LaTeX`, also den Verzeichnisbäumen der installierten LaTeX-Distribution.
- In allen Verzeichnissen, die in der Umgebungsvariable `TEXINPUTS` erscheinen.
- In allen Verzeichnissen, die über den Befehl `\graphicspath` aufgelistet werden.

Verzeichnisse über `TEXINPUTS`

Über das Betriebssystem legen Sie Verzeichnisse mit der Umgebungsvariablen `TEXINPUTS` fest.

Unter Linux können Sie z. B. über den `export`-Befehl die Umgebungsvariable `TEXINPUTS` festlegen:

```
export TEXINPUTS images:/bilder/eps:$TEXINPUTS
```

Die Umgebungsvariable wird dabei mit einem `$`-Zeichen eingebunden.

Unter Windows werden Umgebungsvariablen über den Befehl `set` festgelegt:

```
Set TEXINPUTS images;c:\bilder\eps;%TEXINPUTS%
```

Unter Windows werden Umgebungsvariablen mit dem `%`-Zeichen eingebunden.

Fügen Sie dabei immer den vorherigen Inhalt der Umgebungsvariablen hinzu, da hier schon manche Verzeichnisse vorab eingetragen sein können. Bei der Einrichtung einer Umgebungsvariablen sollten Sie auch bedenken, dass Sie genügend Speicher haben. Insbesondere unter Windows-Systemen ist dieser häufig sehr klein bemessen. Der Weg über die Umgebungsvariable ist der schnellste Weg, mehrere Verzeichnisse anzugeben.

Wenn Sie den Befehl `\graphicspath` verwenden, so lautet der Aufruf wie folgt:

```
\graphicspath{{images}{/bilder/eps}}
```

Ein anderer Vorteil der Umgebungsvariable ist, dass damit die Portabilität des Dokumentes gewahrt bleibt. Die Verwendung des Befehls `\graphicspath` ist dagegen eher langsam.

Wer die grafische LaTeX-Umgebung Texmaker verwendet, kann sich zudem ähnlich wie im Windows-Dateimanager einfach durch die Verzeichnisse klicken und auf diese Weise Grafikdateien sehr bequem einbinden. Um diese Option zu nutzen, muss man lediglich im LaTeX-Menü von Texmaker die Option `\includegraphics{file}` anklicken.

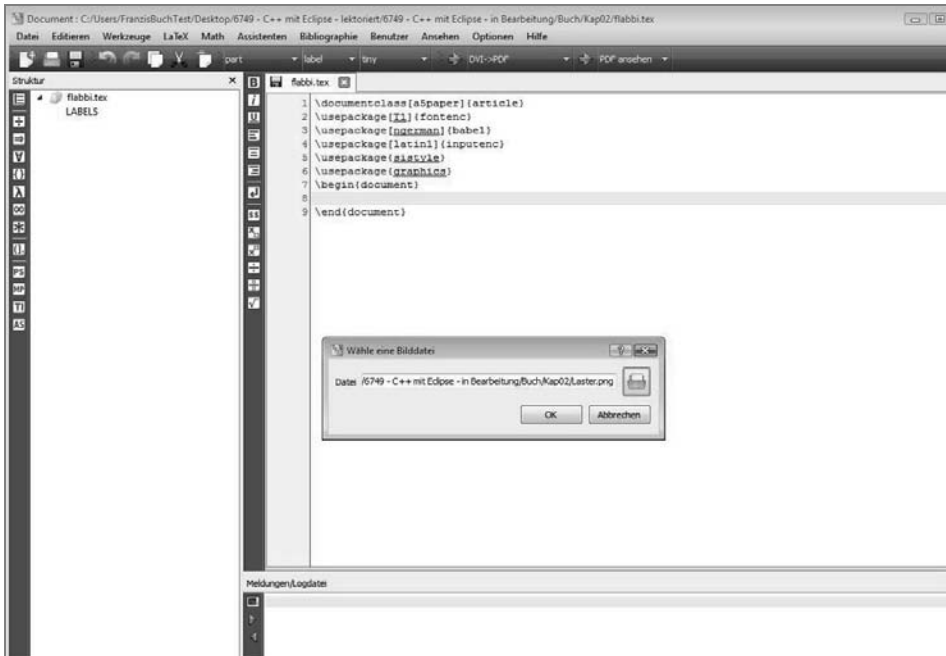


Abbildung 6.6: Bilder einbinden leicht gemacht mit Texmaker

6.2.3 Bilder zentrieren

Sie können Bilder auf verschiedene Arten zentrieren. Die einfachste Möglichkeit ist, Bilder in eine `center`-Umgebung zu packen:

```
\documentclass[4paper]{article}
\usepackage{graphics}
\begin{document}
\begin{center}
\includegraphics[width 1cm, height 2cm]{bild1}
\end{center}
\end{document}
```

Beispiel 6.1: Bilder zentrieren

Die Alternative dazu ist, den Befehl `\centering` in die `figure`-Umgebung einzupassen:

```
\begin{figure}
\centering
\includegraphics[width 1cm, height 1cm]{bild1}
\end{figure}
```

Beispiel 6.2: Bilder mit dem Befehl `\centering` zentrieren

Bilder mit `minipage`-Umgebung nebeneinandersetzen

Möchten Sie mehrere Bilder nebeneinandersetzen, dann verwenden Sie eine `minipage`-Umgebung:

```
\documentclass[a4paper]{article}
\usepackage{graphicx}
\begin{document}
\begin{figure}
\begin{minipage}[b]{.2\linewidth}
\includegraphics[width \linewidth]{bild1}
\end{minipage}
\begin{minipage}[b]{.2\linewidth}
\includegraphics[width \linewidth]{bild1}
\end{minipage}
\end{figure}
\end{document}
```

Beispiel 6.3: Bilder mit `minipage`-Umgebung nebeneinandersetzen

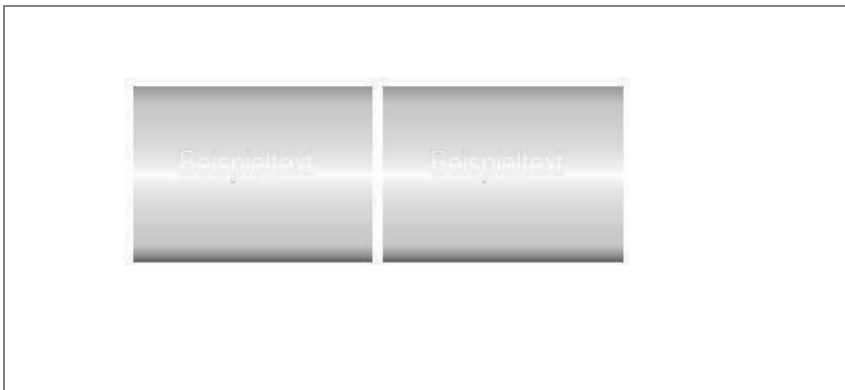


Abbildung 6.7: Bilder nebeneinandersetzen

Mit dem Parameter `.2\linewidth` der `minipage`-Umgebung setzen Sie die Zeilenbreite auf 20 Prozent der Zeilenhöhe des Dokumentes. Mit dem Parameter `b` – für Bottom – werden die Bilder nach unten ausgerichtet.

Tipp: Mit der `minipage`-Umgebung lassen sich auch Tabellen nebeneinandersetzen.

Regeln für die Arbeit mit Bildern

- Sie müssen keine Dateiendungen angeben. Diese werden in LaTeX intern durch den Befehl `\DeclareGraphicsExtensions` festgelegt.
- Schreiben Sie den Befehl `\includegraphics` mit seinen Parametern möglichst in eine Zeile.
- Pfade für Dateien können Sie über die Umgebungsvariable `TEXINPUT` festlegen.

6.2.4 Hintergrundbilder einbinden

Mit dem Paket `eso-pic` lassen sich Hintergrundbilder in ein Dokument einbinden. Der entsprechende Befehl heißt `\AddtoShipOutputPicture`. Mit diesem Befehl können Sie, ähnlich wie mit dem Befehl `\includegraphics`, Bilder skalieren, drehen, stauchen, aufblähen usw.

```
\documentclass[a4paper]{article}
\usepackage{graphicx}
\usepackage{eso-pic}
\AddtoShipoutPicture{\AtTextLowerLeft{\framebox{\LenToUnit{%
\textwidth},\LenToUnit{\textheight}}{\rotatebox[origin c]{%
{ 50}}{\includegraphics[scale 0.9]{bild1}}}}}
\begin{document}
Das ist ein Beispieltext
Das ist ein Beispieltext
\end{document}
```

Beispiel 6.4: Hintergrundbild

Das Ergebnis sieht so aus:



Abbildung 6.8: Hintergrundbild

Als Ergebnis sehen Sie ein um 50 Grad gedrehtes Hintergrundbild. Das Bild selbst wird mit dem Befehl `\includegraphics` eingebunden und mit dem Befehl `\AtTextLowerLeft` in die linke untere Ecke gesetzt. Das Paket `eso pic` enthält noch weitere nützliche Befehle. Nachstehende Tabelle gibt eine grobe Übersicht:

Befehl	Beschreibung
<code>\ClearShipoutPicture</code>	Bewirkt das Gegenteil von <code>\AddToShipoutPicture</code> .
<code>\LenToUnit</code>	Gibt eine Längeneinheit für das Bild an.
<code>\pscoord</code>	Legt den linken oberen Eckpunkt fest.

Tabelle 6.6: Befehle des Pakets `eso pic`

Weitere Informationen zu den Befehlen finden Sie in der Dokumentation des Pakets `eso-pic`.

Tipp: Die Paketdokumentation dazu finden Sie unter:

`ftp://ftp.uni-erlangen.de/mirrors/CTAN/macros/LaTeX/contrib/eso-pic/eso-pic.pdf`

Drehen von Bildern

In einem früheren Kapitel haben Sie schon den Befehl `\rotatebox` kennengelernt. Sie können damit nicht nur Boxen, sondern auch Bilder und andere Objekte drehen. Der allgemeine Aufruf des Befehls lautet:

```
\rotatebox[param]{winkel}{objekt}
```

Die Tabelle gibt eine Übersicht über die Parameter:

<i>Parameter</i>	<i>Beschreibung</i>
<code>X</code>	Drehpunkt für das Bild
<code>Y</code>	Drehpunkt für das Bild
<code>Origin label</code>	Der Referenzpunkt wird an die entsprechende Stelle gesetzt.
<code>Units zahl</code>	Damit wird die Maßeinheit für den Winkel festgelegt. Standardmäßig gilt hier das normale Gradsystem. Sie können aber auch Winkel im Bogenmaß angeben.

Tabelle 6.7: Parameter für den Befehl `\rotatebox`

Das Beispiel zeigt eine Anwendung für den Befehl `\rotatebox`:

```
\usepackage{graphicx}
\begin{document}
Ein Text der \rotatebox{25}{rauf} und wieder
\rotatebox[origin rB]{25}{runter} geht
\end{document}
```

Beispiel 6.5: Der Befehl `\rotatebox`

Das Ergebnis zeigt Abbildung 6.7:

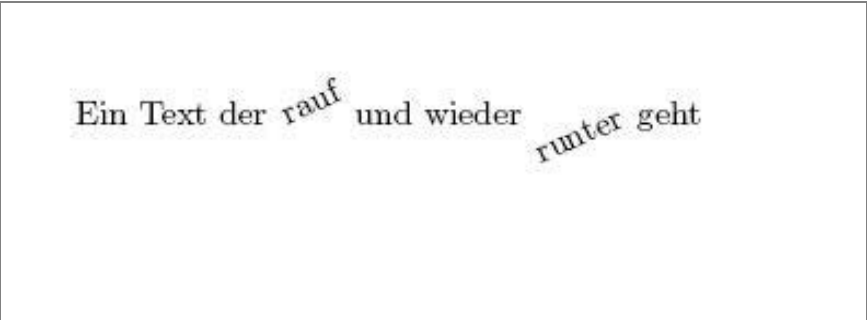


Abbildung 6.9: Beispiel für den Befehl \rotatebox

Skalierung von Bildern

Für die Skalierung von Bildern gibt es verschiedene Befehle. Die Tabelle gibt einen Überblick über die Befehle:

Befehle	Beschreibung
\scalebox	Skaliert ein Objekt
\resizebox	Skaliert ein Objekt nach Breite und Höhe.

Tabelle: 6.8: Befehle für das Skalieren von Objekten

Der einfachste Befehl zum Skalieren von Objekten ist \scalebox. Hier ein Beispiel für den Befehl \scalebox:

```
\scalebox{h scale}[v scale]{Objekt}  
Ein \scalebox{2}[3]{skalierter} Text
```

Der andere Befehl lautet \resizebox:

```
\resizebox{breite}{höhe}{Objekt}  
\resizebox{2cm}{\height}{skalierter Text}
```



Abbildung 6.10: Beispiel für skalierbare Objekte

Das Ergebnis der beiden Befehle zeigt die obige Abbildung.

6.2.5 Objekte in gleitenden Umgebungen drehen

In Kapitel 5 haben Sie schon den Befehl `\rotate` kennengelernt. Neben dem Befehl `\rotate` gibt es noch die Umgebung `rotate`, mit der Sie Objekte um einen bestimmten Winkel drehen können. Der Aufruf der Umgebung `\rotate` lautet wie folgt:

```
\begin{rotate}[winkel]
\end{rotate}
```

So lässt sich ein Text um einen beliebigen Winkel drehen:

```
\begin{rotate}{15}
\fbbox{Ein gedrehter Text..}
\end{rotate}
```

Neben der Umgebung `rotate` gibt es noch die Umgebung `sideways`, die ähnlich aufgerufen wird wie die Umgebung `rotate`. Mit der Umgebung `sideways` lassen sich Texte um 90 Grad drehen:

```
\begin{sideways}
\end{sideways}
```

Hier ein Beispiel für den Aufruf von `sideways`:

```
\begin{sideways}
\fbbox{Ein gedrehter Text..}
\end{sideways}
```

Der Text in der `fbbox` wird dabei um 90 Grad nach links gedreht.

Gedrehte gleitende Umgebungen

Wenn Sie Tabellen oder Bilder um 90 Grad drehen wollen, gibt es die Umgebungen `sidewaystable` und `sidewaysfigure`. Die Umgebung `sidewaystable` dreht Tabellen um 90 Grad. Die Umgebung `sidewaysfigure` dreht Bilder um 90 Grad.

Die Umgebung `sidewaystable` wird wie folgt aufgerufen:

```
\begin{sidewaystable}
\end{sidewaystable}
```

Die Umgebung `sidewaysfigure` wird so aufgerufen:

```
\begin{sidewaysfigure}
\end{sidewaysfigure}
```

Hier ein Beispiel für die Umgebung `sidewaystable`:

```
\begin{sidewaystable}
\begin{tabular}{@{}ld{2.2}@}\toprule
Käse & Herkunftsland \\
\multicolumn{1}{c}{Käse} \\
\cmidrule(r){1 1}\cmidrule(lr){2 2}\cmidrule(l){3 3}
Edamer & Holland \\
Gouda & Holland \\
Ementaler & Holland \\
Andechser Bierkäse & Bayern \\
\bottomrule
\end{tabular}
\end{sidewaystable}
```

In dem Beispiel wird eine Tabelle über Käsesorten erstellt, in der die Tabelle nach rechts gedreht wird.

6.3 Bilder beschriften und nummerieren

Es gibt noch zwei weitere gleitende Umgebungen für Tabellen und Bilder. Diese beiden Umgebungen heißen `subfloats` und `subtables`. Damit lassen sich unter anderem Teiltbilder und Teiltabellen automatisch beschriften und nummerieren. Das ist nützlich, wenn Sie Bilder oder Tabellen, die in einem bestimmten Kapitel stehen, fortlaufend nummerieren wollen, aber die Kapitelnummer aus der Nummerierung ersichtlich sein soll. In so einem Fall werden die Bilder bzw. Tabellen dann mit 1a, 1b, 1c beziehungsweise 1-1, 1-2, 1-3 usw. versehen.

Die einfache fortlaufende Nummerierung erfolgt über die Umgebung `figure`; diese ist bereits im Grundpaket von LaTeX enthalten. Das Beispiel zeigt, dass jedes einzubindende Bild von der `figure`-Umgebung umschlossen wird. Der Befehl `\caption` enthält in geschweiften Klammern die Bildunterschrift, die der entsprechenden Abbildung zugeordnet ist. Im folgenden Beispiel wird der Bildunterschrift »Ein Laster« die Bildnummer 1 zugeordnet, der folgenden »Noch ein Laster« die Nummer 2. Anzumerken ist, dass Sie über `\usepackage` den Parameter `[ngerman]` des `babel`-Pakets einbinden

sollten, denn sonst erhalten Sie statt der deutschsprachigen Angabe »Abbildung 1« die englische Bezeichnung »Figure 1«.

```
\documentclass[a4paper]{article}
\usepackage[ngerman]{babel}
\begin{document}
\begin{figure}
% \includegraphics[bb 0 0 1 1]{bild1.png}
\caption{Ein Laster}
\end{figure}

\begin{figure}
% \includegraphics[bb 70 70 71 71]{bild1.png}
\caption{Noch ein Laster}
\end{figure}
\end{document}
```

Die für die Teilnummerierungen notwendigen Umgebungen `subfloats` und `subtables` werden über den Befehl `\usepackage{subfloat}` bereitgestellt.

Der Aufruf der beiden Umgebungen sieht so aus:

```
\usepackage{subfloat}

\begin{subfloats}
\begin{figure}
\end{figure}
\end{subfloats}
% bzw.

\begin{subtables}
\begin{table}
\end{table}
\end{subtables}
```

Zu dem Paket `subfloat` gibt es auch eine entsprechende Umgebung namens `subfigures`, mit der Sie Bilder mit 1a, 1b, 1c usw. nummerieren.

Hier sehen Sie ein Beispiel für die Umgebung `subfigures`:

```
\begin{subfigures}
\begin{figure}
\includegraphics[width 6cm]{beispiel}
\caption{Ein Beispielbild}
\end{figure}
\begin{figure}
```

```

\includegraphics[width 6m]{beispiel2}
\caption{Noch ein Beispielbild}
\end{figure}
\end{subfigures}

```

In diesem Beispiel werden die zwei Bilder `beispiel` und `beispiel2` untereinander als gleitende Objekte eingefügt. Die beiden Bilder erhalten den Inhalt der `\caption`-Zeile als Bildunterschrift und werden mit den Nummern 1a und 1b versehen. Sie werden dazu jeweils in eine eigene `figure`-Umgebung eingebettet.

6.3.1 Änderung der Beschriftung

Mit dem Paket `subfigure` lässt sich über `\thesubfloatfigure` oder `\themainfigure` das Schema der Nummerierung anpassen. Im vorigen Beispiel wurde das Nummernschema 1a, 1b usw. verwendet. Es gibt aber auch andere Möglichkeiten der fortlaufenden Nummerierung, etwa 1-1, 1-2, 1-3 usw. Der allgemeine Aufruf des Befehls lautet:

```

\thesubfigure
\themainfigure

```

`\themainfigure` repräsentiert hier die Hauptnummer, `\thesubfigure` die Teilnummer. Unten sehen Sie ein Beispiel für diese beiden Befehle:

```

\renewcommand*{\thesubfloatfigure}{\themainfigure \arabic{subfloatfigure}}

```

Mit `\themainfigure` wird die Hauptnummerierung und mit `subfloatfigure` die Teilnummerierung neu definiert. Hier wählen wir die Option `arabic` – also die Darstellung der Teilnummern in arabischen Ziffern. Wollen wir wieder auf Buchstaben zurückwechseln, dann wird anstelle der Option `arabic` der Befehl `alph` benutzt.

Für Tabellen werden die Nummerierungen entsprechend mit `\themaintable` und `\subfloattable` dargestellt.

6.3.2 Neue gleitende Umgebungen

Über das Ergänzungspaket `float` lassen sich eigene gleitende Umgebungen definieren, die sich genauso verhalten wie `table` und die `figure`-Umgebung.

Das Paket `float` hat dabei folgende optionale Parameter:

- `plain`
Damit wird der Standard-LaTeX-Stil gesetzt. Die Beschriftung wird hier unter das Objekt gesetzt.
- `plaintop`
Wie `plain`, jedoch wird die Beschriftung über das Objekt gesetzt.

- `boxed`
Das Objekt wird in eine Box mit einem Rahmen gesetzt. Die Beschriftung wird unterhalb des Kastens gesetzt.
- `ruled`
Die Umgebung wird mit horizontalen Linien begrenzt. Die Beschriftung wird über das Objekt gesetzt.

Mit dem Befehl `\newfloat` definieren Sie die neuen Umgebungen. Der allgemeine Aufruf lautet:

```
\newfloat{<Name>}{<pos>}{<Dateiendung>}[<Gliederungsebene>]
```

Mit dem ersten Parameter wird dabei der Name der neuen gleitenden Umgebung definiert. Über den Parameter `pos` wird die Positionsvoreinstellung definiert. Als mögliche Werte können Sie hier `h`, `t`, `b`, `p` und `H` sowie die Kombination von diesen angeben. Der Parameter `H` sorgt dafür, dass das Objekt genau hier positioniert wird, egal, ob es an dieser Stelle gerade passt oder nicht. Das heißt, das Objekt gleitet nicht. Wenn das Objekt nicht auf die Seite passt, wird entsprechend Leerraum eingefügt und das Objekt auf die nächste Seite verschoben. Über den Parameter `Dateiendung` wird festgelegt, wo die Einträge für die Objektlisten gespeichert werden. Der Dateiname setzt sich dabei aus dem Namen des Hauptdokumentes und der angegebenen Dateiendung zusammen.

Die Objektlisten können dann als Übersicht eingebunden werden. Der letzte Parameter legt fest, auf welcher Gliederungsebene die Nummerierung durchgeführt wird. Des Weiteren stehen diese Befehle zur Verfügung:

```
\floatname{<Name>}{<Label>}
\floatplacement{<Name>}{<pos>}
\floatstyle{<stil>}
\restylefloat{<Umgebung>}
```

Mit `\floatname` wird der Name festgelegt, der vor der Nummerierung bei der Beschriftung gesetzt wird. Mit `\floatplacement` lässt sich die Standardpositionierung nachträglich ändern. Den Stil der gleitenden Umgebung können Sie über den Befehl `\floatstyle` ändern. Dabei können Sie die gleichen Parameter wie beim Paketaufruf angeben. Wenn Sie schon eine Umgebung festgelegt haben, können Sie diese per `\restylefloat` ändern.

```
\floatstyle{ruled}
\newfloat{Diagramm}{htbp}{dia}

\begin{Diagramm}
  \includegraphics{dia}
  \caption{Graphik}
\end{Diagramm}
```

In dem obigen Beispiel wird die neue Umgebung `Diagramm` definiert, in die eine Grafik eingefügt wird. In der ersten Zeile wird dabei der Stil für die neue Umgebung festgelegt. Danach wird die neue Umgebung mit dem Namen `Diagramm` bestimmt.

Das Ergänzungspaket `floatrow`

Das Ergänzungspaket `floatrow` erlaubt es, ähnliche Layouts für leere Floatboxen, Reihen von gleitenden Objekten sowie gleitende Objekte neben der Beschriftung zu definieren. Außerdem können Sie eine gemeinsame Ausrichtung für die ganze Floatbox, die Objekte innerhalb der Floatbox sowie die Inhalte der gleitenden Objekte erwirken. Ferner können Sie ein gemeinsames Layout für alle Floatboxen eines Typs bzw. das Layout ändern.

Die Parameter haben dabei folgende Bedeutung:

- `preamble`
Damit wird die Anordnung der Beschriftung erreicht. Mit `\capsecond` liest der Befehl den Beschriftungsinhalt nach dem Objekthinhalt. Mit `\captop` bzw. `\capbot` wird die Beschriftung vor bzw. nach `\capbeside` neben dem Floatobjekt platziert.
- `captype`
Damit wird der Typ des Floatobjektes fixiert.
- `width`
Damit wird die Breite der Beschriftungsbox festgelegt, wenn die Beschriftung vor oder nach dem gleitenden Objekt angeordnet wird bzw. die Breite des gleitenden Objektes, wenn die Beschriftung neben einem Objekt angeordnet wird.
- `height`
Damit wird die Höhe der Beschriftungsbox angegeben, wenn die Beschriftung vor oder nach einem gleitenden Objekt angeordnet wird bzw. die Höhe des Objektes, wenn die Beschriftung neben dem Objekt angeordnet wird.
- `inner vert pos`
Damit wird die vertikale Ausrichtung des gleitenden Objektes in eine Box festgelegt.
- `caption`
Beschriftungstext
- `object`
Inhalt des gleitenden Objektes.

Ein Beispiel dazu:

```
\begin{table}
\floatbox[\captop]{table}[1.0\FBwidth]
{\caption{Einfache Tabelle mit Beschriftung oberhalb}}
{\begin{tabular}{|l|c|c|}\hline
Stadt & Land & Fluss\\\hline
```

```

Darmstadt & Deutschland & Donau\\
Paris & Frankreich & Seine\\
\dots & &\\hline
\end{tabular}}
\end{table}
\end{document}

```

Wichtig ist hier der Parameter `1.0\FBwidth`. Die Zahl vor `\FBwidth` gibt an, wie die Breite der Beschriftung im Verhältnis zur Tabelle aussieht. Hier haben wir den Faktor 1.0 gewählt, die Beschriftung darf somit ebenso breit sein wie die Tabelle selbst. Wählen wir einen Faktor unter 1, wird die Beschriftung schmaler, wählen wir einen größeren, wird sie entsprechend breiter. Das Ergebnis ist im folgenden Bild zu sehen:

Tabelle 1: Einfache Tabelle mit Beschriftung oberhalb

Stadt	Land	Fluss
Darmstadt	Deutschland	Donau
Paris	Frankreich	Seine
...		

Abbildung 6.11: Die Tabelle wurde komplett mit der Beschriftung erzeugt.

Wenn Sie mehrere Umgebungen nebeneinander anordnen wollen, erreichen Sie dies mit der Umgebung `floatrow`:

```

\begin{floatrow}[<anzahl der Gleitobjekte]
\floatbox...
\floatbox...
\end{floatrow}

```

6.3.3 Umgebungen drehen

Neben der Möglichkeit, neue gleitende Umgebungen zu erstellen, lassen sich auch ganze Umgebungen drehen. Mit dem Ergänzungspaket `rotfloat` werden die Pakete `float` und `rotating` kombiniert.

Das Paket `rotfloat` stellt keinen eigenen Parameter bereit, es werden aber alle Parameter an das Paket `rotating` weitergereicht.

Über den Befehl `\newfloat` definieren Sie wie beim `float`-Paket neue gleitende Umgebungen:

```
\newfloat{<Name>}{<pos>}{<Dateiendung>}[<Gliederungsebene>]
```

Hier ein Beispiel für diesen Befehl:

```
\newfloat{Diagramm}{htbp}{dia}
```

Daneben stehen noch diese Umgebungen zur Verfügung:

- `Diagramm`
Damit wird eine neue gleitende Umgebung mit dem Namen `Diagramm` erzeugt.
- `Diagramm*`
Wie `Diagramm`, jedoch für zweispaltigen Text.
- `sidewaysDiagramm`
Damit wird eine neue gleitende Umgebung mit dem Namen `sidewaysDiagramm` bewirkt, mit der der Inhalt um 90 Grad gedreht wird.

Außerdem stehen die Befehle `\floatname`, `\floatstyle`, `\floatplacement` und `\restylefloat` zur Verfügung, die dieselben Funktionen wie im `float`-Paket haben. Mit dem Befehl `\restylefloat` ändern Sie einen bereits definierten Stil.

Das folgende Beispiel erstellt zunächst ein Diagramm mit der Beschriftung »Eine Grafik«. Die Dateiendung `dia` im Befehl `\newfloatstyle` zeigt an, dass wir hier eine Grafik einbinden wollen, die mit dem freien Grafikprogramm `dia` erstellt wurde (im Web zu finden unter <http://live.gnome.org/Dia>). Das damit erzeugte Schaubild wird nun zweimal eingebunden – einmal als Standard und einmal um 90 Grad gedreht.

```
\floatstyle{ruled}
\newfloatstyle{Diagramm}{htbp}{dia}
\begin{Diagramm}
  \includegraphics{width 3cm}{dia2}
  \caption{Eine Grafik}
\end{Diagramm}

\begin{sidewaysdiagramm}
  \includegraphics[width 3cm]{dia2}
  \caption{Noch eine Grafik}
\end{sidewaysdiagramm}
```

6.3.4 Vorhandene Umgebungen anpassen

Über das `rotfloat`-Paket lassen sich auch vorhandene Umgebungen wie `table` oder `figure` entsprechend anpassen.

```
\floatstyle{boxed}  
\restylefloat{table}
```

Dies stellt sicher, dass die Umgebungen `table`, `table*`, `sidewaystable` und `sidewaystable*` so angepasst werden, dass sie zukünftig auch mit einer Box dargestellt werden können.

6.3.5 Nichtgleitende Umgebungen

Hin und wieder ist es erwünscht, ein Objekt mit einer fortlaufenden Nummerierung und einer Beschriftung wie bei einem Gleitobjekt zu versehen, es jedoch genau auf dem Platz zu positionieren, an dem es im Quelltext auftritt. Über das Ergänzungspaket `nofloat` legen Sie eine neue Umgebung fest, die dort positioniert wird, wo sie auftritt, und die Nummerierung einer gleitenden Umgebung fortführt:

```
\begin{nofloat}[<bezug>]  
\end{nofloat}
```

In dem nachfolgenden Beispiel wird ein Bild in einer `nofloat`-Umgebung platziert. Dabei wird die Nummerierung der `figure`-Umgebung fortgeführt:

```
\begin{figure}  
  \includegraphics[width 4cm]{dia}  
  \caption{Ein Diagramm}  
\end{figure}  
  
\newpage\null\newpage%Seitenumbruch erzwingen  
  
\begin{nofloat}{figure}  
  \includegraphics[width 4cm]{dia}  
  \caption{Noch ein Diagramm}  
\end{nofloat}
```

In diesem Beispiel wurde das Bild `dia` in einer `figure`-Umgebung platziert. Danach erfolgt ein Seitenumbruch. In der Umgebung `nofigure` wird dann das Bild noch einmal eingebunden und dort platziert, wo es im Quelltext auftritt.

Gleitobjekte ans Ende verschieben

Redakteure von Artikeln wünschen sich hin und wieder, dass alle gleitenden Umgebungen automatisch an das Ende des Kapitels geschoben werden, sodass eine strikte Trennung zwischen Text und Bildern bzw. Tabellen entsteht. Über das Ergänzungspaket

`endfloat` können Sie dies realisieren. Weitere Informationen finden Sie in der Dokumentation des Pakets.

Gleitobjekte auf Doppelseiten positionieren

In vielen Büchern werden Bilder und Tabellen auf Doppelseiten positioniert. Über das Ergänzungspaket `dpfloat` lässt sich dies auch mit LaTeX erreichen. Weitere Informationen finden Sie in der Paketdokumentation.

6.4 Textumflossene Objekte

Viele Textverarbeitungsprogramme und auch HTML bieten die Möglichkeit, Objekte mit Text umfließen zu lassen. Für LaTeX gibt es verschiedene Ergänzungspakete, mit denen dies möglich ist. Alle diese Pakete haben dabei eine andere Intention und ein anderes Einsatzgebiet, das jeweils nachfolgend kurz dargestellt wird.

6.4.1 Bilder und Tabellen im Absatz

Es gibt verschiedene Ergänzungspakete, mit denen Sie Bilder in Tabellen und Absätze einfügen können.

- Mit dem Ergänzungspaket `picins` lassen sich kleinere Bilder in einem Absatz positionieren und vom Text umfließen.
- Für das Setzen von Bildern und Tabellen im Absatz gibt es außerdem noch die Pakete `picinpar`, `wrapfig`, `floatflt` usw.

Das Paket `picins`

Mit dem Paket `picins` lassen sich kleinere Bilder am Beginn eines Absatzes einfügen. Es liefert unter anderem den Befehl `\parpic`, der es erlaubt, zu Beginn eines Absatzes (rechts oder links) ein Bild (oder ein anderes Objekt) zu platzieren:

```
\parpic(<breite>,<höhe>)(<x offset>,<y offset>)[<optionen>][<pos>]{<Bild>}
```

Mit den ersten beiden Parametern `breite` und `höhe` wird die Breite und Höhe des Gleitobjektes bestimmt. Der nachfolgende Text wird um die Breite plus einen kleinen Abstand eingerückt. Die Höhe bestimmt, wie viele Zeilen des umfließenden Textes eingerückt werden sollen. Mit den Parametern `x offset` bzw. `y offset` wird das Bild innerhalb des Rahmens verschoben.

Über den Parameter `optionen` bestimmen Sie Position und Gestaltung des Bildrahmens. Es sind folgende Einstellungen möglich:

- `l` (für left): Das Bild wird auf der linken Seite des Absatzes platziert.
- `r` (für right): Das Bild wird auf der rechten Seite des Absatzes platziert.

- `f` (für frame): Das Bild wird mit einem Rahmen umzogen.
- `d` (für dashed): Das Bild wird mit einem gestrichelten Rahmen umzogen.
- `o` (für oval): Die Ecken des Rahmens werden abgerundet.
- `s` (für shadow): Der Rahmen erhält einen Schatten.
- `x` Das Bild wird mit einem 3-D-Kasten umrahmt.

Text neben dem Bild

Mit dem Befehl `\picskip` können Sie die Anzahl von Textzeilen bestimmen, die neben das Bild gesetzt werden sollen. Der Befehl `\picskip` wird allgemein so aufgerufen:

```
\picskip{n}
```

Mit dem Parameter `n` legen Sie die Anzahl der Zeilen fest. Wenn Sie hier 0 angeben, wird der aktuelle Absatz abgebrochen und der nachfolgende Absatz unterhalb des Bildes fortgeführt. Wenn der Wert von `n` größer ist als die Anzahl der Zeilen, die für das Bild benötigt werden, werden die Zeilen nach dem Bild trotzdem eingerückt.

Anpassen der Strichstärke

Die Strichstärken und -längen der Rahmen für die Bilder lassen sich mit diesen Befehlen verändern:

```
\linethickness{<dicke>}
\dashlength{<länge>}
\shadowthickness{<dicke>}
\boxlength{<länge>}
```

Die Stärke der Linie wird dabei mit dem Befehl `\linethickness` festgelegt. Der Standardwert ist hier 0.4 pt. Die Strichlänge wird über `\dashlength` bestimmt. Wenn Sie eine Schattenbox verwenden, wird über `\shadowthickness` die Stärke des Schattens festgelegt. Wenn Sie eine 3-D-Box verwenden, können Sie über `\boxlength` die Tiefe der Box bestimmen.

Bilder bei Doppelseiten positionieren

Mit dem Befehl `\picchangemode` können Sie Bilder auf Doppelseiten entsprechend einer geraden oder ungeraden Seite positionieren. Bei ungeraden Seiten (das sind im Allgemeinen rechte Seiten) wird die Voreinstellung von `\parpic` belassen (wenn für `\parpic` die Option `l` gewählt wurde, bedeutet das, dass das Bild auf der linken Seite des Absatzes positioniert wird). Bei geraden (also linken) Seiten wird diese Anordnung vertauscht (das Bild kommt also auf der rechten Seite des Absatzes zu liegen). Dieser Modus ist so lange aktiv, bis der Befehl `\nopicchangemode` aufgerufen wird.

Bilder zwischen Absätzen

Mit dem Befehl `\hpic` können Sie Bilder zwischen zwei Absätzen positionieren.

Der allgemeine Aufruf lautet:

```
\hpic(<breite>,<höhe>)(<x offset>,<y offset>)[<optionen>][<pos>]{<Bild>}
```

Es gibt folgende Parameter für die Optionen:

- `t` (für top): Aufeinanderfolgende Bilder werden an der Oberkante ausgerichtet.
- `b` (für bottom): Aufeinanderfolgende Bilder werden an der Unterkante ausgerichtet.
- `f` Das Bild wird mit einem Rahmen versehen.
- `d` Das Bild wird mit einem gestrichelten Rahmen versehen.
- `o` Die Ecken des Rahmens werden abgerundet.
- `s` Der Rahmen enthält einen Schatten.
- `x` Das Bild wird mit einem 3-D Kasten umrandet.

Ohne die Parameter `t` und `b` werden die Bilder nacheinander vertikal gesetzt.

Das Paket `wrapfig`

Mit dem Ergänzungspaket `wrapfig` können Sie Bilder und Tabellen ganz einfach im Absatz positionieren. Der Vorteil von diesem Paket ist, dass Tabellen und Bilder auch in den Rand hineinragen können. Dieses Paket stellt die beiden Umgebungen `wrapfigure` und `wraptable` zur Verfügung.

Der allgemeine Aufruf lautet:

```
\begin{wrapfigure}[<Zeilen>]{<pos>}[<Randüberhang>]{<Breite>}
\end{wrapfigure}
```

Der Aufruf von `wraptable` erfolgt analog.

Mit dem Parameter `Zeilen` wird angeordnet, wie viele Zeilen das Bild umfließen sollen. Wenn Sie hier keinen Wert angeben, so wird dieser Wert aus der Bildhöhe berechnet. Über den Parameter `pos` wird die Position des Bildes bestimmt. Hier können Sie `l` für links und `r` für rechts, `i` für innen und `o` für außen angeben. Die entsprechenden Parameter in Großbuchstaben sind dafür verantwortlich, dass das Objekt entsprechend gleitet. Der Parameter `Randüberhang` gibt den Wert an, der bestimmt, um welchen Wert das Bild in den Rand hineinragen soll.

6.4.2 Initiale erstellen

In LaTeX sind Initiale nichts anderes als textumflossene Objekte. Wenn Sie Initiale in einen Text einbinden wollen, benötigen Sie das Paket `lettrine`. Das Paket `lettrine`

stellt den Befehl `\lettrine` zur Verfügung, der als Parameter zwei Argumente hat. Erstens den Buchstaben, der Initial werden soll, und zweitens den nachfolgenden Text, der dargestellt werden soll:

```
\lettrine{B}{enjamin Blümchen}
```

Das Aussehen des Initials können Sie über verschiedene Parameter beeinflussen. So können Sie z. B. angeben, über wie viele Zeilen sich der Anfangsbuchstabe erstrecken und wie groß der Abstand zum nachfolgenden Text sein soll. Hier ein Beispiel:

```
\usepackage{lettrine}
...
\begin{document}
\lettrine[lines 2, lraise 0.0]{D}{er große Häuptling} der Cherusker spricht
und ruft seine Mannen dazu auf, tapfer gegen die einfallenden Römer zu
streiten und zu siegen.
\end{document}
```

Die Parameter in den eckigen Klammern geben an, dass sich der Anfangsbuchstabe über zwei Zeilen erstrecken (`lines 2`) und gegenüber dem nachfolgenden Text nicht nach oben oder unten verschoben sein soll (`lraise 0.0`). Als Initial hervorgehoben wird der erste Buchstabe in den geschweiften Klammern, der nachfolgend in geschweiften Klammern enthaltene Text wird in Kapitälchen gesetzt. Hier das Resultat:

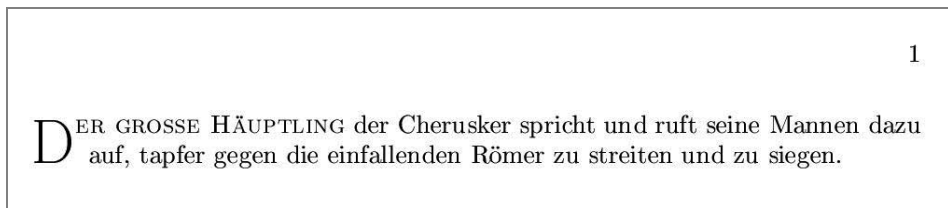


Abbildung 6.12: Fertiges Initial im Druckbild

7 Eigene Grafiken erstellen

Neben den Möglichkeiten, bestehende Grafikdateien einzubinden, lassen sich mit LaTeX eigene Bilder und Zeichnungen erstellen. In Kapitel 3 haben Sie schon die `picture`-Umgebung kennengelernt, mit der Sie einfache Objekte zeichnen können. Neben der `picture`-Umgebung gibt es auch noch das Paket `PSTricks`, das Ihnen in diesem Kapitel gezeigt wird.

7.1 Bilder zeichnen mit der `picture`-Umgebung

Die `picture`-Umgebung ist eine einfache Umgebung zum Zeichnen von einfachen Grafiken wie Linien, Kreisen, Dreiecken usw. Die einzelnen Befehle werden in der `picture`-Umgebung aufgerufen, und zwar in Kombination mit dem Befehl `\put`.

```
\begin{picture}(x,y)
\put %Befehle
\end{picture}
```

Bei dem einleitenden Aufruf der `picture`-Umgebung werden zwei Koordinaten in runden Klammern beigegeben, die die Breite (x) und Höhe (y) der nachfolgenden Zeichnung definieren. Zuvor muss allerdings noch die Längeneinheit bestimmt werden, nach der sich diese Größe bemisst. Dies geschieht mit dem Befehl:

```
\setlength{\unitlength}{Längeneinheit}
```

An die Stelle von `Längeneinheit` können Sie verschiedene Werte setzen, es empfiehlt sich, hier `1cm` zu verwenden – nicht zuletzt, weil eigentlich jeder eine recht gute Vorstellung davon hat, wie lang ein Zentimeter ist. Wenn also die `picture`-Umgebung so eingeleitet wird, ist klar, dass die Zeichnung vier Zentimeter breit und zwei Zentimeter hoch sein soll:

```
\setlength{\unitlength}{1cm}
\begin{picture}(4,2)
```

7.1.1 Linien zeichnen

Die einfachste geometrische Figur, die mit der `picture`-Umgebung gezeichnet wird, ist eine Linie.

Der allgemeine Aufruf lautet:

```
\line(delta x, delta y){<länge>}
```

Der Befehl `\line` erzeugt einfache Linien. Unten sehen Sie ein simples Beispiel für eine Linie:

```
\begin{document}
\setlength{\unitlength}{1cm}
\begin{picture}(4,2)
\put(0,0){\line(1,0){4}}
\end{picture}
\end{document}
```

Beispiel 7.1: Einfache Linie

Sie sehen, dass der `\line`-Befehl in den bereits erwähnten `\put`-Befehl eingebettet wurde. `\put` legt die Ausgangsposition der Linie fest. Das ist in diesem Fall der Punkt mit den Koordinaten $(0, 0)$, der linke untere Eckpunkt unserer Zeichnungsfläche.



Abbildung 7.1: Einfache Linie

7.1.2 Linien mit Pfeilen

Mit dem Befehl `\vector` können Sie auch Geraden mit Pfeilen, also Vektoren, zeichnen. Der allgemeine Aufruf lautet:

```
\vector(delta x, delta y){<länge>}
```

Die Pfeilspitze befindet sich dabei stets am Ende der Linie. Hier ein Beispiel für den Befehl `\vector`:

```
\begin{picture}(4,2)
\put(0,0){\vector(0,2){4}}
\put(0,0){\vector(2,0){4}}
\end{picture}
```

Beispiel 7.2: Zwei Pfeile

7.1.3 Kreise

Mit dem Befehl `\circle` können Sie einfache Kreise zeichnen. Der Befehl `\circle` wird so aufgerufen:

```
\circle{<radius>}  
\circle*{<radius>}
```

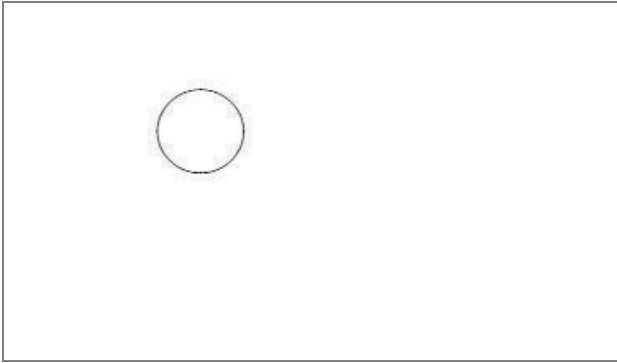


Abbildung 7.2: Einfacher Kreis

Mit dem Befehl `\circle*` wird der Kreis in einer Füllfarbe gezeichnet:

```
\begin{picture}(4,2)  
\put(1,0.5){\circle{2}}  
\put(1,0.5){\circle*{2}}  
\end{picture}
```

Beispiel 7.3: Einfacher Kreis



Abbildung 7.3: Kreis mit Füllfarbe

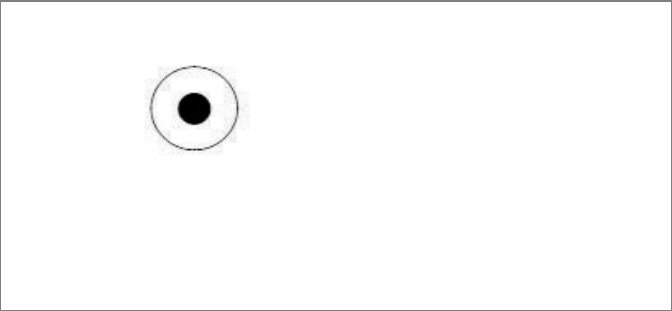


Abbildung 7.4: Zwei Kreise

7.1.4 Rechtecke

Mit den Befehlen `\framebox`, `\makebox` und `\dashbox` können Sie Rechtecke zeichnen. Die Befehle dazu:

```
\makebox(<x>,<y>)[<pos>]{<text>}
\framebox(<x>,<y>)[<pos>]{<text>}
\dashbox(<dashline>)[<x>,<y>]{<text>}
```

Die Parameter `<x>` und `<y>` geben jeweils die Koordinaten an. Der Parameter `<pos>` bestimmt die Positionierung eines Textes innerhalb des Rechtecks. Folgende Werte können für den Parameter `<pos>` angegeben werden:

Parameter	Beschreibung
t (top)	Der Text wird oben angezeigt.
b (bottom)	Der Text wird unten angezeigt.
l (left)	Der Text wird links angezeigt.
r (right)	Der Text wird rechts angezeigt.
s (stretched)	Der Text wird vertikal zentriert.
- (ohne)	Der Text wird normal zentriert angezeigt.
tl (top left)	Der Text wird links oben angezeigt.
tr (top right)	Der Text wird rechts oben angezeigt
bl (bottom left)	Der Text wird unten links angezeigt.
br (bottom right)	Der Text wird unten rechts angezeigt.

Tabelle 7.1: Parameter für das Rechteck

Hier ein paar Beispiele für die verschiedenen Parameter:

```
\begin{picture}
\put(0,0){\framebox(3,1)[t]{Beispieltext}}
\end{picture}
```

Beispiel 7.4: Rechteck mit Text

Das Ergebnis zeigt folgende Abbildung:

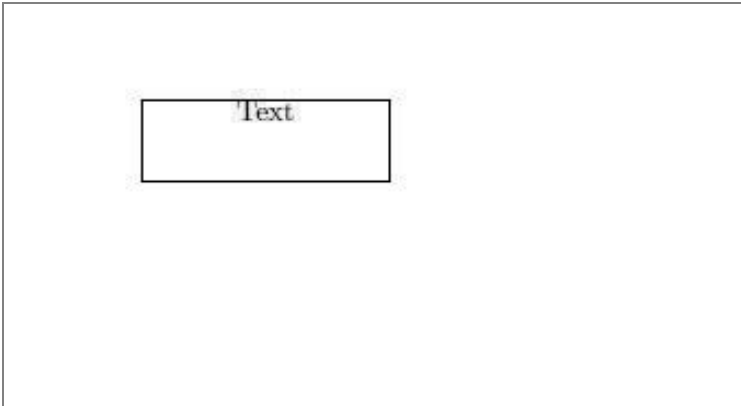


Abbildung 7.5: Rechteck mit Text

Die anderen Parameter werden ähnlich aufgerufen:

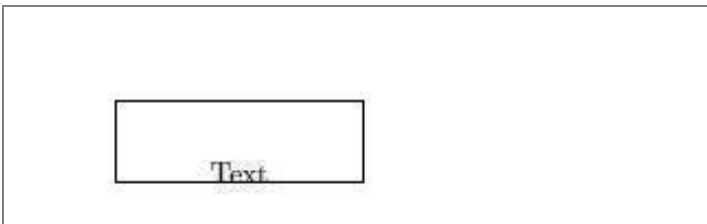


Abbildung 7.6: Parameter b – der Text rutscht nach unten.

Mit dem Befehl `\dashbox` können Sie ein Rechteck mit gestrichelten Linien erzeugen:

```
\begin{picture}(3.1,1.1)
\put(0,2){\dashbox{0.2}(3,1){Text}}
\end{picture}
```

Beispiel 7.5: Rechteck mit einer gestrichelten Umrisslinie

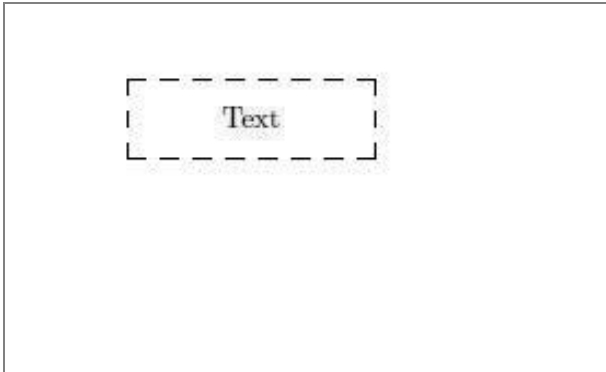


Abbildung 7.7: Rechteck mit gestrichelten Linien

7.1.5 Ovale und gerundete Ecken

Mit dem Befehl `\oval` können Sie ein Rechteck mit abgerundeten Ecken zeichnen. Der allgemeine Aufruf lautet:

```
\oval(<x>,<y>)[<teil>]
```

Die Koordinaten im `\put`-Befehl legen hier den Mittelpunkt des Ovals fest.

Im folgenden Beispiel wird ein einfaches Oval gezeichnet:

```
\begin{picture}(4,2)
\put(0,0){\oval(4,2)}
\end{picture}
```

Beispiel 7.6: Oval

Das Ergebnis zeigt folgende Abbildung:

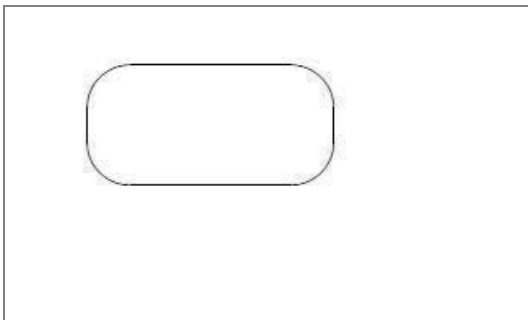


Abbildung 7.8: Einfaches Oval

Mit dem Parameter `<teil>` können Sie festlegen, welcher Teil des Ovals gezeichnet werden soll. Die folgende Tabelle gibt eine Übersicht über die möglichen Parameter:

Parameter	Beschreibung
b (bottom)	Der untere Teil wird gezeichnet.
t (top)	Der obere Teil wird gezeichnet.
l (left)	Der linke Teil wird gezeichnet.
r (right)	Der rechte Teil wird gezeichnet.
br (bottom right)	Der untere rechte Teil wird gezeichnet.
bl (bottom left)	Der untere linke Teil wird gezeichnet.
tr (top right)	Der obere rechte Teil wird gezeichnet.
tl (top left)	Der obere linke Teil wird gezeichnet.

Tabelle 7.2: Parameter des Befehls `oval`

Hier ein Beispiel für die oben genannten Parameter:

```
\begin{picture}(4,2)
\put(5,0){\oval(3,2)[b]}
\end{picture}
```

Beispiel 7.7: Parameter `b` – die untere Hälfte des Ovals wird ausgegeben.

Das Ergebnis zeigt folgende Abbildung:

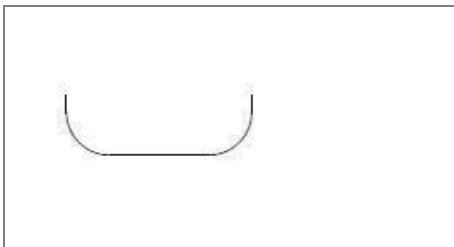


Abbildung 7.9: Nach oben geöffnetes Oval

In Kombination mit dem Befehl `\makebox` können Sie auch Texte in das Oval integrieren:

```
\begin{picture}(4,2)
\put(0,0){\oval(4,2)}
\put(0,0){\makebox(2,2){Text}}
\end{picture}
```

Beispiel 7.8: Oval mit Text

Das Ergebnis sieht so aus:

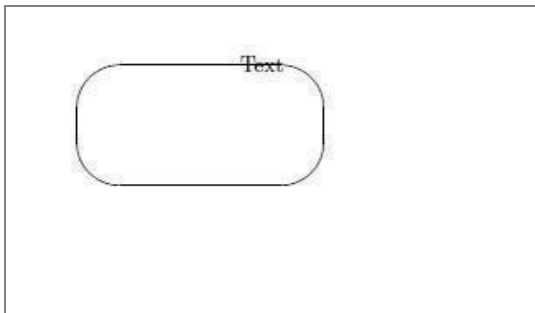


Abbildung 7.10: Oval mit Text

7.1.6 Bézierkurven

Eine Kurve lässt sich durch die Angabe von verschiedenen festen Punkten zeichnen. Eine Bézierkurve lässt sich durch die Angabe von mindestens drei verschiedenen Punkten definieren.

Der allgemeine Aufruf lautet:

```
\qbezier(<x>,<y>)(<x2>,<y2>)(<x3>,<y3>)
```

Durch die drei Koordinatenpaare `<x>,<y>`, `<x2>,<y2>` und `<x3>` und `<y3>` wird die Kurve definiert. Das erste Koordinatenpaar gibt den Anfang der Kurve an. Das zweite Paar stellt den Kontrollpunkt für die Kurve dar und das letzte markiert das Ende der Kurve. Da ja das erste Koordinatenpaar den Anfang der Kurve repräsentiert, ist hier kein `\put`-Befehl zur Positionierung notwendig.

Im folgenden Beispiel wird eine einfache Bézierkurve gezeichnet:

```
\begin{picture}(4,2)
\qbezier(0,0)(2,2)(4,1)
\end{picture}
```

Beispiel 7.9: Bézierkurve

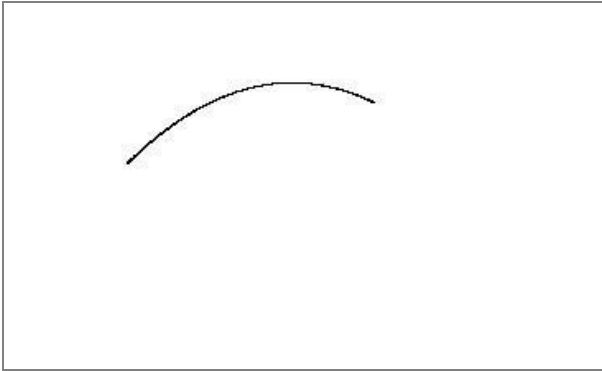


Abbildung 7.11: Bézierkurve

Linienstärke

Mit den Befehlen `\thicklines` und `\thinlines` können Sie die Linienstärke beeinflussen. Alle bisherigen Linien wurden mit der Strichstärke 0,4 pt gezeichnet. Mit dem Befehl `\linethickness` können Sie die Linienstärke festlegen:

```
\thicklines
\thinlines
\linethickness{<stärke>}
```

Im folgenden Beispiel werden Bézierkurven mit unterschiedlichen Strichstärken gezeichnet:

```
\begin{picture}(7,5)
\thicklines
\qbezier(0,0)(2,3)(7,5)
\thinlines
\qbezier(0,1)(2,4)(7,4)
\end{picture}
```

Beispiel 7.10: Linien mit unterschiedlicher Stärke

Das Ergebnis sieht so aus:

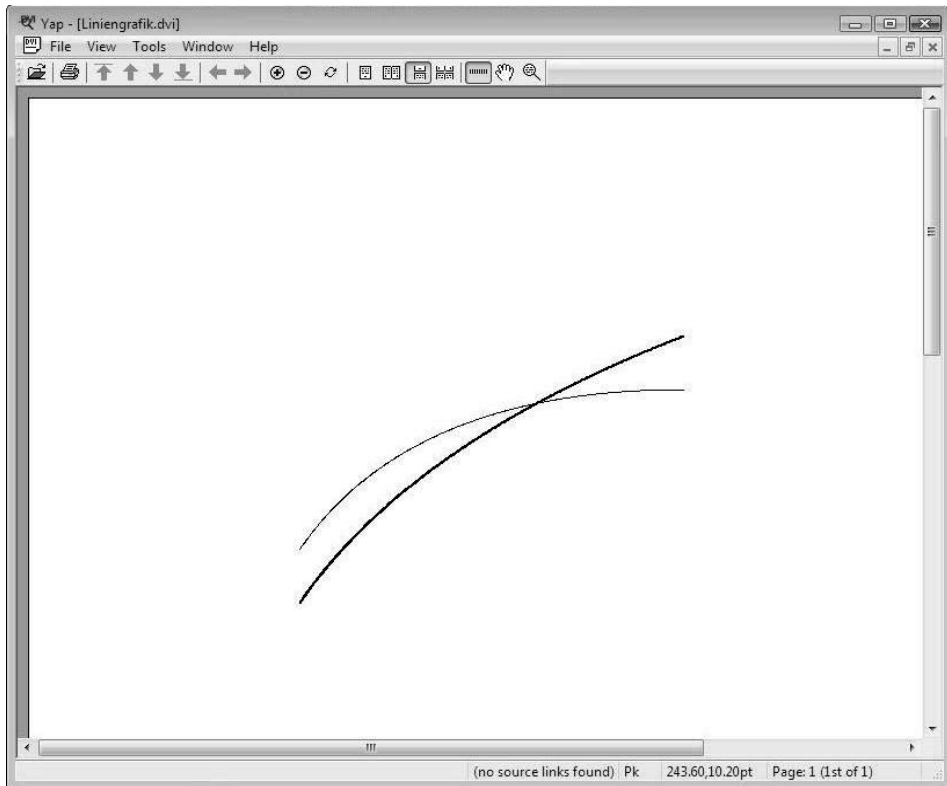


Abbildung 7.12: Ergebnis zu Beispiel 7.9

Im folgenden Beispiel werden verschieden starke Linien mit dem Befehl `\linethickness` gezeichnet:

```
\begin{picture}(4,2)
\linethickness{1pt}
\put(0,0){\line(1,0){4}}
\linethickness{2pt}
\put(0,1){\line(1,0){4}}
\end{picture}
```

Beispiel 7.11: Verschieden starke Linien

Das Ergebnis sieht dann so aus:

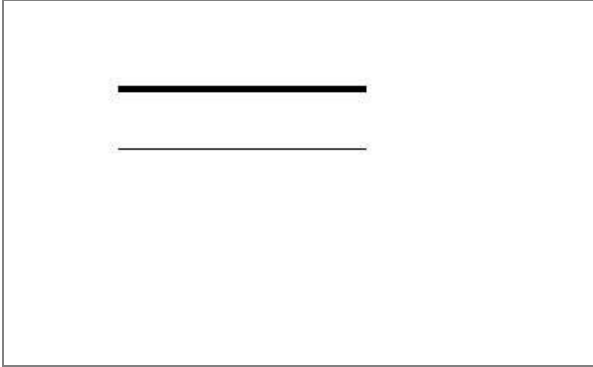


Abbildung 7.13: Ergebnis zu Beispiel 7.10

7.1.7 Mehrere Figuren zeichnen

Mit dem Befehl `\multiput` können Sie Objekte mehrmals zeichnen. Der Befehl `\multiput` ist eine Variante des Befehls `\put`, mit dem Sie grafische Objekte platzieren können.

```
\multiput(<x>,<y>)(<delta x>,<delta y>){<anzahl>}{<Befehl>}
```

Mit den Parametern `<x>` und `<y>` legen Sie die Koordinaten fest. Mit den Parametern `<delta x>` und `<delta y>` bestimmen Sie die Verschiebung des Objektes um die angegebenen Werte. Mit dem Parameter `anzahl` legen Sie die Anzahl der Objekte fest.

Im folgenden Beispiel wird – nur aus horizontalen und vertikalen Linien, die jeweils um eine halbe Längeneinheit gegeneinander verschoben werden – ein kleines Quadratbrett gezeichnet:

```
\begin{picture}(4,2)
\multiput(0,0)(0.5,0){7}{\line(0,1){3}}
\multiput(0,0)(0,0.5){7}{\line(1,0){3}}
\end{picture}
```

Beispiel 7.12: Quadratbrett

Das Ergebnis sieht so aus:

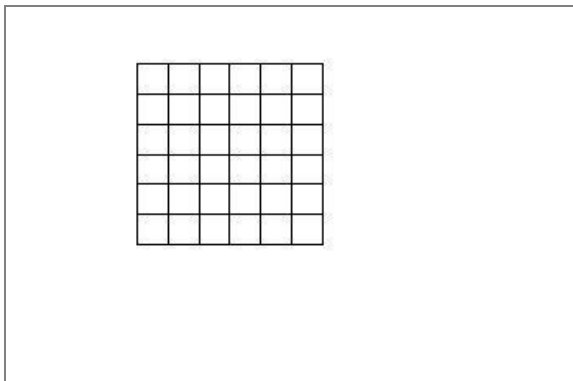


Abbildung 7.14: Quadratbrett

7.1.8 Text mit Bildern

Mit dem Befehl `\parbox` können Sie einen Text mit Zeichnungselementen mischen. Im folgenden Beispiel wird ein einfacher Text mit einem Vektor als Pfeil angezeigt:

```
\begin{picture}(4,2)
\put(0.5,1){\parbox[b]{2cm}{Ein Text mit Bild}}
\put(0,0){\vector(1,2){0.5}}
\end{picture}
```

Beispiel 7.13: Text mit Bild

Das Ergebnis zeigt folgendes Bild:

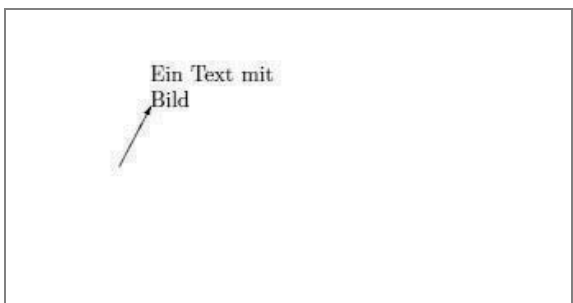


Abbildung 7.15: Ergebnis zu Beispiel 7.12

Mit dem Parameter `[b]` für bottom des Befehls `\parbox` wird der Bezugspunkt des Pfeils unter den Text gesetzt. Wenn Sie den Parameter weglassen, zeigt der Pfeil links in die Mitte des Textes. Verwenden Sie den Parameter `[t]`, dann zeigt der Pfeil in die erste Zeile.

```
\begin{picture}(4,2)
\put(0.5,1){\parbox{2cm}{Ein Text mit Bild}}
\put(0,0){\vector(1,2){0.5}}
\end{picture}
```

Beispiel 7.14: Text mit Grafik

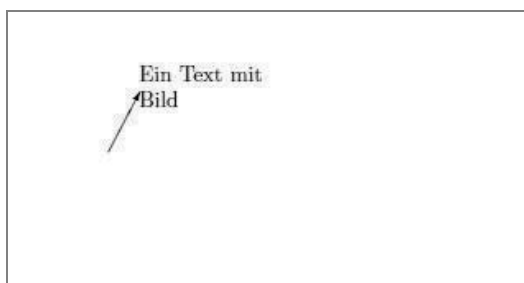


Abbildung 7.16: Text mit Grafik

7.1.9 Text vertikal zeichnen

Mit dem Befehl `\shortstack` können Sie Text vertikal, z. B. an einer Linie entlang, laufen lassen. Der Befehl `\shortstack` gilt auch außerhalb der `picture`-Umgebung.

Der allgemeine Aufruf lautet:

```
\shortstack[<pos>]{<Text>}
```

Über den Parameter `<pos>` wird die Ausrichtung des Textes bestimmt. Die folgende Tabelle gibt eine Übersicht über die möglichen Parameter:

Parameter	Beschreibung
l	Links
r	Rechts
c	Zentriert

Tabelle 7.3: Parameter für die Textausrichtung

Unten sehen Sie ein Beispiel für den Befehl `\shortstack`:

```
\begin{picture}(4,2)
\put(1,0.5){\line(0,1){2.5}}
\put(1.1,1.5){\shortstack[l]{\\L\\i\\n\\i\\e}}
\end{picture}
```

Beispiel 7.15: Vertikaler Text

Das Ergebnis zeigt folgende Abbildung:

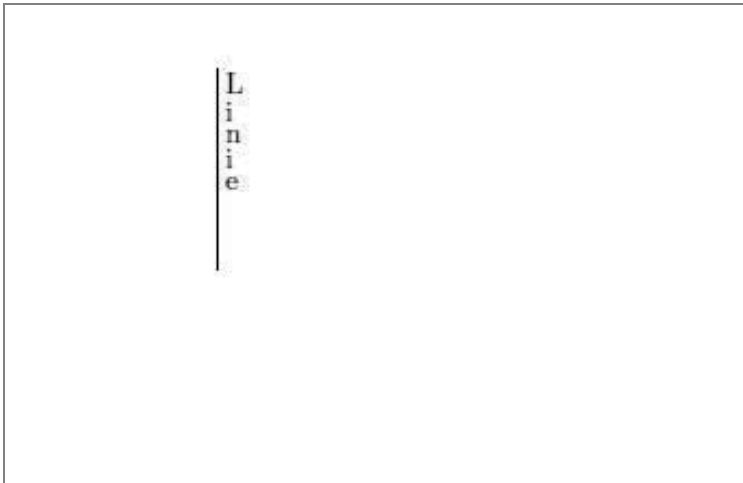


Abbildung 7.17: Vertikal ausgerichteter Text

7.1.10 Grafiken verschachteln

Wie bei anderen Umgebungen bzw. Befehlen auch lassen sich Grafiken verschachteln. Bei dem Beispiel ineinander gezeichneter Kreise haben Sie schon ein Beispiel für eine verschachtelte Grafik gesehen.

Unten sehen Sie ein weiteres Beispiel für eine verschachtelte Grafik:

```
\begin{picture}(4,2)
\put(0,0){\circle{3}}
\put(0,0){\circle{2}}
\put(0,0){\circle{1}}
\end{picture}
```

Beispiel 7.16: Verschachtelte Grafiken

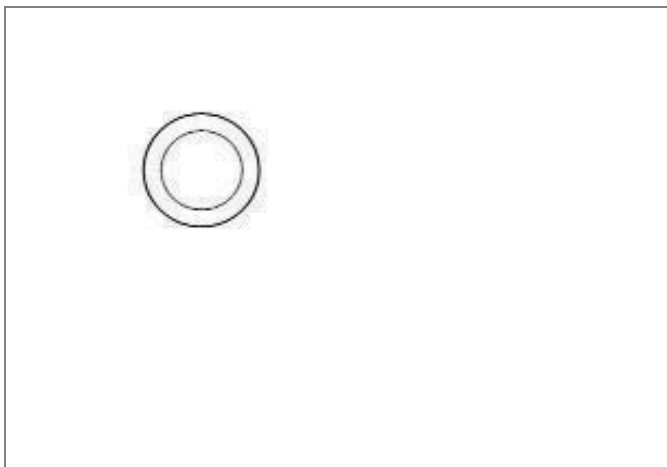


Abbildung 7.18: Verschachtelte Grafiken – konzentrische Kreise

7.1.11 Grafiken für die Wiederverwendung speichern

Grafikobjekte, die Sie häufig verwenden, speichern Sie am besten in einer Datei. Dazu gibt es die Befehle `\newsavebox`, `\savebox` und `\usebox`. Der allgemeine Aufruf lautet:

```
\newsavebox{<nameTeilbild>}  
\savebox{<nameTeilbild>}{<x>,<y>}[<pos>]{<Teilbild>}  
\usebox{<nameTeilbild>}
```

Der Befehl `\newsavebox` erstellt einen neuen Namen für das Teilbild. Der Befehl `\savebox` speichert das Bild und `\usebox` lädt das Bild wieder.

Der Befehl `\savebox` hat vier Parameter. Der erste ist das gespeicherte Bild. Der zweite speichert die Breite und Höhe des Bildes. Der dritte Parameter `<pos>` speichert die Ausrichtung des Bildes.

In dem folgenden Beispiel wird ein Kreis als Bild gespeichert:

```
\documentclass[a4paper]{article}  
\setlength{\unitlength}{1cm}  
\newsavebox{\Kreis}  
\savebox{\Kreis}(0,0){  
  \put(0,0){\circle{3}}  
  \put(0,0){\circle{2}}  
  \put(0,0){\circle{1}}  
}  
\begin{document}
```

```
\begin{picture}(4,2)
\put(0,0){\usebox{\Kreis}{\makebox(2,0){Kreis}}}
\end{picture}
\end{document}
```

Beispiel 7.17: Kreis als Bild gespeichert

7.1.12 Erweiterung für die picture-Umgebung

Das Ergänzungspaket `graphpap` ist eine Erweiterung für die `picture`-Umgebung. Der Name leitet sich von »graphpaper« ab, und das ist es auch, was diese Erweiterung tut – sie zeichnet Gitternetze, ähnlich dem Millimeterpapier, das Konstrukteure nur zu gut kennen. Das Paket `graphpap` stellt den Befehl `\graphpaper` zur Verfügung. Das Paket selbst wird wie folgt eingebunden:

```
\usepackage{graphpap}
```

Der allgemeine Aufruf des Befehls `\graphpaper` lautet so:

```
\graphpaper[<num>](<x>,<y>)(<xb>,<yb>)
```

Die Parameter `<x>` und `<y>` geben dabei die Breite und Höhe an, die Parameter `<xb>` und `<yb>` die Breite und Höhe des Gitternetzes. Mit dem Parameter `<num>` können Sie festlegen, bei wie vielen Längeneinheiten jeweils ein Gitterstrich gezeichnet werden soll.

Standardmäßig werden hier zehn Koordinateneinheiten verwendet. Im nachfolgenden Beispiel wird ein einfaches Gitternetz gezeichnet:

```
\usepackage{grahpap}
\setlength{\unitlength}{1cm}
\begin{document}
\begin{picture}(4,2)
\graphpaper(50,0)(400, 150)
\end{picture}
\end{document}
```

Beispiel 7.18: Einfaches Grid

7.2 Das Paket PSTricks

Das Paket `PSTricks` ist ein Ergänzungspaket für Zeichnungen und Grafiken. Das Paket ist sehr leistungstark. Es ist derzeit in der Version 1.5 verfügbar.

Der Name `PSTricks` steht für Post Script Tricks. Das Paket steht nur für DVI-Treiber zur Verfügung und kann daher nicht direkt mit `pdflatex` verwendet werden.

Weitere interessante Informationen zu psticks finden Sie unter www.pstricks.de oder unter <http://userpage.fu-berlin.de/~latex/PSTricks/DANTE2006.pdf>. Das Dokument zeigt auch, über welche Umwege aus psticks PDF-Dateien erstellt werden können.

7.2.1 Das PSTricks-Paket einbinden

Das Paket PSTricks wird wie folgt eingebunden:

```
\usepackage{pst-all}
\usepackage{pst-add}
```

Das Paket pst-add sollte als Letztes eingebunden werden. Einfacher ist aber folgende Lösung, die ebenfalls funktioniert:

```
\usepackage{pstricks}
```

Tipp: Der einfachste Weg, Grafiken mit PSTricks zu erzeugen, ist, beim Compilerlauf direkt den DVI-Betrachter als Ziel anzugeben bzw. direkt nach DVI zu konvertieren.

7.2.2 PSTricks-Standardwerte festlegen

Ähnlich wie bei der picture-Umgebung können Sie bei dem Paket psticks mit dem Befehl \psset Standardwerte festlegen. Der allgemeine Aufruf lautet:

```
\psset{param1 wert, param2 wert,...}
```

Für Längeneinheiten geben Sie für die einzelnen Koordinaten verschiedene Werte an. Die Einstellungen des Befehls gelten so lange, bis der Befehl erneut mit neuen Werten aufgerufen worden ist. Die Tabelle enthält eine Übersicht über die möglichen Werte:

<i>Parameter</i>	<i>Beschreibung</i>
Xunit dim	Längeneinheit für die x-Achse
Yunit dim	Längeneinheit für die y-Achse
Runit dim	Längeneinheit für die Radialrichtung

Tabelle 7.4: Parameter für den Befehl psset

Der Standardwert für diese Parameter ist jeweils 1cm. Wenn Sie z. B. die Längeneinheit auf 2 cm setzen wollen, rufen Sie den Befehl so auf:

```
\psset{xunit 2cm, yunit 2cm, runit 2cm}
```

7.2.3 Die Umgebung

Die PSTricks-Befehle können Sie alle in den normalen Text integrieren. Diese Befehle ändern jedoch nicht den aktuellen Punkt von LaTeX, was dazu führt, dass Text und Zeichnungen mit einer Box mit der Dimension null dargestellt werden. Das bedeutet, dass sich die Zeichnung und der Text den gleichen Platz teilen und jeweils übereinander liegen.

Damit dies verhindert wird, stellt das Paket `pspicture` eine dezidierte Umgebung mit dem Namen `pspicture` zur Verfügung:

```
\begin{pspicture}(<x0>,<y0>)(<x1>,<y1>)  
%Befehle  
\end{pspicture}
```

Mit der Umgebung `pspicture` wird ein Bild definiert. Die ersten beiden Parameter `<x0>,<y0>` geben dabei jeweils die linke untere Ecke an. Die Parameter `<x1>` und `<y1>` geben die Breite und die Höhe des Bildes an. Wird nur ein Koordinatenpaar angegeben, geht `pspicture` davon aus, dass der Nullpunkt des Koordinatensystems auch gleichzeitig der Ausgangspunkt des Bildes ist.

Die `pspicture`-Umgebung hat das gleiche Koordinatensystem wie die `picture`-Umgebung (siehe Kapitel 3 und Abschnitt 7.1).

Im folgenden Beispiel wird ein einfaches Bild mit einer Breite von 2 cm und einer Höhe von 2 cm definiert:

```
\psset{xunit 1cm, yunit 1cm, runit 1cm}  
\begin{pspicture}(2,2)  
\end{pspicture}
```

7.2.4 Farben

Das Paket `pspicture` definiert die folgenden Farben:

Farbwert	Beschreibung
Black	Schwarz
Gray	Grau
Darkgray	Dunkelgrau
Lightgray	Hellgrau
White	Weiß
Red	Rot

<i>Farbwert</i>	<i>Beschreibung</i>
Green	Grün
Cyan	Cyan
Magenta	Magenta
Yellow	Gelb

Tabelle 7.5: Farbwerte von PSTricks

Für diese Farben stehen folgende Befehle zur Verfügung:

<i>Farbwert</i>	<i>Befehl</i>
Gray	\gray
Lightgray	\lightgray
Darkgray	\darkgrey
White	\white
Red	\red
Green	\green
Cyan	\cyan
Magenta	\magenta
Yellow	\yellow

Tabelle 7.6: Befehle für Farbwerte

Unten sehen Sie ein Beispiel für rot gefärbten Text:

```
{\red Dieser Text ist in Rot}
```

Neue Farbwerte definieren

Neben den oben genannten einfachen Befehlen können Sie auch neue Farbwerte definieren:

- `\newgray{<color>}{<num>}`
Damit wird ein neuer Grauwert erzeugt.
- `\newrgbcolor{<color>}{num1 num2 num3}`
Damit wird ein neuer RGB-Farbwert erzeugt. Die Parameter `num1`, `num2` und `num3` stehen dabei für die RGB-Werte (Rot, Grün und Blau).
- `\newhsbcolor{<color>}{num1 num2 num3}`
Damit wird ein neuer HSB-Farbwert nach dem HSB-Modell definiert.

- `\newcmykcolor{<color>}{num1 num2 num3}`

Damit wird ein neuer Farbwert nach dem CMY-Modell definiert. Die Werte `num1`, `num2` und `num3` stehen jeweils für C, M und Y (Cyan, Magenta und Gelb).

Die einzelnen `num`-Werte können jeweils zwischen 0 und 1 liegen.

Beispiele:

```
\newgraycolor{gray}{0.25}
\newrgbcolor{red}{0 1 0}
```

Die einzelnen Werte können dabei mit folgenden Befehlen gesetzt werden:

- `setgraycolor`
Setzt einen Grauwert.
- `setrgbcolor`
Setzt einen RGB-Farbwert.
- `sethsbcolor`
Setzt einen HSB-Farbwert.
- `setcmykcolor`
Setzt einen CMY-Farbwert.

7.2.5 Linienstile

Für verschiedene geometrische Figuren und Objekte, wie z. B. Linien, können Sie unter anderem den Linienstil festlegen.

Für den Linienstil gibt es verschiedene Parameter. Die folgende Tabelle gibt eine kurze Übersicht über die möglichen Linienstile:

<i>Linienstil</i>	<i>Beschreibung</i>
<code>linestyle style</code>	Der Linienstil an sich. Gültige Angaben sind <code>solid</code> (durchgehend), <code>none</code> (keine Linie), <code>dashed</code> (gestrichelt) und <code>dotted</code> (gepunktet).
<code>dotsep dim</code>	Der Abstand zwischen zwei Punkten
<code>border dim</code>	Ein Rand in der Dimension <code>dim</code>
<code>bordercolor color</code>	Legt die Rahmenfarbe fest.
<code>doubleline true/false</code>	Zeichnet eine Doppellinie, wenn der Wert <code>true</code> ist. Die Doppellinie wird durch Leerraum getrennt.
<code>doublecolor color</code>	Legt die Farbe für den Zwischenraum einer Doppellinie fest.

<i>Liniensstil</i>	<i>Beschreibung</i>
shadow true/false	Zeichnet einen Schatten mit der Distanz shadowsize mit dem Winkel shadowangle, wenn der Wert wahr ist.
shadowsize	Siehe shadow
shadowangle	Siehe shadow
shadowcolor	Siehe shadow

Tabelle 7.7: Liniensstile

Beispiele

Hier ein paar Beispiele für Liniensstile:

```
1\psline(0,0)(1.8,3)
2\psline[border 1pt]{>}(0,3)(1.8,0)
3\psline[linecolor red,linewidth 1.5pt]{<>}(2.2,0)(3.8,3)
```

In den obigen Beispielen werden verschiedene Linien mit diversen Liniensstilen gezeichnet (dazu mehr in Abschnitt 7.2.9). Die Linie 2 wird mit einem Rand von 1 pt Breite gezeichnet. In der Zeile 3 wird eine Linie in der Farbe Rot und der Strichstärke 1.5 pt gezeichnet.

Die beiden Linien 2 und 3 sind zudem als Pfeile ausgeführt. Die Linie 2 erhält durch den Parameter {>} eine nach rechts weisende Spitze, die Linie 3 ist mit dem Parameter {<>} sogar als Doppelpfeil ausgeführt. Dazu jedoch später mehr.

7.2.6 Füllstile

Neben Liniensstilen gibt es auch verschiedene Stile, die festlegen, wie Objekte gefüllt werden. So können Sie z. B. Rechtecke, Kreise usw. auf verschiedene Weise füllen.

Die Tabelle gibt eine Übersicht über verschiedene Füllstile:

<i>Füllstil</i>	<i>Beschreibung</i>
fillstyle style	Der Füllstil. Gültige Angaben sind none (keine Füllung), solid (gleichmäßige Füllung), vlines (vertikale Schraffierung), vlines* (vertikale Schraffierung plus Hintergrundfarbe), hlines (horizontale Schraffierung), hlines* (horizontale Schraffierung plus Hintergrundfarbe), crosshatch (Gitternetz), crosshatch* (Gitternetz mit Hintergrundfarbe) und boxfill.
fillcolor color	Die Füllfarbe. Die Hintergrundfarbe für den jeweiligen Füllstil.
hatchwidth dim	Die Breite der Schraffurlinien

Füllstil	Beschreibung
<code>hatchsep dim</code>	Die Breite des Abstandes zwischen den Schraffurlinien
<code>hatchcolor color</code>	Die Farbe der Schraffurlinie
<code>hatchangle angle</code>	Der Winkel der Schraffurlinie
<code>addfillstyle style</code>	Erlaubt die Verwendung von zwei Füllstilen für eine Linie.

Tabelle 7.8: Füllstile

7.2.7 Koordinatensysteme

Das Paket `psricks` benutzt standardmäßig das gleiche Koordinatensystem wie die `picture`-Umgebung. Mit folgenden Parametern kann man das Koordinatensystem beeinflussen:

- `origin {coord}`
Damit lässt sich der Ursprung des Koordinatensystems verschieben.
- `swapaxes true`
Damit können die Achsen des Koordinatensystems vertauscht werden.

7.2.8 Achsen

Mit dem Befehl `\psaxes` lassen sich die Achsen eines Koordinatenpaares zeichnen. Der allgemeine Aufruf lautet:

```
\psaxes[<param>]{<arrow>}( <x0,y0>)( <x1,y1>)(x2, y2)
```

Im folgenden Beispiel werden die Achsen eines Koordinatensystems gezeichnet.

```
\begin{pspicture}( 0.5, 0.5)(3, 2)
\psaxes{ >}(0,0)(3,2)
\end{pspicture}
```

Diese Parameter beeinflussen die Beschriftung der Achsen:

- `0x num, 0y num`
Damit wird die Beschriftung im Ursprung festgelegt (x und y): Standard 0.
- `Dx num, Dy num`
Zählschritte für die Beschriftung (x und y): Standard 1.
- `dx dim, dy dim`
Damit wird der Abstand zwischen den Beschriftungen festgelegt (x und y): Standard 0pt.
- `labels all/x/y/none`
Damit wird festgelegt, welche Achsen beschriftet werden sollen.

- `showorigin true/false`
Damit wird festgelegt, ob der Ursprung beschriftet werden soll.
- `ticks all/x/y/none`
Damit wird festgelegt, welche Achsen mit Teilstrichen beschriftet werden sollen: Standard: `all`.
- `tickstyle full/top/bottom`
Damit wird festgelegt, ob die Teilstriche oberhalb, unterhalb oder auf beiden Seiten der Achsen gezeichnet werden sollen.
- `ticksiz dim`
Damit wird der Abstand der Teilstriche von der Achse festgelegt (Standard: 3pt).
- `labelsep dim`
Damit wird der Abstand zwischen Teilstrichen und den Beschriftungen festgelegt.
- `axesstyle axes/frame/none`
Damit wird festgelegt, ob die Achsen als Achsen, Rahmen oder gar nicht gezeichnet werden sollen.

Mit den beiden Befehlen `\pshlabel` und `\psvlabel` legen Sie die jeweilige Schriftart für die Achsenbeschriftung fest.

7.2.9 Linien, Polygone und Rechtecke

Mit dem Befehl `\psline` wird eine einfache Linie gezeichnet.

Der allgemeine Aufruf lautet:

```
\psline[<param>]{<arrows>}<x0>,<y0>(<x1>,<y1>),...(<xn>,<yn>)
```

Die Linie wird durch einen Polygonzug dargestellt. Über den Parameter `param` können Sie entsprechende Parameter angeben. Diese können Sie auch global über den Befehl `\psset` definieren. Die Tabelle gibt eine Übersicht über die möglichen Parameter:

Parameter	Beschreibung
<code>linewidth dim</code>	Damit wird die Stärke einer Linie festgelegt.
<code>linecolor color</code>	Damit wird die Zeichenfarbe festgelegt.
<code>linearc dim</code>	Damit wird der Radius für die Eckpunkte einer <code>\psline</code> oder eines <code>\pspolygone</code> festgelegt.

Tabelle 7.9: Parameter für `\psline`

Der Parameter `arrows` legt den Anfangs- und den Endpunkt der Linie genauer fest. Mit der Angabe von `»-«` können Sie eine Linie ohne besondere Endpunkte einrichten.

Im folgenden Beispiel wird eine einfache Linie mit drei Koordinaten gezeichnet:

Um die Zeichnung besser nachvollziehen zu können, wird zusätzlich ein Gitternetz mit Koordinaten für den Befehl `\psgrid` über die Zeichnung gelegt:

```
\begin{pspicture}( 0.5,0.5)(4,2)
  \psgrid[subgriddiv 1,griddots 10,gridlabels 7pt](0,0)(4,2)
  \psline[linewidth 2pt]{ }(0,0)(2,2)(4,0)
\end{pspicture}
```

Gerade Linien

Mit dem Befehl `\qline` zeichnen Sie gerade Linien:

```
\qline(coor0)(coor1)
```

Beispiel:

```
\begin{pspicture}( 0.5,0.5)(4,2)
  \qline(0,0)(4,2)
\end{pspicture}
```

Polygone

Mit dem Befehl `\pspolygon` zeichnen Sie einfache Vielecke. Der allgemeine Aufruf lautet:

```
\pspolygon [par](x0,y0)(x1,y1)(x2,y2)... (xn,yn)
```

In den runden Klammern stehen die Koordinatenpaare für die Eckpunkte der Polygone. Ein Vieleck mit n Ecken hat daher n Koordinatenpaare. In den eckigen Klammern finden sich Angaben zur Strichstärke (etwa `linewidth 2pt`), die Linienfarbe (`linecolor red`) oder den Radius der Eckpunkte (`linearc`). Diese Angaben sind jeweils durch Kommata getrennt.

Es gibt auch eine Sternvariante dieses Befehls (`\pspolygon*`). Bei diesem Befehl wird der Innenraum des Vielecks ausgefüllt.

Beispiel:

```
\pspolygon[linewidth 2pt](0,2)(1,2)(3,3)
\pspolygon*[linearc .2,linecolor darkgray](1,0)(1,2)(4,0)(4,2)
```

Rechtecke

Mit dem Befehl `\psframe` können Sie Rechtecke zeichnen. Der allgemeine Aufruf lautet:

```
\psframe [par](x0,y0)(x1,y1)
```

Hier sind lediglich zwei Koordinatenpaare notwendig, nämlich für die Ecke unten links und die Ecke rechts oben. Es gibt auch eine Sternvariante des Befehls (`\psframe*`). Bei dieser wird die umschlossene Fläche des Rechtecks ausgefüllt.

Ein Beispiel:

```
\psframe*[linewidth 3pt,framearc .2,fillstyle solid,
fillcolor lightgray](4,2)
```

Andere Figuren

Neben einfachen normalen geometrischen Objekten können Sie auch zusammengesetzte geometrische Objekte zeichnen:

- `\psdiamond`
Hiermit werden Rautenformen gezeichnet.
- `\pstriangle`
Mit diesem Befehl lassen sich Dreiecke zeichnen.

Beispiele

Hier Beispiele, die Sie ausprobieren sollten:

```
\psdiamond[framearc .2,fillstyle solid,
fillcolor lightgray](2,2)(1.5,1)
```

```
\pstriangle*[gangle 10](2,.4)(4,1)
```

Mit `psdiamond` und `pstriangle` zeichnen Sie eine Raute beziehungsweise ein Dreieck. Beide Befehle haben die Eigenschaften gemein, dass das erste Koordinatenpaar in runden Klammern für den Mittelpunkt der jeweiligen Figur steht und das zweite Paar die Breite und Höhe angibt. Zudem kann ein Parameter namens `gangle` hinzugefügt werden (wie im `pstriangle`-Beispiel zu sehen), der es ermöglicht, die Figur um die angegebene Gradzahl zu drehen. Die Sternvarianten dieser Befehle sind wiederum ausgefüllt.

7.2.10 Kreise und Ellipsen

Weitere einfache geometrische Objekte sind Kreise und Ellipsen. Ein Kreis wird durch einen Mittelpunkt und einen Radius definiert, eine Ellipse durch ihren Mittelpunkt und ihre horizontale sowie vertikale Halbachse. Die entsprechenden Befehle machen sich diese geometrischen Grundlagen zunutze:

- `\pscircle[par](x0,y0){radius}`
Mit diesem Befehl wird ein Kreis gezeichnet. Das Koordinatenpaar legt den Mittelpunkt fest und der Radius wird für den Parameter `radius` eingesetzt.

- `\qdisk(x0,y0){radius}`
Der Befehl ist eine Variante des Kreisbefehls, die einen gefüllten Kreis erzeugt.
- `\pswedge[par](x0,y0){radius}{angle1}{angle2}`
Dieser Befehl zeichnet einen Halbkreis mit dem angegebenen Radius und zwei Winkeln, dem Startwinkel `angle1` und dem Endwinkel `angle2`.
- `\psellipse[par](x0,y0)(x1,y1)`
Dieser Befehl zeichnet eine Ellipse mit den angegebenen Koordinaten. Das erste Koordinatenpaar bezeichnet den Mittelpunkt, `x1` steht für die horizontale und `y1` für die vertikale Halbachse.
- `\psarc[par]{arrows}(x,y){radius}{angleA}{angleB}`
Dieser Befehl zeichnet einen Halbkreis mit Radius und zwei Winkeln.
- `\psarcn[par]{arrows}(x,y){radius}{angleA}{angleB}`
Dieser Befehl ist ähnlich wie `psarc`. Er zeichnet einen Halbkreis mit Radius und zwei Winkeln.
- `\psellipticarc*[par]{arrows}(x0,y0)(x1,y1){angleA}{angleB}`
Der Befehl zeichnet einen elliptischen Halbkreis.

Hier ein paar Beispiele:

```
\pscircle[linewidth 1pt](.5,.5){1.5cm}
```

Damit wird ein Kreis mit dem Radius 1.5 Zentimeter in der Linienstärke 1 pt gezeichnet.

```
\psset{linecolor gray}
\begin{pspicture}( 0.5, 0.5)(3,2)
  \qdisk(1,2){1cm}
\end{pspicture}
```

Hier wird ein ausgefüllter Kreis mit dem Mittelpunkt (1,2) und dem Radius 1 Zentimeter gezeichnet.

```
\pswedge[linecolor gray,linewidth 1pt,fillstyle solid]{2}{0}{70}
```

Damit wird ein elliptischer Bogen in Grau mit der Linienstärke 1 pt gezeichnet.

7.2.11 Kurven

Neben Kreisen und Ellipsen lassen sich mit dem PSTricks-Paket auch Kurven zeichnen. Zu solchen Kurven gehören Bézierkurven und komplexere Kurven:

- `\psbezier[par]{arrows}(x0,y0)(x1,y1)(x2,y2)(x3,y3)`
Damit wird eine Bézierkurve gezeichnet. Die Kurve wird in diesem Fall durch vier Kontrollpunkte beschrieben. Es gibt eine Sternvariante des Befehls (`\psbezier*` mit gleicher Befehlssyntax); hierbei wird die Kurve mit der Linienfarbe ausgefüllt.

- `\parabola[par]{arrows}(x0,y0)(x1,y1)`
Dieser Befehl zeichnet eine Parabel, die am Punkt (x_0, y_0) startet und bei x_1, y_1 ihren Scheitelpunkt erreicht. Auch für diesen Befehl gibt es eine Sternvariante (`\parabola*` mit gleicher Befehlssyntax), die eine ausgefüllte Parabel erzeugt.
- `\pscurve[par]{arrows}(x1,y1)... (xn,yn)`
Dieser Befehl beschreibt eine Kurve durch die angegebenen Punkte.
- `\psecurve[par]{arrows}(x1,y1)... (xn,yn)`
Dieser Befehl ist wie `pscurve`, das erste und das letzte angegebene Koordinatenpaar werden jedoch nur als Kontrollpunkte interpretiert und nicht gezeichnet.
- `\psccurve[par]{arrows}(x1,y1)... (xn,yn)`
Dieser Befehl zeichnet eine geschlossene Kurve, die durch die angegebenen Punkte bestimmt wird.

Hier ein paar Beispiele:

```
%Eine Bézierkurve
\psbezier[linewidth 1pt,showpoints true]{>}(0,0)(1,2)(2,2)(4,3)

%Eine einfache ausgefüllte Parabel
\parabola*(1,1)(2,2)
\psset{xunit .01}
\parabola{<>}(200,3)(200,0)
```

Anbei ein Beispiel einer Bézierkurve, die in einem einfachen Koordinatensystem gezeichnet wird (der Befehl `\psgrid` wird weiter unten in Abschnitt 7.2.13 näher erläutert):

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\usepackage{pstricks}

\begin{document}
  \psset{unit 1cm}
  \begin{pspicture}(0, 0)(10,10)
    \psbezier(3,6)(1,4)(3,2)(7,6)
    \psgrid(0,0)(0,0)(10,10)
  \end{pspicture}
\end{document}
```

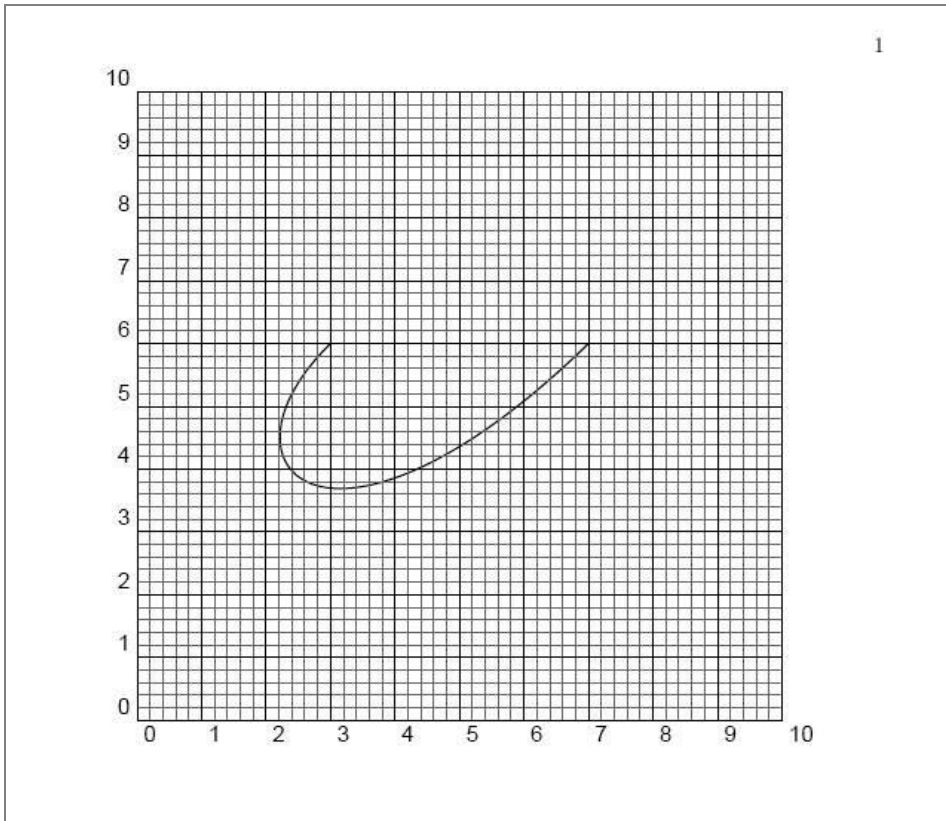


Abbildung 7.19: Simple Bézierkurve in einem Koordinatensystem

7.2.12 Punkte

Mit den Befehlen `\psdot` und `\psdots` lassen sich Punkte unterschiedlichen Stils gestalten. Mit dem Parameter `dotstyle style` können Sie den Stil der Punkte festlegen. Das Paket `psstricks` bietet eine Fülle von verschiedenen Punktformen:

- `\psdot[par](x1,y1)`
Zeichnet einen einzelnen Punkt.
- `\psdots[par](x1,y1)(x2,y2)... (xn,yn)`
Zeichnet eine Punktmenge, die durch die angegebenen Punktkoordinaten bestimmt wird.

Mit dem Parameter `dotsize dim` können Sie die Größe der Punkte beeinflussen. Der Parameter `dotscale num1 num2` streckt die Punkte horizontal (`num1`) und vertikal (`num2`). Der Parameter `dotangle angle` beeinflusst den Winkel der Punkte. Das ist vor

allem dann sinnvoll, wenn der Punkt als kleines Dreieck oder eine andere geometrische Figur dargestellt werden soll.

7.2.13 Grid

Mit dem Befehl `\psgrid` wird ein Gitternetz (Grid) gezeichnet.

- `\psgrid(x0,y0)(x1,y1)(x2,y2)`
Der Befehl `\psgrid` zeichnet ein Grid mit den opponierenden Ecken $(x1,y1)$, $(x2,y2)$. Die Koordinaten sind immer kartesisch.

Die Tabelle gibt eine Übersicht über die Parameter von `\psgrid`:

<i>Parameter</i>	<i>Beschreibung</i>
<code>gridwidth dim</code>	Breite des Grids
<code>gridcolor color</code>	Farbe des Grids
<code>griddots num</code>	Anzahl der Gridpunkte
<code>gridlabels dim</code>	Anzahl der Gridlabels
<code>subgriddiv</code>	Die Anzahl der Gridunterteilungen
<code>Subgridcolor color</code>	Die Farbe der Gridunterteilungen
<code>Subgridwidth dim</code>	Die Breite der Gridunterteilungen
<code>subgriddots num</code>	Anzahl der Punkte der Gridunterteilungen

Tabelle 7.10: Parameter des Befehls `\psgrid`

Hier ein Beispiel:

```
\psgrid[subgriddiv 1,griddots 20,gridlabels 7pt]( 2, 2)(3,1)
```

7.2.14 Plots

Mit den Befehlen `\psdots`, `\psline`, `\pspolygon`, `\pscurve`, `\psecurve` und `\psccurve` können Sie Grafiken plotten. Zusätzlich zum Paket `pstricks` müssen Sie hierzu aber noch das Plotting-Paket `pst plot` einbinden. Das können Sie in einer Zeile erledigen:

```
\usepackage{pstricks,pst plot}
```

Generell werden beim Plotten zunächst die Daten generiert und die Koordinaten angegeben. Danach wird die eigentliche Zeichnung ausgeführt. Mit dem Parameter `plotstyle style` können Sie verschiedene Kurven plotten.

Zulässige Parameter sind `line`, `curve`, `polygon`, `ecurve` und `ccurve`.

Plotting-Funktionen

Mit den folgenden Plotting-Funktionen können Sie einfache Daten drucken:

- `\fileplot[par]{file}`
Der Befehl `\fileplot` ist der einfachste Plot-Befehl. Sie benötigen lediglich eine Datendatei, wie sie z. B. von Mathematic oder Maple generiert wird. Der Dateiname wird dann in den geschweiften Klammern eingetragen.
- `\dataplot[par]{commands}`
Der Befehl `\dataplot` funktioniert ähnlich wie der Befehl `\fileplot`. Auch hier können Sie eine Datei als Datenmenge angeben. Die Daten müssen allerdings erst über die Befehle `\savedata{command}[data]` generiert beziehungsweise mit `\readdata{command}[file]` aus einer Datei ausgelesen werden.
- `\listplot[par]{list}`
Der Befehl `\listplot` ist ein weiterer Befehl, um Listen von Daten zu plotten. In diesem Fall werden die Daten in Form von Koordinatenpaaren geplottet.
- `\psplot`
Dieser Befehl kann dazu benutzt werden, eine Funktion $f(x)$ zu plotten. Die Funktion $f(x)$ muss dabei mit PostScript berechnet werden. Beispiele für $f(x)$ sind $\sin(x)$ und $\cos(x)$ usw.
- `\parametricplot`
Dieser Befehl wird dazu benutzt, die parametrisierte Funktion $x(t)$ oder $y(t)$ zu plotten.

Das folgende Beispiel zeigt, wie die Befehle `\readdata` und `\dataplot` eingesetzt werden. Der Befehl `\readdata` liest die Datei `foo.data` bzw. die Datei `bar.data`. Der Befehl `\dataplot` plottet dann die Daten von `data2`.

```
\readdata{\data1}{foo.data}
\readdata{\data2}{bar.data}
\dataplot{\data1\data2}
\dataplot[origin {0,1}]{\data2}
```

Das folgende Beispiel stellt den Befehl `\listplot` vor. Der Befehl funktioniert analog zu `\dataplot`; auch er greift auf Punktdaten zurück, die zuvor per `\savedata` in einer Datei hinterlegt wurden.

```
\listplot[plotstyle curve,showpoints true,
dotstyle triangle]{\mydata}
```

Abschließend noch ein Beispiel für den Befehl `\psplot`:

```
\psplot[plotpoints 200]{0}{720}{x cos}
```

Der Befehl `\psplot` plottet hier eine Kosinus-Funktion im Intervall von 720 Grad.

Hier noch ein Beispiel für eine Kurve, die in einem kleinen Koordinatensystem geplottet wird:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\usepackage{pstricks, pst plot}
\begin{document}
\psset{unit 1cm}
\savedata{\mydata}[{{0,0}{0.5,1}{1,2}{1.5,1.5}{2,2}{2.5,1.5}{3,2}}]
\begin{pspicture}(0, 0)(10,10)
\psgrid(0,0)(3,2)
\listplot[linewidth 1pt, plotstyle curve]{\mydata} \end{pspicture}
\end{document}
```

Mit dem Befehl `\savedata{\mydata}` werden zunächst die einzelnen Punktkoordinaten in einer Art Datei zwischengespeichert. `\mydata` dient hier quasi als Dateiname oder Label, der es `\listplot` erlaubt, die Daten einzulesen und die Punktkoordinaten zu zeichnen. Anstelle von `\listplot` hätte hier auch `\dataplot` stehen können, die Befehle werden in diesem Zusammenhang praktisch identisch angewandt.

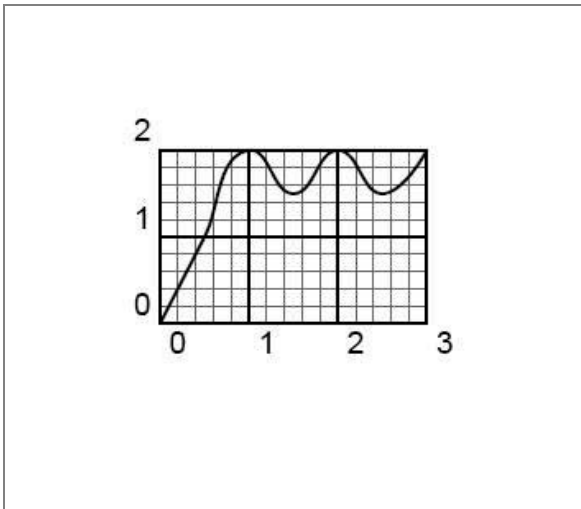


Abbildung 7.20: Mit dem `listplot`-Befehl erzeugte Kurve

7.2.15 Pfeile

Linien und andere Objekte können mit Pfeilen ausgestattet werden. Es gibt verschiedene Formen von Pfeilen, die Sie mit PSTricks formatieren können.

Die Tabelle gibt eine Übersicht über die verschiedenen Befehle und die korrespondierenden Pfeile:

Befehl	Beschreibung
	Kein Pfeil
< >	Pfeilspitzen
> <	Umgekehrte Pfeilspitzen
<< >>	Doppelte Pfeilspitzen
>> <<	Doppelte umgekehrte Pfeilspitzen
	Balken am Linienende
* *	Zentrierte Balken am Linienende
< >	Balken und Pfeilspitzen am Linienende
[]	Eckige Klammern am Linienende
()	Runde Klammern am Linienende

Tabelle 7.11: Befehle für Pfeilspitzen

Es gibt noch eine Reihe anderer Befehle, mit denen sich verschiedene Pfeile darstellen lassen. So können Sie z. B. durch die Angabe von `c c`, `cc cc` gerundete Ecken oder erweiterte runde Ecken an der Linie erzeugen.

Weitere Informationen über Pfeilspitzen finden Sie in der Dokumentation von PSTricks.

Parameter für Pfeilspitzen

Um Pfeile anzupassen, gibt es noch eine Reihe anderer Parameter.

Die Tabelle gibt eine Übersicht über diese Parameter:

Parameter	Beschreibung
<code>arrowsize dim</code>	Die Größe der Pfeilspitze ist <code>dim</code> plus Größe der Linienbreite.
<code>arrowlength num</code>	Die Länge der Pfeilspitze.
<code>arrowinset num</code>	Die Gesamtgröße der Pfeilspitze.

Parameter	Beschreibung
<code>tbar size dim</code>	Die Größe einer eckigen, geschweiften oder runden Klammer.
<code>bracket length num</code>	Die Höhe einer eckigen Klammer.
<code>rbracket length num</code>	Die Höhe einer abgerundeten eckigen Klammer.
<code>arrowscale arrowscale num num</code>	Damit werden die Breite und die Höhe des Pfeils über die <code>num</code> -Parameter angegeben.

Tabelle 7.12: Parameter für Pfeile

7.2.16 Eigene Objekte und Stile

Neben den von PSTricks vorgegebenen Stilen lassen sich Objekte und Stile über `\newpsobject` anpassen. Der allgemeine Aufruf lautet:

```
\newpsobject{name}{objekt}{parameter wert,...}
```

Der erste Parameter ist der Befehl, den Sie erzeugen wollen. Der zweite Parameter ist das jeweilige Objekt, das Sie verändern wollen. Über den dritten Parameter können Sie verschiedene Werte angeben.

Hier ein Beispiel für einen abgewandelten Befehl:

```
\newpsobject{myline}{psline}{linecolor blue,linestyle dotted}
\newpsobject{mygrid}{psgrid}{subgriddiv 1,griddots 10,
gridlabels 7pt}
```

In diesem Fall wird der Befehl `myline` als neue Variante des Befehls `psline` erzeugt. Die Linie selbst wird in der Farbe Blau als gestrichelte Linie angezeigt. Im zweiten Beispiel wird der Befehl `mygrid` erzeugt, der ein neues Gitternetz über den Befehl `psgrid` bildet.

Der Befehl `myline` kann dann wie folgt aufgerufen werden:

```
\myline[linecolor gray,dotsep 2pt](5,6)
```

Eine weitere Möglichkeit, neue Stile zu setzen, ist der Befehl `\newpsstyle`, den Sie so aufrufen:

```
\newpsstyle{mystyle}{linecolor blue,linestyle dotted}
\psline[style mystyle](5,6)
```

7.2.17 Eigene Grafiken

Mit den normalen PSTricks-Befehlen können Sie eine ganze Reihe von vordefinierten grafischen Objekten schaffen. Manchmal gibt es jedoch auch Situationen, in denen man eigene Grafiken erarbeiten möchte oder die vorhandenen Grafiken nicht ausreichen.

In diesem Fall erstellen Sie eigene Grafiken in Form eines Pfades. Ein Pfad ist wie eine mathematische Linie mit zwei Endpunkten zu verstehen und kann verschiedene miteinander verbundene oder nicht verbundene Objekte haben. Der Pfad selbst wird über den Befehl `\pscustom` erzeugt. Der allgemeine Aufruf lautet:

```
\pscustom[parameter]{Befehle}
```

Für den ersten Parameter können Sie verschiedene Werte, für den zweiten jeden möglichen PSTricks-Befehl angeben.

Die Tabelle gibt eine Übersicht über verschiedene Parameter:

Parameter	Beschreibung
0	Offene Kurve mit Pfeilen
1	Offene Kurve mit einem Pfeil am Anfang
2	Offene Kurve mit einem Pfeil am Ende
3	Offene Kurve mit Pfeilen an beiden Enden
1	Geschlossene Kurve ohne symmetrische Segmente
n >1	Geschlossene Kurve mit n symmetrischen Segmenten

Tabelle 7.13: Parameter für den Befehl `\pscustom`

Sichere Tricks

Die folgenden Befehle können Sie sicher in jeder LaTeX-Umgebung benutzen, ohne dass Sie irgendwelche PostScript-Fehlermeldungen bekommen. Alle unten stehenden Befehle können Sie mit `\pscustom` aufrufen:

- `\newpath`
Damit werden der Pfad und der aktuelle Punkt gelöscht.
- `\moveto(koordinaten)`
Bewegt den aktuellen Punkt zu dem angegebenen Koordinatenpaar (x, y).
- `\closepath`
Damit wird der Pfad geschlossen, indem Anfang und Ende miteinander verbunden werden.
- `\stroke[par]`
Damit wird der Pfad gezeichnet. Der Befehl `\pscustom` zeichnet den Pfad automatisch. Sie können den Befehl auch zweimal aufrufen, um z. B. einen Rand zu zeichnen.

- `\fill[par]`
Damit wird die Region gefüllt. `\pscustom` füllt die Region automatisch. Über den Parameter `par` können Sie verschiedene Werte angeben.
 - `\gsave`
Damit wird der aktuelle Grafikzustand gesichert, z. B. Pfad, Linienbreite, Farbe, Koordinaten usw. Der Befehl `\grestore` stellt den Grafikzustand wieder her. `\gsave` und `\grestore` müssen beide hintereinander aufgerufen werden.
 - `\translate(x, y)`
Damit wird das Koordinatensystem um die Punkte (x, y) verschoben. Alle Punkte werden um das Koordinatenpaar x, y verschoben. Aktuell gezeichnete Objekte sind davon nicht betroffen.
 - `\scale{num num}`
Damit wird das Koordinatensystem um `num1` horizontal oder `num2` vertikal skaliert.
 - `\rotate{angle}`
Rotiert das Koordinatensystem um den angegebenen Winkel.
 - `\swapaxes`
Vertauscht die x - und y -Koordinaten.
 - `\msave`
Damit wird das aktuelle Koordinatensystem gespeichert. Mit `\rmsave` wird das Koordinatensystem wieder hergestellt. `\msave` und `\mrestore` können geschachtelt werden.
 - `\openshadow[par]`
Zeichnet einen Schatten für den aktuellen Pfad, indem der Pfad zweimal gezeichnet wird. In den eckigen Klammern können Sie verschiedene Parameter angeben.
 - `\closedshadow[par]`
Zeichnet einen Schatten für die Region, die vom Pfad eingeschlossen wird, als wenn sie eine durchsichtige Region wäre.
 - `\movepath(koordinaten)`
Bewegt den Pfad um das Koordinatenpaar (x, y) . Sie können `\msave` und `\mrestore` benutzen, um den ursprünglichen Pfad wieder herzustellen.
- Neben diesen Befehlen gibt es noch weitere Befehle, die Sie mit `\pscustom` benutzen können, solange es einen aktuellen Punkt gibt.
- `\lineto(koordinaten)`
Dieser Befehl ist eine Kurzfassung des Befehls `\psline`. Damit wird eine Linie zu dem angegebenen Koordinatenpaar x und y gezeichnet.

- `\rlineto(koordinaten)`
Dieser Befehl funktioniert ähnlich wie `\lineto`, allerdings wird die Linie vom aktuellen Punkt zu den relativen Koordinaten gesetzt.
- `\curveto(x1,y1)(x2,y2)(x3,y3)`
Dieser Befehl funktioniert ähnlich wie der Befehl `\psbezier`. Er zeichnet eine Kurve über die gegebenen Koordinatenpaare.
- `\rcurveto(x1,y1)(x2,y2)(x3,y3)`
Dieser Befehl ist ähnlich wie `\curveto`. Die Koordinaten werden jedoch relativ zum aktuellen Punkt bezogen.

7.2.18 Hackertricks

Die folgenden Befehle sind Tricks für »Hacker«, die weitere Möglichkeiten ausnutzen wollen. Sie bieten die Möglichkeit, direkt auf den PostScript-Code einzuwirken, können aber zu Fehlern führen.

- `\code{code}`
Damit können Sie den originalen PostScript-Code einfügen.
- `\dim{dim}`
Damit werden PSTricks-Dimensionen in pts umgewandelt und in den PostScript-Code eingefügt.
- `\coor(x1, y1)(x2, y2)(x3, y3)`
Damit werden Koordinatenpaare in Zahlen konvertiert und in den PostScript-Code eingefügt.
- `\rcoor(x1, y2)(x2, y2)(x3, x3)`
Dieser Befehl ist ähnlich wie `\coor`, jedoch werden die Koordinaten in umgekehrter Reihenfolge eingefügt.
- `\file{Datei}`
Dieser Befehl funktioniert ähnlich wie der Befehl `\code`, jedoch wird der Code »wörtlich« übernommen.
- `\arrows{Pfeile}`
Dieser Befehl definiert die Operatoren `ArrowA` und `ArrowB`, sodass jeweils ein Pfeil gezeichnet wird.
- `\setcolor{Farbe}`
Damit wird die Farbe gesetzt.

8 Aufzählungen

Aufzählungen und Listen sind eine Möglichkeit, Inhalte strukturiert darzustellen. Eine Aufzählung in LaTeX wird ähnlich wie eine Liste dargestellt, d.h. Aufzählungen basieren auf Listen. In Kombination mit Symbolen lassen sich sehr aussagekräftige Listen erstellen. Außerdem können Sie Listen auch mit Zahlen nummerieren.

8.1 Einfache Aufzählungen

Mit der Umgebung `\itemize` lassen sich ganz einfache Aufzählungen erstellen. Die Umgebung `\itemize` wird wie folgt aufgerufen:

```
\begin{itemize}
\item
\item
\end{itemize}
```

Hier eine einfache Liste:

```
\begin{document}
Ein Dokument mit einer Liste:
\begin{itemize}
\item Teil 1
\item Teil 2
\item Teil 3
\end{itemize}
\end{document}
```

Beispiel 8.1: Einfache Liste

Die Listenumgebung wird dabei, wie andere Befehle auch, einfach in das Hauptdokument eingebunden.

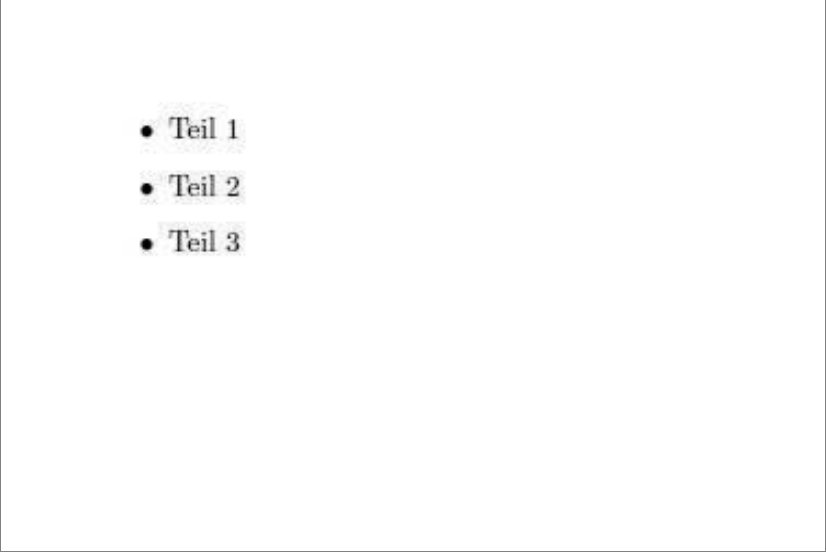
- 
- Teil 1
 - Teil 2
 - Teil 3

Abbildung 8.1: Einfache Liste

Die Liste wird in Form von einfachen Punkten, gefolgt von dem Text, dargestellt. Als Alternative können Sie auch andere Symbole angeben. Das folgende Beispiel zeigt eine Liste mit unterschiedlichen Punktformen:

```
\begin{document}  
\begin{itemize}  
\item Teil 1  
\item [*] Teil 2  
\item \dots  
\end{itemize}  
\end{document}
```

Beispiel 8.2: Liste mit verschiedenen Punkten

Das Ergebnis sieht so aus:

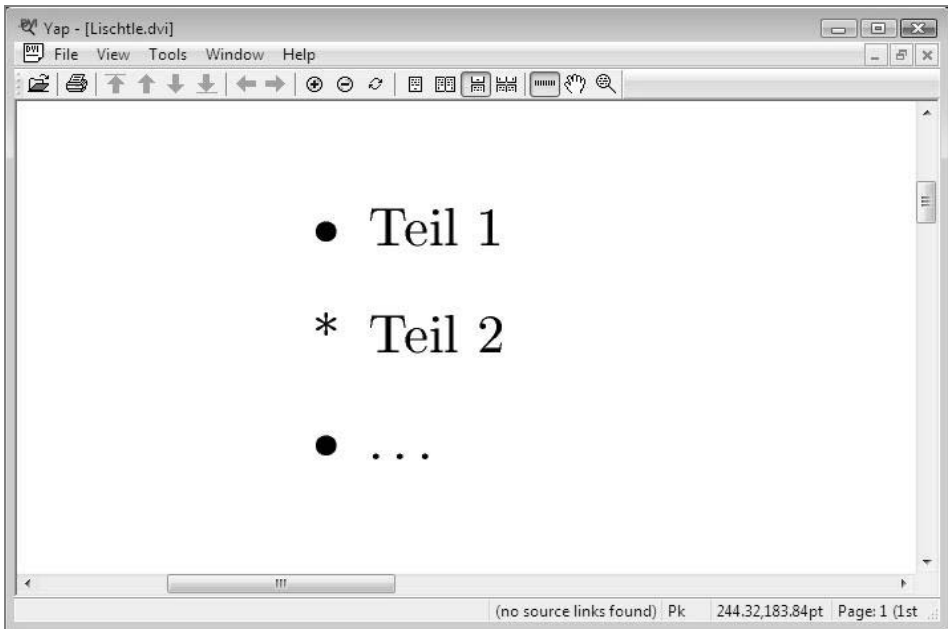


Abbildung 8.2: Ergebnis zu Beispiel 8.2

Fortlaufende Nummerierungen

Mit der Umgebung `enumerate` können Sie eine fortlaufende Nummerierung in die Liste einbauen. Damit wird dann vor jeden Listenelement eine Zahl gesetzt:

Die Umgebung `enumerate` wird wie folgt aufgerufen:

```
\begin{document}
\begin{enumerate}
\item Teil 1
\item Teil 2
\end{enumerate}
\end{document}
```

Beispiel 8.3: Liste mit Nummerierung



Abbildung 8.3: Ergebnis zu Beispiel 8.3

Bei der Umgebung `enumerate` können Sie auch verschiedene Nummerierungsformen über den Parameter `[*]` oder den Befehl `\dots` angeben.

```
\begin{enumerate}
\item Punkt eins
\item [*] Punkt zwei geändert
\item \dots
\end{enumerate}
```

Die Verwendung des Sternparameters in eckigen Klammern sorgt dafür, dass anstelle der Nummerierung ein Sternsymbol erscheint. Die fortlaufende Nummerierung wird dann erst beim folgenden `\item`-Eintrag fortgesetzt. Wenn Sie den Befehl `\item` zusammen mit `\dots` verwenden, setzt LaTeX hinter den Aufzählungspunkt drei Punkte (...).

Auf eine Aufzählungsnummer verweisen

Mit dem Befehl `\label` verweisen Sie auf eine Aufzählungsnummer. Der Verweis auf die entsprechende Markierung erhält dann die jeweilige Aufzählungsnummer:

```
\begin{document}
\begin{enumerate}
\item Bus \label{enum:Bus}
\item Bahn
\item Fahrrad
\end{enumerate}
\end{document}
```

Das Ergebnis zeigt folgende Abbildung:

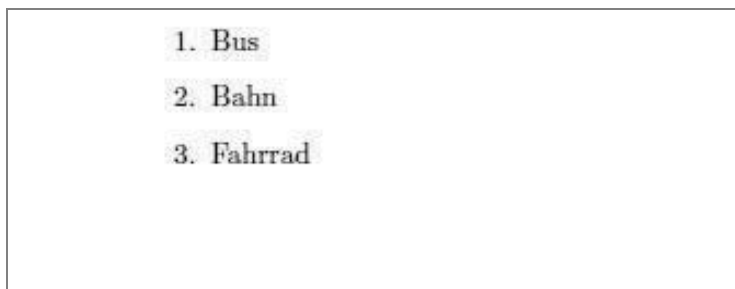
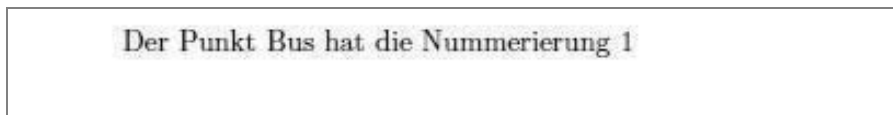
- 
1. Bus
 2. Bahn
 3. Fahrrad

Abbildung 8.4: Ergebnis zu Beispiel 8.4

Mit dem Befehl `\ref` können Sie dann an einer späteren Stelle des Dokumentes auf die Nummerierung zugreifen:

```
Der Punkt Bus hat die Nummerierung \ref{enum:Bus}
```



Der Punkt Bus hat die Nummerierung 1

Abbildung 8.5: Der Befehl `\ref`

Beschreibungen für Auflistungen

Mit der Umgebung `description` können Sie einen beschreibenden Text für einen Auflistungspunkt erzeugen. Das bietet sich unter anderem an, wenn Sie eine Liste von Stichwörtern erstellen möchten, denen Beschreibungen zugeordnet sein sollen. Der allgemeine Aufruf lautet:

```
\begin{description}
\item [Auflistungstext] Text
\end{description}
```

Hier ein Beispiel für die Umgebung `description`:

```
\begin{document}
\begin{description}
\item [Bus] Ein Bus hat 4 Räder
\item [Bahn] Eine Bahn fährt auf Schienen
\end{description}
\end{document}
```

Das Ergebnis zeigt folgende Abbildung:

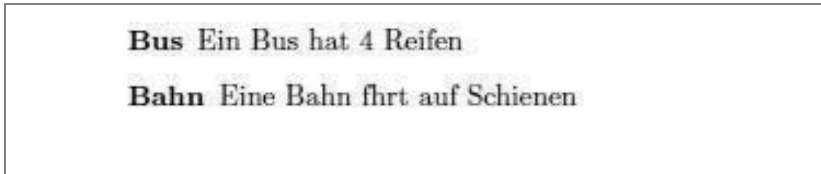


Abbildung 8.6: Beschreibende Listen

8.1.1 Listen verschachteln

Die diversen Auflistungen lassen sich ineinander verschachteln. Ebenso können Sie die verschiedenen Listenformen untereinander mischen. Wenn Sie Listen verschachteln, müssen Sie allerdings jedes Mal die Listen-Umgebung neu aufrufen:

```
\begin{itemize}
\item Teil 1
\item Teil 2
\begin{itemize}
\item Teil 3
\item Teil 4
\end{itemize}
\end{itemize}
```

Beispiel 8.4: Verschachtelte Liste

Das Ergebnis sieht so aus:

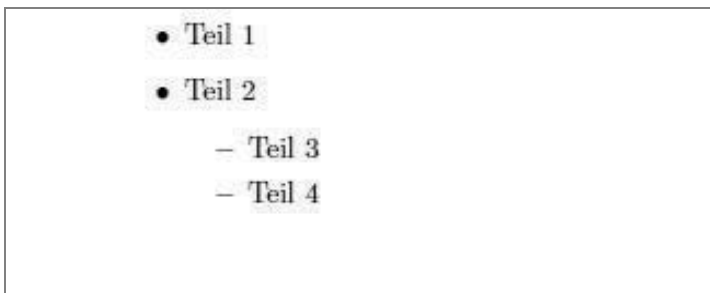


Abbildung 8.7: Verschachtelte Listen

Hierbei muss man darauf achten, dass die Umgebungen jeweils richtig wieder geschlossen werden, andernfalls meldet LaTeX einen Fehler.

Verschiedene Aufzählungspunkte

Das Erscheinungsbild der einzelnen Aufzählungspunkte sowie die Nummernformatierung können Sie über verschiedene Befehle ändern. Dafür stehen die Kommandos `\labelitem...` und `\labelenum...` zur Verfügung. Den Kommandos werden die kleinen römischen Zahlen i, ii, iii und iv nachgestellt, die die jeweilige Gliederungsebene angeben. Hier sehen Sie eine Übersicht über die verschiedenen Befehle:

```
\labelitemi
\labelitemii
\labelitemiii
\labelitemiv
```

`\labelitem...` bezieht sich auf Listensymbole. Wenn es um die Definition der Zahlensymbole für die fortlaufende Nummerierung geht, wird das Kommando `\labelenum...` verwendet:

```
\labelenumi
\labelenumii
\labelenumiii
\labelenumiv
```

Zudem spielen noch die Kommandos `\enumi` bis `\enumiv` eine wichtige Rolle:

```
\enumi
\enumii
\enumiii
\enumiv
```

Wie die Befehle eingesetzt werden, sehen Sie hier. Mit dem Befehl `\renewcommand` ändern Sie die Markierungen. Die betreffende Gliederungsebene wird durch die Befehle `\labelitemi` bis `\labelitemiv` angegeben. Darauf folgt in geschweiften Klammern das gewünschte Symbol:

```
\renewcommand{\labelitemi}{+}
```

In diesem Beispiel wird als Aufzählungszeichen für die oberste Gliederungsebene ein Pluszeichen angezeigt. Bei folgendem Beispiel

```
\renewcommand{\labelitemi}{ }
```

wird ein Spiegelstrich als Aufzählungssymbol gesetzt.

Wenn Sie andere Aufzählungssymbole bzw. Aufzählungszeichen, z. B. römische oder arabische Ziffern, benutzen wollen, wählen Sie folgende Definition:

```
\renewcommand{\labelenum}{\Roman{enumi}}
```

Dadurch werden römische Ziffern für die Nummerierung der Aufzählung verwendet.

Hier noch zwei Beispiele für geänderte Aufzählungspunkte:

```
\renewcommand{\labelitemi}{ }
\renewcommand{\labelitemii}{+}
\renewcommand{\labelitemiii}{*}

\begin{itemize}
\item Eins
\item Zwei
\begin{itemize}
\item Drei
\item Vier
\end{itemize}
\end{itemize}
```

Hier werden Einträge der ersten Gliederungsebene mit einem Spiegelstrich, die der zweiten Ebene mit einem Pluszeichen und die dritte Ebene mit einem Sternsymbol gekennzeichnet.

Im folgenden Beispiel wird die Nummerierung über den Befehl `\labelenumi` geändert:

```
\renewcommand{\labelenumi}{\Roman{enumi}}
\renewcommand{\labelenumii}{\Roman{enumi} \arabic{enumii}}
\renewcommand{\labelenumiii}{\Roman{enumi} \arabic{enumii}\Alph{enumiii}}

\begin{enumerate}
\item Eins
\item Zwei
\begin{enumerate}
\item Zwei Eins
\item Zwei Zwei
\begin{enumerate}
\item Zwei Zwei Eins
\item Zwei Zwei Zwei
\end{enumerate}
\end{enumerate}
\end{enumerate}
```

Was passiert hier? Die oberste Gliederungsebene erhält römische Ziffern (I), die zweite kombiniert römische und arabische Ziffern (I-1) und die dritte setzt sich aus einer römischen und einer arabischen Zahl sowie einem Buchstaben zusammen (I-2A). Die Befehle `\enumi` bis `\enumiv` werden herangezogen, um die Nummernzeichen für die ein-

zelen Gliederungsebenen der fortlaufenden Nummerierung zu definieren. Bei einer Zahlenkombination II-2A definiert `\enumi` die erste Stelle (hier die römische II), `\enumii` die zweite Ziffer (die arabische 2) und `\enumiii` die dritte Stelle (hier den Buchstaben A). Hier das Ergebnis:

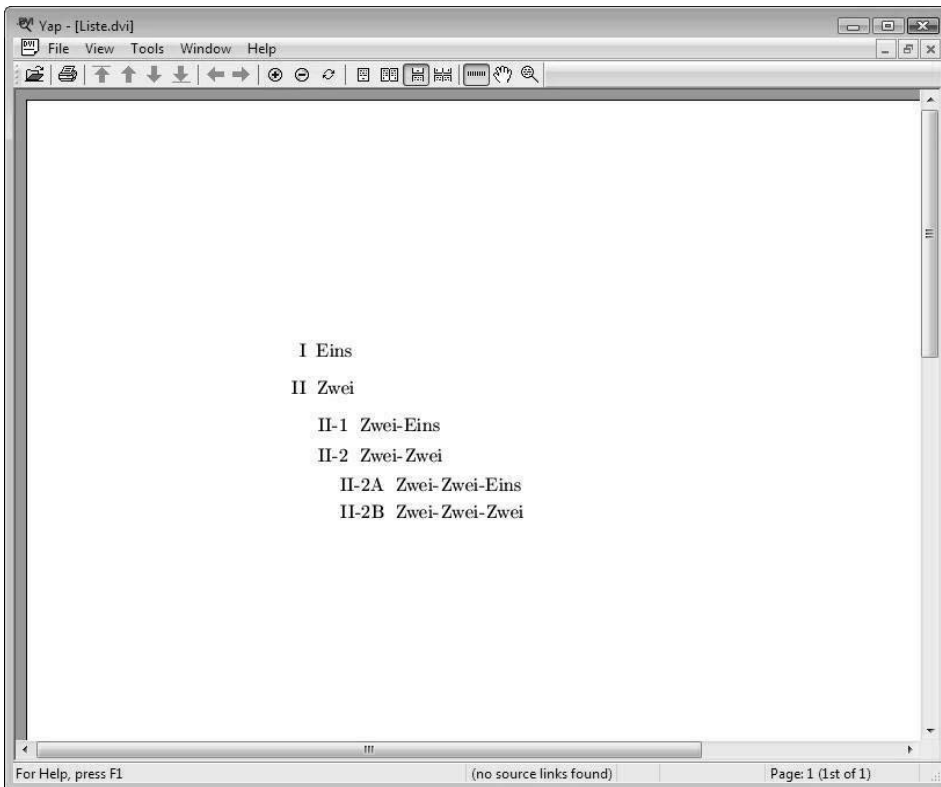


Abbildung 8.8: Aufzählungsebenen im Druckbild

Den Befehlen `\enumi` bis `\enumiv` können diese Varianten zugeordnet werden:

- `\arabic` für arabische Ziffern
- `\roman` für kleine römische Ziffern (i, ii, iii)
- `\Roman` für große römische Ziffern (I, II, III)
- `\alph` für fortlaufende Kleinbuchstaben (a, b, c)
- `\Alph` für fortlaufende Großbuchstaben (A, B, C)

9 Eigene Listen

Außer Standardlisten lassen sich auch eigene Listen einrichten. Alle Listen in LaTeX basieren auf der Listenumgebung `list`. Über den Befehl `\newcommand` können Sie eigene Listen anlegen und Ihren Wünschen gemäß anpassen.

9.1 Die Umgebung `list`

Die Grundumgebung für alle Listen ist `list`. Sie wird wie folgt aufgerufen:

```
\begin{list}{<standardmarke>}{<Listenerklärungen>}
\item {<optional>} Text
\end{list}
```

Der erste Parameter ist die Standardmarke, d. h. der Aufzählungspunkt. Diesen können Sie bei Bedarf mit dem optionalen Parameter überschreiben. In der Listenerklärung ändern Sie dann die Standardeinstellungen für die Liste entsprechend. Dazu gehören unter anderem die Formatierung, die Zahlen sowie die entsprechenden Abstände.

Die Abstände lassen sich mit der Methode `\setlength{\rightmargin}{1cm}` entsprechend setzen.

Nachfolgend finden Sie die Befehle für die Umgebung `list`:

- `\topsep`
Das ist der Zwischenraum zwischen dem vorangegangenen Text und dem ersten Aufzählungszeichen. Außerdem wird der Abstand `\parskip` und `\partopsep` addiert.
- `\parskip`
Das ist der normale Absatzabstand. Diesen können Sie im Dokumentkopf definieren.
- `\partopsep`
Wenn vor oder nach der `list`-Umgebung eine Leerzeile ist, wird zusätzlich der Abstand `\partopsep` eingefügt.
- `\parsep`
Das ist der Abstand zwischen zwei Absätzen in der Aufzählung.

- `\itemsep`
Das stellt den Abstand zwischen den Listenpunkten dar. Dieser wird zusätzlich zu `\parsep` addiert.
- `\leftmargin`
Der linke Rand zum Text
- `\rightmargin`
Der rechte Rand zum Text
- `\itemindent`
Der Abstand, um den die erste Zeile des ersten Abstandes eingerückt wird.
- `\labelwidth`
Die Breite des Aufzählungspunktes
- `\labelsep`
Der Abstand zwischen Aufzählungspunkt und der eingerückten ersten Zeile.

Die folgende Abbildung veranschaulicht die oben genannten Dimensionen der `list`-Umgebung:



Abbildung 9.1: Dimensionen der `list`-Umgebung

Mit dem Befehl `\makeatlabel` vor der `list`-Umgebung können Sie die Marke bzw. den Aufzählungspunkt setzen. Dieser lässt sich umdefinieren, wenn das Aussehen geändert werden soll.

Im folgenden Beispiel wird eine Aufzählung mit neuem Aussehen definiert. Wir wollen, dass den einzelnen Einträgen jeweils die Markierung »Punkt 1«, »Punkt 2« und so fort vorangeht. Der verwendete Zähler wird vorher mit

```
\newcounter{neuerzaehler}
```

entsprechend gesetzt. Um den Zähler zu erhöhen, müssen Sie den Befehl `\usecounter` benutzen.

Das Beispiel lautet dann so:

```
\newcounter{neuerzaehler}
\begin{list}{%
  \textbf{Punkt \arabic{enumi}:}}{%
  \usecounter{enumi}
  \setlength{\labelwidth}{3cm}
  }
\item Essen
\item Duschen
\item Anziehen
\item Ins Büro gehen
\end{list}
```

Nach dem Kompilieren sieht das Ergebnis folgendermaßen aus:

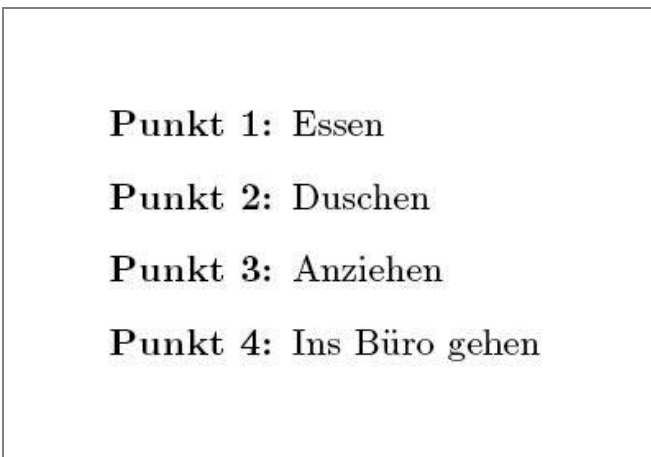


Abbildung 9.2: To-do-Liste nach dem Aufstehen

9.1.1 Neue Listenumgebungen

Auf Basis der Umgebung `list` lassen sich erweiterte Listenumgebungen definieren, die sich an Ihre Bedürfnisse noch besser anpassen lassen. Mit den Befehlen `\newcommand` und `\newenvironment` können Sie sich eigene Listenumgebungen erstellen:

```
\newcommand{}{}
\newenvironment{mylist}{}{}
```

Der Befehl `\newenvironment` erstellt eine neue Listenumgebung, und mit `\setlength` lassen sich weitere Eigenschaften der Liste verändern. In dem folgenden Beispiel wird eine neue Liste erzeugt, die als Listenelement einen Rechtspfeil hat:

```
\documentclass[a4paper]{article}
\usepackage{calc}
\usepackage{ifthen}
\newcommand{\myitemlabel}{$\rightarrow$}
\newenvironment{myitem}{%
  \begin{list}{}{%
    \setlength{\labelwidth}{%
      \widthof{\myitemlabel}}%
    \setlength{\itemsep}{0.3em plus0.3em}%
    \setlength{\labelsep}{0.7\labelwidth}%
    \setlength{\partopsep}{0ex plus0.2ex}%
    \setlength{\leftmargin}{%
      \labelwidth+\labelsep}%
    \setlength{\rightmargin}{0pt}%
    \setlength{\topsep}{0pt}%
    \setlength{\listparindent}{0pt}%
    \renewcommand{\makelabel}[1]{%
      \ifthenelse{\equal{##1}{} }{%
        \myitemlabel%
      }{%
        \parbox[b]{\labelwidth}{%
          \vspace{\parsep}%
          \myitemlabel%
          \makebox[0pt][l]{%
            \hspace{\labelsep}%
            \textbf{\strut##1}\hfill\mbox{}}}%
        }%
      }%
    }%
  }%
}
```

```
}%
\begin{document}
\begin{myitem}
\item [Heute] Brot kaufen
\item Salat machen
\item Wein trinken
\item [Morgen] ausschlafen
\item nichts tun
\end{myitem}
\end{document}
```

Beispiel 9.1: Eigene Listendefinition

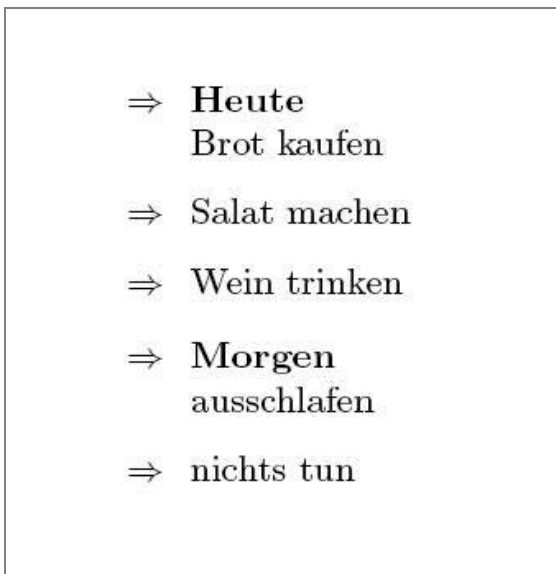


Abbildung 9.3: Guter Plan fürs Wochenende

In diesem Beispiel wird eine neue Liste definiert, bei der die Listenangaben in eckigen Klammern fett dargestellt und die übrigen Punkte mit einem rechten Pfeil versehen werden.

In Zeile 2 wird das Ergänzungspaket `calc` sowie in Zeile 3 das Ergänzungspaket `ifthen` eingebunden.

Dann wird mit `\newcommand` der Befehl `\rightarrow` neu definiert. Anschließend wird die neue Liste mit dem Namen `myenvironment` definiert. Als Grundlage dafür dient die bekannte `list`-Umgebung. Mit dem Befehl `\widthof` wird danach die Länge für den

Aufzählungspunkt festgelegt. In den weiteren Zeilen werden die folgenden Werte definiert.

Im Hauptdokument werden die neue Umgebung aufgerufen und als Parameter für die Liste die oben neu definierte Umgebung `myenvironment` angegeben. Die anderen Parameter erhalten ihr Aussehen entsprechend den oben genannten Definitionen.

9.1.2 Listen als Grundlage für andere Umgebungen

LaTeX stellt neben der oben genannten Umgebung `list` noch weitere Umgebungen zur Verfügung, z. B. die Umgebung `trivlist` (triviale Liste). Diese wird ähnlich aufgerufen wie die Umgebung `list`:

```
\begin{trivlist}
\end{trivlist}
```

Die Listenparameter `\leftmargin`, `\itemindent` und `\labelwidth` können auf 0 gesetzt werden. Die Parameter `\parsep` und `\listparindent` werden dann vom aktuellen Absatz übernommen. Die Umgebung `trivlist` fügt einen vertikalen Abstand zwischen der Liste und den vorangegangenen und nachfolgenden Zeilen ein. Somit lassen sich Textblöcke absetzen.

9.2 Ergänzungspakete für Listen

Es gibt eine Menge von Ergänzungspaketen für Listen. Hier werden einige vorgestellt.

9.2.1 Das Paket `expdlist`

Das Paket `expdlist` ist ein Ergänzungspaket für Listen, das Sie mit dem Befehl `\usepackage{expdlist}` einbinden können. Die Liste wird so aufgerufen:

```
\begin{description}[declarations]
...
\end{description}
```

Dieses Listenpaket enthält folgende Parameter:

- `\setleftmargin`
Hiermit wird der horizontale Abstand des linken Abstandes der Listenpunkte gesetzt.
- `\setlabelphantom`
Dieser Befehl berechnet die Breite des linken Abstandes über die Länge des Textes und addiert den Wert `\labelsssep`.
- `\setlabelstyle`
Damit wird der Stil für die Marken festgelegt.

- `\breaklabel`
Führt den Text in der nächsten Zeile fort, wenn der Auflistungstext länger als die vorgesehene Breite ist.
- `\compact`
Es wird kein zusätzlicher Leerraum zwischen den Aufzählungspunkten eingefügt.

```
\documentclass[10pt,a4paper]{article}
\usepackage[latin1]{inputenc}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{expdlist}
\begin{document}
\begin{description}[\setlabelphantom{Bus}]{\setlabelstyle{\itshape}%
\compact\compact\breaklabel}
\item [Bus] Bus
\item [Bahn] Bahn
\end{description}
\end{document}
```

Beispiel 9.2: Die Umgebung `expdlist`

In diesem Beispiel werden die Umgebung `description` sowie die Listenpunkte `Bus` und `Bahn` definiert. Die Breite der Marke wird hier durch den Text `Bus` erreicht.

Tipp: Weitere Informationen zum Paket `expdlist` finden Sie unter
<http://ftp.gwdg.de/pub/ctan/macros/latex/contrib/expdlist/expdlist.pdf>

9.2.2 Das Paket `paralist`

Das Paket `paralist` stellt ein paar neue Listenumgebungen zur Verfügung und erweitert die Standardumgebung `list`.

```
\begin{itemize}[<label>]
\item
\end{itemize}
```

Über die `enumerate`-Umgebung ändern Sie die Art der Nummerierung:

```
\begin{enumerate}{<format>}
\end{enumerate}
```

Für den Parameter `format` können Sie verschiedene andere Aufzählungen wie z. B. römische oder arabische Ziffern verwenden.

In diesem Beispiel wird als Aufzählungssymbol ein kleines *i* verwendet. Es steht in eckigen Klammern hinter dem einleitenden `enumerate`-Befehl:

```
\begin{enumerate}[i]
\item Eins
\item Zwei
\end{enumerate}
```

Den normalen Text, der nicht als Aufzählungssymbol interpretiert werden soll, müssen Sie dagegen in geschweifte Klammern setzen.

```
\begin{enumerate}[{Beispiel}a:]
\item Eins
\item Zwei
\end{enumerate}
```

Das Paket stellt folgende Befehle bereit:

- `\pointedenum`
Damit wird die Nummerierung auf 1.1, 1.1.1 gesetzt. Hinter der Nummerierung wird ein Punkt gesetzt.
- `\pointlessenum`
Wie `\pointedenum`, jedoch ohne Punkt.
- `\paradescription label`
Damit wird das Format und Aussehen des `description`-Textes festgelegt.
- `\setdefaultitem`
Damit wird das Aussehen für die Labels definiert.

Tipp: Weitere Informationen zu dem Paket finden Sie unter

<http://ftp.gwdg.de/pub/ctan/macros/latex/contrib/paralist/paralist.pdf>

9.2.3 Das Paket `mdwlist`

Das Paket `mdwlist` stellt eine Ergänzung der Umgebung `list` dar, in der Sie verschiedene Parameter angeben können. Außerdem stellt das Paket zusätzlich zu den drei Standardlisten (`itemize`, `enumerate` und `description`) kompakte Versionen zur Verfügung – also mit geringerem Zeilenabstand. Die neue `description`-Umgebung wird folgendermaßen aufgerufen:

```
\begin{basedescript}{<befehle>
\end{basedescript}
```

Im Gegensatz zu den Paketen `expdlist` und `paralist` erlaubt das Paket `mdwlist` zudem mehrzeilige Labels.

Die Umgebung `basedescript` können Sie mit folgenden Befehlen beeinflussen:

- `\desclabelwidth{<breite>}`
Damit wird die Breite des Labels festgelegt. Der folgende Text wird dann um diesen Wert zuzüglich der Länge von `\labelsep` eingerückt.
- `\desclabelstyle{<stil>}`
Damit wird der Stil für das Label festgelegt. Es sind folgende Einstellungen möglich:
 - `\nextlinelabel`
Wenn der Labeltext breiter als die Vorgabe ist, wird der nachfolgende Text in die nächste Zeile geschrieben.
 - `\multilinelabel`
Der Text wird in eine `\parbox` gesetzt und entsprechend ihrer Länge umbrochen.
 - `\pushlabel`
Wenn der Labeltext breiter als die Vorlage ist, wird der nachfolgende Text mit etwas Abstand in derselben Zeile gesetzt.
- `\makelabel`
Mit diesem Befehl werden die Labels gesetzt.
- `\defaultdesc`
Damit wird die Standformatierung für die Umgebung festgelegt.
- `\makecompactlist`
Damit wird eine neue Umgebung erstellt, die auf einer bisherigen Umgebung aufbaut. Der Zeilenabstand zwischen den Auflistungen wird auf 0 gesetzt. Das Paket definiert drei neue Umgebungen, die folgendermaßen aufgerufen werden:

```
\makecompactlist{itemize*}{itemize}
\makecompactlist{enumerate*}{enumerate}
\makecompactlist{description*}{description}
```

Im folgenden Beispiel wird die `itemize*`-Umgebung benutzt:

```
\begin{itemize*}
\item Eins
\item Zwei
\item Drei
\end{itemize*}
```

9.2.4 Das Paket desclist

Das Paket `desclist` stellt eine weitere `description`-Umgebung bereit.

Der allgemeine Aufruf lautet:

```
\begin{desclist}{<vorspann>}{<nachspann>}[<Texte für Breite>]
\end{desclist}
```

Mit dem Parameter `<vorspann>` werden der Text bzw. die Befehle festgelegt, die vor dem eigentlichen Label gesetzt werden. Mit dem Parameter `<nachspann>` werden die Befehle bzw. der Text gesetzt, der nach dem Labeltext erscheint. Über den optionalen dritten Parameter wird der Text übergeben, mit dessen Hilfe die Breite berechnet wird:

```
\begin{desclist}{\bfseries}{}[Eins]
\item [Eins] Punkt Eins
\item [Zwei] Punkt zwei
\end{desclist}
```

9.2.5 Weitere Ergänzungspakete

Neben diesen Paketen gibt es noch weitere Pakete, die für Listen zuständig sind. Dazu zählen unter anderem:

- `eqlist`
Dieses Paket stellt verschiedene Umgebungen zur Verfügung, die sich ähnlich wie die `description`-Umgebung verhalten.

Tipp: Weitere Informationen zu diesem Paket finden Sie unter

<http://ftp.join.uni-muenster.de/pub/software/CTAN/macros/latex/contrib/eqlist/eqlist.pdf>

- `enumerate`
Dieses Ergänzungspaket erweitert die Umgebung `enumerate` um ein optionales Argument, mit dem die Formatierung der Nummerierung festgelegt wird.

Tipp: Weitere Informationen zum Paket `enumerate` finden Sie unter

<http://www.ctan.org/tex-archive/macros/latex/required/tools/enumerate.pdf>

- `enumcount`
Dieses Ergänzungspaket stellt die Umgebung `enumcount` zur Verfügung, mit der die Nummerierung einer vorherigen `enumerate`-Umgebung fortgeführt wird.

- `multienum`

Dieses Ergänzungspaket stellt eine Aufzählungsumgebung zur Verfügung, die die Aufzählung in Spalten anordnet. Die Spaltenzahl kann zwischen einer und vier variieren.

Tipp: Weitere Informationen zum Paket finden Sie unter

<http://sunsite.informatik.rwth-aachen.de/ftp/pub/mirror/ctan/macros/latex/contrib/multenum/multienum.pdf>

10 Unformatierte Texte und Listings einbinden

Neben Bildern, Grafiken und Symbolen können Sie auch Texte in der Originalformatierung in LaTeX-Dokumente einbinden. Das ergibt insbesondere dann Sinn, wenn Sie etwa ein oder mehrere Beispiellistings einer Programmiersprache in das Dokument einbauen möchten.

Für die Einbindung von Text, z. B. mit Sonderzeichen usw., stellt LaTeX die Umgebung `verbatim` zur Verfügung.

10.1 Text einbinden mit der `verbatim`-Umgebung

Die einfachste Möglichkeit, Text einzubinden, ist die `verbatim`-Umgebung. Dabei erscheint der Text in Schreibmaschinenschrift, so wie er eingegeben wurde. Die `verbatim`-Umgebung wird wie folgt aufgerufen:

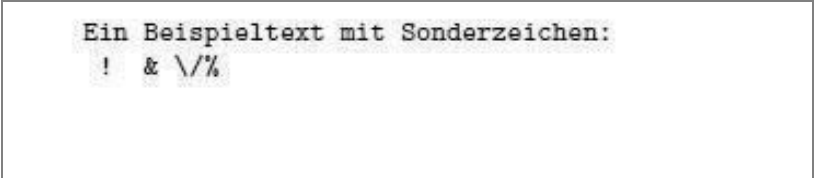
```
\begin{verbatim}  
Ein eingebundener Text  
\end{verbatim}
```

Die `verbatim`-Umgebung steht auch in der Sternvariante zur Verfügung, in der alle Leerzeichen mit einem Unterstrich (`_`) dargestellt werden.

Zwei Beispiele für die `verbatim`-Umgebung:

```
\begin{verbatim}  
Ein Beispieltext mit Sonderzeichen:  
!"$$öä& \/  
\end{verbatim}
```

Beispiel 10.1: Die `verbatim`-Umgebung



```
Ein Beispieltext mit Sonderzeichen:  
! & \%
```

Abbildung 10.1: Einfügen von Text mit Sonderzeichen

Hier noch ein Beispiel für die `verbatim`-Umgebung:

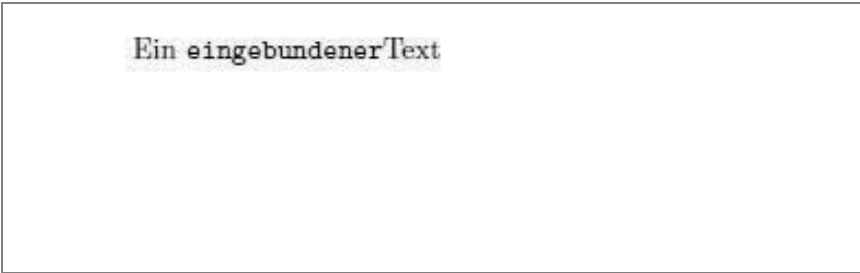
```
\begin{verbatim}  
Heute ist ein schöner Tag zum Aufstehen.  
\end{verbatim}
```

Anstelle der Umgebung `verbatim` können Sie auch den Befehl `\verb` benutzen, mit dem Sie un- oder vorformatierte Textpassagen unverändert in das Dokument einfügen können.

```
Ein \verbleingebundener|Text  
\verb|Heute|ist ein schöner Tag zum Aufstehen.
```

Beispiel 10.2: Einbinden von Text mit dem Befehl `\verb`

Der Befehl `\verb` ist insofern interessant, als er nicht vorschreibt, welches Zeichen benutzt werden soll, um den eingebundenen Text zu markieren. Zur Abgrenzung können Sie grundsätzlich eine Reihe von Zeichen oder sogar eine Kombination mehrerer Zeichen verwenden. In diesem Beispiel wird der Hochstrich verwendet, um den Text einzugrenzen, der in Schreibmaschinenschrift erscheinen soll.



```
Ein eingebundenerText
```

Abbildung 10.2: Eingebundener Text

Der eingebundene Text folgt direkt nach dem Befehl `\verb`. Im zweiten Beispiel sehen Sie, dass Sie sogar die Zeichenkombination `+123` benutzen können, um die eingebundene Textpassage zu markieren. Der Text wird so lange eingegrenzt, bis die Kombination (hier `+123`) wieder auftritt.

```
\verb |Text|  
\verb +123 ... +123
```

Auch für den Befehl `\verb` gibt es eine Sternvariante, die Leerzeichen mit einem Unterstrich darstellt.

Tip: Die Umgebung `verbatim` und der Befehl `\verb` dürfen nicht als Argument in einem Befehl angegeben werden.

Wenn Sie in die `verbatim`-Umgebung sehr lange Texte einbinden, kann es passieren, dass es zu einem Fehler kommt, weil TeX der Speicher ausgeht. Die nachstehend genannten Ergänzungspakete schaffen hier Abhilfe.

10.1.1 Text mit LaTeX-Befehlen einbinden

Neben der Standardumgebung `verbatim` und dem `verb`-Befehl gibt es noch Ergänzungspakete, mit denen Sie Text einbinden können. Die meisten Ergänzungspakete definieren zwar neue Umgebungen oder Befehle, der Aufruf dieser Umgebungen ist jedoch gleich.

Das Ergänzungspaket `alltt`

Das Ergänzungspaket `alltt` stellt die Umgebung `alltt` zur Verfügung. Diese Umgebung verhält sich ähnlich wie `verbatim`. Es sind jedoch normale LaTeX-Befehle erlaubt, die mit dem Backslash und den geschweiften Klammern eingeleitet werden. Diese Befehle werden auch wie gewohnt ausgewertet.

```
\usepackage{alltt}  
\begin{alltt}  
  Die alltt Umgebung  
\end{alltt}
```

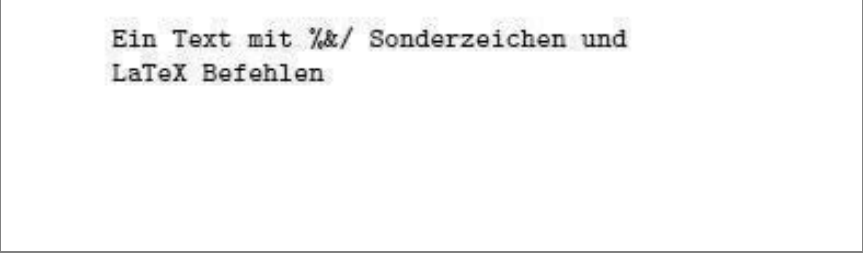
Beispiel 10.3: Die `alltt`-Umgebung

Wenn in dem Text ein Paragraphenzeichen (§) enthalten ist, können Sie dieses mit `\(...\)` einbinden, ansonsten wird es nicht dargestellt.

Nachfolgend sehen Sie ein Beispiel für die `alltt`-Umgebung:

```
\begin{alltt}  
Ein Text mit $%&/ Sonderzeichen und  
LaTeX{} Befehlen  
\end{alltt}
```

Beispiel 10.4: Text mit der `alltt`-Umgebung einbinden



```
Ein Text mit %&/ Sonderzeichen und
LaTeX Befehlen
```

Abbildung 10.3: Text mit der alltt-Umgebung einbinden

```
\begin{alltt}
Heute pack ich, morgen back ich, übermorgen heirate ich.
\end{alltt}
```

10.1.2 Verkürzte Syntax für \verb

Das Ergänzungspaket `shortvrb` stellt eine Verkürzung für den Befehl `\verb` zur Verfügung:

```
\MakeShortVerb{}
\DeleteShortVerb{}
```

Über den Befehl `\MakeShortVerb{||}` wird ein Zeichen definiert, das den einzubindenden Text einschließt. Dieses Zeichen ist `|`. Dieses Zeichen gilt so lange, bis der Befehl `\DeleteShortVerb{}` aufgerufen wird.

```
\MakeShortVerb{||}
\LaTeX{} wird mit ||\LaTeX{}| geschrieben.
\DeleteShortVerb{}
```

```
\MakeShortVerb{|}
Das Wochenende besteht aus |\Samstag| und |\Sonntag|
\DeleteShortVerb{}
```

10.1.3 Das Paket verbatim

Das Ergänzungspaket `verbatim` ist eine Erweiterung für die `verbatim`-Umgebung, das noch eine weitere Umgebung `comment` für Kommentare im LaTeX-Quellcode enthält. Die `verbatim`-Umgebung wird so aufgerufen:

```
\begin{verbatim}
test
test
\end{verbatim}
```

Die `comment`-Umgebung für Kommentare wird wie folgt aufgerufen:

```
\begin{comment}  
\end{comment}
```

```
\begin{comment}  
Aller guten Dinge sind drei.  
\end{comment}
```

Das Paket enthält außerdem den Befehl `\verbatiminput`, mit dem Sie einen Text aus einer Datei einbinden:

```
\verbatiminput{beispiel.tex}
```

Tipp: Weitere Informationen über das Paket finden Sie unter

http://www.ctan.org/tex_archive/macros/latex/required/tools/verbatim.pdf

10.1.4 Originaltext als Befehl einbinden

Mit dem Paket `verbdef` können Sie einen Originaltext als Befehl einbinden. Das Paket stellt den Befehl `\verbdef*` zur Verfügung, der Leerzeichen im Text sichtbar macht:

```
\verbdef\test{Original Text}  
\verbdef*\test{Original Text}
```

Hier sehen Sie ein Beispiel für den Befehl `\verbdef`:

```
\usepackage{verbdef}  
\verbdef\Test |Ein Beispieltext|  
\begin{document}  
\Test  
\end{document}
```

Beispiel 10.5: Das Paket `verbdef`

Noch ein Beispiel:

```
\usepackage{verbdef}  
\verbdef\Test |Dieser Text wird eingebunden|  
\begin{document}  
\Test  
\end{document}
```

10.1.5 Text mit dem Paket `moreverb` einbinden

Das Paket `moreverb` ist ein weiteres Paket, mit dem Text eingebunden werden kann. Das Paket `moreverb` stellt die Umgebungen `verbatimtab`, `verbatimabinput` sowie `verbatimabsize` zur Verfügung.

Die `verbatimtab`-Umgebung ersetzt Tabulatoren im eingebundenen Text durch Leerzeichen. Sie wird wie folgt aufgerufen:

```
\begin{verbatimtab}[<tabwidth>]
\end{verbatimtab}
```

Mit dem Parameter `tabwidth` geben Sie die Zahl der Leerzeichen an, die den Tabulator ersetzen.

Hier ein Beispiel für die `verbatimtab`-Umgebung:

```
\begin{verbatimtab}[2]
London ist die Hauptstadt von England.
\end{verbatimtab}
```

- `\verbatimabinput[<Anzahl Leerzeichen>]{<datei>}`
Dieser Befehl arbeitet ähnlich wie die Umgebung `verbatimtab`, schreibt den Text jedoch in eine Datei.
- `\verbatimabsize`
Dieser Befehl setzt die Anzahl der Leerzeichen.

Hier sehen Sie ein Beispiel für die beiden Befehle:

```
\begin{document}
\verbatimabinput[2]{Einfuegen.tex}
\end{document}
```

Tipp: Weitere Informationen zu dem Paket finden Sie unter <http://ftp.join.uni-muenster.de/pub/software/CTAN/macros/latex/contrib/moreverb/moreverb.pdf>

10.1.6 Text einbinden mit dem Paket `sverb`

Das Paket `sverb` stellt die `listing`-Umgebung zur Verfügung. Die `listing`-Umgebung wird wie folgt aufgerufen:

```
\begin{listing}
Die listing Umgebung
\end{listing}
```

Mit der Umgebung `verbatim` können Sie einen Text in eine Datei schreiben:

```
\begin{verbatim}{Dateiname}
Text in eine Datei schreiben mit der verbatim Umgebung
\end{verbatim}
```

Weiterhin stellt das Paket diese Befehle zur Verfügung:

- `\listingsize`
Damit wird die Textgröße der `listing`-Umgebung festgelegt. Diese wird standardmäßig auf `\small` gesetzt.
- `\listingindent`
Legt die Einrückung des einzuziehenden Textes fest.
- `\listing*`
Mit der Umgebung `listing*` können Sie mit einem Parameter den End-Befehl festlegen, mit dem die Umgebung beendet wird. Damit kann der Befehl `\end{listing}` im Text vorkommen, ohne dass dadurch die Umgebung beendet wird.
- Anbei sehen Sie ein paar Beispiele für die Befehle und Umgebungen:

```
\renewcommand{\listingindent}{2em}
\renewcommand{\listingsize}{\scriptsize}

 $\leftarrow$  linker Rand
\begin{listing}
Hier steht der Originaltext
\end{listing}
 $\leftarrow$ 
```

Tipp: Weitere Informationen zu dem Paket finden Sie unter

http://tug.ctan.org/tex_archive/macros/latex/contrib/mdwtools/sverb.pdf

10.1.7 Text mit `fancyvrb` einbinden

Das Paket `fancyvrb` ist ein weiteres umfangreiches Paket. Der eingebundene Text kann unter anderem mit einem Rahmen und einer Überschrift versehen werden.

- `\verbatimfootnotes`
Mit dem Befehl `verbatimfootnotes` können Sie `verbatim`-Umgebungen und `\verb`-Befehle in Fußnoten verwenden.

- `\DefineShortVerb{}`
Mit dem Befehl `\DefineShortVerb{}` können Sie eine Kurzversion des `verbatim`-Befehls erzeugen. Das Gegenteil dazu ist `\UndefineShortVerb`. Dieser Befehl ist ähnlich wie der bereits genannte Befehl `\makeshortverb`

Hier sehen Sie ein Beispiel für den Befehl `\verbatimfootnotes`:

```
\verbatimfootnotes
Eine Fußnote \footnote{\verb+% Das ist ein Kommentar}
```

```
\DefineShortVerb{
\LaTeX{}} wird mit |\LaTeX{}} geschrieben
```

- `\commentchar`
Definiert ein Zeichen, das einen Kommentar im Originaltext darstellt. Die von diesem Zeichen eingeleitete Zeile wird dann nicht gedruckt.

```
\begin{verbatim}[commentchar \%]
%Das ist ein Kommentar
\end{verbatim}
```

- `gobble`
Mit diesem Parameter können Sie die Anzahl von Zeichen (maximal neun) angeben, die zu Beginn jeder Zeile unterdrückt werden sollen. Das ist bei Listings mit Zeilennummern nützlich.
- `formatcom`
Damit werden ein oder mehrere Befehle definiert, die vor der `verbatim`-Umgebung ausgeführt werden können.
- `\FancyVerbFormatLine`
Diesen Befehl können Sie vor jeder Zeile ausführen. Wenn Sie den Befehl undefinieren, lassen sich Aufzählungspunkte vor jede Zeile setzen bzw. die Farbe jeder Zeile ändern.

```
\begin{verbatim}[gobble 2]
Es werden zwei Zeichen unterdrückt.
\end{verbatim}
```

Anbei einige weitere Beispiele für die verschiedenen Befehle und Umgebungen. Hier wird `formatcom` dazu eingesetzt, den Text in der `verbatim`-Umgebung kursiv zu setzen:

```
\begin{verbatim}[formatcom \itshape]
Ein Beispielttext
\end{verbatim}
```

```
\renewcommand{\FancyverbFormatLine}[1]{%
\makebox[0pt]{$\rightarrow$#1}
```

```
\begin{verbatim}  
Beispieltext  
\end{verbatim}
```

Tipp: Weitere Informationen zu diesem Paket finden Sie in der Dokumentation unter

<ftp://ftp.tu-chemnitz.de/pub/tex/macros/latex/contrib/fancyvrb/fancyvrb.pdf>

10.1.8 Listings einbinden

Mit dem Paket `listings` binden Sie Listings ein. Das sind Programmbeispiele, die einen bestimmten Sachverhalt dokumentieren.

Die Umgebung `lstlisting`

Mit der Umgebung `lstlisting` lassen sich ganz einfach Listings einbinden. Sie besitzt einen optionalen Parameter, der diverse Werte aufnehmen kann. Der allgemeine Aufruf lautet:

```
\begin{lstlisting}[key1 wert, key2 wert,...]  
\end{lstlisting}
```

Hier ein Beispiel für die Verwendung der `lstlisting`-Umgebung:

```
\begin{lstlisting}  
for(int i = 0; i<10; i++)  
    System.out.println("Hello World");  
\end{lstlisting}
```

Beispiel 10.6: Die Umgebung `lstlisting`

```
for(int i = 0; i<10; i++)  
    System.out.println("Hello World");
```

Abbildung 10.4: Die Umgebung `listing`

In diesem Beispiel wird eine `lstlisting`-Umgebung definiert, in der ein Java-Beispielcode eingebettet ist.

- `firstline`
Mit diesem Parameter wird angegeben, ab welcher Zeile der Code dargestellt werden soll.
- `\lstset{key wert,key2 wert,...}`
Damit werden für alle nachfolgenden Befehle die Grundeinstellungen festgelegt.
- `\lstinputlisting`
Dieser Befehl liest das Listing aus einer Datei. Der Befehl verhält sich wie der Befehl `lstlisting`.
- `\lstinline{key1 wert1,key2 wert2,...}<char>Listing char`
Dieser Befehl zeigt den Quelltext im normalen Fließtext an. Er verhält sich ähnlich wie der `\verb`-Befehl. Es sind jedoch alle Einstellungen des `listing`-Pakets möglich.
- `\lstloadlanguages{sprache, sprache2,...}`
Dieser Befehl lädt die entsprechende Sprache, damit sie in der `listing`-Umgebung verwendet werden kann. Die Sprache kann mit dem Befehl `\lstset` aktiviert werden.

Tipp: Weitere Informationen finden Sie in der Dokumentation unter <http://ftp.gwdg.de/pub/ctan/macros/latex/contrib/listings/listings.pdf>

11 Mathematische Formeln

Eine Stärke von LaTeX ist das Setzen mathematischer Formeln. Egal wie kompliziert eine Formel auch sein mag, das gedruckte Ergebnis ist stets ansprechend. Die mathematischen Formeln werden standardmäßig kursiv gesetzt. Viele gängige Mathematikbücher sind mit LaTeX gesetzt worden, was man unter anderem an der typischen Formeldarstellung erkennt. Für die Darstellung mathematischer Ausdrücke gibt es verschiedene Umgebungen und eine Vielzahl von Befehlen, die Sie in diesem Kapitel kennenlernen werden.

11.1 Grundlagen

Das Setzen mathematischer Ausdrücke ist in LaTeX relativ einfach. Um eine Formel setzen zu können, müssen Sie LaTeX in den Mathematikmodus bringen. Dazu gibt es mehrere Möglichkeiten.

11.1.1 Einfache Mathematikschreibweise

Die einfachste Mathematikschreibweise ist, eine Formel zwischen zwei Dollarzeichen zu setzen. Das ist zum Beispiel extrem hilfreich, wenn Sie einen mathematischen Ausdruck in normalen Fließtext einfügen wollen:

```
$Formel$
```

Hier ein konkreteres Beispiel mit dem Satz des Pythagoras:

```
$c^2 = a^2 + b^2$
```

So sieht dann das Ergebnis aus:

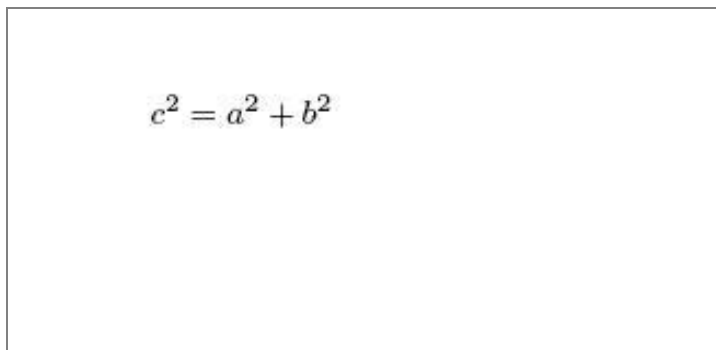

$$c^2 = a^2 + b^2$$

Abbildung 11.1: Einfache Formel

Tipp: Die zwei Dollarzeichen (\$\$) stellen kein normales LaTeX-Kommando, sondern ein plain-TeX-Kommando dar. Das hat Konsequenzen, z. B. für die Abstände zum umgebenden Text. Anstelle dessen sollte man die abkürzende Schreibweise für die `displaymath`-Umgebung verwenden. Siehe unten.

11.1.2 Mathematikumgebungen

Zum einen gibt es die Umgebung `math`, die automatisch in den Mathematikmodus wechselt. Die Umgebung `math` wird wie folgt aufgerufen:

```
\begin{math}  
  Formel  
\end{math}
```

Innerhalb der `math`-Umgebung sind die beiden Dollarzeichen nicht notwendig; alles, was innerhalb des öffnenden und schließenden `math`-Befehls steht, wird automatisch im Mathematikmodus gesetzt. Ausdrücke, die im Mathematikmodus stehen, werden grundsätzlich kursiv gesetzt.

Die `math`-Umgebung stellt die grundlegende Mathematikumgebung dar, mit der Sie ganz einfach mathematische Formeln setzen können. Neben dieser grundlegenden Umgebung für Formeln gibt es auch noch weitere Umgebungen, mit denen Sie verschiedene mathematische Objekte definieren können, wie z. B. Gleichungen und Ähnliches.

11.1.3 Weitere Umgebungen

Neben der Umgebung `math` existieren noch weitere Umgebungen für mathematische Formeln wie `equation` und `displaymath`. Diese beiden Umgebungen dienen dazu, einen mathematischen Ausdruck vom übrigen Text etwas abzusetzen. Sie weisen einer Formel daher einen eigenen Absatz zu. Bei Verwendung der Umgebung `equation` wird der Formel zusätzlich noch eine fortlaufende Nummer zugewiesen. Die Umgebungen werden analog zur Umgebung `math` aufgerufen.

```
\begin{equation}
(a+b)^2 a^2+2ab+b^2
\end{equation}
```

```
\begin{displaymath}
(a b)^2 a^2 2ab+b^2
\end{displaymath}
```

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

Abbildung 11.2: Erste und zweite binomische Formel, oben gesetzt in der `equation`-Umgebung und unten in der `displaymath`-Umgebung.

Die Umgebung `displaymath` lässt sich ähnlich wie die `math`-Umgebung ebenfalls abgekürzt schreiben. Der mathematische Ausdruck wird dabei durch einen Backslash eingeleitet, die Formel folgt dann in eckigen Klammern. Vor der schließenden eckigen Klammer wird dann nochmals ein Backslash gesetzt.

```
\[Formel\]
```

Hier die Notation in der Praxis:

```
\[F a*a\]
```

$$F = a * a$$

Abbildung 11.3: Mit der Kurzform von `displaymath` gesetzte Formel

Setzen von Formeln

Formeln werden in LaTeX standardmäßig horizontal zentriert. Wenn eine Formel (etwa aufgrund der `equation`-Umgebung) nummeriert wird, wird die Formelnummer rechtsbündig angezeigt. Das Aussehen einer Formel kann man zusätzlich über das Paket `amsmath` beeinflussen. Mehr zu dem Paket `amsmath` finden Sie in Abschnitt 11.2.

Setzen im mathematischen Modus

Im mathematischen Modus werden Formeln anders als im Textmodus gesetzt:

- Leerzeichen sind im mathematischen Modus verboten.
- Leerstellen und Zeilenwechsel haben bei der Eingabe keine Bedeutung: Alle Ausdrücke werden automatisch nach der Logik der mathematischen Ausdrücke bestimmt oder müssen durch spezielle Befehle wie `\,` oder `\quad` angegeben werden.
- Jeder einzelne Buchstabe wird als Name einer Variablen betrachtet und entsprechend gesetzt, nämlich kursiv in entsprechendem Abstand. Wenn Sie innerhalb einer Formel einen normalen Text setzen wollen, müssen Sie diesen in `\textnormal{...}` setzen. Das Paket AMS-LaTeX stellt darüber hinaus den Befehl `\text{text}` zur Verfügung. Siehe dazu Abschnitt 11.2.

```
\begin{displaymath}
(a+b)^2 a^2+2ab+b^2\textnormal{Erste Binomische Formel}
\end{displaymath}
```

$$(a + b)^2 = a^2 + 2ab + b^2 \text{ Erste Binomische Formel}$$

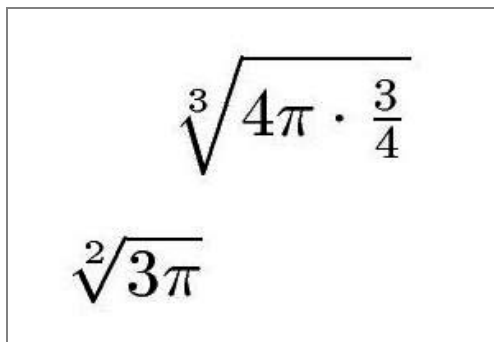
Abbildung 11.4: Binomische Formel mit normal gesetztem Text

11.1.4 Schriftgröße und Schriftarten

Latex wählt die Schriftgröße für Formelelemente selbst aus. Im Text eingebettete Formeln werden dabei platzsparend gesetzt. Ein Beispiel hierfür ist das Integralzeichen. Auch bei Zähler und Nennern in Brüchen und Exponenten und Indizes werden automatisch kleinere Schriftgrößen verwendet.

```
\sqrt[3]{4\pi \cdot \frac{3}{4}}
```

```
\sqrt[2]{3\pi}
```



$$\sqrt[3]{4\pi \cdot \frac{3}{4}}$$

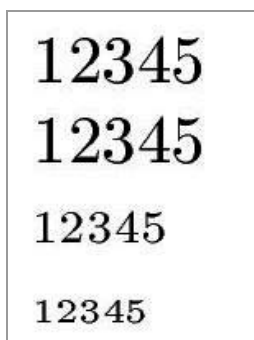
$$\sqrt{3\pi}$$

Abbildung 11.5: Bei mathematischen Ausdrücken wird die Schriftgröße automatisch angepasst.

Bei Doppelbrüchen oder anderen geschachtelten Strukturen kann dies jedoch dazu führen, dass der Text kaum noch lesbar ist. Dieses Problem kann umgangen werden, indem die Schriftgröße im Mathematikmodus explizit gesetzt wird. Dies ist mit einem kleinen Trick möglich.

```
\textstyle 12345$\\ %Schriftgrad in Textformeln
\displaystyle 12345$\\ %Schriftgrad in Absatzformeln
\scriptstyle 12345$\\ %für einfache Umstellungen
\scriptscriptstyle 12345$ %für mehrfache Umstellungen
```

Das Resultat sieht folgendermaßen aus:



$$12345$$

$$12345$$

$$12345$$

$$12345$$

Abbildung 11.6: Schriftgrößen im Mathematikmodus

Falls es notwendig sein sollte, manuell kleinere Schriftgrößen zu verwenden, empfiehlt sich die Verwendung von `\scriptstyle` und `\scriptscriptstyle`.

In mathematischen Ausdrücken können diese Befehle den jeweiligen Zahlenwerten und Variablen vorangestellt werden:

```
\sqrt[\scriptstyle 3]{\displaystyle 4\pi}
```

Schriftformatierungen

Die einzelnen Schriftarten und -formatierungen werden über verschiedene Befehle gesetzt. So gibt es je nach Bedarf Befehle für unterschiedliche Formatierungen. Hier sehen Sie einige Beispiele:

```
%Standardschrift
\[ \mathnormal{A \sum {i 0}^3 3i+5}\]
%Kalligraphische Schrift
\[ \mathcal{A \sum C+B}\]
%Mathematik Roman
\[ \mathrm{A \sum{i 0}^4 3i+5}\]
%Fett
\[ \mathbf{A \sum{i 0}^2 2i+6}\]
```

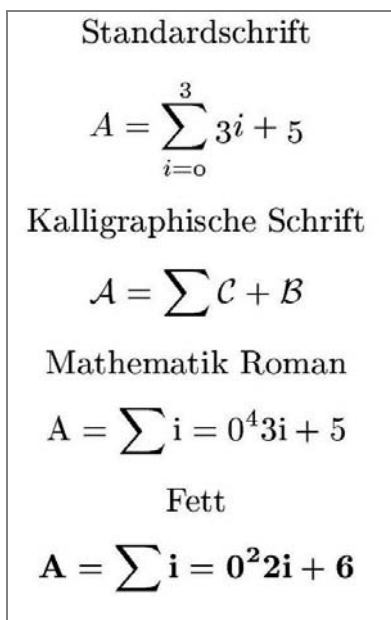


Abbildung 11.7: Mathematik-Schriftschnitte

Neben diesen Schriften gibt es noch andere Schriftschnitte wie zum Beispiel serifenlose und schreibmaschinenartige Schriften (Type Writer) sowie kursive Schriften. Die entsprechenden Befehle sehen so aus:

- `\mathsf`
Serifenlose Schrift
- `\mathtt`
Type Writer-Schrift
- `\mathit`
Kursive Mathematiksschrift

Die Befehle werden alle auf die gleiche Weise aufgerufen, indem man sie zwischen zwei eckige Klammern setzt und darauf die jeweiligen Formeln setzt.

11.1.5 Grundlegende Formelelemente

Eine Formel besteht grundsätzlich aus Variablen und Konstanten sowie verschiedenen Arten von Operatoren, die diese verknüpfen. LaTeX folgt dem internationalen Standard, wonach für mathematische Ausdrücke die Schriftart Roman verwendet wird und Variablen kursiv gesetzt werden.

Mathematische Operatoren und einige weitere Symbole bedürfen für das Setzen in LaTeX keiner weiteren Umschreibung; sie werden einfach über die Tastatur eingegeben:

`+ / : ! ' | [ ] ( )`

Wenn in einem mathematischen Ausdruck geschweifte Klammern (`{ }`) vorkommen sollen, dann muss ihnen ein Backslash vorangestellt werden, damit sie angezeigt werden.

Klammersymbole

Klammern werden in der Mathematik dazu verwendet, Terme zu strukturieren. Zusätzlich zu den einfachen, über die Tastatur verfügbaren runden und eckigen Klammern existieren in LaTeX noch weitere Befehle für Klammersymbole. Diese sehen Sie in der Tabelle 11.1:

<i>Symbol</i>	<i>Eingabe</i>	<i>Beschreibung</i>
<code>(</code>	<code>(</code>	Linke Klammer
<code>)</code>	<code>)</code>	Rechte Klammer
<code>{</code>	<code>\{</code>	Geschweifte Klammer links
<code>}</code>	<code>\}</code>	Geschweifte Klammer rechts
<code>[</code>	<code>\lceil</code>	Geöffnete eckige Klammer links

Symbol	Eingabe	Beschreibung
$\lfloor$	<code>\lfloor</code>	Geöffnete eckige Klammer rechts
$ $	<code> </code>	Balken
$\lceil$	<code>\lceil</code>	Geöffnete Klammer rechts
$\rfloor$	<code>\rfloor</code>	Geöffnete Klammer links
$\backslash$	<code>\backslash</code>	Backslash
$\langle$	<code>\langle</code>	Spitze Klammer links
$\rangle$	<code>\rangle</code>	Spitze Klammer rechts

Tabelle 11.1: Klammersymbole

Mehrzeilige Ausdrücke

Manchmal sind Brüche oder andere Terme, die sich über mehr als eine Zeile erstrecken, Bestandteile mathematischer Ausdrücke. Sollen sie in Klammern eingegeben werden, werden die Kommandos `\left` vor die öffnende und `\right` vor die schließende Klammer gesetzt. Die mit `\left` bzw. `\right` erzeugten Klammern werden automatisch skaliert.

```
\[y = \left(\frac{1}{1+x^3}\right)^2]
```

Der Befehl `\frac` erzeugt in dieser Formel einen Bruchstrich. In LaTeX gesetzt, sieht die Formel folgendermaßen aus:

$$y = \left(\frac{1}{1 + x^3}\right)^2$$

Abbildung 11.8: Die Klammern werden in diesem Fall der Größe des Terms angepasst.

Bei manchen mathematischen Ausdrücken werden lediglich öffnende Klammern verwendet, so etwa bei Fallunterscheidungen. Im folgenden Beispiel betrachten wir eine Gleichung, bei der x in Abhängigkeit vom Wert y einen unterschiedlichen Wert annimmt. Die öffnende Klammer umfasst drei verschiedene Ausprägungen. Um die drei Varianten darzustellen, wurde eine `array`-Umgebung verwendet, die ähnlich wie eine Tabelle funktioniert. Damit die schließende (also rechte) Klammer nicht dargestellt wird, folgt direkt auf das Kommando `\right` ein Punkt.

```
\[y \left\{\begin{array}{r@{\quad\quad}}
1 & x<0\\
0 & x = 0\\
+1 & x>0
\end{array}\right.
\]
```

$$y = \left\{ \begin{array}{ll} -1 & : x < 0 \\ 0 & : x = 0 \\ +1 & : x > 0 \end{array} \right.$$

Abbildung 11.9: Die schließende Klammer wird hier nicht gesetzt.

Neben der automatischen Größenzuweisung gibt es auch die Möglichkeit, die Größe von Klammern manuell zu beeinflussen. Dafür finden folgende Befehle Verwendung:

Öffnende Klammer	Schließende Klammer
\bigl	\bigr
\Bigl	\Bigr
\biggl	\biggr
\Biggl	\Biggr

Tabelle 11.2: Manuelle Größenzuweisungen für Klammern

11.1.6 Mathematische Symbole

Eine mathematische Formel besteht aus Variablen und Symbolen. Symbole können aus verschiedenen Zeichen bestehen, z. B. griechischen oder lateinischen Buchstaben. Griechische Buchstaben werden unter anderem in der Geometrie verwendet, um Winkel zu bezeichnen.

Griechische Buchstaben

Ein griechischer Buchstabe wird durch den entsprechenden Befehl erzeugt. Die folgende Tabelle fasst alle griechischen Buchstaben zusammen.

Buchstabe (klein)	Befehl	Buchstabe (groß)	Befehl
α	\alpha	A	A
β	\beta	B	B
γ	\gamma	Γ	\Gamma
δ	\delta	Δ	\Delta
ε	\epsilon	E	E
ζ	\zeta	Z	Z
η	\eta	H	H
θ	\theta	Θ	\Theta
ι	\iota	I	I
κ	\kappa	K	K
λ	\lambda	Λ	\Lambda
μ	\mu	M	M
ν	\nu	N	N
ξ	\xi	Ξ	\Xi
ο	\o	O	O
π	\pi	Π	\Pi
ρ	\rho	P	P
ς	\varsigma		
σ	\sigma	Σ	\Sigma
τ	\tau	T	T
υ	\upsilon	Υ	\Upsilon
φ	\varphi	Φ	\Phi
χ	\chi	X	X
ψ	\psi	Ψ	\Psi
ω	\omega	Ω	\Omega

Tabelle 11.3: Griechische Buchstaben

Da einige griechische Buchstaben (vor allem die Großbuchstaben) wie lateinische Buchstaben aussehen, werden hier auch lateinische Lettern verwendet. Grundsätzlich gilt: Ein kleiner Buchstabe hinter dem Backslash erzeugt einen Kleinbuchstaben (`\sigma`), ein großer einen Großbuchstaben (`\Sigma`).

\$Ein paar Beispiele für griechische Buchstaben:

```
\psi
\omega
\upsilon
\theta$
```

Kursive griechische Buchstaben

Mit den folgenden Befehlen können Sie griechische Großbuchstaben kursiv darstellen:

<i>Befehl</i>	<i>Großbuchstabe</i>
<code>\varGamma</code>	Γ
<code>\varTheta</code>	Θ
<code>\varXi</code>	Ξ
<code>\varSigma</code>	Σ
<code>\varPhi</code>	Φ
<code>\varOmega</code>	Ω

Tabelle 11.4: Kursive griechische Buchstaben

Hier ein paar Beispiele für kursive griechische Buchstaben:

```
\varGamma
\varXi
\varPhi
```

Tipp: Mit den Paketen `mathptmx` und `mathpazo` können Sie ebenfalls griechische Buchstaben kursiv darstellen. Diese besitzen die Option `[slantedGreek]`.

11.1.7 Mathematische Operatoren

In der Mathematik gibt es die unterschiedlichsten Operatoren für die verschiedensten Aufgaben. So gibt es Operatoren für einfache Arithmetik, Bruchrechnen, Funktionen usw. Für verschiedene mathematische Operatoren, wie z. B. Vergleichsoperatoren usw., stellt LaTeX eine Reihe von Befehlen zur Verfügung. Bei einigen Kommandos gibt es eine Kurzfassung. Für die Kurzversionen der Befehle benötigen Sie das Paket `LaTeXsym`. Die nachfolgende Tabelle fasst alle möglichen Befehle für relationale und Mengenoperatoren zusammen:

Operator	Befehl
<	< (größer als)
>	> (kleiner als)
≤	\le, \leq (größer gleich)
≥	\ge, \geq (kleiner gleich)
=	(gleich)
≠	\ne, \neq (nicht gleich)
≪	\ll
≫	\gg
⊂	\subset (Untermenge von)
⊆	\subseteq (Untermenge von oder gleich)
⊃	\supset (Obermenge von)
⊇	\supseteq (Obermenge von oder gleich)
∈	\in (Element von)

Tabelle 11.5: Vergleichs- und Mengenoperatoren

Unten sehen Sie ein paar Beispiele für die verschiedenen Operatoren:

```
\begin{document}
\setlength{\parindent}{0pt}
Diese Menge ist $\le$ kleiner als die andere Menge
Diese Menge ist $\ne$ ungleich der anderen Menge
Diese Menge ist eine $\subset$ Untermenge der anderen Menge
\end{document}
```

Im Ausdruck sieht das Beispiel so aus:

Diese Menge ist $\leq$ kleiner als die andere Menge

Diese Menge ist $\neq$ ungleich der anderen Menge

Diese Menge ist eine $\subset$ Untermenge der anderen Menge

Abbildung 11.10: Mengenoperatoren im Überblick

Binäre Operatoren

In der Mathematik gibt es verschiedene Operatoren, um zwei Mengen miteinander zu verknüpfen. Die folgende Tabelle listet die wichtigsten Operatoren auf:

Operator	LaTeX-Befehl
+	+
-	
÷	\div
±	\pm
.	\cdot
*	\ast
°	\circ
x	\times
⊕	\oplus
⊗	\otimes

Tabelle 11.6: Binäre Operatoren

Es gibt noch weitere binäre Operatoren, die Sie mit LaTeX darstellen können.

11.1.8 Mathematische Akzente

Viele mathematische Variablen und Symbole haben einen Akzent. Ein bekannter mathematischer Akzent ist z. B. der Vektorakzent. Die unten stehende Tabelle gibt eine Übersicht.



Abbildung 11.11: Mathematische Akzente

Befehle

- `\hat`
Dieser Befehl erzeugt ein Dach über der Variablen: $\wedge$
- `\vec`
Damit wird ein Vektorpfeil über der Variablen erzeugt: $\rightarrow$
- `\dot`
Damit wird ein Punkt über der Variablen erzeugt.
- `\acute`
Damit wird ein Accent aigu (´) über der Variablen erzeugt.
- `\check`
Damit wird ein eingedrücktes Dach über der Variable erzeugt.
- `\tilde`
Damit wird eine Tilde $\sim$ über der Variablen erzeugt.
- `\bar`
Damit wird ein Balken über der Variablen erzeugt.
- `\ddot`
Damit wird ein Doppelpunkt (Trema) über der Variablen erzeugt: $\ddot{}$
- `\grave`
Damit wird ein Accent grave (`) erzeugt: $\grave{}$
- `\breve`
Damit wird ein unterer Halbkreis erzeugt.

Unten sehen Sie ein paar Beispiele für die verschiedenen Funktionen:

```
\begin{document}
Ein \hat Dach.
Ein \acute Akzent aigu.
\end{document}
```

Tipp: Wenn Sie die Buchstaben i und j für Variablen verwenden, können Sie auch die Befehle `\imath` und `\jmath` benutzen. Das bewirkt, dass i und j ohne Punkte dargestellt werden. Die Punkte würden nämlich stören, wenn Sie eine Tilde oder einen Vektorpfeil über die Buchstaben setzen. Die Syntax dafür lautet:

```
 $\vec{\imath} \tilde{\jmath}$ 
```

11.1.9 Exponenten und Indizes

In mathematischen Formeln treten häufig Exponenten und Indizes auf. Exponenten finden sich z. B. in quadratischen oder kubischen Funktionen. Indizes werden unter Variablennamen platziert, um den Stellenwert einer Variablen anzugeben.

Exponenten

Exponenten werden in LaTeX durch ein hochgestelltes $^$ dargestellt.

```
\begin{math}
\mathrm{x}^2+2
\mathrm{x}^3+2\mathrm{x}^2+5\mathrm{x}
\end{math}
```

Indizes

Indizes werden mit einem tiefgestellten $_$ (Unterstrich) dargestellt.

Beispiel

```
\begin{math}
\mathrm{x}^2_2+\mathrm{x}_1
\mathrm{x}_{n+1}
\mathrm{x}^{3n}_{ij}
\end{math}
```

Exponenten und Indizes gleichzeitig darstellen

Wenn Sie verhindern wollen, dass die hochgestellte und die tiefgestellte Zahl übereinander stehen, so können Sie auch nach der tiefgestellten Zahl geschweifte Klammern setzen. Damit wird die hochgestellte Zahl hinter die tiefgestellte Zahl gestellt:

```
$\mathrm{x}_{i\{}}^{n}$
```

Zusammengesetzte Exponenten und Indizes

Wenn Sie mehrgliedrige Exponenten und Indizes benutzen wollen, müssen Sie sie in geschweiften Klammern eingeben.

```
$\mathrm{x}^{1+2\mathrm{x}}$\mathrm{x}_{i+1\{}}^{n-1}$
```

11.1.10 Wurzel

Die Wurzel ist eine mathematische Funktion, mit der sich verschiedene Werte berechnen lassen. Die Umkehrfunktion des Wurzelziehens ist das Potenzieren. Die wichtigste mathematische Wurzel ist die Quadratwurzel. Die Quadratwurzel können Sie in LaTeX über den Befehl `\sqrt` darstellen.

Der allgemeine Aufruf lautet:

```
\sqrt[<n>]{<arg>}
```

Mit dem Parameter <n> (in eckigen Klammern) wird der Exponent der Wurzel angegeben und der Parameter <arg> erhält die Argumente unter der Wurzel. Bei der einfachen Quadratwurzel muss der Exponent der Wurzel n nicht angegeben werden:

```
$\sqrt{50}$
```

```
$\sqrt[3]{8}$
```

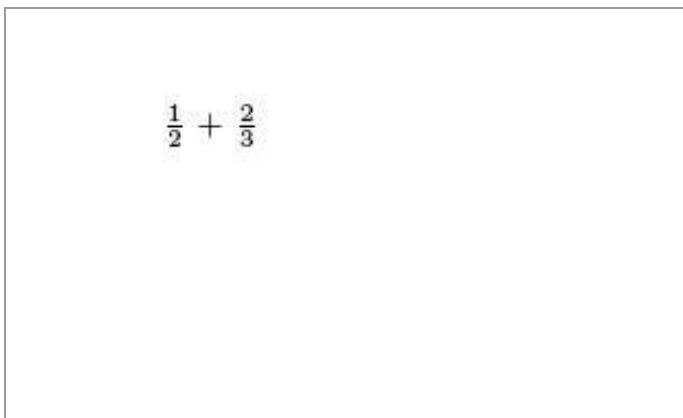
11.1.11 Brüche

Mit dem Befehl `\frac` können Sie Brüche darstellen. Zähler und Nenner stehen dabei jeweils in einer separaten geschweiften Klammer.

```
\frac{<zähler>}{<nenner>}
```

Die Länge des Bruchstriches richtet sich nach der Länge des längeren Argumentes:

```
\frac{1}{2} + \frac{3}{n+4}
```



$$\frac{1}{2} + \frac{3}{n+4}$$

Abbildung 11.12: Zwei Brüche

Zusammengesetzte Brüche

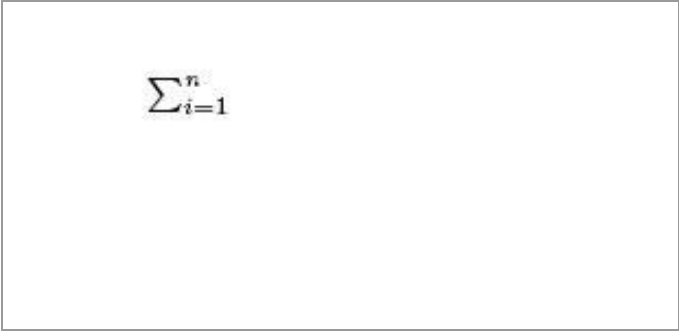
Zusammengesetzte Brüche können durch mehrfachen Aufruf des Befehls `\frac` erzeugt werden: Achten Sie aber darauf, dass jeder Instanz von `\frac` ein Zähler und ein Nenner zugeordnet sind.

```
$\frac{1}{\frac{2x+1}{4}}{3x}$
```

11.1.12 Summen und Integrale

Integrale und Summen sind weitere wichtige Hilfsmittel in der Mathematik. In LaTeX werden Integrale und Summen durch die Befehle `\sum` und `\int` dargestellt. Das Integralsymbol besteht aus einem lang gezogenen S. Das Summenzeichen besteht aus dem griechischen Buchstaben Σ (Sigma).

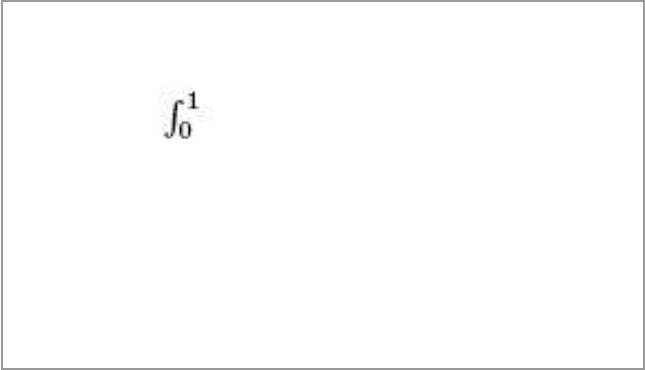
```
\sum {i 1}^n
```



$$\sum_{i=1}^n$$

Abbildung 11.13: Summe

```
\int 0^1
```



$$\int_0^1$$

Abbildung 11.14: Integral

Das Beispiel zeigt ein einfaches Integral in den Integrationsgrenzen 0 bis 1. Für Integrale und Summen gibt es auch eine ausführlichere Schreibweise:

```
\[\int\limits {x 0}^{x 1}\]
```

Mit dem Befehl `\limits` werden die Integrationsgrenzen für das Integral angegeben. Das ist dann sinnvoll, wenn die Integrationsgrenzen durch einen Term dargestellt werden müssen und ein einzelnes Zeichen nicht mehr ausreicht. Die Terme für die Grenzen werden jeweils in geschweiften Klammern notiert. Dem Term am unteren Ende des Integralsymbols geht ein Unterstrich voraus, dem Term am oberen Ende das Exponentenzeichen $^$.

Wenn Sie die Kurzform des `displaymath`-Modus mit den zwei eckigen Klammern verwenden und Formeltext vor einem Integral darstellen möchten, z. B. einen Faktor, können Sie dies wie folgt angeben:

```
\[2 \int_0^1 f(x)dx\]
```

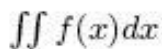
Das Beispiel zeigt ein einfaches Integral in den Integrationsgrenzen 0 bis 1 und der Funktionsvariablen x .

Mehrfachintegrale

Mehrfachintegrale bestehen aus mindestens zwei hintereinandergestellten Integralen. Das Paket AMS LaTeX (der Name steht für American Mathematical Society) stellt für Mehrfachintegrale den Befehl `\iint` zur Verfügung.

Das Paket AMS LaTeX ist in LaTeX-Standardinstallationen verfügbar und wird mit `\usepackage{amsmath}` eingebunden:

```
\usepackage{amsmath}
\[iint_0^1 f(x)dx\]
```



$$\iint f(x)dx$$

Abbildung 11.15: Mehrfachintegrale

11.1.13 Komplexe mathematische Strukturen

Für komplizierte mathematische Sachverhalte benötigt man neben Operatoren und Klammern noch weitere Funktionen.

Funktionsnamen

Die geometrischen Standardfunktionen wie z. B. $\exp$, $\log$, $\sin$, $\cos$ usw. können Sie über entsprechende LaTeX-Befehle setzen. Damit wird gewährleistet, dass die Funktionsnamen in Normalschrift gesetzt werden und nicht kursiv wie etwa Variablennamen. Anbei sehen Sie eine Übersicht über verschiedene mathematische Funktionen:

<i>Funktion</i>	<i>Befehl</i>
Arkuskosinus	<code>\arccos</code>
Arkustangens	<code>\arctan</code>
Kosinus	<code>\cos</code>
Kotangens	<code>\cot</code>
Kosekans	<code>\csc</code>
Determinante	<code>\det</code>
Exponentialfunktion	<code>\exp</code>
Homomorphismen	<code>\hom</code>
Kern	<code>\ker</code>
Grenzwert (Limes)	<code>\lim</code>
Größter Grenzwert (Limes superior)	<code>\limsup</code>
Kleinster Grenzwert (Limes inferior)	<code>\liminf</code>
Logarithmus	<code>\log</code>
Minimum	<code>\min</code>
Sekans	<code>\sec</code>
Sinus Hyperbolicus	<code>\sinh</code>
Tangens	<code>\tan</code>

Tabelle 11.7: Mathematische Funktionen

Einige Funktionen, wie z. B. die `\lim`-Funktion, benötigen noch weitere Angaben, wie z. B. den jeweiligen Grenzwert. Diese werden einfach, wie Indizes, tiefgestellt, wobei sich das genauere Aussehen danach richtet, ob der mathematische Ausdruck in einer Zeile – im Quelltext also innerhalb zweier Dollarzeichen – oder eine abgesetzte Formel sein soll.

```
\lim {x\to\infty}
```

Die liegende 8, die den Wert Unendlich repräsentiert, wird über den Befehl `\infty` dargestellt.

$$\lim x \rightarrow \infty$$

Abbildung 11.16: Darstellung von Grenzwerten in LaTeX

11.1.14 Pfeile über und unter mathematischen Ausdrücken

Mit den Befehlen `\overrightarrow`, `\overleftarrow`, `\underrightarrow` und `\underleftarrow` können Sie beliebige Pfeile über oder unter mathematische Ausdrücke setzen. Das Paket AMS LaTeX bietet darüber hinaus noch die Befehle `\overleftrightharrow` und `\underleftrightharrow`.

```
\[ \overrightarrow{\Psi \alpha(y) \beta y} \]
```

Dehnbare Pfeilsymbole

Eine weitere Möglichkeit sind dehnbare Pfeilsymbole. Mit den beiden Befehlen `\xleftarrow` und `\xrightarrow` können Sie Pfeile erzeugen, deren Länge sich nach der jeweiligen Beschriftung richtet. Hier muss in geschweiften Klammern ein Parameter für die Beschriftung über dem Pfeil angegeben werden. Wenn Sie keine Beschriftung wünschen, kann dieser Parameter leer bleiben. Zusätzlich können Sie noch in eckigen Klammern einen Parameter für die Beschriftung unterhalb des Pfeils angeben:

```
\[ A \xleftarrow{n+\mu-1} B \xrightarrow[n\pm i-1]{T} C \]
```

Über den Befehl `\pm` erzeugen Sie wie hier gezeigt das Zeichen plus/minus.

$$A \xleftarrow{n+\mu-1} B \xrightarrow[n\pm i-1]{T} C$$

Abbildung 11.17: So gibt LaTeX die dehnbaren Pfeilsymbole aus.

11.1.15 Gestapelte Symbole

Mit dem Befehl `\stackrel` können Sie Symbole übereinanderstapeln. Der allgemeine Aufruf lautet:

```
\stackrel{<oberes symbol>}{<unteres symbol>}
```

Dabei wird automatisch für das obere Symbol eine kleinere Schriftart gewählt.

11.1.16 Matrizen

Um eine Matrix in LaTeX zu erzeugen, benötigen Sie die `array`-Umgebung. Diese funktioniert ähnlich wie die `tabular`-Umgebung für Tabellen. Auch bei Matrizen wird durch einen doppelten Backslash (`\\`) das Zeilenende definiert und die Spalten werden durch ein kaufmännisches Und-Zeichen (`&`) voneinander getrennt. Die horizontale Ausrichtung wird wie bei einer Tabelle festgelegt.

Mit den Befehlen `\left` bzw. `\right` werden die Begrenzer, gefolgt von dem jeweiligen Begrenzungszeichen für die Matrix, gesetzt. Diese beiden Befehle müssen immer paarweise auftreten und bewirken, dass die Begrenzungszeichen genau der Größe der Matrix entsprechen:

```
\begin{math}
\left(
\begin{array}{llll}
1 & 0 & 1 & 1 & 2 \\
2 & 2 & 3 & 1 & 6 \\
5 & 3 & 1 & 2 & 1 \\
1 & 2 & 3 & 2 & 1
\end{array}
\right)
\end{math}
```

Mit dem Paket AMS LaTeX können Sie die Erstellung von Matrizen wesentlich vereinfachen. Dazu benötigen Sie das Paket `amsmath`, das einige Umgebungen für die Erstellung von Matrizen bereitstellt. Die jeweiligen Klammern um die Matrix sind in den Umgebungen bereits fest eingebaut.

All diese Umgebungen richten die Spalten horizontal zentriert aus und gehen sparsamer mit dem in der Breite benötigten Platz um als z. B. die `array`-Umgebung. Eine solche Matrix kann standardmäßig bis zu zehn Spalten umfassen.

Wenn Sie Matrizen in kleinerer Form direkt in einen Text einbinden wollen, können Sie die Umgebung `smallmatrix` benutzen.

Bei dieser Umgebung müssen Sie die Klammersetzung selbst vornehmen, wie Sie an den umrahmenden `\bigl`- und `\bigr`-Befehlen, begleitet von den runden Klammern, sehen:

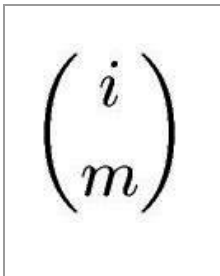
```
$
\bigl(
\begin{smallmatrix}
A & B \\
D & E
\end{smallmatrix}
\bigr)
$
```

11.1.17 Binomialkoeffizienten

Binomialkoeffizienten können in der Form `\{.. \choose .. \}` gesetzt werden. Wollen Sie keine Klammern, verwenden Sie den Befehl `\atop`.

Die nachfolgenden Beispiele zeigen den Unterschied – einmal mit Klammern ...

```
\[ { i \choose m } \]
```

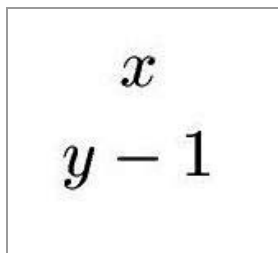


$$\binom{i}{m}$$

Abbildung 11.18: Binomialkoeffizient mit Klammer

... und einmal ohne:

```
\[ {x \atop y 1} \]
```



$$\frac{x}{y - 1}$$

Abbildung 11.19: Binomialkoeffizient ohne Klammer

11.1.18 Abstände in Formeln variieren

Wenn Sie einen größeren oder kleineren Zwischenraum in den Formeln setzen wollen, geschieht dies mit den Befehlen »\,« für einen schmalen Abstand, »\ « für einen mittleren Abstand und \quad und \qquad für große Abstände. Der Befehl »!\,« verkleinert dagegen den Zwischenraum. Welcher Abstand wo am besten aussieht, ist Geschmackssache. Probieren geht hier über Studieren.

Hier zwei Beispiele für eine einfache Formel sowie einen mathematischen Ausdruck mit zwei Integralen:

```
%Formel mit großem Zwischenraum
\[ F {n}    F {n+1} + F {n 2} \quad n \ge 2 \]
%Formel mit engem Abstand zwischen zwei Integralen
\[\int\!\!\!\int {D} dx\,dy\]
```

Formel mit großem Zwischenraum

$$F_n = F_{n+1} + F_{n-2} \quad n \geq 2$$

Formel mit engem Abstand zwischen zwei Integralen

$$\iint_D dx\,dy$$

Abbildung 11.20: Abstände in mathematischen Ausdrücken

11.1.19 Mehrzeilige Formeln und Gleichungssysteme

Für mehrzeilige Formeln und Gleichungssysteme stehen zwei verschiedene Umgebungen zur Verfügung. Bei der Umgebung `eqnarray` erhält jede Zeile des Gleichungssystems eine Nummer. Diese Nummer wird bei `eqnarray*` unterdrückt. Diese Umgebungen werden anstelle der bekannten Umgebungen `equation` oder `displaymath` verwendet.

Beide Umgebungen sind wie eine Tabelle aufgebaut, bei der die erste Spalte rechtsbündig, die zweite zentriert und die dritte linksbündig ist. Das Gleichheitszeichen steht dabei für die zweite Spalte. An dieser richtet sich die Gleichung aus, sodass alle Gleichheitszeichen direkt untereinander stehen. Dadurch wird das Lesen erleichtert. Auch hier werden die Spalten wieder durch ein kaufmännisches Und-Zeichen (&) voneinander getrennt.

```
\begin{eqnarray}
f(x) & & \cos x \\
f'(x) & & \sin x \\
\end{eqnarray}
```

Wenn eine Formel zu lang für eine Zeile ist, so kann LaTeX sie nicht unterbrechen. Daher müssen Sie dem System genau sagen, wo umbrochen und wie die nächste Zeile platziert werden soll:

```
\begin{eqnarray}
\lefteqn{\sin x - \frac{x^2}{2!} + \dots} \nonumber \\
& + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots
\end{eqnarray}
```

Der Befehl `\nonumber` sorgt dafür, dass in dieser Zeile die Gleichungsnummer entfällt. Der Befehl `\lefteqn` passt außerdem die Spaltenausrichtung derart an, dass die zweite Zeile nun unter dem Gleichheitszeichen und nicht mehr unter dem ersten Zeichen des Ausdrucks in der ersten Zeile beginnt.

11.1.20 Feintuning für Nummerierung und mehrzeilige Formeln

Mit den bisher gezeigten Befehlen von LaTeX lassen sich bereits ansehnliche mathematische Dokumente erstellen. Diesen Dokumenten fehlt jedoch noch der letzte Schliff, den Sie mit ein paar Handgriffen anlegen können.

Vertikale Position der Formelnummer

Mit dem Befehl `\raisetag` können Sie die vertikale Position der Formelnummer beeinflussen, wenn diese aus ihrer eigentlichen Position verschoben wurde. Das Paket AMS LaTeX sorgt bei mehrzeiligen Ausdrücken zwar dafür, dass die Formelnummer nie überschrieben wird. Mitunter ist es aber nötig, hier selbst einzugreifen.

Dem Befehl `\raisetag` wird als Argument die Längenangabe in Punkt übergeben.

```
\raisetag{4pt}
```

Im Optimalfall sollten Sie den Befehl `\raisetag` erst dann in das Dokument integrieren, wenn das Dokument bereits fertigverfasst ist. Das hat den Vorteil, dass Sie die Einstellungen nicht immer wieder neu setzen müssen.

Seitenumbrüche

Hin und wieder möchte man zwischen den Zeilen eines mehrzeiligen Ausdrucks mehr Abstand, als das Paket AMS LaTeX standardmäßig vorsieht. Dieser vertikale Abstand lässt sich mit dem Befehl `\\[<abstand>]` beeinflussen. AMS LaTeX erlaubt normalerweise keine automatischen Seitenumbrüche innerhalb einer mehrzeiligen Formel. Die Idee dahinter ist die, dass es dem Autor überlassen sein sollte, ob er die Formel auseinanderreißen oder aber zum besseren Verständnis auf einer Seite belassen will. Wenn Sie einen individuellen Seitenumbruch in eine Gleichung einbauen wollen, können Sie den Befehl `\displaybreak` benutzen.

Dieser Befehl wird am besten vor einem Zeilenumbruch `\\` benutzt, an dem auch der Seitenumbruch stattfinden soll. Ähnlich wie der Befehl `\pagebreak` im Standard-LaTeX hat der Befehl `\displaybreak` auch einen optionalen Parameter. Dieser Parameter kann die Werte 0 bis 4 annehmen, wobei 0 bedeutet, dass ein Seitenumbruch stattfinden kann, aber nicht muss, und 4 bedeutet, dass ein Seitenumbruch erzwungen wird. Fehlt der Parameter, dann findet der Seitenumbruch auf jeden Fall statt.

Wenn Sie einen Seitenumbruch dann einbauen wollen, wenn es gerade notwendig ist, benutzen Sie am besten den Befehl `\allowdisplaybreaks[1]`. Dessen optionaler Parameter kann die Werte 1 bis 4 annehmen. Der Wert 1 bedeutet dabei, dass der Seitenumbruch erlaubt ist, aber soweit wie möglich vermieden werden soll.

Formelnummerierung mit einbeziehen

Wenn Sie die Formelnummerierung in die jeweilige Abschnittsnummer einbeziehen wollen, wählen Sie den Befehl `\theequation`:

```
\renewcommand{\theequation}{\thesection.\arabic{equation}}
```

Hierbei wird der Zähler `equation` im nachfolgenden Abschnitt allerdings nicht automatisch auf 0 gesetzt. Dies müssen Sie mithilfe des Befehls `\setcounter` selbst erledigen.

Wenn Sie das Paket AMS LaTeX verwenden, lässt sich die Nummerierung noch einfacher gestalten. Der Befehl `\numberwithin` integriert ebenfalls die Abschnittsnummer. Der Zähler wird zu Beginn eines Abschnitts auf 0 zurückgesetzt, ohne dass Sie eingreifen müssen.

Der allgemeine Aufruf lautet:

```
\numberwithin{<equation>}{<section>}
```

Querverweise

Um Querverweise einzubinden, integrieren Sie mit dem Paket AMS LaTeX und dem Befehl `\eqref` Verweise auf Formeln. Dieser Befehl erzeugt automatisch Klammern um die Formelnummern. Ein solcher Verweis würde dann wie folgt aussehen: »siehe Formel(1)«. Bei dem normalen `\ref`-Befehl würden die Klammern fehlen. Das Äquivalent würde dann lauten: »Siehe Formel 1«.

```
\eqnref{<labelname>}
```

Das Paket `amsmath` gestattet es außerdem, Formelgruppen mit untergeordneten Nummern zu erzeugen. Damit wird deutlich, dass die Formeln zusammengehören. Mit der Umgebung `subequations` können Sie Formelnummern von der Art (1a), (1b) und so weiter erzeugen. Die einzelnen mathematischen Ausdrücke müssen dabei durch Zeilenumbrüche (`\`) voneinander getrennt sein. Die allgemeine Nummerierung innerhalb des Kapitels wird dabei eingehalten.

11.1.21 Gestaltung von Formeln

Formeln lassen sich auf verschiedenste Weise gestalten. So können Sie etwa Boxen benutzen oder Rahmen um eine Formel herum setzen.

Formeln nebeneinandersetzen

Sie können mehrere Formeln mit einer Box nebeneinandersetzen. Dazu können Sie z. B. den Befehl `\parbox` benutzen oder `\minipage`-Umgebungen verwenden:

```
\parbox{begin{eqnarray*}\dot{x 1} & & x 2\\
\dot{x 2} & & u
\end{eqnarray*}
\parbox{4.0cm}{\[x 1(0) \quad x 2(0) \quad 0\]}
\parbox{4.0cm}{\[x 1(1) \quad x 2(1) \quad 1\]}
```

`\parbox` erzeugt dann eine normale Box mit einfachem Rahmen.

Gerahmte Formeln

Neben der Möglichkeit, einer Box mit `\parbox` einen Rahmen zu spendieren, können Sie sie auch mit dem Befehl `\fbox` einrahmen:

```
\fbox{\parbox{5.0cm}{\int 0^\infty \frac{\cos ax \quad \cos bx}{x} dx. \quad \ln
\frac{b}{a}}}
```

Der Nachteil ist, dass man die Breite der Box selbst definieren muss. Bei komplexeren Formeln ist dies häufig nicht mehr ganz so einfach. Daher muss der richtige Breitenparameter oft durch Ausprobieren gefunden werden.

Mit den Umgebungen `displaymath` und `equation` kann hier jedoch Abhilfe geschaffen werden. Innerhalb dieser Umgebungen wird eine `\fbox` definiert, die den Formeltext umschließt:

```
\begin{displaymath}
  \fbox{$ \displaystyle <Formeltext> $}
\end{displaymath}
```

Anstelle von `displaymath` können Sie auch `equation` verwenden; die Syntax ist dieselbe. Wenn Sie `equation` benutzen, erhalten Sie allerdings noch eine Formelnummer hinzu, die außerhalb des Rahmens steht.

11.2 Das Paket AMS LaTeX

Wie schon erwähnt, hat die Amerikanische Gesellschaft für Mathematik (American Mathematical Society) ein Ergänzungspaket für mathematische Formeln entwickelt. Das Paket `amsmath` wird mit dem Befehl `\usepackage{amsmath}` eingebunden.

Das Paket AMS LaTeX enthält viele Ergänzungsbefehle für mathematische Formeln.

11.2.1 Die Pakete von AMS LaTeX

Das Paket AMS LaTeX enthält folgende Klassen und Pakete:

- `amsbook.sty`
- `amsart.sty`
- `amsproc.sty`

Diese Klassen entsprechen den LaTeX-Standardklassen `book`, `article` und `proc`. Der Vorteil dieser Klassen besteht darin, dass Sie damit die Layoutvorgaben und Erweiterungen des AMS-Pakets benutzen können. Dadurch gehen aber die Vorteile der KOMA-Script-Klassen verloren.

Während der Installation entstehen neben oben genannten Klassen-Dateien folgende Pakete:

- `amsmath.sty`
Hierbei handelt es sich um das Hauptergänzungspaket für AMS LaTeX, in dem die meisten Befehle für die Formelverarbeitung zu finden sind.
- `amsgen.sty`
Dieses Paket wird von einigen der nachfolgenden Ergänzungspakete eingebunden und definiert Makros, die von diesen vorausgesetzt werden.

- `amstext.sty`
Dieses Paket wird von `amsmath.sty` eingebunden. Es stellt den Umschaltbefehl `\text{text}` zur Verfügung, mit dem Sie kurze Textpassagen in das mathematische Dokument einbinden können.
- `amsbsy.sty`
Dieses Paket wird von `amsmath.sty` implizit eingelesen. Dieses Paket stellt den Schriftumschaltbefehl `\boldsymbol` und `\pmb` bereit.
- `amsopn.sty`
Dieses Paket wird von `amsmath.sty` eingebunden. Dieses Paket stellt den Definitionsbefehl `\DeclareMathOperator` zur Einrichtung von Funktionsnamen wie `\sin` oder `\log` bereit.
- `amsthm.sty`
Dieses Paket stellt die `proof`-Umgebung für mathematische Beweise sowie Erweiterungen für den Befehl `\newtheorem` bereit.
- `amscd.sty`
Dieses Paket stellt die `CD`-Umgebung zur Erzeugung von kommutativen Diagrammen zur Verfügung.
- `amsxtra.sty`
Dieses Paket dient zur Kompatibilitätssicherung mit der älteren Version 1.1 des AMS LaTeX-Pakets – momentan ist die Version 2 aktuell.
- `upref.sty`
Dieses Paket bewirkt, dass Kreuzbezüge unabhängig von der momentan aktiven Schrift stets mit `\normalfont` erscheinen.
- `amstex.sty`
Dieses Paket dient für die Rückwärtskompatibilität. Für neue Dokumente sollte es nicht mehr benutzt werden.

Diese Pakete werden alle über den Befehl `\usepackage` eingebunden.

11.2.2 Die Parameter von AMS LaTeX

Das Verhalten von AMS LaTeX können Sie mit verschiedenen Parametern beeinflussen.

Mögliche Parameter sind:

- `centertags` und `tbtags`
Diese beiden Parameter dienen der Ausrichtung der Formelnummer bei mehrzeiligen Formeln, die Sie mit der `split`-Umgebung erzeugen können. Die Standardeinstellung `centertags` richtet die Formelnummer vertikal zentriert über alle dazugehörenden Zeilen aus. Mit der Alternative `tbtags` können Sie die Formelnummer dagegen an der ersten Zeile (in Verbindung mit dem Parameter `leqno` für

linksbündige Formelnummern) bzw. an der letzten Zeile (in Verbindung mit dem Parameter `reqno` für rechtsbündige Formelnummern) ausrichten.

- **`sumlimits` und `nosumlimits`**
Mit diesen beiden Parametern können Sie die Positionierung der Grenzüberschriften in abgesetzten Formeln beeinflussen. Diese Parameter gelten dann für verschiedene Zeichen, wie etwa das Summenzeichen, Mengenzeichen, Funktionszeichen und ähnliche. Die Standardeinstellung `sumlimits` schreibt die Grenzüberschriften direkt über bzw. unter das Zeichen. Der Wert `nosumlimits` platziert diese Angaben dahinter.
- **`intlimits` und `nointlimits`**
Diese beiden Parameter sind für Integralzeichen bestimmt. Bei Integralen schreibt man standardmäßig die Grenzüberschriften hinter das Zeichen. Dies erfolgt mit dem Parameter `nointlimits`. Mit `intlimits` werden die Grenzüberschriften über und unter dem Integralzeichen platziert.
- **`namelimits` und `nonamelimits`**
Einige Funktionen benötigen zusätzliche Angaben, deren Anordnung Sie über die beiden Parameter festlegen können. Über die Standardeinstellung `namelimits` können Sie die Grenzüberschriften direkt unter den Funktionsnamen platzieren, andernfalls dahinter. Dieses Parameterpaar gilt für die Funktionen `\det`, `\gcd`, `\inf`, `\lim`, `\liminf`, `\limsup`, `\max`, `\min`, `\Pr` und `\sup`.
- **`leqno` und `reqno`**
Mit diesen Parametern können Sie Formelnummern ausrichten. Über die Einstellung `leqno` wird die Formelnummer links und mit `reqno` rechts angegeben.
- **`fleqn`**
Mit dieser Einstellung können Sie alle abgesetzten Formeln linksbündig anordnen, wobei durch den Parameter `mathindent` der Einzug festgelegt wird.

11.2.3 Darstellung einzelner Formeln mit AMS LaTeX

Das Paket `amsmath` erweitert die normale Mathematikumgebung von LaTeX und stellt weitere bereit. Alle Umgebungen werden mit `\begin` und `\end` eingeschlossen. Die folgende Tabelle gibt eine Übersicht über die verschiedenen Umgebungen:

Umgebung	Beschreibung
<code>equation</code>	Zentrierte Formeln mit Formelnummer
<code>equation*</code>	Zentrierte Formeln ohne Formelnummer
<code>gather</code>	Zentrierte Formel mit Formelnummer
<code>gather*</code>	Zentrierte Formel ohne Formelnummer

Umgebung	Beschreibung
align	Ausgerichtete Formeln mit Formelnummer
align*	Ausgerichtete Formeln ohne Formelnummer
flalign	Ausgerichtete Formeln mit Formelnummer
flalign*	Ausgerichtete Formeln ohne Formelnummer
alignat	Mehrfach ausgerichtete Formeln mit Formelnummer
alignat*	Mehrfach ausgerichtete Formeln ohne Formelnummer
Multiline	Mehrzeilige Formel mit Formelnummer
multiline*	Mehrzeilige Formel ohne Formelnummer
split	Mehrzeilige Formeln innerhalb einer anderen Umgebung

Tabelle 11.8: Umgebungen des amsmath-Pakets

Anbei sehen Sie einige Beispiele für den allgemeinen Aufruf dieser Umgebungen:

```
% equation Umgebung
\begin{equation}
  a = b
\end{equation}

\begin{equation*}
  a = b
\end{equation*}

% gather Umgebung
\begin{gather}
  a = b
\end{gather}

% align Umgebung
\begin{align}
  a &= b
\end{align}

% flalign Umgebung
\begin{flalign}
  a &= b
\end{flalign}

% multiline Umgebung
\begin{multiline}
  a = b
\end{multiline}
```

```
% split Umgebung
\begin{split}
\end{split}
```

Mehrzeilige Formeln ohne Ausrichtung

Mit der Umgebung `multiline` können Sie mehrzeilige Formeln ohne Ausrichtung setzen. Die `multiline`-Umgebung stellt eine Variation der `equation`-Umgebung dar. Die erste Zeile der Gleichung wird dabei am linken, die letzte Zeile am rechten Rand ausgerichtet. Dieses Verhalten können Sie mit dem Befehl `\multilinegap` beeinflussen, indem Sie die Einrücktiefe definieren. Alle übrigen Zeilen dazwischen werden unabhängig voneinander innerhalb des Anzeigebereichs zentriert, wenn nicht der Parameter `fleqn` gesetzt wird.

Mehrzeilige Formeln mit Ausrichtung

Die `split`-Umgebung dient dazu, mehrzeilige Formeln mit Ausrichtung zu setzen. Der Vorteil dieser Umgebung besteht darin, dass Sie hier mit dem Setzen von Ausrichtungspunkten die Ausrichtung der Gleichungen beeinflussen. Diese Punkte werden mit kaufmännischen Und-Zeichen (&) gesetzt.

Eine weitere Besonderheit dieser Umgebung ist, dass sie keine Nummerierung der Formeln zur Verfügung stellt. Sollen Formeln dennoch nummeriert werden, muss eine `split`-Umgebung in eine `equation`-, `gather`- oder `align`-Umgebung eingebettet werden.

11.3 Sätze und Definitionen

Die Mathematik beruht auf Sätzen und Definitionen. Was ein Satz oder eine Definition ist, lässt sich allerdings nicht immer leicht unterscheiden. Neben Sätzen und Definition gibt es noch Korollare, Lemmata, Axiome und Bemerkungen. Wenn Sie solche mathematischen Grundlagen in Ihrer Publikation verwenden, bietet es sich an, ein durchgängiges Schema zu bestimmen, das Ihnen und dem Leser die Orientierung erleichtert. Dafür steht die `\newtheorem`-Umgebung zur Verfügung.

Diese können Sie mit folgenden Befehlen einsetzen:

- `\newtheorem{<name>}{<marke>}`
- `\newtheorem{<name>}[<name>]{<marke>}`
- `\newtheorem{<name>}{<marke3>}[<section>]`

Alle drei Umgebungen funktionieren sehr ähnlich. Es werden ein Name einer neuen Umgebung und eine Marke definiert, die für die Zählung verwendet werden soll. Mit der zweiten Variante können Sie den Zähler einer bereits definierten Umgebung weiter-

hin benutzen. Mit der dritten Variante lässt sich bestimmen, dass in die Nummerierung Ihrer Umgebungen die Abschnittsnummer eingezogen wird. Außerdem können Sie beim Eintritt in diese definierten Umgebungen noch einen optionalen Text angeben, der automatisch fett und in Klammern gesetzt wird.

Ein Beispiel: Vielleicht wollen Sie einen mathematischen Satz anhand von Beispielen erklären, die durchnummeriert werden sollen. Diesen Beispielen wollen Sie noch einige Aufgaben begeben. Zunächst werden hierzu mithilfe von `\newtheorem` die entsprechenden Umgebungen definiert:

```
\newtheorem{Sa}{Satz}[Satz des Thales]
\newtheorem{Bsp}{Beispiel}
\newtheorem{Auf}{Aufgabe}
\newtheorem{Aufg}[Auf]{Weitere Aufgaben}
```

Die so entstandenen Umgebungen werden nun angewendet. Zuerst folgt der einleitende Satz des Thales:

```
\begin{Sa}
Der Satz des Thales lautet kurzgefasst: Alle Winkel am Halbkreisbogen sind
rechte Winkel.
\end{Sa}
```

Wir fügen nun ein Beispiel an:

```
\begin{Bsp}
Konstruiert man ein Dreieck aus den beiden Endpunkten des Durchmessers
eines Halbkreises (Thaleskreis) und einem weiteren Punkt dieses
Halbkreises, so erhält man immer ein rechtwinkliges Dreieck.
\end{Bsp}
```

Danach setzen wir ein zweites Beispiel und schließen zwei Aufgaben an:

```
\begin{Bsp}
Ein Beispiel...
\end{Bsp}

\begin{Auf}
Eine Aufgabe...
\end{Auf}

\begin{Auf}
Eine Aufgabe...
\end{Auf}
```

Dann fügen wir ein weiteres Beispiel ein:

```
\begin{Bsp}  
Ein Beispiel...  
\end{Bsp}
```

Und zuletzt fahren wir mit den Aufgaben fort:

```
\begin{Aufg}  
Eine weitere Aufgabe...  
\end{Aufg}  
  
\begin{Auf}  
Und noch eine Aufgabe...  
\end{Auf}
```

Der Satz des Thales

Satz 1 *Der Satz des Thales lautet kurzgefasst: Alle Winkel am Halbkreisbogen sind rechte Winkel.*

Beispiel 1 *Konstruiert man ein Dreieck aus den beiden Endpunkten des Durchmessers eines Halbkreises (Thaleskreis) und einem weiteren Punkt dieses Halbkreises, so erhält man immer ein rechtwinkliges Dreieck.*

Beispiel 2 *Ein Beispiel...*

Aufgabe 1 *Eine Aufgabe...*

Aufgabe 2 *Eine Aufgabe...*

Beispiel 3 *Ein Beispiel...*

Weitere Aufgaben 3 *Eine weitere Aufgabe...*

Aufgabe 4 *Und noch eine Aufgabe...*

Abbildung 11.21: Die mit `\newtheorem` definierten Umgebungen in der Praxis.

12 Fehler

Bei jeder Programmiersprache gibt es unendlich viele Möglichkeiten, Fehler zu machen. LaTeX ist ja in gewissem Sinne eine Programmiersprache und meldet Fehler, wenn Sie sich vertippen. Die meisten Fehler bei LaTeX sind auf vergessene Klammern, falsche Klammertypen (eckig statt geschweift und umgekehrt) oder nicht eingebundene Pakete zurückzuführen. In diesem Kapitel werden Ihnen die häufigsten Fehlermeldungen vorgestellt. Außerdem erfahren Sie, wie Sie Fehler vermeiden und aufspüren können.

12.1 Fehler finden

Das Problem bei Fehlern ist häufig nicht, sie zu beheben, sondern die Fehler erst einmal zu finden. Hierbei leistet Ihnen LaTeX durch seinen hervorragenden Compiler gute Dienste.

Das folgende Beispiel produziert einen einfachen Fehler:

```
\documentclass[a4paper]{article}  
\begin{document  
Hallo Welt  
\end{document}
```

Beispiel 12.1: Ein einfacher Fehler

Wenn Sie dieses Skript kompilieren lassen, meldet sich LaTeX bzw. in diesem Fall die grafische Umgebung Texmaker und gibt folgenden Fehler aus:

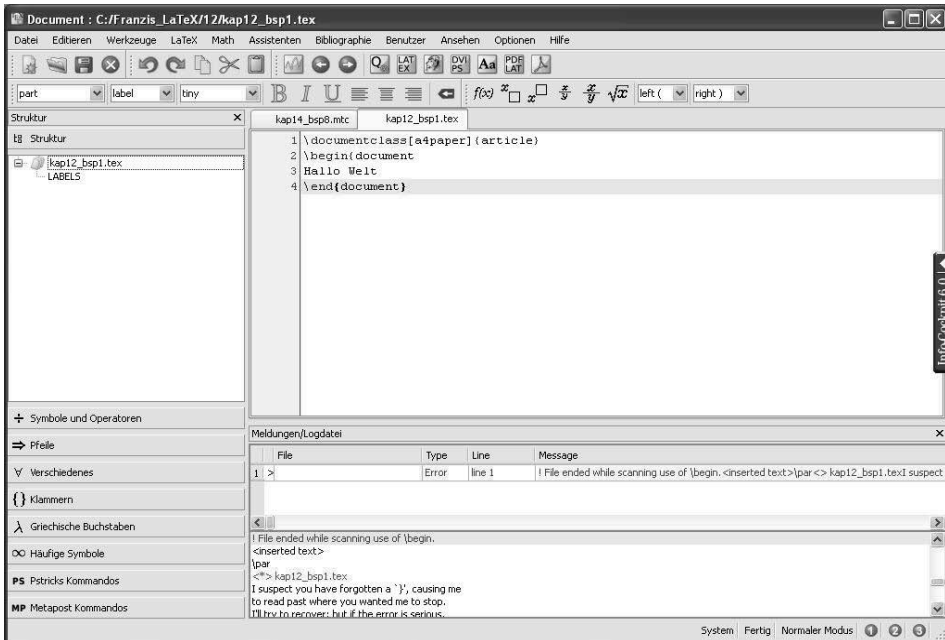


Abbildung 12.1: Fehlermeldung von Texmaker

LaTeX meldet in diesem Fall den Fehler »File ended while scanning use of `\begin`«. Das bedeutet: Die Überprüfung des Dokumentes wurde beendet, weil LaTeX einen Fehler beim Befehl `\begin` gefunden hat. Diese Fehlermeldung wird unterhalb des Editorfensters in Rot angezeigt. Im vorliegenden Fall wurde einfach eine geschweifte Klammer vergessen.

Sprechender Compiler

Ein Vorteil des LaTeX-Compilers ist es, dass es sich um einen »sprechenden« Compiler handelt. Das heißt, LaTeX macht Ihnen unter anderem Vorschläge, wie Sie das Problem am besten beheben können. Ferner schreibt LaTeX alle Fehlerausgaben in eine Logdatei. In der Konsole unterhalb der eigentlichen Fehlermeldung zeigt LaTeX noch weitere Meldungen an, die das Problem genauer analysieren.

```
LOG FILE :
This is pdfTeX, Version 3.1415926 1.40.9 (MiKTeX 2.7) (preloaded
format pdfLaTeX 2009.4.10) 8 MAY 2009 20:08
entering extended mode
```

Darunter erscheint dann in Grün die zu kompilierende Datei:

```
**kap12_bsp1.tex
(kap12_bsp1.tex
```

Es folgt eine genauere Fehleranalyse mit einem Verbesserungsvorschlag:

```
I suspect you have forgotten a `}', causing me
to read past where you wanted me to stop.
I'll try to recover; but if the error is serious,
you'd better type `E' or `X' now and fix your file.
```

In diesem Fall vermutet LaTeX, dass eine schließende geschweifte Klammer (}) fehlt und meldet, dass es deshalb den Compilerlauf angehalten hat. Es folgen weitere Eingabemöglichkeiten für die Verarbeitung der Fehlermeldung. Diese Befehle stehen auf der Konsole zur Verfügung:

- **Return**
Diese Taste fordert den Compiler auf, weiterzuarbeiten. Der entsprechende Befehl oder das entsprechende Wort wird dabei ignoriert.
- **S - scroll mode**
Mit diesem Tastendruck setzt LaTeX die Bearbeitung fort. Bei Folgefehlern werden diese nacheinander am Bildschirm angezeigt.
- **R - run mode**
In diesem Modus setzt LaTeX die Bearbeitung fort und hält auch nicht bei Folgefehlern an.
- **Q - quiet mode**
Dieser Modus ist ähnlich wie der Modus R, es erfolgt jedoch keine weitere Ausgabe. Die Fehler werden aber in die Logdatei geschrieben.
- **I - insert mode**
Beim `insert mode` können Sie den Fehler durch Eingabe des richtigen Wertes oder Wortes beheben. Der Fehler wird jedoch nicht in der originalen Datei behoben.
- **l ...**
Bei der Eingabe einer Zahl kleiner als 100 überspringt der Compiler die nächsten x Zeilen.
- **H - Hilfe**
Zeigt eine Fehlerbeschreibung an und einen Hinweis, wie der Fehler behoben werden kann.
- **X - Exit**
Die Bearbeitung wird abgebrochen.
- **E - Edit**
Bearbeitungsmodus. LaTeX öffnet den Editor und springt in die Zeile, in der der Fehler aufgetreten ist. Dieser Modus ist nicht in jeder LaTeX-Version vorhanden.

Wenn Sie die fehlende }-Klammer eingeben und neu kompilieren lassen, wird das Dokument fehlerfrei übersetzt.

Häufige Fehler

In nahezu jeder Programmier- und Markup-Sprache gibt es eine Topliste der Fehler, die man nie machen sollte, die sich aber immer wieder einschleichen. LaTeX stellt dabei keine Ausnahme dar. Insbesondere als Anfänger macht man häufig diese Fehler, weshalb hier kurz eine Liste der typischen LaTeX-Fehler aufgeführt wird:

- **Klammern schließen**
Die häufigsten Fehler bei LaTeX sind vergessene Klammern. LaTeX ist bei der Verwendung von Klammern sehr kleinlich und meldet sofort einen Fehler, falls Sie irgendwo vergessen haben, eine geöffnete Klammer zu schließen.
- **Falsche Parameter**
Ein anderer beliebter Fehler ist die Übergabe von falschen Parametern bzw. Argumenten. LaTeX meldet auch hier sofort einen Fehler, wenn das Argument eines Befehls nicht richtig ist.
- **Falsche Befehle**
Häufig werden auch Befehle falsch geschrieben. LaTeX merkt sofort, wenn der Befehl falsch ist, und meldet den entsprechenden Fehler.
- **Falsches Umgebungsende**
Wenn Sie Umgebungen schachteln, kann es passieren, dass Sie eine Schachtelungstiefe zu wenig geschlossen haben oder eine Umgebung mit einer falschen Umgebung beendet haben. Auch dies führt sofort zu einem Fehler.

Hier ein Beispiel für die oben genannten Fehler:

```
\begin{document}
%Falscher Befehl
\sqt
%Falsche Klammerschließung
\begin{verbatim}
\end{verbatim}
%Falscher Parameter
\relsize{1}
%Falsches Umgebungsende
\begin{math}
\end{equation}
\end{document}
```

Beim Auftreten eines solchen Fehlers ist LaTeX sofort zur Stelle und bringt eine Fehlermeldung. Je komplizierter die Befehle und Kommandostrukturen werden, desto schwieriger wird es allerdings, den Fehler zu lokalisieren. Es empfiehlt sich daher, komplizierte Formulierungen zu vermeiden und möglichst einfache Konstrukte zu wählen.

12.2 Fehlermeldungen und Warnungen

In diesem Abschnitt finden Sie eine Liste von Fehlermeldungen und Warnungen in alphabetischer Reihenfolge. Weitere Fehlermeldungen finden Sie in der LaTeX-Dokumentation:

- `!Double subscript`
Bei einer mathematischen Formel treten zwei Tiefstellungsbefehle `»_«` hintereinander auf, ohne dass diese geklammert sind.
- `!Double superscript`
Bei einer mathematischen Formel treten zwei Hochstellungsbefehle `»^«` hintereinander ohne Klammerung auf.
- `! Extra }, or forgotten $`
Es wurde eine rechte oder linke Klammer `»{« »}«` vergessen oder es wurde vergessen, den Mathematikmodus zu beenden.
- `Extra alignment tab has been changed to \cr`
Eine Zeile in einer Tabelle in einer `tabular`- oder `array`-Umgebung enthält mehr Spalteneinträge, als Spalten vorhanden sind.
- `I can't find file`
LaTeX kann eine Datei nicht finden. LaTeX sucht dann in den folgenden Verzeichnissen:
 - im aktuellen Verzeichnis
 - in allen Verzeichnissen, die in der Umgebungsvariablen `TEXINPUTS` angegeben sind
 - in der Verzeichnisstruktur des `texmf`-Baumes.
- `Illegal Parameter number in definition of...`
Bei einer Befehlsdefinition mit `\newcommand`, `\renewcommand`, usw. wurde das `#`-Zeichen falsch benutzt.
- `Illegal unit of measure (pt inserted)`
Es wurde eine Längeneinheit erwartet, aber eine Zahl wurde ohne eine Längeneinheit wie `pt` eingegeben.
- `Misplaced alignment tab character &`
Sie haben das `&`-Zeichen in normalem Text benutzt. Dieses Zeichen darf nur in Tabellenumgebungen verwendet werden. Wenn Sie zu viele `&`-Zeichen verwenden, meldet LaTeX ebenfalls einen Fehler.
- `Missing $ inserted`
Sie haben ein Symbol eingegeben, das nur im mathematischen Modus benutzt werden darf.

- `Missing { inserted`
Eine öffnende Klammer fehlt.
- `Missing control sequence inserted`
Bei einer neuen Befehlsdefinition mit `\newcommand` usw. wurde der führende Backslash `\` vergessen.
- `Missing number, treated as Zero`
Sie haben einen Befehl aufgerufen, der einen Parameter erwartet, haben diesen aber nicht angegeben.
- `Not a letter`
Sie haben in der Trennliste des `\hyphenation`-Befehls ein Zeichen verwendet, das nicht als Buchstabe angesehen wird.
- `Paragraph ended before... was complete`
Sie haben bei einem Befehlsargument ein Leerzeichen oder den `\par` Befehl angegeben, der nicht erlaubt ist. Möglicherweise haben Sie auch eine Klammer vergessen.
- `Undefined control sequence`
Sie haben einen Befehl falsch geschrieben oder der Befehl ist nicht vorhanden, weil Sie ein Ergänzungspaket nicht eingebunden haben.
- `!TeX capacity exceeded sorry`
Sie haben die Speichergrenze von LaTeX überschritten. Sie haben dann vermutlich eine Klammer vergessen oder ein zu großes Dokument erzeugt.

LaTeX verwaltet intern folgende Speicher (diese können Sie in einigen Distributionen wie TeXLive in der `texmf`-Datei anpassen):

- `buffer size`
Hier ist die Ursache vermutlich eine zu lange Zeile oder ein falsch erkanntes Newline-Zeichen am Ende der Zeile (dieses Problem tritt meistens dann auf, wenn Sie Texte zwischen verschiedenen Systemen austauschen).
- `exception dictionary`
Die Trennliste ist zu lang.
- `main memory`
In diesem Speicher verarbeitet LaTeX jede einzelne Seite. Wenn dieser Speicher überläuft, haben Sie zu viele Definitionen in der Seite vorgenommen.
- `hash size`
Sie haben zu viele Befehlsdefinitionen oder Referenzmarken in der Eingabedatei eingegeben.
- `pool size`
In diesem Speicher werden die Marken für Befehlsdefinitionen und Querverweise gespeichert. Wenn diese Namen zu lang sind, kommt es zu einem Überlauf.

- `save size`
Wenn Sie Befehle zu tief geschachtelt haben, tritt ein Überlauf in diesem Speicher auf.

LaTeX gibt in der Logdatei Auskunft darüber, wie viel Speicher Sie verwendet haben.

- LaTeX error: Bad `\line` or `\vector` argument
Sie haben ein ungültiges `\line`- oder `\vector`-Argument angegeben. Als Alternative können Sie hier das Paket `pict2e` verwenden.
- LaTeX error: Bad math environment delimiter
Sie sind in den mathematischen Modus gewechselt, obwohl sich LaTeX bereits im mathematischen Modus befindet. Dieser Fehler tritt auch auf, wenn Sie im normalen Modus eine Folge von Kommandos eingegeben haben, die Sie nur im mathematischen Modus eingeben dürfen.
- LaTeX error: can be used only in preamble
Sie haben einen Befehl benutzt, der nur in der Präambel verwendet werden darf.
- LaTeX error: Command... already defined
Sie haben einen neuen Befehl definiert, obwohl dieser schon vorhanden ist.
- LaTeX error: Command... invalid in math mode
Sie haben einen Befehl im mathematischen Modus verwendet, der dort nicht erlaubt ist.
- LaTeX error: Command... undefined in encoding
Sie haben einen Schriftbefehl im falschen Codeattribut verwendet (z. B. in T1, wenn er in OT definiert ist).
- LaTeX error: Counter too large
Dieser Fehler tritt dann auf, wenn ein Zähler mit Buchstaben dargestellt wurde und der Wert größer als 26 ist (oder bei Fußnoten den Wert 9 hat).
- LaTeX error: Environment undefined
Sie haben eine Umgebung verwendet, die Sie vorher nicht definiert haben. Sie haben dann entweder den Namen falsch geschrieben oder das entsprechende Paket nicht eingebunden.
- LaTeX error: file not found
LaTeX konnte eine Datei nicht finden.
- LaTeX error: Floats lost
Sie haben ein gleitendes Objekt in einer `minipage`-Umgebung oder `\par`-Umgebung definiert. LaTeX hat bei der Ausgabe der Seite festgestellt, dass das Objekt nicht gedruckt werden konnte.

- LaTeX error: illegal character in encoding
Sie haben in einer `tabular`- oder `array`-Umgebung ein unbekanntes Spaltenformatierungszeichen eingegeben.
- LaTeX error: lonely \item perhaps a missing list environment
Sie haben einen `\item` Befehl außerhalb der Aufzählungsumgebung benutzt.
- LaTeX error: missing @exp in array arg
Sie haben das `@`-Zeichen in einer Spaltendefinition verwendet, ohne dass dabei Text in geschweiften Klammern eingegeben wurde.
- LaTeX error: missing begin{document}
Sie haben vergessen, `\begin{document}` zu schreiben.

12.3 Fehler mit Hilfsprogrammen auffinden

Darüber hinaus gibt es auch noch verschiedene Hilfsprogramme, mit denen Sie Fehler finden können.

Das Hilfsprogramm lacheck

Das Hilfsprogramm `lacheck` ist ein kleines Konsolenprogramm, mit dem Sie Fehler in LaTeX-Dokumenten finden können. Dieses Programm können Sie über die Internetadresse http://www.ctan.org/tex_archive/support/lacheck/ beziehen.

Das Programm gibt es für die verschiedenen Betriebssysteme in Form einer ZIP-Datei zum Download. Sie können dieses Programm herunterladen und in ein beliebiges Verzeichnis entpacken.

Der allgemeine Aufruf von `lacheck` über die Kommandozeile lautet:

```
lacheck <datei>.tex
```

Kopieren Sie eine TEX-Datei in das Verzeichnis von `lacheck` und streuen Sie dort einen Fehler ein. In folgender TEX-Datei wurde der Befehl `\begin{document}` falsch geschrieben.

```
\documentclass[a4paper]{article}
\begin{document}
\setcounter{enumiii}{3}
Zähler: \arabic{enumiii}
Zähler: \Roman{enumiii}
Zähler: \alph{enumiii}
\end{document}
```

Wenn Sie diese Datei mit `lacheck` aufrufen, findet das Programm den Fehler sofort:

```
lacheck kap13 bsp1.tex
```

```
C:\WINDOWS\system32\cmd.exe
Volumenseriennummer: C49D-A7D6

Verzeichnis von C:\lacheckpc
09.05.2009 09:38 <DIR>      .
09.05.2009 09:38 <DIR>      ..
05.05.2009 12:23           160 kap13_bsp1.tex
29.10.1997 17:58        37.748 lacheck.exe
29.10.1997 20:09        13.842 lacheck.ps
29.10.1997 20:09         5.214 lacheck.txt
29.10.1997 18:58        49.152 lacheckw32.exe
29.10.1997 20:05         1.072 README.pc
        6 Datei(en)          107.188 Bytes
        2 Verzeichnis(se), 235.842.486.272 Bytes frei

C:\lacheckpc>lacheckw32.exe kap13_bsp1.tex
C:\lacheckpc>lacheckw32.exe kap13_bsp1.tex
"kap13_bsp1.tex", line 2: <- unmatched ">"
"kap13_bsp1.tex", line 1: -> unmatched "beginning of file kap13_bsp1.tex"
"kap13_bsp1.tex", line 7: <- unmatched "\end{document}"
"kap13_bsp1.tex", line 1: -> unmatched "beginning of file kap13_bsp1.tex"
"kap13_bsp1.tex", line 7: "end of file kap13_bsp1.tex" found at top level

C:\lacheckpc>
```

Abbildung 12.2: Das Programm `lacheck`

Das Programm `chktx`

Das Hilfsprogramm `chktx` ist ebenfalls ein kleines Konsolenprogramm, das ähnlich funktioniert wie `lacheck`. Sie finden es im Netz unter folgender Adresse: <http://www.ctan.org/tex archive/support/chktx/>

Das Programm `chktx` wird so aufgerufen:

```
Chktx <.tex Datei>
```

Das Programm `chktx` hat eine Reihe von Parametern, die Sie sich über die Hilfefunktion anzeigen lassen können. So erfolgt der Aufruf:

```
Chktx help
```

Daraufhin wird die Hilfe von `chktx` angezeigt.

Tipp: Das Programm `chktx` gibt es nur als TAR ZIP-Datei für Linux-Systeme. Wenn Sie unter Windows arbeiten, reicht das Programm `lacheck` vollkommen aus.

LaTeX FAQ lesen

Viele häufig gestellte Fragen werden auch in der LaTeX-FAQ beantwortet. Den Link zur FAQ finden Sie unter: <http://www.dante.de/faq/de tex faq/html/de tex faq.html>

Mailinglisten

Wenn Sie mit den oben genannten Mitteln nicht weiterkommen, können Sie auch Fragen in Mailinglisten oder Internet-Foren stellen. Ein deutschsprachiges Forum finden Sie zum Beispiel unter www.golatex.de.

13 Längenmaße, Zähler und neue Befehle

In diesem Kapitel dreht sich alles um Längen, Zähler und neue Befehle. LaTeX unterstützt verschiedene Maßeinheiten wie z. B. Zentimeter, Meter und so weiter. Zähler können Sie für die Nummerierung von Kapiteln und Abbildungen einsetzen. In LaTeX lassen sich auch unter anderem Zähler und neue Befehle selbst erzeugen. Letzteres funktioniert ganz einfach über das Kommando `\newcommand`.

13.1 Längen und Maßeinheiten

In LaTeX können Sie diverse Längen und Maßeinheiten konzipieren.

Feste Maße

Ein festes Maß besteht aus einer Zahl plus der dazugehörigen Einheit (also z. B. 5 cm, 12 pt, 15 mm, 7 in). Im Gegensatz dazu gibt es auch elastische Maße; mehr darüber im nächsten Abschnitt. Zentimeter können zum Beispiel wie folgt in cm notiert werden:

20.5 cm 5cm

Die Tabelle zeigt die gängigen Maßeinheiten. Als Dezimaltrenner kann der Dezimalpunkt ».<« oder das Komma », « verwendet werden.

Maßeinheit	Beschreibung
cm	Zentimeter
mm	Millimeter
in	Inches (Zoll=2,54 cm)
pt	Punkte (1 in = 72,27 pt)
pc	Picas (1pc = 12 pt)
bp	Big point (1 in = 72 pt)
dd	Didot (1157 dd = 1238 pt)
cc	Cicero (1 cc = 12 dd)

Maßeinheit	Beschreibung
sp	Scaled point (1 pt = 65336 sp)
em	Breite des Gravierstriches »—« im jeweiligen Zeichensatz
ex	Höhe eines Buchstabens »x« im jeweiligen Zeichensatz

Tabelle 13.1: Maßeinheiten in LaTeX

Für die Festlegung der Maßeinheiten stehen folgende Befehle zur Verfügung:

- `\setlength{\Länge}{<Wert>}`
Damit wird ein Längenregister auf einen bestimmten Wert gesetzt, wie zum Beispiel `\setlength{\textwidth}{10cm}`.
- `\addtolength{\Länge}{<Wert>}`
Damit wird der Wert zum Längenregister hinzuaddiert. Hier kann auch ein negativer Wert angegeben werden, um ein Längenregister zu verkleinern.
- `\addtolength{\textlength}{ 3mm}`
- `\newlength{\Länge}`
Damit wird ein neues Längenregister erzeugt. Die jeweilige Anzahl an möglichen Längenregistern ist jeweils von LateX-Variante zu LaTeX-Variante unterschiedlich. Nach dem Erzeugen des neuen Längenregisters hat dieses dann den Wert 0.

Elastische Maße

Neben den oben genannten festen Maßen gibt es in LaTeX auch elastische Maße. Diese werden mit dem Befehl `\setlength` durch die Angabe von diversen Parametern gebildet. Elastische Maße dienen dazu, ein Layout an die bestehenden Platzverhältnisse anzupassen. Ein Layout kann auf diese Weise innerhalb der vorgegebenen Parameter atmen. Ein elastisches Maß sieht so aus:

```
<Sollwert> plus <Dehnwert> minus <Stauchwert>
```

Mit dem Sollwert wird das Grundmaß angegeben. Sollte mehr Platz vorhanden sein, kann dieses maximal um den Dehnwert erhöht werden. Wenn weniger Platz vorhanden ist, kann es um den Stauchwert verringert werden. LaTeX verwendet solche elastischen Maße an vielen Stellen automatisch, etwa bei der Festlegung von Absatzabständen. Der Befehl `\setlength` wird benutzt, um die elastischen Maße zu definieren:

```
\setlength{\parskip}{1ex plus0.3ex minus0.1ex}
```

Zu beachten ist dabei, dass zwischen `plus` und `minus` und den jeweils zugeordneten Dehn- und Stauchwerten kein Leerzeichen stehen darf.

Neben den oben genannten Befehlen gibt es noch weitere Kommandos:

- `\settoheight{\Länge}{<text>}`
Damit wird die Länge auf die Höhe eines Textes in einer L-R Box gesetzt.
- `\settodepth{\Länge}{<text>}`
Damit wird die Länge auf die Tiefe des Textes gesetzt.
- `\stretch{<faktor>}`
Damit wird eine elastische Länge erzeugt, die ein Vielfaches von `\fill` benutzt.

13.1.1 Zähler

Zähler werden in LaTeX für die Nummerierung von Kapiteln und Abbildungen eingesetzt. Für die Erzeugung von Zählern stellt LaTeX folgende Befehle zur Verfügung:

- `\newcounter{<zählername>}[<rücksetzer>]`
Damit wird ein neuer Zähler definiert. Er erhält die in `zählername` genannte Bezeichnung und wird mit 0 initialisiert. Bei Bedarf kann dieser Zähler an einen anderen, bereits definierten Zähler gekoppelt werden. Dazu muss der Name des bereits bestehenden Zählers angegeben werden; dies geschieht über den optionalen Parameter `rücksetzer` in eckigen Klammern. In diesem Fall wird der neue Zähler wieder auf 0 zurückgesetzt, wenn der andere Zähler mit `\stepcounter` oder `\refstepcounter` um 1 erhöht wird. Die Anzahl an verfügbaren Zählern ist je nach verwendeter LaTeX-Version etwas unterschiedlich.
- `\setcounter{<zählername>}{<wert>}`
Setzt den Zähler auf einen bestimmten Wert.
- `\addtocounter{<zählername>}{<wert>}`
Damit wird der angegebene Wert zu dem Zähler addiert. Ist der Wert negativ, wird er vom Zähler abgezogen.
- `\stepcounter{<zählername>}`
Damit wird der aktuelle Zähler um 1 erhöht. Sofern über `\newcounter` ein anderer Zähler mit dem aktuellen verkoppelt wurde, wird der andere Zähler auf 0 zurückgesetzt.
- `\refstepcounter{<zählername>}`
Dieser Befehl arbeitet genauso wie `\stepcounter`, erhöht also einen Zähler um 1. Dieser Befehl wird dann verwendet, wenn man einen Zählerstand ändern möchte, auf den per `\ref`-Befehl Bezug genommen wurde. Mit `\refstepcounter{figure}` erhöhen Sie den Zähler einer `figure`-Umgebung um 1.

- `\value{<zählername>}`
Liefert den Wert des Zählers.
- `the<zählername>`
Damit wird der Wert des jeweiligen Zählers ausgegeben. In diesem Fall folgt der Zählername direkt ohne die geschweiften Klammern. Für jeden Zähler wird dabei ein jeweiliger Befehl `the<zählername>` definiert.

Ausgeben von Zählern

LaTeX bietet verschiedene Möglichkeiten, Zähler auszugeben. Die Tabelle gibt einen Überblick über die verschiedenen Befehle zur Nummerierung:

Kommando	Beschreibung
<code>arabic</code>	Arabische Ziffern (1,2,3)
<code>roman</code>	Kleine römische Ziffern (i,ii,iii)
<code>Roman</code>	Große römische Ziffern (I,II,III)
<code>alph</code>	Fortlaufende Kleinbuchstaben (a,b,c)
<code>Alph</code>	Fortlaufende Großbuchstaben (A,B,C)
<code>fnsymbol</code>	Liefert Fußnotensymbole für die Zahlen 1 bis 9: *†‡ § ¶ **††‡‡

Tabelle 13.2: Nummerierungsmöglichkeiten

Hier ein einfaches Beispiel für die verschiedenen Befehle:

```
\documentclass[a4paper]{article}
\begin{document}
\setcounter{enumiii}{3}
\arabic{enumiii}
\Roman{enumiii}
\fnsymbol{enumiii}
\end{document}
```

Beispiel 13.1: Befehle für Zähler

Dieses Beispiel erzeugt arabische und römische Nummerierungen sowie ein Fußnotensymbol für die Ziffer 3.

13.1.2 Rechenmöglichkeiten

LaTeX bietet verschiedene Möglichkeiten, um mit Längen und Zählern zu rechnen. Das Erweiterungspaket `calc` erweitert die LaTeX-Standardbefehle `\setcounter`, `\addtocounter`, `\setlength` und `\addtolength` um zusätzliche Funktionen.

Die Neudefinition erlaubt:

- Addition und Subtraktion ohne Einschränkung (mit den Operatoren »+« und »-«).
- Multiplikation und Division mit einer ganzen Zahl (mit den Operatoren »*« und »/«).
- Multiplikation mit einer Kommazahl (mit `\real`) bzw. mit einem Bruch (`\ratio`).

Anbei sehen Sie einfache Beispiele:

```
2cm + 3pt
5cm+2*8
8/2
2pt plus 4pt * \real{0.5}
* \real{2.0} * \real{3.0}
```

Das Paket `calc` enthält noch weitere Befehle:

- `\widthof{<text>}`
Damit wird die Breite in der aktuellen Schrift ermittelt.
- `\heightof{<text>}`
Damit wird die Höhe in der aktuellen Schrift ermittelt.
- `\depthof{<text>}`
Damit wird die Tiefe des aktuellen Textes ermittelt.

13.1.3 Vergleichen von Zahlen und Längen

LaTeX bietet, ähnlich wie auch andere Programmiersprachen, verschiedene Konstrukte bzw. Befehle für den Vergleich von Zahlen und Längen. Für den Vergleich von Zahlen und Längen steht das Kommando `\ifthenelse` zur Verfügung. Dazu muss das Ergänzungspaket `ifthen` eingebunden werden.

Der Befehl wird so aufgerufen:

```
\ifthenelse{<test>}{<true Befehl>}{<false Befehle>}
```

Wenn die Bedingung `<test>` wahr ist, dann werden die `true`-Befehle ausgeführt, wenn die Bedingung `<test>` falsch ist, werden die `false`-Befehle ausgeführt. Hier sehen Sie die möglichen Ausprägungen für den Parameter `<test>`:

- `<zahl> < <zahl>`
Bedingung »kleiner als«.
- `<zahl> > <zahl>`
Bedingung »größer als«.

- `<zahl> <zahl>`
Bedingung »gleich«.
- `\isodd`
Prüft, ob der Wert ungerade ist.
- `\isundefined{\befehl}`
Prüft, ob ein Befehl vorhanden ist oder nicht.
- `\equal`
Prüft, ob zwei Zeichenketten gleich sind.
- `\lengthtest{<länge> <länge>}`
Prüft, ob die Länge einen bestimmten Wert unterschreitet, z. B. `\lengthtest{\textwidth < 10cm}`.
- `lengthtest{<länge> > <länge>}`
Prüft, ob die Länge einen bestimmten Wert überschreitet, z. B. `\lengthtest{\textwidth < 10cm}`.
- `lengthtest{<länge> <länge>}`
Prüft, ob zwei Längen gleich lang sind.
- `\boolean{<name>}`
Boolescher Wahrheitswert – true oder false.
- `\(\)`
Erlaubt die Klammerung von logischen Ausdrücken.
- `\not<test>`
Die logische Verneinung.
- `\and<test>`
Das logische Und.
- `\or<test>`
Das logische Oder.
- `\newboolean{<name>}`
Damit wird ein neuer boolescher Ausdruck erzeugt.

Es sind folgende boolesche Ausdrücke vordefiniert:

`hmode`

Gibt true zurück, wenn LaTeX im horizontalen Bearbeitungsmodus ist.

`vmode`

Gibt true zurück, wenn LaTeX im vertikalen Bearbeitungsmodus ist.

mmode

Gibt true zurück, wenn LaTeX im mathematischen Modus ist.

- `\provideboolean{<name>}`
Damit wird ein neuer boolescher Ausdruck erzeugt, falls er noch nicht vorhanden ist.
- `\setboolean{<name>}{<true>|<false>}`
Damit wird der boolesche Wert <name> auf true oder false gesetzt.
- `\whiledo{<test>}{<Befehle>}`
Damit wird eine whiledo-Schleife erzeugt. Die Schleife wird ausgeführt, solange die <test>-Bedingung wahr ist.

```
\usepackage{ifthen}
%im Hauptdokument
\setcounter{enumiii}{1}
\whiledo {\value{enumiii} < 10 }{%
\arabic{enumiii}
\addtocounter{enumiii}{1} }
```

Beispiel 13.2: Die whiledo-Schleife

Die obige Schleife erzeugt alle Zahlen von 1 bis 10.

13.2.1 Neue Befehle erzeugen

In LaTeX können Sie Befehle neu definieren. Dies geschieht in der Regel über den Befehl `\newcommand`. Wenn ein bestimmter Befehl schon vorhanden ist, können Sie ihn über `\renewcommand` neu festlegen. Wenn Sie nicht sicher sind, ob der Befehl schon vorhanden ist, können Sie dies mit `\providecommand` überprüfen.

Die drei Kommandos werden wie folgt aufgerufen:

```
\newcommand{\<Befehl>}[<Anzahl Parameter>]
[<standard>]{<Definition>}

\renewcommand{\<Befehl>}[<Anzahl Parameter>]
[<standard>]{<Definition>}

\providecommand{\<Befehl>}[<Anzahl Parameter>]
[<standard>]{<Definition>}
```

Mit dem Parameter `\Befehl` wird das neu zu definierende Kommando festgelegt. Er enthält also den Namen des neuen Befehls. Über den Parameter `<Anzahl Parameter>` wird die Zahl der Parameter des Befehls angegeben. Wenn Sie den Parameter `<standard>`

angeben, können Sie einen optionalen Parameter vergeben. In geschweiften Klammern folgen dann die Befehle, die unter dem neuen Kommando zusammengefasst werden sollen.

Im folgenden Beispiel wird ein neuer Befehl ohne Parameter erzeugt:

```
\newcommand{\bs}{\textbackslash}
```

Das aktuelle Datum wird mit dem Befehl `\bs today` erzeugt.

In diesem Beispiel wird der Befehl `\bs` definiert, der einen Backslash erzeugt. Damit wird der Befehl `\today` mit einem echten Backslash geschrieben.

Einen etwas komplexeren Befehl definieren wir im folgenden Beispiel:

```
\documentclass[a4paper]{article}
\usepackage{pifont}
\usepackage{ifthen}
\newcounter{mycounter}
\newcommand{\schnipp}[2][\ding{34}]{%
\setcounter{mycounter}{#2}
\whiledo{\value{mycounter} > 0 }{%
#1 \addtocounter{mycounter}{ 1 }
}}
\begin{document}
\schnipp{10}\
\schnipp[\ding{36}]{10}
\end{document}
```

Beispiel 13.3: Eingaben für einen neuen Befehl

Der neu festgelegte Befehl `\schnipp` enthält als optionalen Parameter das Scherensymbol aus dem Schriftsatz »Zapf Dingbats«. Diese Symbole werden dann über den Befehl `\ding` und die entsprechende Nummer des Symbols angesprochen. Die `whiledo`-Schleife sorgt dafür, dass der Befehl, der das Scherensymbol Nr. 34 aus Zapf Dingbats ausgibt, zehnmal ausgeführt wird (`\schnipp` startet unmittelbar nach dem Befehl `\begin{document}` mit dem Wert 10, und bei jedem Schleifendurchlauf wird der Wert 1 abgezogen. Das wiederholt sich beim zweiten `\schnipp`-Befehl, nur wird diesmal das Scherensymbol Nr. 36 ausgegeben.

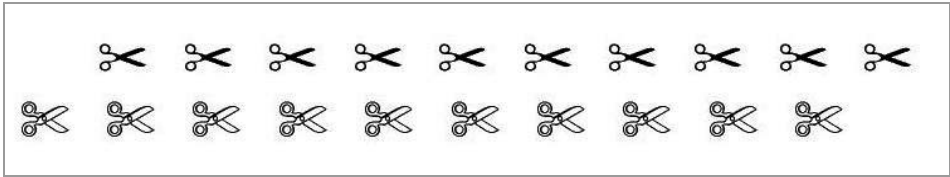


Abbildung 13.1: Über den neu definierten Befehl werden Scherensymbole ausgegeben.

Weitere Parameter übergeben

Mit dem Paket `keyval` ist es möglich, zusätzliche Parameter an Befehle zu übergeben. Die einzelnen Parameter werden dabei durch Kommata getrennt. Das Paket wird von anderen Paketen automatisch geladen und braucht daher nicht manuell vom Benutzer eingebunden zu werden. Das Paket `keyval` bzw. `xkeyval` wird zum Beispiel sehr oft vom Paket `graphics` verwendet.

```
\myline[farbe blau, laenge 20cm,pos c,debug]
```

Für jeden Parameter muss dazu ein Key definiert werden.

Der allgemeine Aufruf lautet:

```
\define@key{<gruppe>}{<key name>}[<standard>]{<Definition>}
```

Mit dem Parameter `gruppe` wird die Zugehörigkeit zu einer Gruppe festgelegt. Hier können Sie einen Namen vergeben, der noch nicht verwendet worden ist, da viele Pakete das Paket `keyval` einbinden. Der Parameter `<key name>` legt den Namen des Parameters fest. Über den Parameter `<standard>` wird der Wert festgelegt, der verwendet werden soll, wenn vorher kein Wert zugewiesen worden ist (ähnlich wie bei `debug`).

Mit dem Befehl `\setkeys` werden die einzelnen Parameter zugeordnet. Unten sehen Sie ein Beispiel für die Verwendung des Pakets `keyval`. Dafür wird der Befehl `\parbox` mit folgenden Parametern umdefiniert:

Parameter	Beschreibung
pos	Position
hohe	Die Höhe der Box
ipos	Die Höhe innerhalb der Box
breite	Die Breite in der Box
debug	Zeigt die eingestellten Werte am Bildschirm an.

Tabelle 13.3: Parameter für den Befehl `myparbox`

Um den Befehl neu festzulegen, werden zunächst die benötigten Pakete eingebunden:

```
\usepackage{keyval}
\usepackage{ifthen}
```

Danach wird mit dem Befehl `\newcommand` der Befehl `\myparbox` mit seinen Parametern neu definiert:

```
\newcommand{\myparbox@pos}{t}
\newcommand{\myparbox@hohe}{1cm}
\newcommand{\myparbox@ipos}{c}
\newcommand{\myparbox@breite}{}
\newcommand{\myparbox@debug}{false}
```

Anschließend werden über das Makro `define@Befehlsname` die einzelnen Parameter des Befehls definiert:

```
\define@key{myparbox}{pos}[\myparbox@pos]{%
\renewcommand{\myparbox@pos}{#1}}
\define@key{myparbox}{hohe}[\myparbox@hohe]{%
\renewcommand{\myparbox@hohe}{#1}}
\define@key{myparbox}{ipos}[\myparbox@ipos]{%
\renewcommand{\myparbox@ipos}{#1}}
\define@key{myparbox}{breite}[\myparbox@breite]{%
\renewcommand{\myparbox@breite}{#1}}
\define@key{myparbox}{debug}[true]{%
\renewcommand{\myparbox@debug}{#1}}
```

Danach wird dann der eigentliche Befehl `\myparbox` definiert:

```
\newcommand{\myparbox}[2][{}]{%
\setkeys{myparbox}{#1}
\ifthenelse{\equal{\myparbox@debug}{true}}{%
{\myparbox@debugshow}}{}
\parbox[\myparbox@pos]%
[\myparbox@hohe]%
[\myparbox@ipos]{\myparbox@breite}{#2}
}
```

Der Befehl `\myparbox` selbst hat zwei Parameter. Einen festen Parameter und einen optionalen. Die Parameter werden dann mit dem Befehl `\setkeys` analysiert und verarbeitet. Mit dem Befehl `\ifthenelse` wird danach der Wert von `debug` abgefragt. Wenn dieser `true` ist, so werden die einzelnen Parameterwerte ausgegeben.

```
\newcommand{\myparboxset}[1]{\setkeys{myparbox}{#1}}
```

Mit dem Befehl `\myparboxset` werden dann die Parameter global gesetzt:

```
\newcommand{\myparbox@debugshow}{%
\typeout{DEBUG myparbox}
\typeout{pos \myparbox@pos}
\typeout{hoehe \myparbox@hohe}
\typeout{ipos \myparbox@ipos}
\typeout{breite \myparbox@breite}
\typeout{debug \myparbox@debug}
\typeout{
}
\makeatother
```

Damit das @-Zeichen verwendet werden kann, müssen zuvor die Befehle `\makeatletter` und `\makeatother` verwendet werden.

Abschließend wird noch der Parameter `debugshow` definiert, der die aktuellen Werte am Bildschirm ausgibt. Danach können Sie den neuen Befehl in dem Hauptdokument aufrufen:

```
\begin{minipage}{\linewidth}
\myparboxset{breite 3cm,hohe 2cm}
\fbbox{
\myparbox{center}
\myparbox{ipos t}{top}
\myparbox{ipos b, debug}{bottom}
}
\end{minipage}
```

Der Befehl selbst wird in eine `minipage`-Umgebung eingebettet.

Leerraum zwischen neue Befehle setzen

LaTeX setzt standardmäßig einen Leerraum zwischen normalen Text und Befehlen. Mit dem Paket `xspace` fügen Sie einen Leerraum nach einem Befehl ein. Das Paket `xspace` stellt den Befehl `xspace` zur Verfügung:

```
{\small Deutschland}\xspace ist ein schönes Land zum Leben.
```

Dieser Befehl ist eine Alternative für `\` bzw. `{}` bei manchen Befehlen. Eine weitere Möglichkeit, einen Leerraum zu erzeugen, ist die Tilde `~`.

Zeilenumbrüche einfügen

Die Definitionen neuer Befehle bringen es mit sich, dass sich die dazugehörigen Kommandos mitunter über mehrere Zeilen erstrecken. Zur besseren Übersicht wäre hier im Listing ein Zeilenumbruch oft wünschenswert. Wenn Sie aber normale Zeilenumbrüche mit der Return-Taste herbeiführen, werden sie von LaTeX standardmäßig

ignoriert und als Leerraum interpretiert. Mit dem Prozentzeichen (%) können Sie dagegen einen Zeilenumbruch bei einem neuen Befehl erzeugen.

Gültigkeitsbereich von neuen Befehlen

Neue Befehle gelten global, wenn sie im Vorspann, d. h. in der Präambel des Dokuments, definiert werden. Andernfalls gelten neue Befehle nur lokal in einer Umgebung, sofern sie dort definiert werden.

In einer Umgebung gilt der neue Befehl nur, solange die Umgebung gültig ist. Wird die Umgebung verlassen, ist der neue Befehl auch nicht mehr gültig.

Argumentbegrenzung von neuen Befehlen

Wie für viele LaTeX-Kommandos gibt es auch für die Befehle `\newcommand`, `\renewcommand` und `\providecommand` eine Sternvariante. Diese Variante sorgt dafür, dass die übergebenen Parameter die Grenze des Absatzes nicht überschreiten. Dies dient dazu, Fehler zu vermeiden. Bei den normalen Varianten dieser Befehle, d. h. ohne Sternvariante, dürfen die Parameter beliebig lang sein. Das kann diverse Fehler nach sich ziehen. Konkret gesagt dürfen die Sternvarianten von `\newcommand`, `\renewcommand` und `\providecommand` keinen Leerschritt, den Befehl `par` oder absatzüberschreitende Strukturen enthalten.

13.2.2 Neue Umgebungen erzeugen

Mit dem Befehl `\newenvironment` können Sie neue Umgebungen definieren. Das Kommando `\renewenvironment` dient zur Umarbeitung einer existierenden Umgebung. Der allgemeine Aufruf lautet:

```
\newenvironment{<Umgebungsname>}[<anzahl Parameter>]
[<standard>]{<start Definition>}{<ende Definition>}

\renewenvironment{<Umgebungsname>}[<anzahl Parameter>]
[<standard>]{<start Definition>}{<ende Definition>}
```

Der Befehl `\newenvironment` hat fast die gleichen Parameter wie der Befehl `\newcommand`. Mit `<Umgebungsname>` wird die neue Umgebung bezeichnet. Mit `<anzahl Parameter>` werden die Parameter angegeben, mit `<start Definition>` und `<ende Definition>` wird definiert, welche Befehle beim Start und beim Ende der Umgebung ausgeführt werden sollen. Es können ein fester sowie neun optionale Parameter vergeben werden.

```
\documentclass[a4paper]{article}
\usepackage{xcolor}

\newcounter{counter}
\newenvironment{Kommentar}[1][
```

```
{\begin{sloppypar}\noindent{\large\textcolor{red}{Kommentar: #1}}}%
\begin{quote}}
{\stepcounter{counter}\hfill(\arabic{counter}\end{quote}\end{sloppypar}}

\begin{document}

\begin{Kommentar}[Wichtig]
Eine neu definierte Umgebung
\end{Kommentar}
\end{document}
```

In diesem Beispiel wird eine Umgebung `Kommentar` definiert, die linksbündig gesetzt wird. Der eigentliche Text des Kommentars wird in Rot ausgegeben. Der eigentliche Kommentar wird in eine `quote`-Umgebung gesetzt.

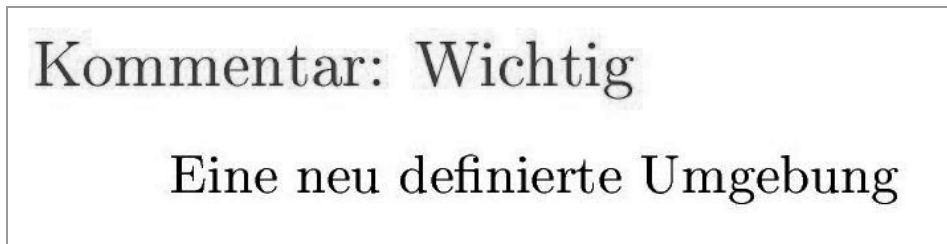


Abbildung 13.2: Die neu definierte Kommentariumgebung fasst die Formatierungsbefehle zusammen.

Parameter im Endbereich

Ein Parameter kann nur im ersten Bereich (Befehl vor der neuen Umgebung) an eine Umgebung übergeben werden. Im zweiten Bereich (Befehle nach der Umgebung) stehen diese Parameter nicht mehr zur Verfügung.

Speichern von Befehlen

Mit den Befehlen `\newsavebox`, `\savebox` und `\usebox` können neue Befehle gespeichert werden. Vor der neuen Umgebung wird eine Art Box erzeugt, in der der Parameter gesichert wird. Die Box wird im ersten Teil, wo der Parameter zur Verfügung steht, gespeichert und steht dann im zweiten Teil bereit.

Verwendung von `keyval`

Genauso wie bei Befehlen können Sie auch bei Umgebungen das Paket `keyval` verwenden.

14 Bücher und wissenschaftliche Arbeiten

Mit LaTeX lassen sich nicht nur kleinere Dokumente erarbeiten, sondern natürlich auch umfangreiche Dokumente wie Bücher, Diplomarbeiten, Dissertationen etc. Der Aufbau eines Buches in LaTeX sieht allerdings etwas anders aus als derjenige der bisher behandelten Dokumente. Längere Publikationen und wissenschaftliche Texte enthalten unter anderem ein Inhaltsverzeichnis, eine differenzierte Gliederung, einen Index und Ähnliches. Dafür stehen verschiedene Hilfsprogramme zur Verfügung.

14.1 Teildokumente

Bei Erstellung eines größeren Dokumentes ist es sinnvoll, dieses in kleinere Teildokumente aufzuteilen. Diese werden dann über entsprechende Befehle in das Hauptdokument eingebunden. Ein Teildokument selbst kann dann wieder eigene Überschriften, Abschnitte etc. enthalten.

Ein Teildokument darf allerdings nicht mit einem eigenen Dokumentkopf, d. h. einer Präambel, versehen sein.

14.1.1 Einbinden von Teildokumenten

Ein Teildokument wird über den Befehl `\input` eingebunden. Der allgemeine Aufruf lautet:

```
\input{Dateiname}
```

Der Befehl `\input` fügt automatisch die Dateiendung `.tex` an. Das Besondere an diesem Befehl ist, dass er rekursiv, d. h. verschachtelt, angewendet werden kann. Das bedeutet, dass ein Teildokument wiederum ein weiteres Teildokument einbinden kann. Das ist unter anderem dann nützlich, wenn mehrere Autoren an einer Publikation arbeiten. Ein spezialisierter Autor kann zum Beispiel einen Abschnitt eines Buchkapitels beisteuern. Für diesen Abschnitt legt er dann ein Teildokument an, das wiederum in das Teildokument des betreffenden Buchkapitels eingebunden wird.

Einbinden von Teildokumenten mit include

Eine weitere Möglichkeit, Teildokumente einzubinden, ist der Befehl `\include`. Der Befehl `\include` darf jedoch im Gegensatz zum Befehl `\input` nicht verschachtelt angewendet werden.

Der Vorteil des Befehls `\include` liegt darin, dass man damit besser steuern kann, welche Dokumente eingebunden werden sollen. Vor dem Einbinden eines Dokumentes wird nämlich immer ein `\clearpage`-Befehl gesetzt. Dadurch wird erreicht, dass immer nur abgeschlossene Dokumente bzw. Abschnitte eingebunden werden.

Eine Variante des Befehls `\include` ist der Befehl `\includeonly`. Dieses Kommando darf im Dokumentkopf verwendet werden.

```
\includeonly{<dateiname>, <dateiname>, ...}
```

Bei `\includeonly` werden nur diejenigen Dateien eingebunden, die durch Komma getrennt angegeben wurden. Wenn Teildokumente mit `\include` eingebunden werden, dürfen sie ihrerseits beliebig viele mit `\input` einbinden.

Tipp: Als Gegenstück zu `\includeonly` gibt es auch den Befehl `\excludeonly`. Dieser sorgt dafür, dass alle Dokumente eingebunden werden, die eben nicht mit diesem Befehl markiert worden sind.

Beispiele

Unten sehen Sie ein Beispiel für eine mögliche Hauptdatei:

```
\documentclass[a4paper]{article}
\include{Kapitel1}
\include{Kapitel2}
\include{Kapitel3}
```

Titelseite einfügen

Jedes Dokument kann in verschiedene Bereiche aufgeteilt werden, z. B. Titel, Name des Autors, Datum usw. Für den Titel gibt es den Befehl `\title`, mit dem Sie einen einfachen Titel einfügen können.

```
\title{Mein erstes Dokument}
\author{Alexander Schunk}
\datum{10.03.2009}
```

Bei dem Befehl `\author` lassen sich in einer geschweiften Klammer mehrere Autorennamen angeben, die durch `\and` verknüpft werden. Wenn die Autorennamen untereinander stehen sollen, müssen Sie anstatt `\and` den doppelten Backslash `\\` verwenden. Mit dem Befehl `\Datum` können Sie ein aktuelles Datum angeben.

Mit `\maketitle` wird die Seite gesetzt. Die Dokumentenklassen `book`, `scrbook`, `screport` und `report` sehen eine eigene Seite für den Titel vor. Dann erhält die folgende Seite die Seite »1«. Bei den Dokumentenklassen `article` bzw. `scrartcle` wird dagegen lediglich ein Titelvorspann erzeugt. Über die Umgebung `titlepage` kann allerdings auch hier eine eigene Seite erzwungen werden.

```
\title{Java Grundlagenbuch}  
\author{Alexander Schunk \and John Doe}  
\begin{document}  
\maketitle  
\end{document}
```

Beispiel 14.1: Titelseite

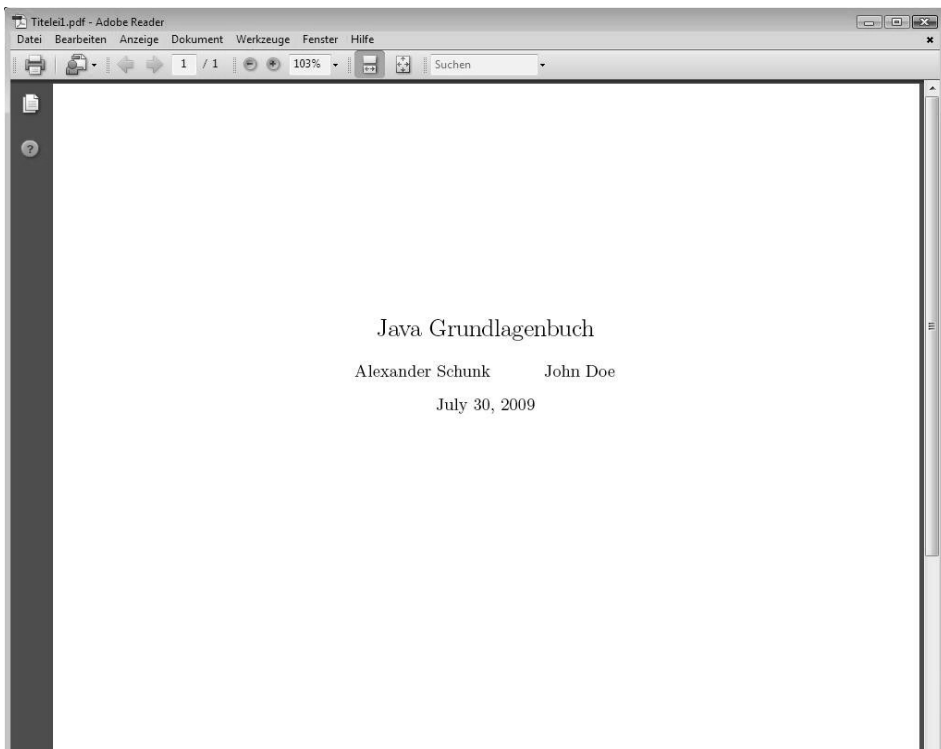


Abbildung 14.1: Ergebnis zu Beispiel 14.1

Den Autorennamen können Sie über \and noch Adressen hinzufügen:

```
\title{Java Grundlagenbuch}  
\author{Alexander Schunk \and Margeritenweg 2}  
\begin{document}  
\maketitle  
\end{document}
```

Beispiel 14.2: Titelseite mit Anschrift

Das Ergebnis zeigt Abbildung 14.2:

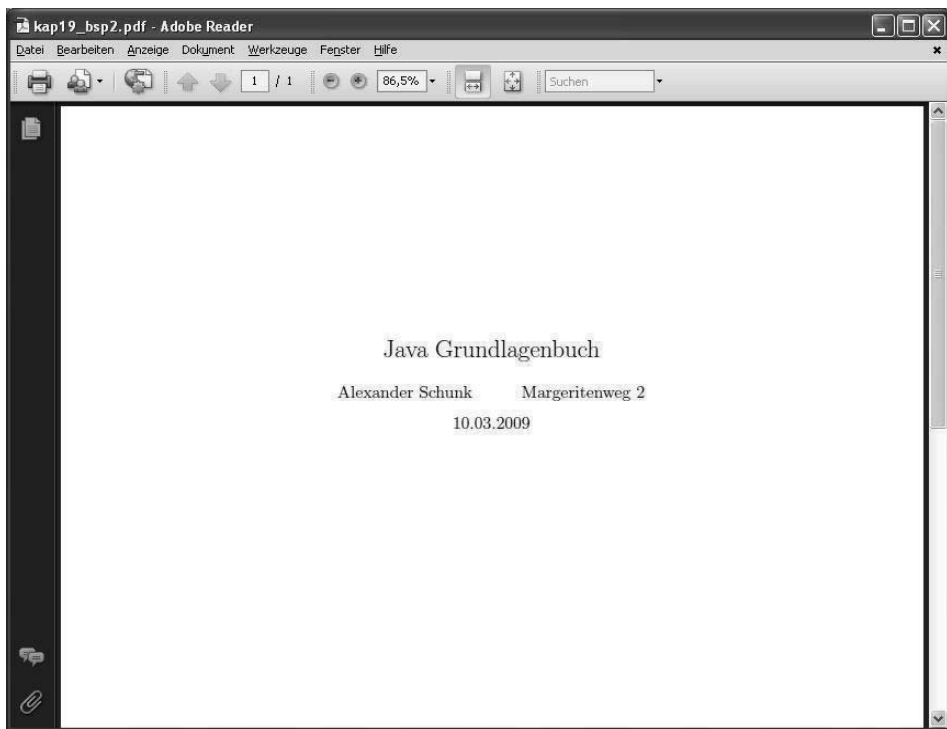


Abbildung 14.2: Ergebnis zu Beispiel 14.2

Titelseite frei gestalten

Mit der bereits erwähnten Umgebung `titlepage` können Sie die Titelseite etwas freier gestalten.

Der allgemeine Aufruf lautet:

```
\begin{titlepage}

\end{titlepage}
```

Auf diese Weise lässt sich die Titelseite nach eigenen Vorstellungen gestalten. Hier eine einfache Variante:

```
\begin{document}
  \begin{titlepage}
    \begin{center}
      {\Huge Java\[\[2ex] Grundlagenbuch\par }
      \vspace{2cm}
      {\large Alexander Schunk\par}
      \vspace{1cm}
      Köln, den \today
    \end{center}
  \end{titlepage}
\end{document}
```

Beispiel 14.3: Selbstgestaltung der Titelseite mit `titlepage`



Abbildung 14.3: Selbst gestaltete Titelseite

Sonstige Pakete für die Titelseite

Außer `\title` bzw. `\titlepage` gibt es noch zwei weitere Pakete für die Gestaltung von Titelseiten.

- `titlesec`
Das Paket erlaubt ebenfalls die einfache Gestaltung von Titelseiten.
- `titles`
Erlaubt die komfortable Gestaltung von Titelseiten.

14.2 Buchgliederungen

Für Bücher sind außer der Gliederung in Kapitel und Abschnitte noch weitere Untergliederungen möglich. So gibt es z. B. Vorspann, Hauptteil und Nachspann:

Befehl	Beschreibung
<code>\frontmatter</code>	Damit wird der Buchvorspann eingeleitet und es werden kleine römische Zahlen (i,ii,iii) als Seitennummerierung verwendet.
<code>\mainmatter</code>	Damit wird der Hauptteil eingeleitet. Hier beginnt die Seitennummerierung wieder mit 1, die mit arabischen Ziffern dargestellt wird.
<code>\backmatter</code>	Damit wird der Nachspann eingeleitet. Die Seitennummerierung des Hauptteils wird hier fortgeführt.

Tabelle 14.1: Gliederungsebenen für Bücher

Die einzelnen Teildokumente innerhalb dieser Bereiche können z. B. mit dem Kommando `\include` eingebunden werden.

```

\frontmatter
\include{Vorwort}
\include{Inhaltsverzeichnis}

\mainmatter
\include{Kapitel1}
\include{Kapitel2}

\backmatter
\include{Anhang}
\include{Index}

```

Beispiel 14.4: Gliederung eines Buches

14.2.1 Die Umgebung `abstract`

Mit der Umgebung `abstract` können Sie Ihrem Artikel oder Buch eine Zusammenfassung des Inhalts voranstellen. Insbesondere bei wissenschaftlichen Artikeln, aber auch bei Business-Dokumenten ist ein derartiger Abstract beziehungsweise ein sogenanntes Management Summary üblich.

Der allgemeine Aufruf lautet:

```

\begin{abstract}
\end{abstract}

```

Bei den Dokumentenklassen `article` bzw. `scrarticle` wird die Schriftgröße mit dem Befehl `\small` gesetzt und beidseitig eingerückt. Darunter erscheint dann der Text. Bei der Umgebung `titlepage` wird stattdessen eine eigene Seite verwendet.

Bei den Dokumentenklassen `report` bzw. `screport` wird der Text hingegen in der Standardgröße gesetzt und nicht eingerückt. Auch hier wird eine eigene Seite dafür vorgesehen.

Bei den Dokumentenklassen `book` bzw. `scrbook` wird nicht zusammengefasst.

Mit dem Paket `abstract` können Sie die Zusammenfassung auf verschiedene Weise beeinflussen und formatieren.

14.3 Verzeichnisse

In jeder umfangreicheren Publikation gibt es verschiedene Verzeichnisse, z. B. Inhaltsverzeichnisse, Stichwortverzeichnisse und Ähnliches, die es Lesern erlauben, sich schnell zu orientieren und die gewünschte Information zu finden.

14.3.1 Inhaltsverzeichnis

Mit dem Befehl `\tableofcontents` haben Sie die Möglichkeit, ein Inhaltsverzeichnis von Kapiteln und Abschnitten zu erstellen. Der Inhalt wird in eine eigene Datei geschrieben, die die Endung `.toc` hat. Damit die Einträge mit den aktuellen Seitenzahlen übereinstimmen, ist es erforderlich, mindestens zwei Kompilierungsläufe in LaTeX zu starten.

Tipp: Damit die folgenden Beispiele funktionieren, müssen Sie als Dokumenttyp die Klasse `book` verwenden.

Hier ein Beispiel für den Einsatz des Befehls `\tableofcontents`:

```
\documentclass[a4paper]{book}
\begin{document}
  \tableofcontents
  \chapter{Kapitel 1}
  \section{Erster Absatz}
  \chapter{Kapitel 2}
  \section{Zweiter Absatz}
\end{document}
```

Beispiel 14.5: Inhaltsverzeichnis erstellen

Das so erzeugte Inhaltsverzeichnis steht dann in der dazugehörigen TOC-Datei, die wie folgt aussieht:

```
\contentsline {chapter}{\numberline {1}Kapitel 1}{3}
\contentsline {section}{\numberline {1.1}Erster Absatz}{3}
\contentsline {chapter}{\numberline {2}Kapitel 2}{5}
\contentsline {section}{\numberline {2.1}Zweiter Absatz}{5}
```

Beispiel 14.6: TOC-Datei

In der TOC-Datei wird der Befehl `\contentline` aufgerufen; dahinter stehen die jeweiligen Kapitel bzw. Unterabschnitte mit den dazugehörigen Namen, den Gliederungsnummern sowie der Seitennummerierung.

```
\contentline{<Gliederungsname>}{\numberline}{<Gliederungsnummer>}{<Gliederungstext>}{<Seitennummer>}
```

Mit dem Gliederungsnamen werden die einzelnen Abschnitte eines Dokumentes wie z. B. Kapitel, Abschnitte und Ähnliches angesprochen. Die Gliederungsnummer stellt die automatisch erzeugte Nummerierung der Überschrift dar, gefolgt von dem Text der Überschrift. Wenn bei der Überschrift die Kurzform angegeben wird, so wird diese verwendet. Als Letztes folgt die Seitenzahl des entsprechenden Eintrags. Diese Nummer ist allerdings erst nach zwei Durchläufen des LaTeX-Compilers korrekt.

Nummerierungstiefe

Je nachdem, welche Dokumentenklasse verwendet wird, fällt die Nummerierungstiefe unterschiedlich aus:

Dokumentenklasse	Nummerierungsebene bis
book bzw. scrbook	\subsection (tocdepth 2)
report bzw. scrreprt	\subsection (tocdepth 2)
article bzw. scrartcl	\subsubsection (tocdepth 3)
proc	\subsubsection (tocdepth 3)

Tabelle 14.2: Gliederungsebene

Mit dem Zähler `tocdepth` wird angegeben, bis zu welcher Gliederungsebene die Einträge in die TOC-Datei aufgenommen werden sollen.

Der allgemeine Aufruf lautet:

```
\setcounter{tocdepth}{<nummer>}
```

Mit dem Zähler `tocdepth` wird festgelegt, bis zu welcher Gliederungsebene die Nummerierung durchgeführt werden soll. Bei den gängigen Dokumentenklassen `book` bzw. `scrbook`, `article` bzw. `scrartcl`, `report` bzw. `scrreport` werden die Gliederungsebenen von `\part` bis zu `\subsection` nummeriert. Der Wert `-1` steht dabei für `\part`, `0` steht für `\chapter`, `1` für `\section` und so weiter.

Bei den Dokumenttypen `article` bzw. `scrartcl` existiert die Ebene `\chapter` nicht. Der Gliederungsebene `\part` ist daher der Wert `0` zugeordnet.

Im folgenden Beispiel wird die Nummerierungstiefe auf den Wert `1` gesetzt, wodurch die Gliederungsebenen bis `\section` ins Inhaltsverzeichnis aufgenommen werden.

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

Beispiel 14.7: Inhaltsverzeichnis bis `\section`

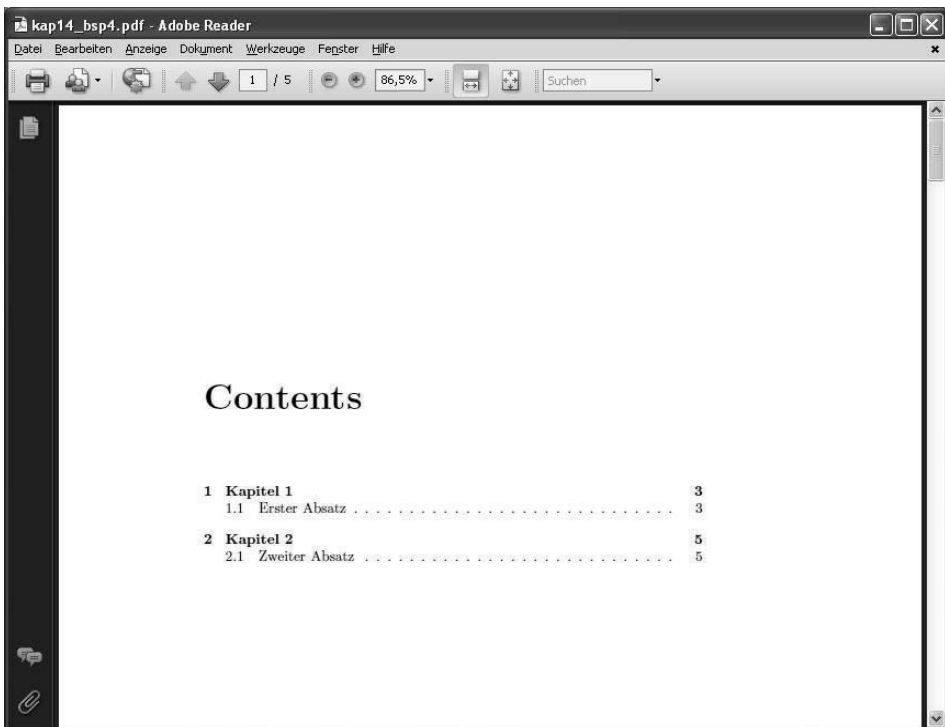


Abbildung 14.4: Ergebnis zu Beispiel 14.6

Weitere Einträge

Normalerweise erzeugt der Befehl `\tableofcontents` die Einträge in der TOC-Datei. Wenn Sie die Sternvariante `\tableofcontents*` verwenden, werden diese Einträge nicht erzeugt. Wollen Sie dennoch Einträge in die TOC-Datei aufnehmen, können Sie entweder den Befehl `\addcontentsline` oder `\addtocontentsline` benutzen:

```
\addcontentsline{toc}{<Gliederungsname>}{<Text>}
\addtocontentsline{toc}{<Eintragstext>}
```

Damit wird der Gliederungsname in die TOC-Datei übernommen. Als Alternative können Sie auch eine Nummer einfügen:

```
\addcontentsline{toc}{<Gliederungsname>}{\protect\numberline{<Gliederungsnummer>}{<Text>}}
```

Der Befehl `\protect` sorgt dafür, dass der nachfolgende Befehl geschützt in die TOC-Datei geschrieben wird.

Mit dem Befehl `\addtocontents` können Sie einen beliebigen Befehl in die TOC-Datei schreiben. So können Sie z. B. einen Seitenumbruch erzeugen und die aktuelle Seite des Inhaltsverzeichnisses etwas vergrößern.

```
\addtocontents{toc}{\protect\newpage}
\addtocontents{toc}{\protect\enlargethispage{0.3cm}}
```

Auch hier können Sie den Befehl `\protect` verwenden, um das Kommando in die TOC-Datei zu schreiben:

```
\tableofcontents
\addcontents{toc}{\protect\newpage}
\chapter{Erstes Kapitel}
\addcontentsline{toc}{section}{\protect\numberline{A}{Befehle}}
\addcontentsline{toc}{section}{\protect\numberline{B}{Umgebungen}}
\addcontentsline{toc}{section}{\protect\numberline{C}{Sonstiges}}
```

Beispiel 14.8: Inhaltsverzeichnis erweitern

Contents

1	Erstes Kapitel	3
A	Befehle	3
B	Umgebungen	3
C	Sonstiges	3

Abbildung 14.5: Ergebnis zum \addtocontents-Beispiel

Das Inhaltsverzeichnis selbst gestalten

LaTeX zieht standardmäßig jeden neuen Untergliederungspunkt unterschiedlich weit ein, um optisch anzuzeigen, dass es sich um eine neue Gliederungsebene handelt. Mit den Dokumentenklassenoptionen `tocleft` und `tocindent` in KOMA-Script können Sie hier selbst Hand anlegen und die Einzüge der Gliederungspunkte beeinflussen:

```
\documentclass[a4paper, tocleft]{book}
```

Contents

1	Kapitel 1	3
1.1	Erster Absatz	3
2	Kapitel 2	5
2.1	Zweiter Absatz	5

Abbildung 14.6: Ergebnis zu `tocleft`

Gliederungsnummern

Mit dem Befehl `\numberline` werden die Gliederungsnummern gesetzt. Die Gestaltung der Gliederungsnummer lässt sich beeinflussen, indem Sie das Kommando `\numberline` neu definieren:

```
\renewcommand{\numberline}[1]{\makebox[1em][l]{\textbf{#1}}}
```

In diesem Beispiel wird die Nummer in eine Box mit 1cm Breite gepackt. Die Gliederungsnummer selbst wird links positioniert und fett ausgezeichnet.

Seitennummern

Die Seitennummern können Sie genauso formatieren. Um etwas mehr Platz für die Seitennummern zu arrangieren, nutzen Sie diesen Befehl:

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

Gliederungstext

Den Gliederungstext gestalten Sie mit folgenden Befehlen:

```
\l@part{2}
\l@chapter{2}
\l@section{2}
\l@subsection{2}
\l@paragraph{2}
\l@subparagraph{2}
```

Der erste Parameter stellt den Gliederungstext und der zweite Parameter die Seitenzahl dar.

Wenn Sie die Einrückungstiefe beeinflussen möchten, können Sie dies im LaTeX-Kern entsprechend anpassen:

```
\renewcommand*\l@section{\@dottedtocline{1}{1.3em}{2.0em}}
\renewcommand*\l@subsection{\@dottedtocline{2}{3.0em}{3.1em}}
\renewcommand*\l@subsubsection{\@dottedtocline{3}{6.0em}{4.0em}}
```

Mit dem Befehl `\dottedtocline` wird zusätzlich eine kleine gepunktete Linie erzeugt. Als Parameter enthält der Befehl die Gliederungsebene, gefolgt von der Einrückungstiefe und der Breite der Gliederungsnummer (die Einrückungstiefe und die Breite der Gliederungsnummer ergeben die Breite der nächsten Ebene).

Anbei ein Beispiel. Das Inhaltsverzeichnis soll so formatiert werden, dass auf der Kapitelebene unterhalb des Textes eine Linie über die gesamte Textbreite verläuft. Der Text soll in diesem Bereich größer und in Fettdruck erscheinen. Alle anderen Ebenen sollen eingerückt und die Seitenzahlen ganz nach rechts gesetzt werden.

```

\documentclass[a4paper]{book}
\makeatletter

%Gliederungsnummer
\renewcommand{\numberline}[1]{%
  \makebox{0.9cm}[1][#1]\#hspace{1mm}}

%Kapitel (Chapter)
\renewcommand{\l@chapter}[2]{%
  \addvspace{2ex}%           vertikaler Abstand
  \pagebreak[3]%           Seitenumbruch erlauben
  \noindent%               nicht einrücken
  \makebox[0pt][1]{%       Box für die Linie
    \rule[ 3pt]{\textwidth}{0.5pt}} Linie über d. Textbreite
    {\large\textbf{#1}}\hfill#2% Text und Nummer
  \par%                   Zeilenumbruch
  \nopagebreak%           kein Seitenumbruch
\addvspace{1ex}%           vertikaler Abstand
}

%Abschnitt (Section)
\renewcommand{\l@section}[2]{%
  \addvspace{0.5ex}%       vertikaler Abstand
  \noindent\hspace{1cm}%   horizontales Einrücken
  #1\hfill#2%             Text und Nummer
  \par%                   Zeilenumbruch
\nopagebreak[2]%         Seitenumbruch vermeiden
}

%Unterabschnitt (Subsection)
\renewcommand{\l@subsection}[2]{%
  \addvspace{0.2ex}%       vertikaler Abstand
  \noindent\hspace{2cm}%   horizontales Einrücken
  #1\hfill#2%             Text und Nummer
  \par%                   Zeilenumbruch
}

\makeatother

\begin{document}

\tableofcontents

```

```

\chapter{Die Revolutionszeit}
\section{Das zweite Triumvirat}
\section{Der Bruch mit Antonius}
\chapter{Der Prinzipat des Augustus}
\section{Die Kampagnen des Drusus}
\section{Die Nachfolgefrage}

\end{document}

```

Beispiel 14.9: Selbst gestaltetes Inhaltsverzeichnis

Inhaltsverzeichnis		
1	Die Revolutionszeit	3
1.1	Das zweite Triumvirat	3
1.2	Der Bruch mit Antonius	3
2	Der Prinzipat des Augustus	5
2.1	Die Kampagnen des Drusus	5
2.2	Die Nachfolgefrage	5

Abbildung 14.7: Selbst gestaltetes Inhaltsverzeichnis

In diesem Beispiel werden die Ebenenüberschriften `\chapter` und `\section` sowie `\subsection` durch Neudefinitionen ersetzt. Zwischen den einzelnen Kapiteln soll eine Linie eingefügt werden. Damit sich die neuen Befehle definieren lassen bzw. auf das @-Zeichen zugegriffen werden kann, müssen die Neudefinitionen zwischen die Befehle `\makeatletter` und `\makeatother` gesetzt werden.

Dann wird die Gliederungsnummer wie im Beispiel oben neu definiert. Es wird eine Box mit 0,9 cm Größe sowie ein Zwischenraum von 1 mm erzeugt.

Dann wird der Befehl für das Kapitel, `\chapter`, neu definiert. Zunächst werden ein vertikaler Zeilenumbruch und anschließend eine Box mit der Breite 0 pt erzeugt, die eine horizontale Linie über die ganze Breite zieht. Die Linie selbst wird dabei um 3 pt unterhalb der Basislinie gezeichnet. Der Gliederungstext wird anschließend in der entsprechenden Formatierung gesetzt.

Tipp: Neudefinitionen von Befehlen wie oben können Sie in Style-Dateien (mit der Endung `.sty`) definieren, damit Sie diese wie Ergänzungspakete mit `\usepackage{mystyle}` einbinden können.

Kapitelinhaltsübersicht – die minitoc-Umgebung

Mit dem Ergänzungspaket `minitoc` können Sie ein Inhaltsverzeichnis pro Kapitel erstellen.

Im folgenden Beispiel wird für das erste Kapitel ein Inhaltsverzeichnis erstellt:

```
\documentclass[10pt,a4paper]{book}
\usepackage[latin1]{inputenc}
\usepackage[ngerman]{babel}
\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{makeidx}
\usepackage[german]{minitoc}

\begin{document}
\dominitoc
\tableofcontents
\chapter{Die Revolutionszeit}

\minitoc
\section{Das zweite Triumvirat}
Abschnitt 1
\section{Der Bruch mit Antonius}
Abschnitt 2
\end{document}
```

Beispiel 14.10: minitoc-Umgebung

Kapitel 1

Die Revolutionszeit

Inhaltsangabe

1.1	Das zweite Triumvirat	3
1.2	Der Bruch mit Antonius	3

1.1 Das zweite Triumvirat

Abschnitt 1

1.2 Der Bruch mit Antonius

Abschnitt 2

Abbildung 14.8: Mini-Inhaltsverzeichnis am Kapitelbeginn

In der zweiten Zeile wird die `minitoc`-Umgebung eingebunden. Mit dem Parameter `german` werden die deutschen Spracheigenschaften für die Umgebung festgelegt. Nach dem Beginn des Hauptdokumentes wird der Befehl `\dominitoc` aufgerufen, womit die Erstellung des Inhaltsverzeichnisses beginnt. Bei der Erstellung des Inhaltsverzeichnisses werden die Inhalte in MTC-Dateien geschrieben. Der Befehl `\minitoc` muss an die Stelle geschrieben werden, an der die Inhaltsangabe beginnen soll.

Tipp: Weitere Möglichkeiten des Pakets `minitoc` finden Sie in der Paketdokumentation unter <http://www.ctan.org/tex-archive/macros/latex/contrib/minitoc/minitoc.pdf>

Sonstige Ergänzungspakete

- `shorttoc`
Das Paket erlaubt es, ein weiteres Inhaltsverzeichnis mit anderen Gliederungsnummern zu erzeugen und an beliebiger Stelle einzubinden.

- `tocloft`
Mit diesem Paket lässt sich auf einfache und komfortable Weise das Inhaltsverzeichnis verändern.
- `titlesec` und `titletoc`
Mit diesen beiden Ergänzungspaketen können Sie das Aussehen einer Seite sowie von Überschriften und Inhaltsverzeichnissen ändern.

14.3.2 Tabellen- und Abbildungsverzeichnisse

Tabellen- und Abbildungsverzeichnisse werden nach demselben Prinzip wie Inhaltsverzeichnisse erstellt. Für jeden `\caption-`, `\figure-` oder `\table-`Befehl wird ein Eintrag in eine Datei geschrieben. Der Befehl `\listoffigures` erzeugt eine Datei mit der Endung `.lof` (list of figures) und generiert ein Abbildungsverzeichnis. Der Befehl `\listoftables` erzeugt ein Tabellenverzeichnis, das in eine Datei mit der Endung `.lot` (list of tables) geschrieben wird.

Für die Tabellen und Abbildungsverzeichnisse können Sie auch über die Befehle `\addcontents` bzw. `\addcontentline` Inhalte in die jeweiligen Verzeichnisse einfügen. Die Definition erfolgt analog zu den oben beschriebenen Beispielen. Anstelle der Dateinamenendung `.toc` geben Sie aber hier einfach `.lof` bzw. `.lot` an.

Durch eine Umdefinition der Befehle im LaTeX-Kern lässt sich die Ansicht anpassen. Diese Definition sieht so aus:

```
\newcommand*\l@figure{@dottedtocline{1}{1.5em}{2.3em}}
\newcommand*\l@table{@dottedtocline{1}{1.5em}{2.3em}}
```

14.3.3 Literaturverzeichnis

Einen wichtigen Teil eines jeden wissenschaftlichen Werkes bildet das Literaturverzeichnis. Für die Erstellung eines Literaturverzeichnisses bietet LaTeX die Umgebung `thebibliography`.

Die Umgebung wird allgemein wie folgt aufgerufen:

```
\begin{thebibliography}{<mustermarke>}
  \bibitem{<bezug>}<literaturangabe>
\end{thebibliography}
```

Die Umgebung kann an jeder beliebigen Stelle im Dokument aufgerufen werden. Sie findet ihren Platz jedoch üblicherweise am Ende eines Dokumentes. Meistens ruft man sie vor dem Index oder danach auf. Mit dem Parameter `<bezug>` können Sie eine Abkürzung für das jeweilige Werk angeben, das Sie zitieren. Der Parameter `<literaturangabe>` steht dann für das konkrete Werk, das Sie zitieren möchten.

Hier ein einfaches Beispiel für ein Literaturverzeichnis:

```
\begin{thebibliography}{2007}  
\bibitem{aschunk06}DirectX Grafikprogrammierung, BTV Verlag  
\bibitem{aschunk07}Alexander Schunk, Java 6 Grundlagenbuch, Databecker  
\end{thebibliography}
```

Beispiel 14.11: Literaturverzeichnis

Das Ergebnis sieht so aus:



Abbildung 14.9: Literaturverzeichnis

Literaturverzeichnis mit Abkürzungen

Mit einer weiteren Option vor der <bezug>-Option erstellen Sie Literaturverzeichnisse mit Abkürzungen:

```
\begin{thebibliography}{2007}  
\bibitem[aschunk06]{aschunk06}DirectX Grafikprogrammierung, BTV Verlag  
\bibitem[aschunk07]{aschunk07}Alexander Schunk, Java 6 Grundlagenbuch,  
Databecker  
\end{thebibliography}
```



Abbildung 14.10: Literaturverzeichnis mit Abkürzungen

14.3.4 Literaturverzeichnisse mit BibTeX

Ein weiteres nützliches Hilfsprogramm ist das Programm BibTeX, das ebenfalls für Literaturverzeichnisse zuständig ist. Um mit BibTeX ein Literaturverzeichnis zu erstellen, müssen folgende Dateien ausgelesen werden:

Datei	Beschreibung
.bib	Enthält die eigentliche Literaturdatenbank.
.bst	Enthält Layoutvorgaben für das Literaturverzeichnis.
.aux	Enthält die Einträge aus der Datenbank, die im Dokument benötigt werden.

Tabelle 14.3: Dateien von BibTeX

Es sind mehrere Übersetzungsläufe nötig, um mithilfe von BibTeX ein Literaturverzeichnis erstellen zu können. Der Ablauf ist wie folgt:

Beim ersten LaTeX-Durchlauf wird die AUX-Datei erzeugt, in der die benötigten Einträge aus der Literaturdatenbank gelistet werden.

Beim BibTeX-Durchlauf werden dann die benötigten Einträge aus der BIB-Datei gelesen. Dabei wird nach den Formatierungsangaben der BST-Datei eine thebibliography-Umgebung erzeugt.

Jetzt muss LaTeX noch einen zweiten und dritten Durchlauf absolvieren. Beim zweiten Lauf wird die thebibliography-Umgebung eingelesen. Beim dritten Lauf werden die Labels an die richtigen Verweisstellen gesetzt.

Folgende Auflistung fasst die Schritte zusammen:

1. Erster LaTeX-Lauf: AUX-Datei erzeugen.
2. BibTeX-Lauf: Die benötigten Einträge aus der BIB-Datei lesen und die Umgebung `thebibliography` erzeugen.
3. Zweiter LaTeX-Lauf: `thebibliography`-Umgebung einlesen.
4. Dritter LaTeX-Lauf: Label an die richtigen Stellen setzen.

Das Dateiformat BibTeX

Das BibTeX-Dateiformat besteht aus folgenden Einträgen:

<i>Eintrag</i>	<i>Beschreibung</i>
<code>@string</code>	Definiert Abkürzungen, die in BibTeX-Einträgen verwendet werden können.
<code>@preamble</code>	Legt fest, wie bestimmte Einträge formatiert werden.
<code>@comment</code>	Damit werden Kommentare in der BibTeX-Datei erzeugt.

Tabelle 14.4: Einträge einer BibTeX-Datei

Die eigentlichen Einträge sind dann Referenzen zum Beispiel zu Artikeln, Büchern usw. Die folgende Tabelle gibt eine Übersicht über die verschiedenen Einträge:

<i>Eintrag</i>	<i>Beschreibung</i>
<code>article</code>	Artikel
<code>booklet</code>	Gebundenes Druckwerk, z. B. ein Vorlesungsskript
<code>book</code>	Ein Buch
<code>conference</code>	Tagungsbericht
<code>inbook</code>	Teil eines Buches, z. B. ein Kapitel
<code>incollection</code>	Teil eines Buches, z. B. ein Beitrag in einem Sammelband
<code>inproceedings</code>	Teil eines Konferenzbandes
<code>phdthesis/masterthesis</code>	Doktorarbeit, Diplomarbeit
<code>misc</code>	Keines der Genannten

Tabelle 14.5: BibTeX-Einträge

Die Tabelle gibt eine Übersicht über die zwingenden Angaben:

<i>Eintrag</i>	<i>Beschreibung</i>
article	author, title, journal, year
book	author oder editor, title, publisher, year
inbook	author oder editor, title, publisher, year, chapter und/oder pages
incollection	author, title, booktitle, publisher, year
proceedings	Title, year
inproceedings	author, title, booktitle, year
phdthesis, masterthesis	Author, title, school, year
misc	Keines von allen

Tabelle 14.6: Zwingende Angaben

Die folgende Tabelle gibt eine Übersicht über die optionalen Argumente:

<i>Eintrag</i>	<i>Optionale Angaben</i>
article	volume, number, pages, month, note
book	volume oder number, series, address, edition, month, note
inbook	volume oder number, series, address, edition, month, note
incollection	editor, volume oder number, series, type, chapter, address, edition, month, note
proceedings	editor, volume oder number, series, address, publisher, month, note, organization
nproceedings	editor, volume oder number, series, type, chapter, address, edition, month, organization, publisher, note
phdthesis, masterthesis	type, address, month, note
Misc	author, title, howpublished, month, year, note

Tabelle 14.7: Optionale Angaben

BibTeX-Tools

Neben dem eigentlichen BibTeX-Programm gibt es noch eine Reihe von Tools, die zusammen mit BibTeX verwendet werden können:

<i>Tool</i>	<i>Beschreibung</i>
Bib2x	BibTeX-Konverter. Konvertiert BibTeX-Formate.
BibTool	Tool zur Manipulation von .bib-Dateien.
BibDesk	Grafisches Tool zur Verarbeitung von BibTeX-Dateien.
BibShare	Framework für die Verwendung von BibTeX mit Textverarbeitungsprogrammen.
Biblatex	Reimplementierung der bibliografischen Funktionen.
BibTeXML	XML-Erweiterung für BibTeX.

Tabelle 14.8: BibTeX-Tools

14.3.5 Index

Ein Index ist ein wichtiger Teil eines Werkes. In einem Index findet man wichtige Stichwörter zusammen mit einem Verweis auf die entsprechende Seitenzahl.

In LaTeX können Sie Indexeinträge mit dem Befehl `\index` erzeugen:

```
\index{<Seitenzahl>}
\index{<Seitenzahl>!<Unterstichwort>}
\index{<Seitenzahl>!<Unterstichwort>!<Unterstichwort>}
```

Ein Stichwort kann in bis zu drei Ebenen aufgeteilt werden. Die einzelnen Ebenen werden durch ein Ausrufezeichen (!) getrennt.

Eine Indexdatei in LaTeX hat die Endung `.idx`. Darin stehen unsortierte Einträge des LaTeX-Dokumentes. Das Stichwortverzeichnis selbst wird durch den Befehl `\makeindex` erstellt:

```
\makeindex
```

Der Befehl `\makeindex` muss in der Präambel vor dem Hauptdokument erscheinen. Dadurch startet LaTeX beim Übersetzungslauf die Indexdatei `IDX`. Die Einträge in die Indexdatei haben diesen Aufbau:

```
\indexentry{<Stichwort>}{<Seite>}
```

Der Parameter `<Stichwort>` enthält den Indexeintrag, der im Index erscheinen soll. Der zweite Parameter `<Seite>` enthält die Seitenzahl, die dem Indexeintrag zugeordnet ist.

Der Indexprozessor `makeindex`

Das Programm `makeindex` ist ein Tool, das für verschiedene Dateiformate verwendet werden kann. Als Ausgangsmaterial dient dem Programm eine unsortierte Indexdatei. Das Programm sortiert die angegebenen Indexeinträge. Das Format der Indexdatei ist dabei unwichtig.

Das Format der Eingabedatei in LaTeX ist eine IDX-Datei. Das Format kann in einer Style-Datei geändert werden. Die Style-Datei legt außerdem das Aussehen der Indexdatei fest.

Um mit dem Programm `makeindex` eine sortierte Indexdatei zu erstellen, sind folgende Schritte notwendig:

1. Zuerst müssen im LaTeX-Dokument die einzelnen Stichwörter mit dem `\index`-Befehl ausgewählt und markiert werden.
2. Mit dem Befehl `\documentstyle` wird die Dokumentstiloption `makedx` ergänzt.
3. In den Vorspann des Dokumentes wird der Befehl `\makeindex` eingefügt.
4. An der Stelle, an der der Index erscheinen soll, meistens am Ende des Buches, wird der Befehl `\printindex` eingefügt.
5. Dann lassen Sie das Dokument in einem ersten Durchlauf von LaTeX bearbeiten. Dabei erzeugt LaTeX eine unsortierte Indexdatei.
6. In dieser Datei wird dann jedes Mal, wenn LaTeX auf einen `\index`-Befehl trifft, ein `\indexentry`-Eintrag erzeugt.
7. Diese Indexdatei mit der Endung `.idx` enthält dann die zuvor ausgewählten Stichwörter und dient als Ausgangsmaterial für das Programm `makeindex`.
8. Das Programm sortiert dann die einzelnen Einträge und erstellt eine neue Indexdatei mit der Endung `.ind`. Das Programm `makeindex` wird folgendermaßen über die Kommandozeile aufgerufen:

```
makeindex beispieldatei.idx
```

9. Falls Sie die grafische Umgebung Texmaker verwenden, finden Sie das Programm `makeindex` über den Menüpfad `Werkzeuge > Makeindex`. Alternativ können Sie das Programm mit der Funktionstaste F12 starten.
10. In dieser neu erstellten Datei werden dann die einzelnen Stichwörter in einer Umgebung namens `theindex` mit den Befehlen `\item`, `\subitem` und `\subsubitem` aufgeführt.
11. Abschließend müssen Sie noch einmal einen LaTeX-Durchlauf starten. Dann wird der Index an der Stelle eingefügt, an der der Befehl `\printindex` steht.

Sortieren von Indexeinträgen

Die Indexeinträge werden von dem Programm `makeindex` sortiert. Dabei können folgende Sortiermöglichkeiten verwendet werden:

- Lexikalisch
- Nach Symbolen
- Nach Zahlen (numerisch aufsteigend sortiert oder umgekehrt)
- Nach Buchstaben (alphabetisch sortiert, Großbuchstaben vor Kleinbuchstaben und umgekehrt)
- Lexikografisch nach DIN 5007.

Makeindex und die Behandlung von Umlauten

`Makeindex` ist ein außerordentlich nützliches Programm. Es hat jedoch vordergründig einen kleinen Nachteil: Bei Indexeinträgen, die Umlaute enthalten, werden die Sonderzeichen im Stichwortverzeichnis entstellt wiedergegeben. Dieses Problem lässt sich jedoch beheben.

Wenn es sich nur um ein kurzes Dokument handelt, und es nur um wenige Einträge geht, hilft es bereits, die Umlaute in den Indexeinträgen (und nur dort, also nicht im Haupttext) durch doppelte obere Anführungszeichen (") zu markieren (z. B. `\index{R""omer}`, `\index{""Agypten}`, `\index{""Ubertakten}`). Ein ""a wird in diesem Fall als ä gelesen, ein ""o als ö und ein ""u als ü. Das ß-Zeichen wird durch ""s wiedergegeben.

Dies ist jedoch bestenfalls eine Quick & Dirty-Teillösung, weil `makeindex` in der Grundkonfiguration Stichwörter, die Umlaute enthalten, separat sortiert. Das fällt besonders bei Einträgen auf, die mit Umlauten beginnen; diese werden im Stichwortverzeichnis ganz nach vorne gestellt. Zudem ist das Eingeben der Umlaute mit doppelten Anführungszeichen nicht sonderlich komfortabel.

Es gibt jedoch eine andere Methode, die lexikografisch korrekt sortiert. Die Notierung der Umlaute ist darüber hinaus etwas einfacher. Die Sonderzeichen in den Indexeinträgen müssen hierfür wie folgt angegeben werden: "a für ä, "A für Ä, "o für ö, "O für Ö und "u für ü, "U für Ü sowie "s für ß.

`Makeindex` beherrscht zwar wie oben erwähnt die Indexsortierung nach DIN-Regeln, muss jedoch dazu durch die Eingabe des Aufrufparameters `g` (für german ordering) überredet werden. Wenn Sie dies durch die Eingabe von `makeindex g` versuchen, erhalten Sie allerdings eine Fehlermeldung. Vereinfacht ausgedrückt liegt das an einer Inkompatibilität von `makeindex` mit der Art, wie TeX und LaTeX die deutschen Sonderzeichen intern behandeln.

Die Lösung liegt darin, `makeindex` mithilfe einer sogenannten Stildatei mit der Endung `.ist` (Index Style) dazu zu bringen, diese Inkompatibilität beizulegen. Anbei ein Beispiel für eine solche Stildatei:

```
quote '\''
preamble "\\begin{theindex}\\n\\n\\small\\n"
postamble "\\n\\end{theindex}\\n"

delim_0 "\\dotfill"
delim_1 "\\dotfill"
delim_2 "\\dotfill"

headings_flag 1
symhead_positive "\\Symbole"
numhead_positive "\\Nummern"
heading_prefix "\\heading{"
heading_suffix "\\n"
```

Abbildung 14.11: Steuerdatei für die korrekte Indexierung

Welchen Namen Sie dieser Datei geben, ist sekundär; wichtig ist die Endung `.ist`. Die Inkompatibilität wird dadurch beseitigt, dass in der ersten Zeile die Behandlung der Sonderzeichen mit dem Befehl `quote '\'` umdefiniert wird. Darüber hinaus enthält die Datei noch einige Formatierungsanweisungen für den Index.

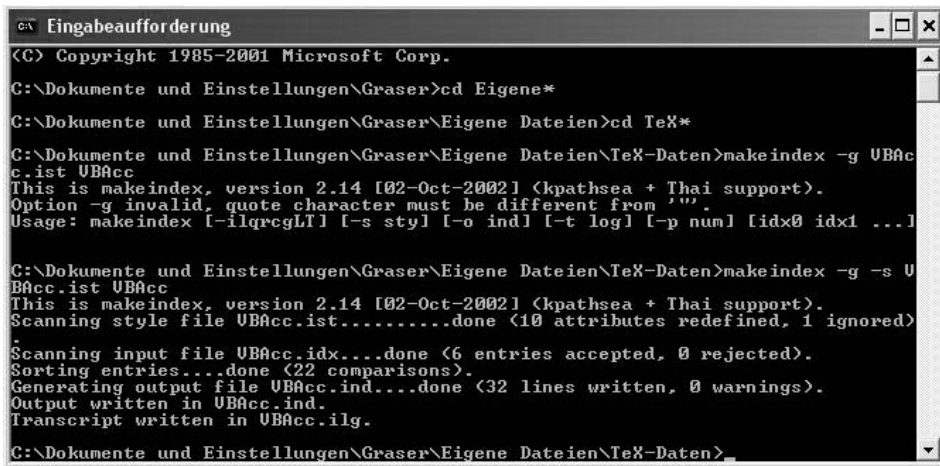
Sie können diese Datei mit einem normalen Texteditor erstellen und als simple Textdatei abspeichern. Achten Sie allerdings auf die Endung `.ist`. Platzieren Sie diese Datei im selben Verzeichnis, in dem das TeX-File liegt, das Sie indexieren wollen.

Um `makeindex` mitzuteilen, dass es beim Indexieren die deutschen Sortierregeln beachten und gleichzeitig die Anweisungen der Stildatei befolgen soll, geben Sie an der Kommandozeile Folgendes ein:

```
makeindex g s <Name Stildatei.ist> <Name des TeX Dokuments>
```

Der Parameter `s` teilt `makeindex` mit, dass die Anweisungen der Stildatei beim Indexieren berücksichtigt werden müssen. Die Endung `.ist` der Stildatei muss beim Befehlsaufruf von `makeindex` angegeben werden, die Endung des Dateinamens des TeX-Dokuments können Sie jedoch weglassen.

Wenn alles richtig läuft, erhalten Sie diese Mitteilung:



```

(C) Copyright 1985-2001 Microsoft Corp.
C:\Dokumente und Einstellungen\Graser>cd Eigene*
C:\Dokumente und Einstellungen\Graser\Eigene Dateien>cd TeX*
C:\Dokumente und Einstellungen\Graser\Eigene Dateien\TeX-Daten>makeindex -g UBAcc.ist UBAcc
This is makeindex, version 2.14 [02-Oct-2002] (kpathsea + Thai support).
Option -g invalid, quote character must be different from '""'.
Usage: makeindex [-ilqrcgLI] [-s sty] [-o ind] [-t log] [-p num] [idx0 idx1 ...]

C:\Dokumente und Einstellungen\Graser\Eigene Dateien\TeX-Daten>makeindex -g -s UB
BAcc.ist UBAcc
This is makeindex, version 2.14 [02-Oct-2002] (kpathsea + Thai support).
Scanning style file UBAcc.ist.....done (10 attributes redefined, 1 ignored)
Scanning input file UBAcc.idx....done (6 entries accepted, 0 rejected).
Sorting entries....done (22 comparisons).
Generating output file UBAcc.ind....done (32 lines written, 0 warnings).
Output written in UBAcc.ind.
Transcript written in UBAcc.ilg.
C:\Dokumente und Einstellungen\Graser\Eigene Dateien\TeX-Daten>

```

Abbildung 14.12: Mission geglückt – der Indexlauf ist gelungen.

Jetzt müssen Sie nur noch einmal den LaTeX-Compiler starten, um die Ergebnisse der Indizierung und Sortierung in den Text einbinden zu können. Das Ergebnis sehen Sie hier:

Index	
A	
Ägypten	1
C	
Compiler	1
I	
Instanz	1
R	
Rückrufe	1
S	
Schnösel	1
U	
Übertaktung	1

Abbildung 14.13: Der Index ist fertiggestellt – komplett mit Umlauten.

Der Clou dieser Methode liegt darin, dass Sie die notwendige Steuerdatei nur einmal zu erstellen brauchen und für alle Indizierungsvorgänge wieder verwenden können. Sie müssen lediglich darauf achten, dass Sie die IST-Datei im selben Verzeichnis platzieren, in dem auch Ihr TeX-Dokument liegt, und dass Sie die Datei bei Aufruf von `makeindex` richtig angeben.

Sie können dieses Verfahren auch anwenden, wenn Sie eine grafische Oberfläche wie Texmaker verwenden. Dazu müssen Sie lediglich unter `Optionen > Texmaker` konfigurieren den Eintrag in der Zeile »Makeindex« etwas abändern:



Abbildung 14.14: Makeindex-Konfiguration in Texmaker

In diesem Fall ist die Stildatei mit `VBAcc.ist` angegeben. Beim Aufruf von `makeindex` über F12 wird die Stildatei berücksichtigt und `makeindex` befolgt die enthaltenen Anweisungen.

Tipp: Weitere Informationen zum Programm `makeindex` finden Sie unter <http://www.lrz.muenchen.de/services/software/textverarbeitung/makeindex/>

Spezielle Formatierungen für Indexeinträge

Indexeinträge lassen sich mit speziellen Befehlen formatieren.

- Mit dem Befehl `see` können Sie nicht auf eine Seite, sondern auf einen anderen Eintrag verweisen – auf Deutsch: »siehe«. Dem `see`-Befehl geht der vertikale Balken (|) voraus. Der Befehl wird direkt im Indexeintrag nach dem | angegeben:
`\index{Augustus|see{Caesar}}.`
- (und)
 Durch die Angabe der runden Klammern (und) können Sie den Anfangs- (und den Endbereich) von zusammenhängenden Indexeinträgen festlegen. Normalerweise werden Indexeinträge zusammengefasst, wenn sie auf mehreren aufeinanderfolgenden Seiten auftauchen, z. B. 8-10 oder 11-13. Diese Zusammenfassung wird aber unterbrochen, wenn sich Seiten dazwischen befinden. Mit der Angabe von (...) lässt sich dieses Verhalten ändern. Die öffnende Klammer nach dem Vertikalbalken markiert den Anfangsbereich eines zusammenhängenden Indexeintrags, die schließende Klammer den Endbereich: `\index{Augustus|(...)index{Augustus|}}.`

Symbole sortieren

Außer Schlagworten können auch Symbole in einen Index aufgenommen werden. Das ist vor allem dann nützlich, wenn es sich um mathematische Symbole handelt. Symbole

werden dabei im Index in die Gruppe der Symbole einsortiert. Um ein Symbol aufzunehmen, wird das Symbol @ im `\index`-Befehl verwendet:

```
\index{<Sortiereintrag>@<Druckeintrag>}
```

Im Hauptdokument sieht das dann so aus:

```
\index{Summe@$sum$}  
\index{Integral@$int$}
```

Steuerbefehle selbst erzeugen

Mit dem Befehl `\newcommand` lassen sich neue Steuerbefehle erzeugen. So können Sie den Befehl `\see` z. B. wie folgt ändern:

```
\newcommand*\see[2]{\emph{\seename}#1}
```

Der Befehl `\see` erhält in diesem Fall zwei Parameter. Der erste Parameter ist das Stichwort, der zweite die Seitenzahl. Mit dem Befehl `\seename` wird das Wort »siehe« kursiv in der entsprechenden Sprache dargestellt. Wenn Sie nur einen Parameter angeben, wird lediglich die Seitenzahl verwendet.

Mit den Befehlen `\underline`, `\textbf` und `\textit` können Sie Indexeinträge nach dem oben gezeigten Muster unterstreichen, fett formatieren oder kursiv darstellen.

Weitere Ergänzungspakete

Außer `makeindex` gibt es noch einige Ergänzungspakete, mit denen Sie verschiedene Verzeichnisse erstellen können, z. B. Stichwortverzeichnisse oder Personenverzeichnisse.

- Mit dem Paket `showidx` können Sie Indexeinträge am Seitenrand anzeigen lassen. Das Paket muss nur per `\usepackage` in das Dokument eingebunden werden. Mehr ist nicht nötig.
- Mit dem Ergänzungspaket `splitidx` und dem Programm `splitindex` lassen sich mehrere Indexverzeichnisse erstellen. Für die verschiedenen Indexverzeichnisse existieren verschiedene Ergänzungspakete, die alle Dateien erzeugen und offenhalten. Während eines LaTeX-Laufs können allerdings maximal 16 Dateien offengehalten werden. Mit dem Paket `splitindex` können diese Dateien wiederum in eine einzelne Datei umgewandelt werden.

Das Paket `\splitidx` wird über den Befehl `\usepackage{splitidx}` eingebunden und hat verschiedene Parameter, die in eckigen Klammern vorangestellt werden:

Parameter	Beschreibung
<code>makeindex</code>	Damit wird das Programm <code>makeindex</code> aktivieren.
<code>idxcommands</code>	Damit stehen verschiedene Befehle für die Erzeugung von Gruppen zur Verfügung.

Tabelle 14.9: Parameter des Pakets `splitindex`

Tipp: Weitere Informationen zu `splitidx` finden Sie unter [www.ctan.org/tex archive/macros/latex/contrib/splitindex/splitidx.pdf](http://www.ctan.org/tex-archive/macros/latex/contrib/splitindex/splitidx.pdf)

Indexgruppen erstellen

Mit dem Befehl `\newindex` wird eine Gruppe von Indexeinträgen erstellt. Der allgemeine Aufruf lautet:

```
\newindex{<Überschrift>}{<Gruppe>}
```

Mit dem Parameter `<Überschrift>` geben Sie die Überschrift an und mit dem Parameter `<Gruppe>` die Gruppe.

Weitere Tools

Es gibt noch weitere Tools, mit denen Sie Indexverzeichnisse erstellen können:

- `Splitindex`: Dieses Programm gehört zum Paket `splitidx` und erzeugt aus verschiedenen Dateien ein Stichwortverzeichnis. Das Programm `splitindex` gibt es in verschiedenen Versionen, z. B. in Java, C oder Perl für die unterschiedlichen Betriebssysteme.
- `Xindy`: Dies ist ein alternativer Stichwortprozessor, der zusätzliche Möglichkeiten bietet. Das Programm ist allerdings weitaus komplexer in der Bedienung als `makeindex`. `Xindy` bietet allerdings den Vorteil, dass es nicht nur LaTeX-Dateien, sondern auch z. B. XML-Dateien verarbeiten kann.

14.3.6 Glossar

Ein Glossar enthält ähnlich wie ein Stichwortverzeichnis verschiedene Einträge mit Schlagworten, zu denen es eine Erklärung gibt. Für die Erstellung eines Glossars gibt es das Paket `glossar`.

Ein Glossar in LaTeX hat die Dateiendung `.glo`. Die Einträge in diesem Glossar sind, ähnlich wie in einem Index, zunächst unsortiert. Für die Erstellung eines Glossars kann ebenfalls das Programm `makeindex` benutzt werden. Für die Erstellung eines Glossars gibt es folgende Befehle:

- `\makeglossary`
Mit diesem Kommando werden alle Einträge in eine Datei geschrieben. Dieser Befehl arbeitet ähnlich wie `\makeindex`.
- `\gentry`
Damit wird ein Eintrag in das Glossar erzeugt. Der allgemeine Aufruf lautet:

```
\gentry{<Stichwort>}{<Erklärungstext>}
```
- `\glspace`
Dieser Befehl wird aufgerufen, um die Seitenzahl auszugeben. Die Seitenzahl wird standardmäßig unterdrückt, da Seitenzahlen bei Glossareinträgen unüblich sind.
- `\glsgroup`
Damit wird ein Buchstabe für eine neue Gruppe erzeugt.
- `\glshead`
Damit wird die Überschrift bzw. der Kopfbereich für das Glossar festgesetzt.
- `\printglossary`
Damit wird das Glossar im Dokument platziert.

14.4 Kopf- und Fußzeilen

Kopf- und Fußzeilen wurden allgemein bereits in Kapitel 4 behandelt. Neben den generellen Befehlen für Kopf- und Fußzeilen gibt es noch weitere Befehle, mit denen Sie diese formatieren können:

```
\pagestyle{<Seitenstil>}
\thispagestyle{Seitenstil>}
```

Mit dem Befehl `\pagestyle` wird der Stil für alle folgenden Seiten definiert. Mit dem Befehl `\thispagestyle` wird der Stil nur für die aktuelle Seite festgelegt.

Es stehen folgende Stile zur Verfügung:

- `empty`
Seitenkopf und Fuß bleiben leer.
- `plain`
Die Kopfzeile bleibt leer. Die Fußzeile enthält die Seitennummerierung.

- `headings`
Im Seitenkopf erscheinen normalerweise die aktuelle Überschrift und die Seitenzahl, während die Fußzeile leer bleibt. Bei der Klasse `book` werden auf geraden Seiten die Kapitelüberschriften und die Seitennummern gesetzt. Auf ungeraden Seiten befinden sich die Abschnittsüberschriften.
- `myheadings`
Damit können Sie die Kopfzeile selbst festlegen. Es stehen folgende Befehle zur Verfügung:

```
\markboth{<linker Kopf>}{<rechter Kopf>}
\markright{<rechter Kopf>}
```

Mit dem Befehl `\markboth` können Sie die Kopfzeile für doppelseitige Texte setzen, mit `\markright` für einseitige Texte. Bei Dokumenten, deren Rückseiten nicht bedruckt werden, bietet sich `\markright` an.

Weitere Befehle mit KOMA-Script

Neben den oben genannten Befehlen stellt KOMA-Script noch weitere Kommandos zur Verfügung:

- `\pagemark`
Mit diesem Befehl wird die Seitennummer dargestellt. Die Seitennummer kann mit folgendem Kommando formatiert werden:
- ```
\setkomafont{pagenumber}{<werte>}
```
- `\headmark`  
Mit diesem Befehl können Sie auf den Kolumnentitel zugreifen.
  - `\clearscrheadings`  
Damit werden alle Einstellungen für `scrheadings` gelöscht.
  - `\clearscrplain`  
Damit werden alle Einstellungen für `scrplain` gelöscht.
  - `\clearscrheadfoot`  
Damit werden alle Einstellungen für Kopf- und Fußzeilen gelöscht.
  - `\manualmark`  
Damit wird die automatische Aktualisierung der Kolumnentitel abgeschaltet.
  - `\automark{<rechte Seite>}{<linke Seite>}`  
Damit wird die automatische Aktualisierung der Kolumnentitel eingeschaltet. Als Parameter können Sie hier `chapter`, `section`, `subsection` usw. eintragen.
  - `\headfont`  
Damit wird der Font für den Seitenkopf und -fuß festgelegt.

- `\setheadwidth[<Verschiebung>]{<Breite>}`  
Damit können Sie die Breite der Kopfzeile verändern. Die Breite lässt sich nach außen verschieben. Als Parameter für die Breite können Sie folgende Optionen angeben:  
  
`paper`  
Breite des Papiers  
  
`page`  
Breite der Seite  
  
`text`  
Breite des Textbereichs  
  
`textwidthmarginpar`  
Breite des Textbereichs inklusive Seitenrand  
  
`head`  
Aktuelle Breite des Seitenkopfes  
  
`foot`  
Aktuelle Breite des Seitenfußes
- `\setfootwidth[<Verschiebung>]{<Breite>}`  
Damit wird die Breite für den Seitenfuß festgelegt.
- `\setheadtopline[<Länge>]{<Dicke>}`  
Damit können Sie den Wert für die Linie über dem Seitenkopf ändern.
- `\setheadsetpline[<Länge>]{<Dicke>}`  
Damit wird der Wert für die Linie zwischen Kopf- und Textbereich geändert.
- `\setfootbotline[<Länge>]{<Dicke>}`  
Damit wird die Dicke der Linie unter dem Seitenfuß angepasst.
- `\setfootsepline[<Länge>]{<Dicke>}`  
Damit wird der Wert für die Linie zwischen Textbereich und Fuß festgelegt.

### Weitere Ergänzungspakete

Außerdem gibt es noch weitere Ergänzungspakete, mit denen Sie Kopf- und Fußzeilen formatieren können.

Das Paket `fancyhdr` bietet unter anderem die Möglichkeit, einen Daumenindex zu gestalten. Ein Daumenindex ist ein kleiner Balken, der ein Kapitel am Rand markiert. Telefonbücher, aber auch manche Lexika und andere Nachschlagewerke verfügen über einen solchen Daumenindex.

### 14.4.1 Fußnoten

Wie Fußnoten erstellt werden, haben Sie bereits in Kapitel 4 grundsätzlich kennengelernt. Mit dem Ergänzungspaket `footmisc` können Sie weitere Formatierungen für Fußnoten erstellen:

#### Das Paket `footmisc`

Das Paket `footmisc` bietet verschiedene Optionen für die Gestaltung von Fußnoten:

- `perpage`  
Diese Option erlaubt die Zählung der Fußnoten auf jeder Seite ab 1.
- `para`  
Mit dieser Option werden die Fußnoten am Ende einer Seite zu einem Absatz zusammengefasst. Wenn am Ende der Seite viele kurze Fußnoten stehen, ist das sehr hilfreich und spart Platz.
- `side`  
Mit dieser Option werden die Fußnoten an den Rand gesetzt. Der zur Verfügung stehende Platz ist dann jedoch vergleichsweise gering.
- `symbol`  
Diese Option verwendet Symbole für die Fußnotennummerierung. Es steht jedoch nur eine begrenzte Anzahl an Symbolen zur Verfügung.
- `symbol*`  
Ähnlich wie `symbol`, nur zur Verwendung mit der Option `perpage`.

**Tipp:** Weitere Informationen zu dem Paket `footmisc` finden Sie unter <ftp://ftp.tu-chemnitz.de/pub/tex/macros/latex/contrib/footmisc/footmisc.pdf>.

#### Weitere Ergänzungspakete

Wenn das Paket `footmisc` zusammen mit anderen Erweiterungen verwendet wird, kann es unerwünschte Nebeneffekte geben. Um diese zu vermeiden, gilt es einige Regeln zu beachten:

- Mit dem Paket `setspace` können Sie den Abstand bei den Fußnotentrennlinien verändern. Damit `footmisc` diese Änderungen beachtet, muss `setspace` vor `footmisc` geladen werden.
- Mit dem Paket `hyperref` können Sie Hyperlinks zwischen den Fußnotenmarkierungen setzen. Dadurch wird `footmisc` negativ beeinflusst. Daher muss bei gleichzeitiger Verwendung das Paket `hyperref` wie folgt eingebunden werden:

```
\usepackage[hyperfootnotes false,...]{hyperref}
```

- `manyfoot`  
Die Pakete `footmisc` und `manyfoot` vertragen sich nicht immer, da sich mit `manyfoot` verschiedene unabhängige Fußnotensequenzen erstellen lassen. Diese Probleme waren zum Zeitpunkt der Drucklegung dieses Buches noch nicht endgültig gelöst.

## 14.5 Juristische Texte

Juristische Texte bestehen aus Gesetzen, Paragraphen und Absätzen. Für die Erstellung und Formatierung von juristischen Texten gibt es verschiedene Pakete. Das Paket `juramisc` besteht aus verschiedenen Klassen und Paketen, deren Grundlage wiederum das Paket `jurabase` ist:

### 14.5.1 Das Paket `jurabase`

Das Paket `jurabase` enthält verschiedene Befehle, die für unterschiedliche Arten von juristischen Texten brauchbar sind:

- `\juraenum`  
Damit werden Aufzählungen erzeugt, die die in der Rechtswissenschaft üblichen Nummerierungen aufweisen.
- `\indentoff`  
Damit wird der voreingestellte Erstzeileneinzug ausgeschaltet. Diese Einstellung kann mit `\resetindent` wieder zurückgenommen werden.
- `\antrag`  
Ermöglicht spezielle Formatierungen für juristische Anträge.

### 14.5.2 Die Klasse `jurabook`

Die Klasse `jurabook` definiert einige Elemente der Klasse `book` um:

- Es werden immer die tatsächlichen Überschriften in das Inhaltsverzeichnis eingetragen, auch wenn Sie eine optionale Kurzform angegeben haben.
- Jedes Kapitel wird mit einem §-Zeichen eingeleitet. Sie können dieses Verhalten jedoch bei Bedarf abändern.
- Das Fußnotenlayout entspricht dem von juristischen Fachbüchern.
- Sie können Kopfzeilen anpassen.

Außerdem stehen einige weitere Befehle zur Verfügung, die das Verfassen von juristischem Text vereinfachen.

Wenn Sie mit der Klasse `jurabook` arbeiten wollen, müssen Sie die Pakete `fancy` und `remreset` einbinden, wobei Sie mindestens die Version 2.0 der Datei `fancyhdr.sty` installieren müssen. Die Klasse `jurabook` lädt automatisch das Paket `jurabase`, das wiederum die Pakete `ifthen`, `calc` und `xspace` voraussetzt.

Die Klasse `jurabook` kümmert sich nicht um die Zeilenbreite und Randeinstellungen. Diese Einstellungen können Sie mit dem Paket `geometry` festlegen.

### Weitere Formatierungen

Neben diesen Formatierungen können Sie juristischen Text ähnlich wie ein normales Dokument gliedern und formatieren. So lässt sich z. B. juristischer Text wie ein Buch in Vorspann, Hauptspann und Nachspann gliedern.

Kapitel und Abschnitte werden ebenfalls mit den Befehlen `\section` und `\chapter` definiert. Die Abschnitte werden mit einem Paragraphenzeichen (§) eingeleitet.

Ein Anhang wird mit dem Befehl `\appendix` eingefügt. Alle nachfolgenden Kapitel werden dann mit Großbuchstaben gezählt.

### 14.5.3 Die Klasse `jurautl`

Die Klasse `jurautl` ist für das Verfassen von zivilrechtlichen Urteilen vorgesehen. Für die Erstellung eines entsprechenden Urteils (Rubrum, Tenor, Tatbestand, Entscheidungsgründung) gibt es entsprechende Umgebungen:

```
\begin{document}
\az{34 C 159/00}
\lmv{27.05.09}

\begin{rubrum}
\end{rubrum}
\begin{tenor}
\end{tenor}
\begin{tatbestand}
\end{tatbestand}
\begin{gruende}
\end{gruende}
\end{document}
```

Innerhalb des Deckblatts eines Urteils (des Rubrums) werden die notwendigen Daten über entsprechende Befehle eingefügt:

- `\gericht{<gericht>}`  
Urteilendes Gericht

- `\richter{<namen>}`  
Urteilender Richter
- `\az{<aktenzeichen>}`  
Aktenzeichen
- `\lmv{<datum>}`  
Datum der letzten mündlichen Verhandlung
- `\urteilsart`  
Art des gesprochenen Urteils

Die `tenor`-Umgebung beinhaltet den Entscheidungstenor. Der Text wird dabei links eingerückt und mit Abständen gesetzt.

Die Umgebung `tatbestand` erzeugt automatisch eine zentrierte, unterstrichene Überschrift »Tatbestand«. Innerhalb der Umgebung können die einzelnen Anträge der Streitparteien über den Befehl `\antrag{<einleitung>}{<antragsinhalt>}` mit den richtigen Einrückungen gesetzt werden.

Die Umgebung `gruende` erzeugt ebenfalls eine zentrierte und unterstrichene Überschrift »Entscheidungsgründe«.

Im Folgenden sind weitere Befehle aufgelistet:

- `\lebenslauf`  
Lebenslauf des Angeklagten
- `\sachverhalt`  
Angaben zum Sachverhalt
- `\beweismuenderung`  
Rechtliche Beweiswürdigung
- `\strafzumessung`  
Strafzumessung
- `\kosten`  
Kostenentscheidungen

#### 14.5.4 Die Klasse `juraovw`

Die Klasse `juraovw` basiert auf der KOMA-Script-Klasse `scrartcl`. Mit dem Befehl `\settheme{<thema>}{<datum>}` können Sie die Überschrift eines Artikels fett und zentriert setzen. Das Erstellungsdatum wird in einer Fußnote gesetzt. Außerdem stehen weitere Möglichkeiten von `scrartcl` zur Verfügung.

## 14.6 Arbeiten mit make-Dateien

In der Unix- und Linux-Welt hat sich das Tool `make` durchgesetzt. Make ist ein Tool, das mehrere Dateien in Abhängigkeit ihres letzten Änderungsdatums kompilieren kann.

Wenn Sie größere Bücher schreiben und den Text in verschiedene Dateien auslagern, können Sie sich mit dem Tool `make` die Arbeit erleichtern.

### Aufbau einer make-Datei

Der Aufbau einer `make`-Datei ist relativ einfach. Eine solche Datei besteht aus Regeln, Parametern und den Dateien, die kompiliert werden sollen. Unten sehen Sie einen einfachen Aufbau einer `make`-Datei:

```
Ziel [weitere Ziele] : [Abhängig von]

[<tab> Befehle]
[<tab> Befehle]
```

Über die Angabe des Ziels wird dabei die Einheit festgelegt, z. B. die zu erstellende PDF-Datei. Ferner können Sie angeben, von welchen anderen Zielen, d. h. Dateien, diese Einheit abhängig ist. Diese werden dann vorher von `make` überprüft und eventuell ausgeführt. Danach folgen die Befehle, die für dieses Ziel notwendig sind.

Mehrere Ziele trennen Sie durch Semikola.

Hier ein Beispiel für eine einfache `make`-Datei:

```
MAIN laanw
MAINDIR /var/Franzis Latex
TEXFILES $(wildcard*.tex)

TEXINPUTS: $(MAINDIR)/:$(MAINDIR/Bilder:.$(TEXINPUTS))

all: buch
 Acroread $(MAIN).pdf
buch: $(MAIN).pdf

$(MAIN).pdf: $(MAIN).tex franzis.sty $(TEXFILES)
 Pdflatex $ <
```

In der ersten Zeile wird die Variable `MAIN` definiert, in der der Name der Haupt-LaTeX-Datei und somit der Name für die PDF-Datei stehen.

In Zeile 2 wird dann die Variable `MAINDIR` definiert, die auf das Verzeichnis mit allen LaTeX-Dateien verweist.

Die Variable `TEXFILES` verweist auf alle Dateien mit der Dateiendung `.tex`. Dazu wird die Funktion `wildcard*` verwendet, die die entsprechenden Dateien im aktuellen Verzeichnis ermittelt.

In der vierten Zeile wird dann die Variable `TEXINPUTS` definiert, die festlegt, aus welchen Verzeichnissen LaTeX Dateien aussuchen soll, wenn diese z. B. mit dem Befehl `\include` eingebunden werden.

In der fünften Zeile wird die erste Regel definiert. Diese hat den Namen `all` und wird standardmäßig von `make` aufgerufen, wenn kein Parameter angegeben wird. Diese Regel ist von der Regel `buch` abhängig und führt, wenn diese Regel erfüllt ist, das Programm `acoread` aus. Dieses Programm öffnet dann die entsprechende PDF-Datei. Die PDF-Datei lautet dann `laawn.pdf`.

In der letzten Zeile wird `pdflatex` aufgerufen, wobei als Parameter das Symbol `$ <` angegeben wird. Dieses stellt den Parameter der ersten abhängigen Datei dar.

Das Programm `make` selbst wird so aufgerufen:

```
Make oder make f Makefile.laawnw
```

Das Programm `make` erhält als Datei die Datei `laawn`, die dann von `make` verarbeitet wird.

## 14.7 Arbeiten im Team

Wenn Sie größere Projekte wie Bücher oder Diplomarbeiten mit mehreren Autoren erstellen, müssen Sie darauf achten, dass die Änderungen an dem Dokument auch richtig erfasst werden.

Auch für LaTeX-Dateien gibt es Programme wie Subversion oder CVS, um die Dateien aktuell zu halten. Diese Programme haben folgende Vorteile:

- Es können mehrere Personen gleichzeitig an einem Buch schreiben. Wenn ein und dieselbe Datei von mehreren Personen bearbeitet wird, meldet das System einen Konflikt, wenn es diesen nicht automatisch beseitigen kann.
- Wenn Sie entsprechende Tags in einem Dokument verwenden, werden sie beim Einchecken automatisch mit dem aktuellen Datum versehen. So können Sie z. B. die Versionsnummer, das Datum der letzten Änderung und den Autor anzeigen lassen.
- Alle Änderungen lassen sich auch rückgängig machen, da das Versionssystem Buch über alle Änderungen und Modifikationen führt.
- Wenn das Versionssystem auf einem anderen Rechner installiert ist, haben Sie zusätzlich eine Art Back-up-System, da dort alle Dateien nochmals gespeichert sind.



# 15 Grundlagen von Zeichensätzen

Da die Qualität des gesetzten Textes häufig von der jeweiligen Schrift abhängt, ist es nicht unwesentlich, etwas über die Grundlagen von Zeichensätzen zu wissen.

## 15.1 Grundaufbau eines Zeichensatzes

In Kapitel 2 haben Sie bereits den grundlegenden Aufbau eines Zeichensatzes kennengelernt. Ein Zeichensatz besteht im Wesentlichen aus folgenden Bestandteilen:

- Familienattribut  
Über das Attribut `\fontfamily` wird die Schriftfamilie festgelegt.
- Serienattribut  
Über das Serienattribut `\fontseries` wird die Stärke (weight), Weite und Breite einer Schrift festgelegt.
- Formattribut  
Der `\fontshape`-Befehl legt die Schriftform fest.
- Größenattribut  
Der Befehl `\fontsize` definiert das Größenattribut einer Schrift.

## 15.2 Schriftinitialisierung

Für die Initialisierung einer Schrift stellt LaTeX eine Reihe von Befehlen zur Verfügung. Es folgt hier eine kurze Übersicht. Die meisten Befehle können Sie über `\renewcommand` neu definieren:

- `\encodingdefault`  
Damit wird die Standardcodierung festgelegt. Über die Einbindung des Pakets `\usepackage[T1]{fontenc}` wird T1 als Schrifttyp festgelegt.
- `\seriesdefault`  
Damit wird die Standardserie festgelegt. Diese können Sie über `\renewcommand{\seriesdefault}{m}` neu definieren.

- `\shapedefault`  
Damit wird das Formattribut festgelegt. Dieses lässt sich über `\renewcommand{\shapedefault}{n}` neu definieren.
- `\rmdefault`  
Damit wird die Schrift für `\rmfamily` und `\textrm` festgelegt. Diese wird mit `\renewcommand{\rmfamily}{cmr}` neu definiert.
- `\sfdefault`  
Damit wird die Schrift für `\sffamily` und `\textsf` festgelegt. Diese Festlegung können Sie über `\renewcommand{\sfdefault}{cmss}` ändern.
- `\ttdefault`  
Damit wird die Schrift für `\ttfamily` und `\texttt` festgelegt. Eine Änderung erfolgt analog zu `\sfdefault` mit `\renewcommand{\ttdefault}{cmtt}`.
- `\bfdefault`  
Damit wird die Schrift für `\bfseries` und `\textbf` festgelegt.
- `\mddefault`  
Damit wird die Schrift für `\mdseries` und `\textmd` festgelegt.
- `\itdefault`  
Damit wird die Schrift für `\itshape` und `\textit` festgelegt.
- `\sldefault`  
Damit wird die Schrift für `\slshape` und `\textsl` festgelegt.
- `\scdefault`  
Damit wird die Schrift für `\scshape` und `\textsc` festgelegt.
- `\updefault`  
Damit wird die Schrift für `\upshape` und `\textup` festgelegt.

## 15.3 Fonttypen

Das Design einer Schrift kann auf einem Computer auf verschiedene Weise definiert werden.

### 15.3.1 Pixelfonts vs. Type1-Fonts

LaTeX verwendet standardmäßig die Fonttypen *Computer modern*. Diese werden intern als Pixelraster dargestellt. Die Schriftart selbst wird durch das Programm Metafont erzeugt und als Kurve beschrieben. Bei der Umrechnung wird immer die Auflösung des Ausgabemediums berücksichtigt. Bei der Umsetzung in die DVI-Datei wird geprüft, ob der jeweilige Font schon vorhanden ist.

Im Gegensatz dazu werden Type1-Schriften nicht als Pixelraster, sondern in Vektorform gespeichert. Dadurch kann der Buchstabe beliebig vergrößert werden, ohne dass dies zu einem treppenartigen Erscheinungsbild führt. Die Type1-Fonts sind sehr gut geeignet für die Darstellung auf dem Drucker und dem Bildschirm.

### 15.3.2 Fonttyp festlegen

Für die Darstellung eines Fonts benötigt LaTeX lediglich die Metriken, d. h. die Tabellen, die z. B. bestimmen, wie breit und hoch ein Zeichen ist. Welcher Fonttyp dabei verwendet wird, ist grundsätzlich egal. Erst der Treiber, der die Ausgabe erzeugen soll, kümmert sich um den eigentlichen Fonttyp.

Wenn Sie z. B. `.dvi` als Ausgabeformat benutzen, kümmert sich der Treiber *dvips* um den Font. Über den Parameter `P` für `<printer>` wird der Drucker beziehungsweise das Ausgabeformat festgelegt und damit die Steuerdatei, die bestimmt, welcher Fonttyp in welcher Auflösung benutzt werden soll. Wenn Sie den Parameter nicht angeben, wird als Steuerdatei die Datei `config.ps` verwendet. Andernfalls wird anstelle von `ps` der Druckername verwendet, z. B. `config.www`, wenn Sie `P www` angegeben haben.

Die Konfigurationsdateien sind von TeX-Version zu TeX-Version unterschiedlich. In dieser Datei werden die speziellen Eigenschaften für *dvips* festgelegt. Insbesondere, welcher Font verwendet wird.

So eignet sich z. B. die Datei `config.www` für Dokumente im World Wide Web:

```
% Thomas Esser, 1998, 2002, public domain.
% Usage: dvips Pwww
% Purpose: create ps files "for the web". Output is send to a file and
% not to any printer. Outline fonts are used whenever available
% since we do not know at which resolution the user will print
% out output file.
o
p psfonts t1.map
```

**Beispiel 15.1:** Auszug aus der Konfigurationsdatei `config.www`

Über den Parameter `P` legen Sie fest, welche Font-Map-Datei verwendet werden soll. In dieser Datei wird dann für jeden Font festgelegt, welche Fontdatei und somit welcher Fonttyp verwendet werden soll:

```
cmbsy5 CMBSY5 <cmbsy5.pfb
cmbsy7 CMBSY7 <cmbsy7.pfb
cmmib5 CMMIB5 <cmmib5.pfb
cmmib7 CMMIB7 <cmmib7.pfb
```

```
euex10 EUEX10 <euex10.pfb
eufb10 EUFB10 <eufb10.pfb
```

Beispiel 15.2: Auszug aus einer map-Datei

Bei pdf<sub>l</sub>atex wird standardmäßig die Datei pdf<sub>t</sub>ex.map verwendet. Weitere Informationen über die vorhandenen Fonts und deren Konfigurationsdateien finden Sie in der Dokumentation Ihrer jeweiligen Distribution. Ein weiteres nützliches Dokument ist auch der fontinstallationguide. Sie finden ihn im Web unter [http://www.ctan.org/tex\\_archive/info/Typelfonts/fontinstallationguide/](http://www.ctan.org/tex_archive/info/Typelfonts/fontinstallationguide/).

15.3.3 Ergänzungspakete für Schriften

Für Schriften gibt es eine Reihe von Ergänzungspaketen, die Sie ganz einfach einbinden können. Hier sehen Sie eine Auswahl davon:

Das Paket psnfss

Das Paket psnfss enthält mehrere einzelne Pakete, um verschiedene Schriften zu verwenden. Das Paket psnfss enthält die Pakete helvet, mspatho, mathptmx und pifont. Weitere Informationen zu psnfss finden Sie in der Dokumentation unter [www.ctan.org/macros/latex/required/psnfss/psnfss2e.pdf](http://www.ctan.org/macros/latex/required/psnfss/psnfss2e.pdf).

| Paket    | Serifenschrift | Serifenlose Schrift | Schreibmaschinenschrift |
|----------|----------------|---------------------|-------------------------|
| -        | CM Roman       | CM Sans Serif       | CM Typewriter           |
| mathptmx | Times          |                     |                         |
| mathpazo | Palatino       |                     |                         |
| helvet   |                | Helvetica           |                         |
| avant    |                | Avant Garde         |                         |
| courier  |                |                     | Courier                 |
| chancery | Zapf Chancery  |                     |                         |
| bookman  | Bookman        | Avant Garde         | Courier                 |
| utopia   | Utopia         |                     |                         |
| charter  | Charter        |                     |                         |

Tabelle 15.1: Ergänzungspakete für PS-Schriften

Einen Font einbinden

Ein Font des Pakets psnfss wird über den Befehl \usepackage eingebunden:

```
\usepackage[Parameter]{helvet}
```

Über die Option `Parameter` können Sie verschiedene Parameter angeben.

### Das Paket `lmodern`

Das Paket `lmodern` verwendet den Font *Latin modern*. Der Font *Latin modern* entspricht weitgehend dem Font *Computer modern*, liegt aber im TrueType-Format vor.

## 15.3.4 Zeichen eines Fonts

Über den Befehl `\symbol` wählen Sie jedes beliebige Zeichen eines Fonts aus:

```
\symbol{<nummer>}
```

### Beispiele

Hier ein Beispiel, wie Sie mit dem Befehl `\symbol` Symbole des aktuellen Fonts ansprechen können:

```
\symbol{1}
\symbol{2}
\symbol{3}
\symbol{4}
```

**Beispiel 15.3:** Symbole des aktuellen Fonts ansprechen



**Abbildung 15.1:** Symbole anzeigen mit `\symbol`

### 15.3.5 Symbolfonts

Es gibt noch weitere Symbole, auf die Sie zugreifen können. Eine Liste von Symbolen finden Sie in dem Dokument `symbols a4.pdf`, das Sie unter [www.ctan.org/tex-archive/info/symbols/comprehensive/symbols a4.pdf](http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols a4.pdf) finden.

### 15.3.6 Der Font Zapf Dingbats

Über das Paket `pifont` und den Befehl `\ding{<nummer>}` können Sie auf Symbole des Zeichensatzes Zapf Dingbats zugreifen. Der Befehl `\ding` wird wie folgt aufgerufen:

```
\ding{1}
```

- *dinglist*

Mit der Umgebung `dinglist` wird eine Liste erstellt, der Symbole vorangestellt werden:

```
\begin{dinglist}
\end{dinglist}
```

Ein Beispiel:

```
\begin{dinglist}{164}
\item Text 1
\item Text 2
\item Text 3
\end{dinglist}
```



Abbildung 15.2: Darstellung von Dingbats-Symbolen

- *dingautolist*

Mit der Umgebung `dingautolist` lässt sich eine Aufzählungsumgebung (von 1 bis 9) erstellen, die die Aufzählungssymbole enthält. Die Anfangsnummer wird dann als Parameter der Umgebung angegeben:

```
\begin{dingautolist}{200}
\item Text 1
```

```
\item Text 2
\item Text 3
\end{dingautolist}
```

- `\dingfill`

Dieser Befehl verwendet Dingbats-Symbole für die Auffüllung einer Zeile:

```
Text \dingfill{233} Text
```

- `\dingline`

Dieser Befehl erzeugt eine Linie aus Dingbats-Symbolen.

```
\dingline{34}
```

### 15.3.7 Weitere Symbolfonts

Neben den oben beschriebenen Paketen gibt es noch weitere Pakete für Symbolschriften:

- `latexsym`

Mit dem Paket `latexsym` können Sie verschiedene mathematische und wissenschaftliche Symbole darstellen. Hier eine Liste der Befehle des Pakets:

- `\mho`  
Erzeugt ein umgedrehtes Ohm-Zeichen.
- `\Join`  
Erzeugt ein technisches Verbindungszeichen.
- `\Box`  
Erzeugt ein quadratisches Kästchen.
- `\Diamond`  
Erzeugt eine Raute.
- `\leadsto`  
Erzeugt einen Pfeil mit gezacktem Strich.
- `\sqsubset`  
Erzeugt ein Mengensymbol einer Untermenge.
- `\sqsupset`  
Erzeugt ein Mengensymbol einer Obermenge.
- `\lhd`  
Erzeugt einen linksgerichteten Pfeil.
- `\unlhd`  
Erzeugt einen linksgerichteten Pfeil mit Unterstrich.

- \rhd  
Erzeugt einen rechtsgerichteten Pfeil.
- \unrhd  
Erzeugt einen rechtsgerichteten Pfeil mit Unterstrich.
- wasysym  
Das Paket `wasysym` enthält Symbole für verschiedene Bereiche:
  - \Join  
Erzeugt ein technisches Verbindungssymbol.
  - \Box  
Erzeugt ein leeres quadratisches Kästchen.
  - \Diamond  
Erzeugt eine Raute.
  - \leadsto  
Erzeugt einen Pfeil mit gezacktem Strich.
  - \sqsubset  
Erzeugt ein linksgerichtetes Mengensymbol einer Untermenge.
  - \sqsupset  
Erzeugt ein rechtsgerichtetes Mengensymbol einer Obermenge.
  - \lhd  
Erzeugt einen Linkspfeil.
  - \unlhd  
Erzeugt einen Linkspfeil mit Unterstrich.
  - \LHD  
Erzeugt einen in Schwarz gefärbten Linkspfeil.
  - \rhd  
Erzeugt einen Rechtspfeil.
  - \unrhd  
Erzeugt einen Rechtspfeil mit Unterstrich.
  - \RHD  
Erzeugt einen schwarz ausgefüllten Rechtspfeil.
  - \apprle  
Erzeugt ein Kleiner-als-Zeichen mit einer Tilde darunter.
  - \apprge  
Erzeugt ein Größer-als-Zeichen mit einer Tilde darunter.

- `\wasypropto`  
Erzeugt ein offenes Unendlich-Symbol.
- `\invng`  
Erzeugt die obere linke Ecke eines Rechtecks.
- `\ocircle`  
Erzeugt einen Kreis.
- `\logof`  
Erzeugt einen Kreis mit einem Männchen darin.

Darüber hinaus enthält `wasysm` Symbole aus den Bereichen Mathematik, Physik, Polygone und Sterne, Notenzeichen, Kreise und astronomische Symbole sowie Zeichen, die in der Programmiersprache APL eine Rolle spielen.

**Tipp:** Weitere Informationen zum Paket `wasysm` erhalten Sie unter

<http://ftp.gwdg.de/pub/ctan/macros/latex/contrib/wasysm/wasysm.pdf>

### Das Paket `textcomp`

Das Paket `textcomp` enthält verschiedene Symbole für den Textmodus. Leider können nicht alle Symbole in jedem Font dargestellt werden.

Die Symbole können Sie über verschiedene Befehle wie z. B. `textacutedbl` oder `textasciibreve` ansprechen. Weitere Informationen entnehmen Sie der Paketdokumentation ([www.ctan.org/pub/tex archive/macros/latex/base/loutenc.dtx](http://www.ctan.org/pub/tex/archive/macros/latex/base/loutenc.dtx)).

### Das Paket `marvosym`

Das Paket `marvosym` stellt ebenfalls Symbole für diverse Kategorien zur Verfügung:

- Kommunikation – darunter Symbole für Telefon, Fax und Ähnliches
- Engineering
- Laundry – unter anderem Symbole für Waschanweisungen
- Währungssymbole
- Sicherheit – etwa Symbole für Radioaktivität, Explosionsgefahr u. Ä.
- Navigation – darunter die von Videorecordern bekannten Symbole
- Computer
- Mathematik
- Biologie – etwa Zeichen für männlich und weiblich

- Astronomie – hier vor allem Planetensymbole
- Astrologie – hier vor allem die Symbole des Tierkreises
- Sonstige – unter anderem religiöse Symbole

Diese Symbole können Sie über verschiedene Befehle ansprechen, wie z. B. `\Pickup`, `\Letter`, `\Beam` usw.

### 15.3.8 Fontdokumentationen

Für Fonts gibt es einige Dokumente, die für Sie von Interesse sein könnten:

- [www.ctan.org/tex\\_archive/macros/latex/doc/fntguide.pdf](http://www.ctan.org/tex_archive/macros/latex/doc/fntguide.pdf)
- Der Fontinstallation-Guide: [http://tug.ctan.org/tex\\_archive/info/Type1fonts/fontinstallationguide/](http://tug.ctan.org/tex_archive/info/Type1fonts/fontinstallationguide/)

# 16 Praktische Anwendungen

In den bisherigen 15 Kapiteln wurden die theoretischen Grundlagen von LaTeX gezeigt. In diesem Kapitel erfahren Sie, wie Sie LaTeX praktisch einsetzen. Es werden Lösungswege aufgezeigt, wie Sie z. B. Briefe, Briefköpfe, Musterdokumente und Ähnliches erstellen. Ausgegangen wird von einem einfachen Grunddokument, das je nach Anforderung entsprechend geändert wird.

## 16.1 Ein einfacher Artikel mit LaTeX

In Kapitel 2 und 3 haben Sie bereits gesehen, wie Sie einfache Dokumente setzen können. Die grundlegende Dokumentenklasse, die für dieses Beispiel verwendet wird, ist die Klasse `article`. Es wird also ein Artikel geschrieben, der für deutsche Sprache und deutsche Konventionen formatiert werden soll:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{babel}
\setlength{\headheight}{10pt}
\setlength{\footskip}{10pt}
\setlength{\marginparwidth}{20pt}
\setlength{\textheight}{100pt}
\setlength{\textwidth}{100pt}
\begin{document}
%Artikel
\end{document}
```

**Beispiel 16.1:** Ein einfacher Artikel.

Dieses Beispiel ist dem Beispiel aus Kapitel 2 sehr ähnlich. Es werden die wichtigsten Pakete für die deutsche Sprache und die deutschen Konventionen eingebunden. Anschließend werden mit dem Kommando `\setlength` die benötigten Einstellungen für das Dokumentenlayout vorgenommen. Mit den genannten Einstellungen sollte man im professionellen Einsatz experimentieren, um das gewünschte Ergebnis zu erhalten.

### Titel und Autor

Mit den Befehlen `\title` und `\author` können Sie den Titel und den Autor eines Dokumentes hinzufügen:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{babel}
\setlength{\headheight}{10pt}
\setlength{\footskip}{10pt}
\setlength{\marginparwidth}{20pt}
\setlength{\textheight}{100pt}
\setlength{\textwidth}{100pt}
\title{Mein Dokument}
\author{Alexander Schunk}
\begin{document}
%Dokument
\end{document}
```

**Beispiel 16.2:** Einfügen von Titel und Autor.

### Abschnitte und Unterabschnitte

In diesen Grundaufbau eines Dokumentes fügen Sie nun Abschnitte und Unterabschnitte ein:

```
\documentclass[a4paper]{article}
\usepackage[T1]{fontenc}
\usepackage[ngerman]{babel}
\usepackage[latin1]{babel}
\setlength{\headheight}{10pt}
\setlength{\footskip}{10pt}
\setlength{\marginparwidth}{20pt}
\setlength{\textheight}{100pt}
\setlength{\textwidth}{100pt}
\begin{document}
Dies ist ein Beispielartikel mit Abschnitten und Unterabschnitten.
\section{Einführung}
In diesem Abschnitt stehen einleitende Sätze.
\section{Hauptteil}
In diesem Abschnitt steht der Hauptteil des Artikels.
```

```
\section{Dritter Abschnitt}
In diesem Abschnitt stehen abschließende Bemerkungen.
\end{document}
```

**Beispiel 16.3:** Gliederung eines Dokuments.

Diese Vorlage benutzen Sie für verschiedene Dokumenttypen wie z. B. einfache Artikel oder wissenschaftliche Artikel usw.

## 16.2 Ein einfacher Brief mit LaTeX

Mit der Umgebung `letter` lässt sich ein einfacher Brief in LaTeX abfassen. Die Umgebung `letter` wird mit dem Befehl `\usepackage{letter}` eingebunden und stellt die Befehle `\address`, `\signature`, `\closing` und `\enclosure` zur Verfügung.

### Einfacher Brief

Hier der Aufbau für einen einfachen Brief:

```
\documentclass[a4paper]{article}
\usepackage{letter}
\address{Alexander Schunk, Sonnenscheinweg 1, 50001
 Sonnenscheinland}
\signature{Alexander Schunk}
\begin{document}
 \begin{letter}{Business Center Sonnenmarkt, Sonnenstr. 1,
 Etage, 50667 Sonnenstadt}
 \opening{Betr: Büroanmietung}
 Sehr geehrte Damen und Herren,
 ich möchte hiermit ein Büro bei Ihnen anmieten.
 \closing{Mit freundlichen Grüßen}
 \end{letter}
\end{document}
```

**Beispiel 16.4:** Aufbau eines Briefs.

In diesem Beispiel wird ein einfacher Brief aufgesetzt mit Absenderadresse, Empfängeradresse, Betreffzeile und Grußformel. Die Absenderadresse wird mit dem Befehl `\address` eingeleitet. Die Empfängeradresse wird mit der Umgebung `letter` definiert. Mit dem Befehl `\closing` bauen Sie eine Grußformel ein. Mit dem Befehl `\enclosing` werden Anlagen eingefügt.

```

\documentclass[a4paper]{article}
\usepackage{letter}
\address{Alexander Schunk, Sonnenscheinweg 1, 50001
 Sonnenscheinland}
\signature{Alexander Schunk}
\begin{document}
 \begin{letter}{Business Center Sonnenmarkt, Sonnenstr. 1,
Etage, 50667 Sonnenstadt}
 \opening{Betr: Büroanmietung}
 Sehr geehrte Damen und Herren,
 ich möchte hiermit ein Büro bei Ihnen anmieten.
 \closing{Mit freundlichen Grüßen}
 \enclosing{Anlage}
 \end{letter}
\end{document}

```

**Beispiel 16.5:** Brief mit Anlage.

In diesem Beispiel erhält der Brief die Anlage Anlage.

### 16.2.1 Hübsche Kleinigkeiten

Für besondere Schriftstücke, wie zum Beispiel Grußkarten oder Urkunden, bietet LaTeX in Verbindung mit dem Paket `shapepar` ein paar Spielereien. Mit den vordefinierten Befehlen aus diesem Ergänzungspaket können Sie Absätze rautenförmig, herzförmig oder quadratisch gestalten.

```

\diamondpar{<Text des Absatzes>}
\heartpar{<Text des Absatzes>}
\squarepar{<Text des Absatzes>}

```

Hier ein Beispiel, wie Sie einen Absatz besonders formatieren:

```

\usepackage{shapepar}
\begin{document}
\heartpar{Dieser Absatz wird in Herzform formatiert. Als Ergebnis sehen Sie
ein großes Herz, in dem der Text dargestellt wird. Optimalerweise nutzt man
diesen Befehl über mehrere Absätze hinweg.}
\end{document}

```

**Beispiel 16.6:** Ein Absatz in Herzform.

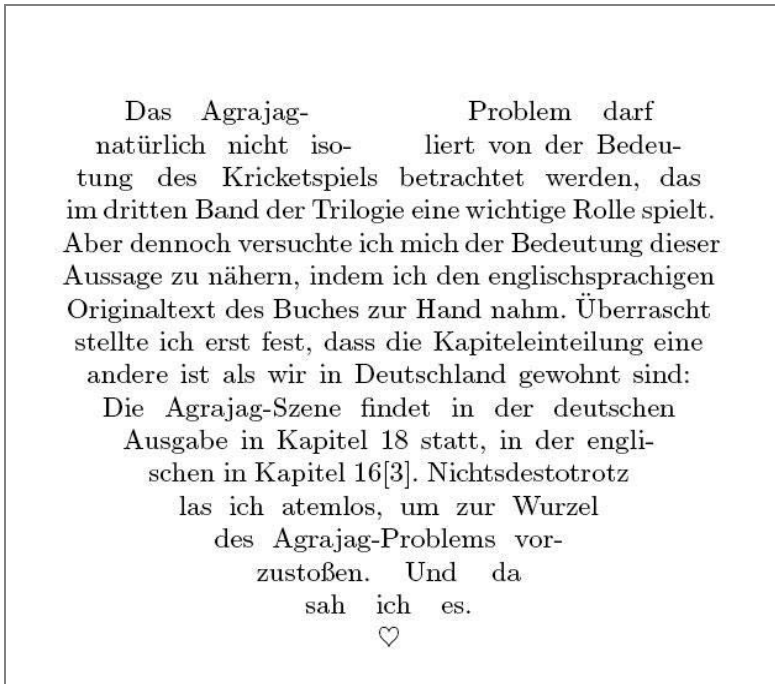


Abbildung 16.1: Absatz in Herzform

### Brief nach DIN-Norm

Mit der `letter`-Umgebung lassen sich zwar einfache Briefe erstellen, nicht jedoch Briefe nach der DIN-Norm.

Dieses Problem löst die Umgebung `g brief`, die Sie über `\usepackage{g brief}` einbinden.

```
\documentclass[a4paper]{article}
\usepackage{g brief}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}

\Name{Alexander Schunk}
\Strasse{Sonnenstr. 1}
\Postleitzahl{50667 Sonnenstadt}
\Email{alexanderschunk@web.de}
\Adresse{Alfred Schunk\
Mondstr. 3, 20411 Mondtadt}
\Unterschrift{Alexander Schunk}
```

```
\begin{document}
```

```
\end{document}
```

### Beispiel 16.7: Ein Brief nach DIN-Norm.

Als Ergebnis erhalten Sie einen Brief nach DIN-Norm. Die Umgebung `g brief` stellt diverse Befehle zur Verfügung, mit denen Sie verschiedene Angaben im Briefkopf machen, wie z. B. Name, Anschrift, Webadresse, Telefon, E-Mail-Adresse usw. Die einzelnen Angaben können Sie mit den entsprechenden Befehlen setzen, z. B. mit `\Email` die E-Mail-Adresse, mit `\HTTP` die Website-Adresse, mit `\Telefon` die Telefonnummer usw.

### Briefe mit KOMA-Script

Das Paket `scr1ttr2` definiert ebenfalls eine Reihe von Befehlen, um einen Brief entsprechend der deutschen Norm zu setzen:

```
\documentclass[DIN, pagenumber false, parskip half,%
 fromalign left, fromphone true,%
 fromemail true, fromurl true, %
 fromlogo true, fromrule false]{scr1ttr2}

\usepackage[latin1]{inputenc}
\usepackage{german}
\RequirePackage{graphicx}

\setkomavar{fromname}{Übersetzer \\\ Hans Meyer }
\setkomavar{fromaddress}{Mondallee 1 \\\ 23422 Bremen}
\setkomavar{fromphone}{02214 63495}
\setkomavar{fromemail}{mustermann@muster.de}
\setkomavar{fromurl}{www.muster.de}
\setkomavar{signature}{Hans Meyer}
\setkomavar{fromlogo}{\includegraphics*[width 2cm]{logo}}

\begin{document}

\begin{letter}{Herr \\\ Alexander Schunk \\\
 Sonnenstr. 13 \\\ 50667 Stadt
 }
\opening{Sehr geehrter Herr ,}
hiermit gebe ich bekannt, dass.....
```

```
\closing{Mit freundlichem Gruss}
\end{letter}

\end{document}
```

**Beispiel 16.8:** Brief mit KOMA-Script.

In diesem Beispiel werden über die Dokumentenklasse verschiedene Einstellungen für das Dokument gesetzt, wie z. B. Seitennummer, Absatzabstand usw. Mit dem Befehl `\setkomavar` werden dann die einzelnen Angaben zu Beruf, Anschrift, Telefonnummer, E-Mail, Webadresse etc. angegeben.

Der Brief selbst wird dann mit der Umgebung `letter` gesetzt.



# 17 PDFLaTeX und PostScript

Dieses Kapitel beschäftigt sich mit den Möglichkeiten, zwischen `pdflatex` und PostScript zu konvertieren. Es existiert eine Reihe von Hilfsprogrammen, die es erlauben, Konvertierungen zwischen PostScript und PDF sowie zwischen PostScript und EPS (Encapsulated PostScript) durchzuführen.

## 17.1 Von PostScript nach PDF konvertieren

Die einfachste Möglichkeit für die Konvertierung von PDF zu PostScript ist das Tool `epstopdf`, das es aktuell in der Version 2.9.9gw gibt. Sie finden die aktuelle Variante im Internet unter [http://www.ctan.org/tex\\_archive/support/epstopdf/](http://www.ctan.org/tex_archive/support/epstopdf/).

### Das Programm `epstopdf`

Das Programm `epstopdf` konvertiert eine EPS-Grafik in eine PDF-Grafik. Der allgemeine Aufruf lautet:

```
epstopdf outfile grafikdatei.pdf grafikdatei.eps
```

Das Bild selbst kann dann über den Befehl `\includegraphics` in `pdflatex` eingebunden werden. Weitere Informationen dazu finden Sie in Kapitel 6.

### Das Programm `ps2pdf`

Mit dem Programm `ps2pdf` konvertieren Sie eine PostScript- in eine PDF-Datei.

Der allgemeine Aufruf des Programms lautet:

```
Ps2pdf grafikdatei.ps
```

Als Resultat entsteht eine PDF-Datei namens `grafikdatei.pdf`.

**Tipp:** Von dem Programm `ps2pdf` gibt es auch eine Onlineversion, die Sie unter [www.ps2pdf.com](http://www.ps2pdf.com) finden.

### PDFLaTeX und `ghostscript`

Die Programme `ps2pdf` und `epstopdf` rufen intern das Programm `ghostscript` auf, um in das PostScript-Format zu konvertieren. Dabei können einige Probleme auftreten. So kann es zum Beispiel passieren, dass der Inhalt der resultierenden PDF-Datei um 90

Grad gedreht wird. Dem können Sie entgegenwirken, indem Sie für die `ghostscript`-Option den Parameter `AutoRotatePages` auf `none` setzen:

```
ps2pdf d/AutoRotatePages /None datei.ps datei.pdf
GS OPTIONS AutoRotatePages none
epstopdf outfile datei.pdf datei.eps
```

Die von `ps2pdf` erzeugte PDF-Datei weist in der Regel einen weißen Rand auf. Mit dem Hilfsprogramm `pdfcrop` wird dieser entfernt.

Der allgemeine Aufruf des Programms lautet:

```
Pdfcrop clip grafikdatei.pdf grafikdatei neu.pdf
```

## 17.2 Temporäre Auslagerung von PostScript-Code

Eine weitere interessante Konvertierungslösung ist `ps4pdf`. Dieses Programm sammelt gewissermaßen PostScript-Teile in einem LaTeX-Dokument ein und konvertiert sie nach PDF. Dazu lagert es den PostScript-Code intern aus, übersetzt ihn anschließend und bindet zuletzt das Resultat wieder ein, meist als Bild. Eine Schlüsselrolle spielt hier der Makrobefehl `\PSforPDF{}`. Er bekommt den Code, der PostScript enthält (zum Beispiel PostScript-Bilder), als Parameter übergeben.

Der Konvertierungsprozess läuft wie folgt ab:

1. Der Befehl `\PSforPDF{}` wird in das LaTeX-Dokument eingefügt, um PostScript-Elemente zu markieren. Beim Kompilieren des Dokumentes durch LaTeX wird dieser PostScript-Code übersetzt und in eine eigene Seite ausgelagert. Daraus entsteht dann eine DVI-Datei, die so viele Seiten enthält, wie `\PSforPDF`-Befehle verwendet worden sind.

```
Latex Beispieldatei.tex
```

2. Dann wird die so erzeugte DVI-Datei in eine PostScript-Datei umgewandelt.

```
Dvips o beispieldatei2.ps beispieldatei.dvi
```

3. Danach wird die PostScript-Datei mithilfe von `ps2pdf` in eine PDF-Datei konvertiert.

```
ps2pdf beispieldatei2.ps beispieldatei.pdf
```

4. Anschließend wird das eigentliche PDF-Dokument mit `pdflatex` erzeugt. Das Paket `ps4pdf` bindet dabei alle während des Konvertierungsprozesses erzeugten Bilder automatisch mit dem Befehl `\includegraphics` ein.

```
pdflatex beispieldatei .tex
```

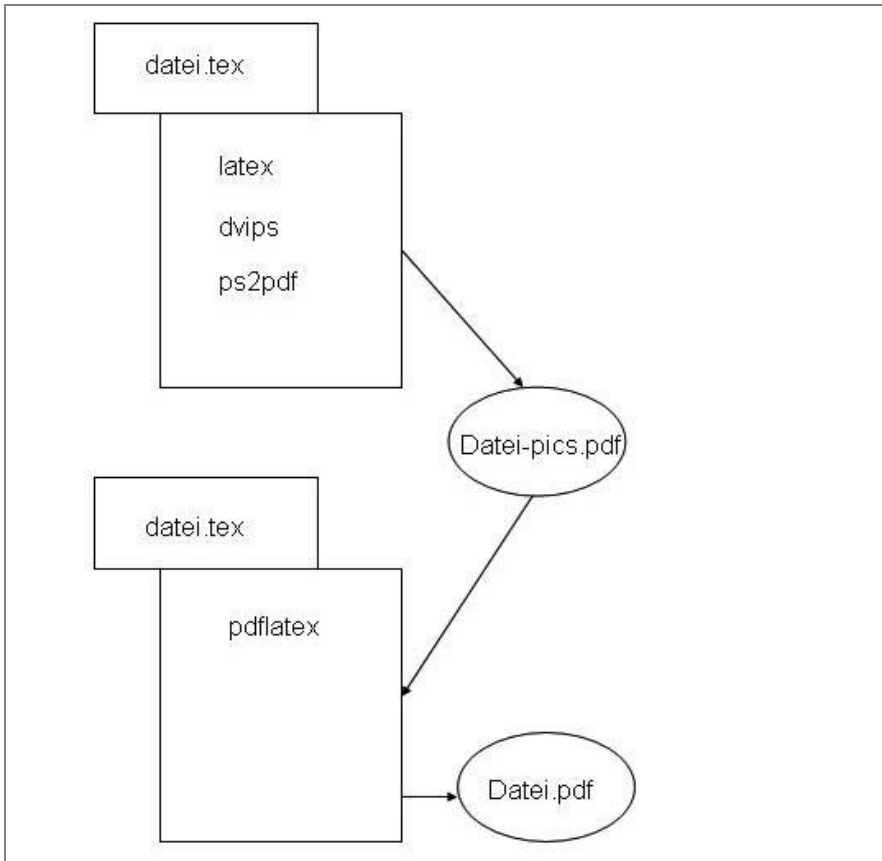


Abbildung 17.1: Von PostScript nach PDF

### Ein Bild extrahieren: Das Paket `preview`

Wesentlicher Bestandteil des Konvertierungsprozesses ist das Paket `preview`. Neben der Möglichkeit, eine ganze Datei umzuwandeln, bietet das Paket `preview` die Möglichkeit, einzelne Bereiche einer LaTeX-Datei als Bild zu extrahieren.

Das nachfolgende Beispiel zeigt, wie dies funktioniert:

```
\documentclass[a4paper]{article}
\usepackage{displaymath, active, tightpage}{preview}[2004/04/11]
\begin{document}
\section{Ein Text}
Ein Abschnitt
\begin{equation}
\frac{1}{1 x^2}
```

```

\end{equation}
Abschnitt
\end{document}

```

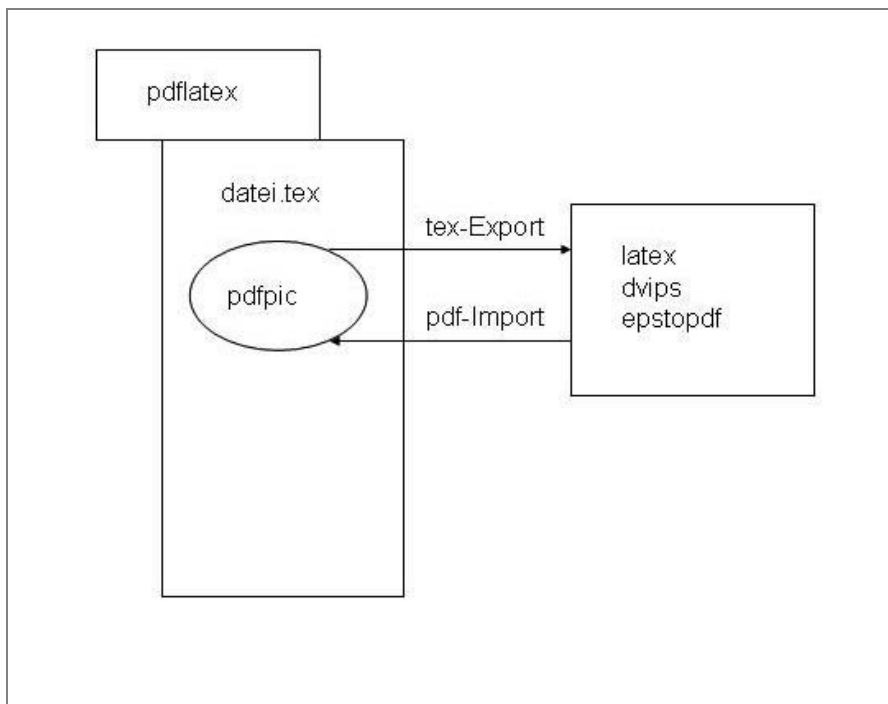
**Beispiel 17.1:** Eine Formel wird als Bild extrahiert

### Das Paket pdftricks

Als letzte Alternative gibt es noch das Paket `pdftricks`. Hier werden alle PostScript-Befehle in die Umgebung `pdfpic` eingebettet. Beim anschließenden `pdflatex`-Durchlauf wird dann der Inhalt der Umgebung in eine externe Datei geschrieben und diese anschließend mit den Befehlen `latex`, `dvips` oder `epstopdf` weiterbearbeitet, sodass man dann am Schluss die PDF-Grafik erhält:

Zunächst wird die LaTeX-Datei erstellt. Dann wird das Paket `pdfpic` aufgerufen. Dabei findet ein Austausch zwischen dem Paket `pdfpic` und den Programmen `latex`, `dvips`, `epstopdf` statt.

Die nachfolgende Abbildung veranschaulicht noch einmal den Prozess:



**Abbildung 17.2:** Funktionsweise von `pdfwrite`

Am Ausgangspunkt steht `pdflatex`. Damit wird die Datei `datei.tex` kompiliert und anschließend `pdfpic` aufgerufen. Darauf erfolgt ein Export nach `latex`, `dvips` oder `epstopdf` und umgekehrt

## 17.3 PostScript-Code in eine externe Datei auslagern

Eine weitere Möglichkeit besteht darin, den Code, den PostScript benutzt, in eine externe Datei auszulagern und diese dann mit dem Befehl `latex` oder Ähnlichem in PDF-Bilder zu übersetzen. Diese so erzeugten Bilder können Sie im Hauptdokument mithilfe von `\includegraphics` einbinden.

Wenn Sie den Übersetzungsvorgang mithilfe des Tools `make` steuern, werden immer nur die Dateien übersetzt, die geändert worden sind. Damit wird der Übersetzungsvorgang bei Änderungen deutlich beschleunigt. Die folgende Grafik veranschaulicht diesen Prozess:

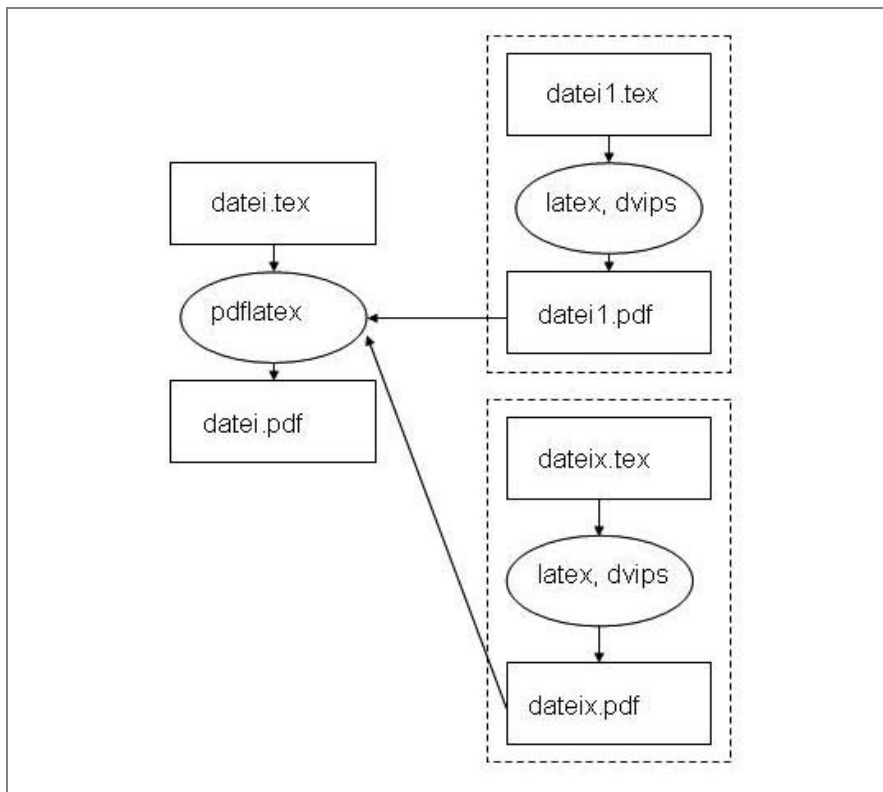


Abbildung 17.3: Funktionsweise beim Auslagern von PostScript-Code



# 18 Fremdformate konvertieren

Wenn Sie mit OpenOffice oder Word arbeiten, ist es möglich, diese Dateien über verschiedene Hilfsprogramme in ein LaTeX-Dokument umzuwandeln. Hier soll kurz das Programm Writer2LaTeX vorgestellt werden. Die aktuellste Version dieses Programms ist Version 0.93, die auch die Konvertierung in XHTML-Dateien erlaubt.

## 18.1 Von OpenOffice nach LaTeX

Mit Writer2LaTeX konvertieren Sie OpenOffice-Dokumente nach LaTeX. Das Programm ist ein Java-Programm, das von Henrik Just entwickelt wurde. Es erweitert OpenOffice und lässt sich über die Kommandozeile aufrufen. Bisher war Writer2LaTeX Bestandteil von OpenOffice.org, inzwischen wird es laut dem OpenOffice-Wiki als separate Extension verteilt. Das Tool kann unter <http://writer2latex.sourceforge.net/> heruntergeladen werden; ausführliche Informationen finden Sie im OpenOffice-Wiki unter <http://www.oowiki.de/Writer2LaTeX>.

Das Programm Writer2LaTeX benötigt eine aktuelle Java-Laufzeitumgebung, die Sie sich vorher installieren sollten. Dann müssen Sie den Pfad zu Writer2LaTeX über den Befehl `set` setzen.

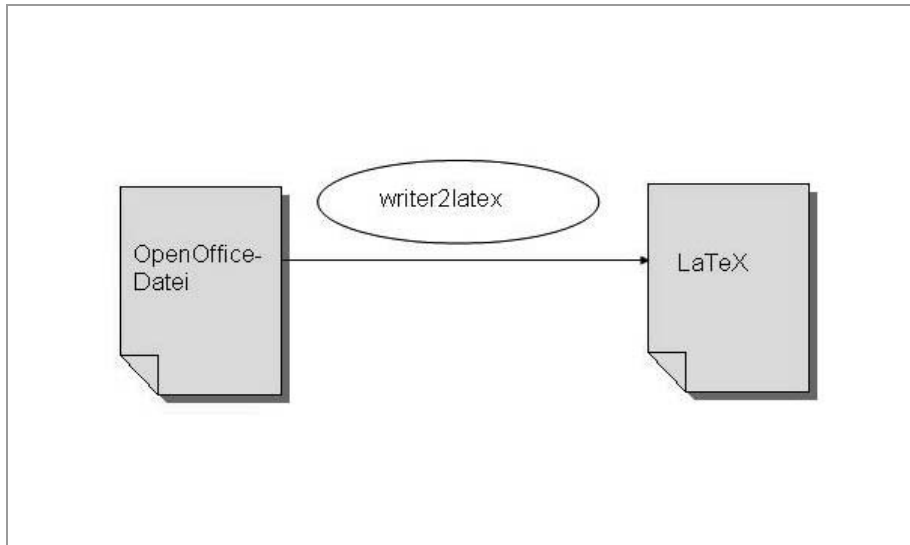


Abbildung 18.1: Von OpenOffice nach LaTeX

## 18.2 Installation von Writer2LaTeX unter Windows und Linux

Die Installation von Writer2LaTeX läuft unter Windows und Linux nahezu identisch ab. Die genannten Path-Variablen finden Sie in der Batch-Datei `w2l.bat`.

### Installation unter Windows

1. Laden Sie sich die Datei `writer2LaTeX1.0.zip` von dem oben angegebenen Link herunter und entpacken Sie sie in ein beliebiges Verzeichnis.
2. Fügen Sie dieses Verzeichnis der Umgebungsvariablen `PATH` hinzu.
3. Öffnen Sie dann die Datei `w2l.bat` mit einem Texteditor, z. B. Notepad, und ändern Sie das Verzeichnis entsprechend ab:  
`set W2LPATH "c:\writer2LaTeX\"`
4. Sollte das Programm dann immer noch nicht funktionieren, müssen Sie sicherstellen, dass der Pfad zum Java-Interpreter gesetzt ist:  
`Set JAVAEXE "c:\j2sdk\bin\Java"`

Dann können Sie das Programm über `w2l` aufrufen.

## Installation unter Linux

Die Installation unter Linux läuft nahezu identisch ab:

1. Laden Sie sich die Datei `writer2LaTeX10.zip` vom oben angegebenen Link in ein beliebiges Verzeichnis.
2. Fügen Sie auch hier dieses Verzeichnis der Umgebungsvariablen `PATH` hinzu. Alternativ können Sie auch einen symbolischen Link zu einem Verzeichnis erstellen, der in der Umgebungsvariablen `PATH` enthalten ist.

Öffnen Sie auch hier die Datei `w2l.bat` mit einem Texteditor. z. B. `Kyle`, und ändern Sie den Pfad entsprechend ab:

```
set W2LPATH "/home/username/writer2LaTeX10"
```

Falls `Writer2LaTeX` dann noch nicht funktioniert, müssen Sie den Pfad zum Java-Interpreter überprüfen:

```
set JAVAEXE "/path/to/Java/executable"
```

3. Abschließend müssen Sie über `chmod` noch die Ausführungsrechte für die Datei setzen:

```
chmod +x w2l
```

Danach können Sie das Programm aufrufen. Weitere Informationen zur Installation finden Sie auf der Programm-Homepage: <http://writer2latexsourceforge.net> unter »Documentation«.

## Aufruf des Programms

Das Programm `Writer2LaTeX` hat verschiedene Parameter, die Sie ihm übergeben können.

```
w2l [ultraclean | clean | pdfscreen | pdfprint | article] [config
<configdatei>]
<zukonvertierende-datei> <ausgabe>
```

Über den Parameter `ultraclean` legen Sie fest, wie OpenOffice-Dateien in LaTeX-Formate übernommen werden sollen. Dabei ist `ultraclean` eine Abkürzung für `config ultraclean`.

| Kurzversion             | Langversion                    | Beschreibung                             |
|-------------------------|--------------------------------|------------------------------------------|
| <code>ultraclean</code> | <code>config ultraclean</code> | Alle Formatierungen werden entfernt.     |
| <code>clean</code>      | <code>config clean.xml</code>  | Es werden viele Formatierungen entfernt. |

| Kurzversion | Langversion          | Beschreibung                                                                            |
|-------------|----------------------|-----------------------------------------------------------------------------------------|
| pdfscreen   | config pdfscreen.xml | Die Ausgabe wird für eine Bildschirmpräsentation formatiert.                            |
| pdfprint    | config pdfprint.xml  | Die Ausgabe wird für eine Druckausgabe formatiert.                                      |
| article     | config article.xml   | Es wird die Klasse article verwendet und es bleiben sehr viele Formatierungen erhalten. |

Tabelle 18.1: Parameter für das Programm Writer2LaTeX

Ein Beispiel: Konvertierung von OpenOffice nach LaTeX

Für das folgende Beispiel wurde eine einfache OpenOffice-Datei erstellt. Diese Datei enthält einen simplen Text sowie eine Tabelle. Das fertige OpenOffice-Dokument sieht so aus:

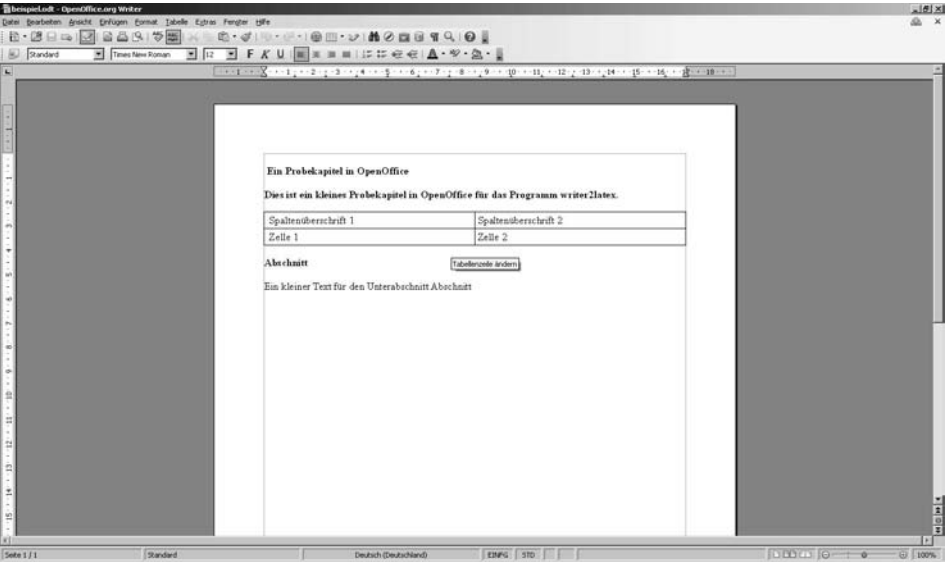


Abbildung 18.2: OpenOffice-Dokument

Diese Datei wird dann mit dem Programm Writer2LaTeX übersetzt:

```
W2l ultraclean beispiel.odt beispiel.tex
```

Als Ergebnis erhält man die Datei `beispiel_ultraclean.tex`:

```
% This file was converted to LaTeX by writer2LaTeX ver 0.3 3g

%see http://www.hj.gym.dk/~hj/writer2LaTeX for more info

\documentclass[12pt]{article}
\usepackage[ascii]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[english]{babel}
\usepackage{amsmat, amssymb, amsfonts, textcomp}
\usepackage{calc}
\usepackage{longtable}
\usepackage{graphicx}
\usepackage{ooomath}
```

Der Text des Dokumentes wird dann wie gewohnt im Hauptdokument dargestellt:

```
\begin{document}
\section{Ein Probekapitel in OpenOffice}
Dieses ist ein kleines Probekapitel in OpenOffice für das Programm
writer2LaTeX.

\begin{longtable}[c]{lp{3.1999cm}|p{4.4430003cm}|p{8.757cm}|}
\hline
\befseries
\end{document}
```

Das Programm Writer2LaTeX macht aus dem OpenOffice-Dokument eine entsprechende LaTeX-Datei, in der die benötigten Pakete eingebunden und die benötigten Befehle aufgerufen werden.

Das Dokument wird dabei mit der Klasse `article` in Schriftgrad 12 formatiert. Die Überschriften werden mit dem Befehl `\section` formatiert. Für Tabellen wird das Paket `longtable` eingebunden.

### Anpassung der Konfiguration

Sie können die Konfiguration von Writer2LaTeX anpassen, indem Sie die Datei `ultraclean.xml` als Basis nehmen. Die Datei `ultraclean.xml` ist so aufgebaut:

```
<?xml version "1.0" encoding "ISO 8859 1"?>
<config>
 <option name "create user config" value "false">
 <! Alle Konfigurationseinstellungen werden in einer XML
 Datei gespeichert >
```

```
<! Weitere Konfigurationen >
</config>
```

In dieser XML-Datei stehen einige Definitionen und Attribute, die für das Programm Writer2LaTeX verbindlich sind. Diese Datei können Sie nach eigenen Wünschen anpassen und die Optionen entsprechend verändern.

Mit einer neu definierten `ultraclean.xml`-Datei können Sie das obige Beispiel nun ändern:

```
W2l config myconfig.xml beispiel.odt beispiel myconfig.tex
```

Wenn Sie die so erzeugte TeX-Datei mit `pdftex` übersetzen, erhalten Sie ein PDF-Dokument mit zwei Seiten.

### Writer2LaTeX als OpenOffice-Plugin

Das Programm Writer2LaTeX gibt es auch als OpenOffice-Plugin, das Sie sich über die Download-Seite von OpenOffice herunterladen können (<http://extensions.services.openoffice.org/project/writer2latex>). In OpenOffice können Sie dieses Programm auch über den Extension Manager beziehen. Klicken Sie einfach auf den Link »Hier erhalten Sie weitere Extensions.«

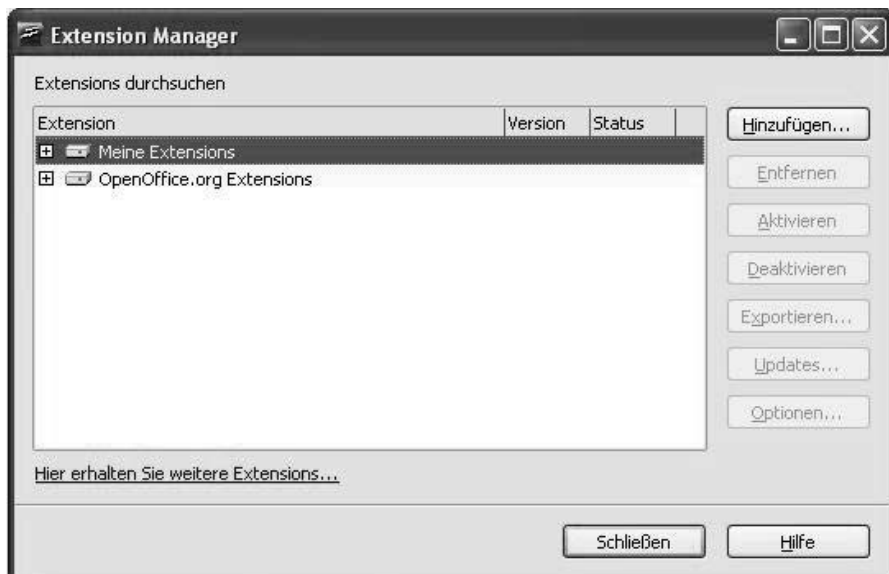


Abbildung 18.3: Extension Manager in OpenOffice

Um das Programm als OpenOffice-Extension zu installieren, gehen Sie wie im Folgenden beschrieben vor. Wichtig dabei ist, dass Ihre OpenOffice-Installation Java unterstützt.

1. Installieren Sie zunächst OpenOffice ganz normal.
2. Im zweiten Schritt müssen Sie in das Installationsverzeichnis wechseln. Das ist je nach Betriebssystem unterschiedlich bezeichnet.
3. Rufen Sie im Installationsordner das Programm `jvmsetup` auf, um die Java-Unterstützung zu aktivieren – unter Linux müssen Sie dazu root-User sein.
4. Bei der Standardinstallation wird der Filterexport normalerweise nicht mit installiert. Er lässt sich aber über das Setup-Programm leicht nachrüsten. Dazu können Sie das Programm Setup mit dem Parameter `reinstall` aufrufen. Wenn Sie hier vorher eine Netzinstallation vorgenommen haben, geben Sie den Parameter `net` an. Anschließend wird die Auswahl »Filter für mobile Geräte« aktiviert. Danach wird das Paket `xmerge` verwendet.

Anschließend müssen die entsprechenden `w2l`-Dateien kopiert werden:

Die Dateien `writer2LaTeX.jar`, `xmergefix.jar` und die Konfigurationsdatei `writer2LaTeX.xml` (aus dem XML-Verzeichnis) werden in das Unterverzeichnis `<OpenOffice/program/classes>` von OpenOffice kopiert. Unter Linux müssen Sie darauf achten, dass Leserechte für alle Benutzer eingerichtet sind:

Die Datei `xmerge.jar` muss in `oldxmerge.jar` umbenannt werden.

Die Datei `xmergefix.jar` wird in `xmerge.jar` umbenannt.

Die Datei `w2lfilter.zip` muss in das Unterverzeichnis `OpenOffice/share/uno_packages` kopiert werden.

5. Nun müssen Sie OpenOffice noch mitteilen, dass neue Filter vorhanden sind. Dazu wird das Programm `pkgchk` mit dem Parameter `shared` im Verzeichnis `OpenOffice/program` aufgerufen. Dabei darf OpenOffice selbst nicht gestartet sein.

Nun können Sie OpenOffice starten und den Text als LaTeX-Datei exportieren. Dazu wird im Menü *Datei* die Option *Export* aufgerufen. Unter *Dateiformat* wählen Sie dann LaTeX2e (.tex) aus. Als Ergebnis erhalten Sie eine TeX-Datei.

Normalerweise verwendet `w2l` die Einstellungen, die in der Datei `writer2LaTeX.xml` eingestellt sind. Wenn Sie z. B. die Konfiguration aus der Datei `clean.xml` verwenden wollen, müssen Sie diese Datei in das Home-Verzeichnis mit dem Namen `writer2LaTeX.xml` kopieren.

### Von DOC/RTF nach LaTeX

Wenn Sie mit Word arbeiten und Dokumente nach `.tex` konvertieren wollen, müssen Sie einen kleinen Umweg gehen, da es derzeit keinen aktuellen Konverter von Word nach LaTeX gibt.

Als Umweg öffnen Sie eine DOC- bzw. RTF-Datei in OpenOffice und speichern diese als OpenOffice-Datei. Dann können Sie diese OpenOffice-Datei über Writer2LaTeX konvertieren.

**Tipp:** Mit dem Programm `tex2rtf` können Sie von LaTeX nach RTF konvertieren.

## 18.3 HTML nach LaTeX

Es gibt auch die Möglichkeit, HTML-Dokumente nach LaTeX zu konvertieren. Dazu gibt es z. B. das Programm `html2LaTeX`. Von diesem Programm ist derzeit die Version 1.0 aktuell. Das Programm `html2LaTeX` finden Sie auf <http://html2LaTeX.sourceforge.net/> als Download.

Das Programm `html2LaTeX` wird wie folgt aufgerufen:

```
html2LaTeX b hr dante faq de.html
```

Als Ergebnis erhalten Sie dann eine TEX-Datei, die die entsprechenden Formatierungsbefehle enthält.

Als Beispiel wird die FAQ-Seite des DANTE-Vereins verwendet. Das Programm `HTML2LaTeX` wird dafür wie folgt aufgerufen:

```
Htm12LaTeX b hr dante faq de.html
```

Als Ergebnis erhält man dieses LaTeX-Dokument:

```
\documentclass[10pt]{article}
\usepackage{fullpage, graphicx, url}
\setlength{\parskip}{1ex}
\setlength{\parindent}{0ex}
\title{DE Tex FAQ: German FAQ about TeX and DANTE e.V.}
\begin{document}
\section*{DE Tex FAQ Fragen und Antworten \ "uber TeX, LaTeX und DANTE
e.V.}

\begin{itemize}
\item Inhaltsverzeichnis
```

```

\item Teil 1: Allgemeines
\item Teil 2: Anwendervereinigungen, Tagungen, Literatur
\item Teil 3: Textsatzsysteme TeX \text{"übersicht}
\end{itemize}
\end{document}

```

Die HTML-Datei wird in eine entsprechende LaTeX-Datei umgewandelt. LaTeX bindet die benötigten Pakete ein und formatiert die Überschrift mit dem `\section`-Befehl. Die Auflistungen werden mit einer `\itemize`-Umgebung formatiert.

Wenn Sie diese Datei dann anschließend mit `pdfLaTeX` übersetzen, erhalten Sie ein PDF-Dokument. So können Sie über den Umweg über LaTeX aus einer HTML-Seite eine PDF-Datei erzeugen.

## 18.4 Von DocBook nach LaTeX

Auch für das DocBook-Format gibt es die Möglichkeit, diese Dateien nach LaTeX zu konvertieren. DocBook ist ein populäres Format zur Erstellung von Dokumentationen. Mithilfe von XSL-Transformationen lassen sich DocBook-Dokumente in nahezu alle Formate umwandeln. Diese Stylesheets finden Sie auf <http://db2LaTeX.sourceforge.net/>.

Neben diesen Stylesheets gibt es noch das Perl-Programm `dblup`, das Sie auf der Seite <http://dblup.sourceforge.net/> finden.

## 18.5 Notlösungen

Wenn Sie kein entsprechendes Hilfsprogramm zur Verfügung haben, bleibt noch die Notlösung, den Text per Hand in die TeX-Datei einzufügen. Dieses Vorgehen können Sie nach dem bewährten Copy-and-Paste-Verfahren vornehmen.

Abschließend sei noch erwähnt, dass es für Windows-User kaum Konvertierungsprogramme gibt, da das OpenOffice-Plugin `Writer2LaTeX` auf die Linux-Welt zugeschnitten ist. Abhilfe können Sie hier aber schaffen, indem Sie sich OpenOffice unter Windows installieren und alle oben genannten Schritte mit dem Programm `Writer2LaTeX` unter Windows durchführen.



# 19 LaTeX und PDF-Dokumente

Ein Ziel bei der Arbeit mit LaTeX ist es, PDF-Dokumente zu erstellen, da sich PDF mittlerweile als Quasi-Standard für Printdokumente herauskristallisiert hat. Damit das Ergebnis auch ansprechend aussieht, gibt es ein paar Besonderheiten zu beachten.

## 19.1 Links in PDF-Dokumente einbinden

Es gibt eine einfache Möglichkeit, HTML-Links in PDF-Dokumente einzubinden. Mit dem Paket `hyperref` können Sie z. B. automatisch solche Hyperlinks erstellen. Außerdem lassen sich auch in PDF spezifische Einstellungen vornehmen.

Das Paket `hyperref` wird, wie alle anderen Pakete auch, über `\usepackage{hyperref}` eingebunden. Das Paket sollte das letzte Paket in der Liste sein, das eingebunden wird, weil es sich auf die vorher eingebundenen Pakete einstellt und die Seite entsprechend anpasst.

Das Paket selbst steuern Sie über die Paketoptionen oder über den Befehl `\hypersetup`. Als Alternative können Sie die Einstellungen auch über die Konfigurationsdatei `hyperref.cfg` vornehmen.

Nachfolgend finden Sie eine Übersicht über die Parameter:

### Treiberoptionen

Das Paket `hyperref` umfasst, genau wie das Paket `graphicx`, eine Reihe von Treiberoptionen: `hypertex`, `dvips`, `dvipsone`, `ps2pdf`, `tex4ht`, `pdftex`, `dvipdf`, `dvipdfm`, `dviwindo`, `vtex` und `textures`.

### Allgemeine Optionen

Nachfolgend die allgemeinen Optionen des Pakets. Alle Optionen sind boolesche Parameter:

- `draft`  
Alle Hypertextoptionen sind ausgeschaltet (Standard: `false`).
- `final`  
Alle Hypertextoptionen sind angeschaltet (Standard: `true`).

- `debug`  
Es werden weitere Meldungen ausgegeben (Standard: `false`).
- `verbose`  
Ähnlich wie `debug`. Weitere Meldungen werden ausgegeben. (Standard: `false`).
- `implicit`  
Damit werden die internen Strukturen von LaTeX undefiniert (Standard: `true`).
- `hypertextnames`  
Es werden verständliche Namen für Links verwendet (Standard: `true`).
- `naturalnames`  
Es werden natürliche Namen für Links verwendet (Standard: `false`).
- `a4paper`  
DIN A4 wird als Standardformat verwendet. Die Option wird nur dann benutzt, wenn sie nicht schon als Klassenoption angegeben wurde (Standard: `true`).
- `a5paper`  
Es wird DIN A5 als Standardformat verwendet (Standard: `false`).
- `b5paper`  
Es wird DIN B5 verwendet (Standard: `false`).
- `letterpaper`  
Es wird US-Letter-Papier als Format verwendet (Standard: `false`).
- `executivepaper`  
Es wird US-Executive-Papier als Format verwendet (Standard: `false`).
- `setpagesize`  
Setzt die Papiergröße über spezielle Befehle (Standard: `true`).

### Konfigurationsparameter

Darüber hinaus gibt es noch einige Konfigurationsparameter. Alle angegebenen Parameter sind boolesche Parameter:

- `raiselinks`  
Alle Links, die über mehrere Zeilen gehen, werden automatisch umbrochen. Für jede Zeile wird dabei ein eigener Link erzeugt (Standard: `false`).
- `pageanchor`  
Legt fest, ob jede einzelne Seite einen Textanker in der linken oberen Ecke enthält. Wenn diese Option ausgeschaltet ist, enthält das `\tableofcontents`-Verzeichnis keine Hyperlinks (Standard: `true`).

- `plainpages`  
Formatiert die Textanker zu Beginn einer Seite mit einer Seitenzahl (Standard: `true`).
- `nesting`  
Erlaubt die Verschachtelung von Links. Wird jedoch bisher nicht unterstützt. (Standard: `false`).

**Tipp:** Weitere Parameter finden Sie in der ausführlichen Dokumentation unter [http://tug.ctan.org/tex\\_archive/macros/latex/contrib/hyperref/doc/manual.pdf](http://tug.ctan.org/tex_archive/macros/latex/contrib/hyperref/doc/manual.pdf)

### Befehle des Pakets `hyperref`

Das Paket `hyperref` bietet noch ein paar zusätzliche Befehle:

- `\href{url}{text}`  
Stellt den Text des Hyperlinks als URL dar.
- `\url{url}`  
Entspricht `\href{url}{url}`.
- `\nolinkurl{url}`  
Schreibt die URL als Standardtext, ohne einen Link zu erzeugen.

## 19.2 Das Paket *ifpdf* für pdfTeX

Mit dem Paket `ifpdf` können Sie LaTeX von Fall zu Fall ausführen. Das bedeutet, dass bestimmte Pakete nur dann eingebunden und gewisse Befehle nur dann ausgeführt werden, wenn das TeX-Dokument mit pdfLaTeX bearbeitet wird und PDF-Dokumente erzeugt werden sollen. Ist das der Fall, dann werden die Befehle befolgt, die unter `\ifpdf` stehen. Übernimmt ein anderes System als pdfLaTeX die Bearbeitung oder befindet sich pdfLaTeX im DVI-Modus, dann werden die unter `\else` gelisteten Anweisungen befolgt. Das Paket wird mit der Anweisung `\usepackage` eingebunden. Die Verzweigung wird mit dem Befehl `\fi` beendet. Anbei ein Beispiel für die Syntax:

```
\usepackage{ifpdf}

\ifpdf
\usepackage[...]{...}
\else
\usepackage[...]{...}
\fi
```

**Beispiel 19.1:** Fallweise Ausführung von LaTeX

## 19.3 In PDF-Dokumenten suchen

Mit dem Paket `cmap` werden PDF-Dokumente besser durchsuchbar, die mit `pdfLaTeX` erzeugt wurden. Dieses Paket wird wie gewohnt über `\usepackage{cmap}` eingebunden:

```
\usepackage{cmap}
```

## 19.4 Optischer Randausgleich und Zeichendehnung

Mit dem Paket `pdfcpot` aktivieren Sie den automatischen Randausgleich. Das Paket wird über die Anweisung `\usepackage[activate]{pdfcpot}` eingebunden.

Daneben gibt es auch das Paket `microtype`, mit dem Sie ebenfalls einen optischen Randausgleich erzeugen können. Das Paket `microtype` ist in der Version 2.3 verfügbar.

**Tipp:** Weitere Informationen zum Paket `microtype` finden Sie auf dem CTAN-Server unter

<http://ftp.gwdg.de/pub/ctan/macros/latex/contrib/microtype/microtype.pdf>

## 19.5 Einbinden von PDF-Seiten

Mit dem Paket `pdfpages` werden externe PDF-Seiten eingebunden. Dieses Paket funktioniert nur mit `pdflatex`.

Das Paket `pdfpages` liefert diverse Optionen und Befehle. Unten sehen Sie eine Auflistung der verschiedenen Optionen und Befehle:

- `final`  
Die PDF-Seiten werden eingefügt.
- `draft`  
Anstelle der Seiten werden nur Boxen in der entsprechenden Größe sowie die Dateinamen eingefügt.
- `\includepdf[<key val...>]{filename}`  
Bindet ein PDF-Dokument entsprechend der Einstellungen ein. Es stehen folgende Parameter zur Verfügung:

– `pages {...}`

Damit wird festgelegt, welche Seiten eingebunden werden sollen. Die Seiten werden dabei durch Kommata getrennt, z. B. `pages {1,8,10}`.

Mehrere Seiten lassen sich durch einen Bindestrich zwischen den Seitenangaben angeben: `pages {1 10,{},15,20 22}`. Eine leere geschweifte Klammer besagt, dass eine Leerseite an dieser Stelle eingebunden werden soll. Mit einem einfachen Minuszeichen ( `-` ) werden alle Seiten eingebunden.

– `nup <xnup>x<ynup>y`

Dieser Parameter fasst mehrere externe Seiten auf einer Seite zusammen. Es muss jeweils die Anzahl an Seiten in der x- und y-Richtung, also horizontal und vertikal, angegeben werden.

– `landscape`

Damit wird das Papierformat auf Querformat ausgerichtet.

– `delta x y`

Damit wird ein Leerraum zwischen den logischen x- und y-Seiten eingefügt, z. B. `delta 20 20`.

– `offset x y`

Der Ursprungspunkt für das Einbinden der Seiten wird um den entsprechenden x- und y-Wert, z. B. `offset 15 15`, verschoben.

– `frame`

Es wird ein Rahmen um die Seite eingefügt.

– `column`

Die Seiten werden spaltenweise angeordnet. Standardmäßig werden die Seiten zeilenweise angeordnet.

Hier ein Beispiel:

1	2	3
4	5	6
7		

Zeilenweise Anordnung

1	4	7
2	5	
3	6	

Spaltenweise Anordnung

– `columnstrict`

Hiermit wird festgelegt, dass für die letzte Seite ein striktes Spaltenlayout eingerichtet wird. Dabei wird zunächst die erste Spalte komplett aufgefüllt, bevor die neue Spalte beginnen kann:

1	4	
2	5	
3		

`Columnstrict true`

1	3	5
2	4	

`Columnstrict false`

– `openstrict`

Es wird eine leere logische Seite eingefügt. Damit lässt sich die erste logische Seite einer rechten Seite platzieren..

**Tipp:** Weitere Informationen zu dem Paket `pdfpages` finden Sie in der ausführlichen Dokumentation unter [www.ctan.org/macros/latex/contrib/pdfpages/pdfpages.pdf](http://www.ctan.org/macros/latex/contrib/pdfpages/pdfpages.pdf)

### Beispiel

Das Paket `pdfpages` wird über den Befehl `\usepackage` eingebunden. Dann können Sie im Hauptdokument den Befehl `\includepdf` wie folgt angeben:

```
\usepackage{pdfpages}
\begin{document}
\includepages[pages,landscape,frame]{pdfpage.pdf}
\end{document}
```

**Beispiel 19.2:** Das Paket `pdfpages` einbinden

## 19.6 In PDF zeichnen

Das Paket `pgf` erlaubt es Ihnen, in PDF-Dokumenten zu zeichnen und ähnelt somit `PStricks`. Zu diesem Paket gibt es noch einige Erweiterungspakete:

- `pgfarrows.sty`  
Es werden verschiedene Arten von Pfeilspitzen zur Verfügung gestellt.
- `pgfnodes.sty`  
Hiermit lassen sich verschiedene Knotentypen und Verbindungen zwischen ihnen herstellen.
- `pgfshade.sty`  
Hiermit sind verschiedene Formen von Schattenverläufen möglich.

### Beispiele

Das Paket `pgf` wird über den Befehl `\usepackage{pgf}` eingebunden. Dann können Sie über verschiedene Umgebungen und Befehle in `TEX`-Dokumenten zeichnen, die dann als PDF ausgegeben werden.

```
\documentclass[a4paper]{article}
\usepackage{pgf}
\begin{document}
\begin{pgfpicture}{0cm}{0cm}{10cm}{10cm}
\pgfrect[stroke]{\pgfpoint{0cm}{0cm}}{\pgfpoint{2cm}{10pt}}
% \pgfcircle[fill]{\pgfpoint{3cm}{1cm}{10pt}}
\end{pgfpicture}
\end{document}
```

#### Beispiel 19.3: Ein Rechteck in PDF

Das Ergebnis sieht so aus:



**Abbildung 19.1:** Beispiel für das Paket `pgf` – ein gezeichnetes Rechteck

**Tipp:** Weitere Informationen zu diesem Paket finden Sie auf dem CTAN-Server unter [http://tug.ctan.org/tex archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf](http://tug.ctan.org/tex_archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf)

## 19.7 Einbetten von Multimedia-Objekten

Über das Paket `media` lassen sich verschiedene Multimedia-Objekte wie Audio- und Videodateien in PDF-Dokumente einbetten. Das Multimedia-Objekt wird wie folgt integriert:

```
\includemovie[options]{width}{height}{media file}
```

Dabei müssen Sie die Höhe und Breite des Anzeigenbereichs fest vorgeben. Über den optionalen Parameter `mimetype` wird das Dateiformat definiert. Gemäß der PDF-Spezifikation werden folgende Formate von Acrobat Reader unterstützt:

<i>Format</i>	<i>Beschreibung</i>
aiff	Audio Interchange Format
au	NeXT/Sun Audio Format
avi	AVI (Audio/VideoInterleaved)
mid	MIDI (Musical Instrument Digital Interface)
mov	Quick Time
mp3	MPEG 3 Audio Layer
mp4	MPEG 4 Video
mp4	MPEG 4 Video
mpeg	MPEG 2 Video
smil	Synchronized Multimedia Integration Language
swf	Adobe Flash

**Tabelle 19.1:** Mediaformate

## 19.8 Wasserzeichen

Mit dem Paket `pdfdraftcopy` lassen sich Wasserzeichen bzw. Hintergrundbilder einfügen. Es funktioniert allerdings nur mit `pdfLaTeX`.

Das Paket `pdfdraftcopy` hat zwei Parameter:

- `draft`  
Es wird ein Text als Hintergrund verwendet.
- `watermark`  
Dadurch wird ein Wasserzeichen festgelegt.
- `\draftstring`  
Damit wird der Text für den Hintergrund festgelegt.

z. B. `\draftstring{Beispiel}`

- `\draftfontsize`  
Damit wird die Schriftgröße festgelegt, z. B. `\draftfontsize{80}`
- `\draftfontfamily`  
Damit wird die Schrift festgelegt, z. B. `\draftfontfamily{pcr}`
- `\draftangle`  
Damit wird der Drehwinkel des Textes festgelegt, z. B. `\draftangle{50}`
- `\draftcolor`  
Damit wird die Farbe festgelegt, z. B. `\draftcolor{blue}`
- `\draftfontattrib`  
Damit wird das Schriftattribut festgelegt.
- `\watermarkgraphic`  
Damit wird das Hintergrundbild festgelegt, z. B. `watermarkgraphic{beispiel}`.

Die Größe lässt sich entsprechend skalieren.

## 19.9 PDF-Präsentationen

Mit dem Paket `beamer` lassen sich einfache kurze Präsentationen auf PDF-Basis erzeugen. Die aktuellste Version ist die Version 3.07. Es kann unter <http://sourceforge.net/projects/latex-beamer/files/> geladen werden.

**Tipp:** Weitere Informationen zu dem Paket gibt es unter  
<http://ftp.gwdg.de/pub/ctan/macros/latex/contrib/beamer/doc/beameruserguide.pdf>



## 20 Anhang

Dieses Kapitel beschäftigt sich mit dem internen Aufbau und der Installation von LaTeX auf verschiedenen Betriebssystemen. Die innere Struktur kann sich von Variante zu Variante unterscheiden.

### 20.1 Installation von LaTeX

LaTeX gibt es für die unterschiedlichsten Betriebssysteme. In diesem Buch wurde die Distribution MikTeX unter Windows benutzt. Für Linux-Benutzer ist die Distribution TeX Live interessant (die auch für Windows verfügbar ist) und für Mac die Distribution MacTex.

#### Installation von LaTeX unter Windows

Die einfachste Distribution unter Windows ist MikTeX (<http://miktex.org/>). Diese gibt es in zwei Varianten: zum einen die Basic-Installation, die etwa 82 Megabyte umfasst, die gängigsten Pakete und Klassen enthält und einfach per Setup-Datei installiert wird.

Zum anderen gibt es die Komplettinstallation, deren Download nur knapp 3 Megabyte umfasst; während des Setup-Prozesses lädt das Programm jedoch den kompletten Funktionsumfang herunter. Das kann je nach System und Internetverbindung ein paar Stunden dauern. Nach dem Download lässt sich die Distribution über das Setup-Programm installieren.

Eine weitere einfache Distribution ist ProText, die auf MikTeX basiert und ebenfalls für Windows zur Verfügung steht. ProText ist eine Kombination aus MikTeX und verschiedenen Tools für Windows und umfasst rund 750 Megabyte. ProText ist unter der Webadresse <http://www.tug.org/protext/> erhältlich.

Daneben steht TeX Live für Windows zur Verfügung. Das Setup-Paket ist lediglich 4 Megabyte groß. Die Installation wird durch Doppelklick auf das Symbol der Batch-Datei `install tl.bat` angestoßen. Dadurch wird ein Skript gestartet, das dem Benutzer zahlreiche Auswahlmöglichkeiten bietet, für die er sich durch Anklicken entscheiden kann. Dann startet der Download der ausgewählten Optionen; auch dieser Vorgang kann mehrere Stunden dauern.

### Installation von LaTeX unter Linux

TeX Live ist eine populäre Distribution, die praktisch für alle Unix-Varianten sowie für Windows und Mac OS X existiert. Das Installationspaket `install tl.tar.gz` kann von der Website <http://www.tug.org/texlive/acquire.html> heruntergeladen werden. Mit knapp über einem Megabyte Umfang ist es sogar noch kompakter als sein Windows-Gegenstück. Um es zu installieren, müssen sich Benutzer an der Unix-Shell zum Administrator ernennen (über den Befehl `sudo su` und die Eingabe des Passworts). Ist das Installationspaket entpackt, müssen die Benutzer die Installationsroutine über `./install tl` starten. Auch hier startet ein Skript, das den Anwendern zahlreiche Auswahlmöglichkeiten anbietet. Der Download des umfangreichen Pakets kann auch hier einige Stunden dauern.

In gewisser Weise hat TeX Live unter Linux das Erbe der früher sehr beliebten *tetex*-Distribution angetreten, die nicht mehr weiterentwickelt wird. In älteren Versionen von SuSe Linux ist *tetex* allerdings noch enthalten. Diese können Sie über das Programm *Yast* als Paket installieren. Rufen Sie dazu das Programm *Yast* über das Startmenü – unter SuSe Linux der grüne Startknopf – auf. Gehen Sie dann auf *Software installieren* und wählen Sie die entsprechenden Pakete aus. Das Paket *tetex* finden Sie unter `productivity=>TeX`.

### Installation von LaTeX unter Mac OS X

Für den Mac ist MacTeX, das auf TeX Live basiert, die derzeit wichtigste Distribution. Sie finden diese Variante im Netz unter <http://www.tug.org/mactex/>. Das Installationspaket `MacTeX.mpkg.zip` umfasst 1,2 Gigabyte; nach dem Download genügt ein Doppelklick, um die Installation zu starten.

### Weitere Systeme

Wenn Sie ein anderes, z. B. unixbasiertes, System haben, können Sie auf die TeXLive-Version für Linux zurückgreifen und die Installationsdatei `install tl` in der Shell ausführen. Voraussetzung ist eine funktionierende Internetverbindung.

## 20.2 Der Aufbau von LaTeX

Jede LaTeX-Distribution speichert die unterschiedlichen Dateien wie Konfigurationsdateien, Fonts usw. in einer bestimmten Struktur. Unter Linux wird z. B. eine Struktur namens `texmf` erzeugt. MikTeX unter Windows verwendet eine davon abweichende Struktur.

### Paketinformationen

Pakete werden unter Windows als CAB-Dateien vertrieben. Informationen zu Paketen finden Sie unter MikTeX entweder über das Programm *Browse Packages* oder im Ordner `tex>latex`.

Unter Linux liegen diese Ordner im `texmf`-Baum im Verzeichnis `tex>latex`. Alle weiteren Angaben zur Struktur von LaTeX gelten für den `texmf`-Baum analog. Der `texmf`-Baum wird unter Linux im Verzeichnis `/usr/share/texmf` bzw. unter `/usr/local/share/texmf` angelegt.

### Dokumentation

Die Dokumentation finden Sie jeweils im Ordner `D00`. Dort finden Sie dann meistens PDF-Dateien, die Informationen zu einzelnen Paketen enthalten.

### Sonstige Ordner

Ein weiterer wichtiger Ordner ist der Ordner `Font`, in dem die Schriftarten selbst sowie die dazugehörigen Informationen untergebracht sind.

### Dateistruktur unter MikTeX

Wenn Sie z. B. neue Pakete installieren, können Sie über die MikTeX-Optionen die Dateistruktur neu anpassen. Anbei sehen Sie einen Screenshot des Programms MikTeX Options, das Sie über `Start >Programme >MikTeX >Settings` erreichen:

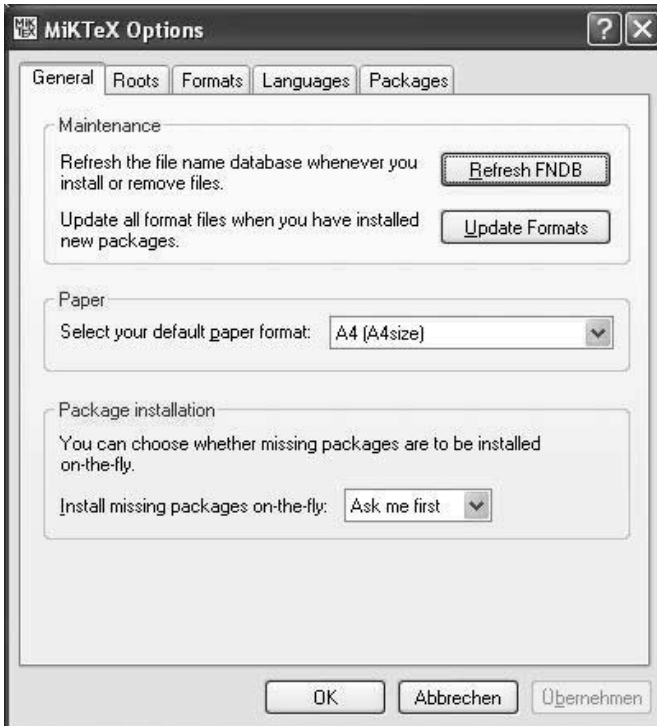


Abbildung 20.1: MikTeX Options

### 20.2.1 Pakete installieren

LaTeX-Pakete enthalten eine Fülle von Dateien – unter anderem gehören dazu Style-Dateien, Konfigurationsdateien und Ähnliches. Die Style-Dateien sind in DTX-Dateien gepackt. Die Steuerinformationen eines LaTeX-Pakets stehen in der INS-Datei.

Unter Windows werden Pakete automatisch über das Programm `Browse Packages` der Distribution MikTeX installiert. Unten sehen Sie einen Screenshot dieses Programms. Das Programm `Browse Packages` dient als Paketmanager von MikTeX:

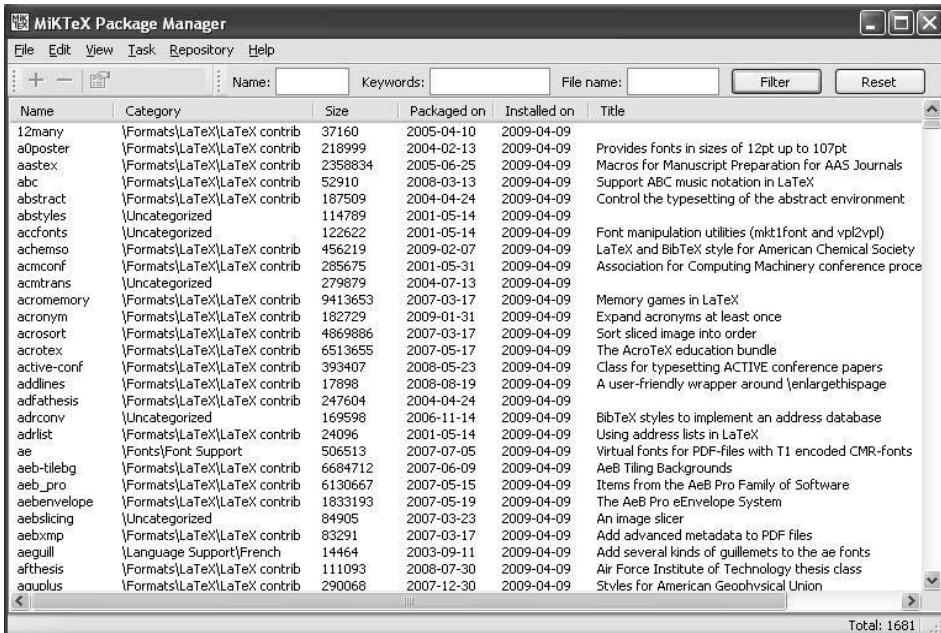


Abbildung 20.2: Das Programm Browse Packages

Unter Linux müssen neue Pakete in das jeweilige `texmf`-Verzeichnis kopiert werden. Danach muss der `texmf`-Baum aktualisiert werden. Unter Linux ruft man dazu das Programm `mktxlsr` auf.

### 20.2.2 Paketdokumentationen anzeigen lassen

Mit dem Hilfsprogramm `texdoc` lassen Sie sich Paketdokumentationen anzeigen. Dazu ruft das Programm den entsprechenden Viewer, z. B. Adobe Acrobat oder Yap, auf. Folgende Formate werden unterstützt: `.dvi`, `.pdf`, `.html`, `.ps` und `.txt`.

Das Programm wird so aufgerufen:

```
texdoc <name>
```

Das Programm ruft dann automatisch den passenden Viewer auf und zeigt die Dokumentation an.

## 20.3 Paket-Updates

Auch LaTeX entwickelt sich immer weiter. Viele der in diesem Buch vorgestellten Pakete wurden auf einen neuen Stand gebracht. Wenn es eine neue Version eines Pakets gibt, so wird dies vor Ort im jeweiligen Kapitel erwähnt.

Am Basissystem von LaTeX und seinem grundlegenden Aufbau hat sich allerdings seit der letzten Aktualisierung dieses Buches nichts Grundsätzliches verändert. Weitere Informationen zu Paket-Updates finden Sie in den jeweiligen Kapiteln bzw. auf der entsprechenden Paket-Homepage auf dem CTAN-Server.

**Tipp:** Unter [http://tug.ctan.org/pub/tex\\_archive/info/l2tabu/german/l2tabu.pdf](http://tug.ctan.org/pub/tex_archive/info/l2tabu/german/l2tabu.pdf) gibt es eine kleine Liste von Dingen, die man bei LaTeX nicht (mehr) tun sollte.

## 20.4 Dateiformate von LaTeX

LaTeX erzeugt bei jedem Compilerlauf Dateien, die unterschiedliche Funktionen und Namen haben. So gibt es z. B. Konfigurationsdateien, Style-Dateien, TOC-Dateien und noch einige mehr.

### 20.4.1 Von LaTeX erzeugte Dateien

Unten sehen Sie eine Liste von Dateien, die von LaTeX bzw. dessen Hilfsprogrammen erzeugt werden:

- `.aux`  
Datei, die bei jedem LaTeX-Lauf erzeugt wird. In dieser Datei stehen Informationen über die Kapitelgliederung, Labels und Referenzen.
- `.bbl`  
In dieser Datei steht nach einem BibTeX-Lauf die fertige Bibliografie.
- `.dvi`  
Die fertige DVI-Ausgabedatei.
- `.idx`  
Enthält die Indexeinträge, die von dem Programm `makeindex` sortiert werden können.
- `.ilg`  
Hier stehen die Fehlermeldungen von LaTeX.

- `.ind`  
Darin stehen nach einem Lauf von `makeindex` die sortierten Indexeinträge.
- `.lof`  
Enthält die Einträge des Abbildungsverzeichnisses.
- `.log`  
Enthält die Logdatei von LaTeX.
- `.lot`  
Enthält die Einträge des Tabellenverzeichnisses.
- `.tex`  
Die eigentliche TeX-Datei.
- `.toc`  
Enthält die Inhaltsverzeichnis-Einträge.

## 20.4.2 Ein- und Ausgabe mit LaTeX

LaTeX kann während eines Compilerlaufs Meldungen ausgeben bzw. Parameter über die Tastatur einlesen.

Mit dem Befehl `\typeout` lassen sich Rückmeldungen von Befehlen auf der Konsole ausgeben. Wenn Sie z. B. einen neuen Befehl definiert haben und dessen Statusmeldungen ausgeben möchten, geschieht dies über `\typeout`.

Der allgemeine Aufruf von `\typeout` lautet:

```
\typeout <text>
```

Das Gegenstück zu `\typeout` bildet `\typein`. Damit können Sie Tastatureingaben von der Tastatur einlesen. Der allgemeine Aufruf lautet:

```
\typein[<Befehl>][<text>]
```

Dabei wird zuerst der Text ausgegeben und dann der eingegebene Text im Befehl gespeichert. Der Text kann dann beliebig verwendet werden.

Im nachfolgenden Beispiel sehen Sie eine Verwendungsmöglichkeit von `\typein`. Der Befehl wird dazu benutzt, die Eingabe des Fonts und dessen Parameter über die Tastatur zu ermöglichen.

```
\typeout{*****}
\typeout{* Bitte zu verwendenden Font eingeben *}
\typeout{*****}
\typein[\fFamilie]{Font Familie (put):}
\typein[\fKode]{Kodierung (T1):}
```

```
\typein[\fSerie]{Font Serie (m):}
\typein[\fFormat]{Font Format (n):}
\typein[\fSize]{Font Size (12):}
```

**Beispiel 20.1:** Ein- und Ausgabe in LaTeX

### 20.4.3 Weitere Informationen

Im Internet finden Sie auf verschiedenen Seiten Informationen zu LaTeX. Neben Internetseiten gibt es auch Mailinglisten und Newsgroups, in denen Fragen zu LaTeX beantwortet werden.

#### Internetseiten

Die erste Anlaufstelle im deutschsprachigen Raum ist mit Sicherheit die Seite der deutschsprachigen Anwendervereinigung TeX e.V. (DANTE), [www.dante.de](http://www.dante.de). Auf dem Dante-Server finden Sie einen ausführlichen FAQ-Bereich, auf dem hilfreiche Informationen bereitstehen.

Eine weitere interessante Seite ist der Server des CTAN (Comprehensive TeX Archive Network), den Sie über [www.ctan.org](http://www.ctan.org) erreichen. Auf CTAN finden Sie immer aktuelle Informationen zu den existierenden Paketen.

#### Newsgroups

Eine interessante Newsgroup findet sich in den Google Groups unter <http://groups.google.de/group/de.comp.text.tex/topics>. Dort werden verschiedenste Fragen zu LaTeX und TeX beantwortet.

#### Mailinglisten

Auf dem Dante-Server finden Sie einige deutschsprachige Mailinglisten, auf denen Fragen zu LaTeX beantwortet werden. Auf der Seite <http://cms.dante.de/dante/hilfe/Mailinglisten.html> steht eine Anleitung, wie man zum Administrator einer Mailingliste Kontakt aufnimmt.

# Stichwortverzeichnis

## Symbole

.pdf 30  
.tex 30

## A

Absatzbox 89  
Absätze 78  
abstract 285  
addcontentline 296  
addcontents 296  
addcontentsline 289  
addtocontentsline 289  
addtocounter 268  
addtolength 268  
allowdisplaybreak 245  
alltt 213  
AMS Latex 241  
amsmath 241, 247  
array 104  
Artikel 17  
Aufzählungen 45, 189  
Ausgabetrener 109  
Außensteg 21  
author 280  
Autorotatepages 338  
Axiome 251

## B

Beam 328  
Bemerkungen 251  
Bézier Kurve 160, 178  
Bezugspunkte 49

Bib2x 301  
BibDesk 301  
Biblatex 301  
Bibliographie 63  
BibShare 301  
BibTeX 298  
BibTeXML 301  
BibTool 301  
Bilder 37, 42, 44, 125  
Bildformat 125  
Binomialkoeffizienten 242  
Bitmap 38  
booktabs 106  
Bounding Box 45, 125  
Box 85, 125  
Boxen-Modell 76  
Briefe 17  
Browse Packages 365  
Bücher 17  
Buchstabenfolgen 68  
Bundsteg 21

## C

calc 203, 268  
ccurve 181  
chapter 288  
chktex 263  
circle 155  
circle\* 155  
clearpage 280  
Codierung 38  
Codierungsattribut 38

comment 214  
 Compiler 23  
 contentline 287  
 curve 181

## D

dashboard 156, 157  
 Dateiendung 130  
 Daumenindex 311  
 dblup 351  
 dcolumn 109  
 Definitionen 251  
 DeleteShortVerb 214  
 description 193  
   Umgebung 46  
 descrlist 208  
 det 249  
 Dezimaltrenner 109  
 ding 272, 324  
 dingautolist 324  
 displaybreak 245  
 displaymath 223  
   Umgebung 48  
 Distributionen 26  
 docbook 351  
 Dokumente 17  
 dominitoc 295  
 dotstyle 180  
 dottedtocline 291  
 draftangle 361  
 draftcolor 361  
 draftfontfamily 361  
 draftfontsize 361  
 draftstring 361  
 dvi 30  
 dvipdf 126, 353  
 dvipdfm 126, 353  
 dvips 126, 353  
 dvipsone 353  
 dviwndo 353

## E

ecurve 181  
 Editoren 19  
 Eingabetrenner 109  
 Einzüge 78  
 Engine 27  
 enlargethispage 59  
 Entwurfsmodus 64  
 enumerate 191, 208  
 enumerate-Umgebung 45  
 epstopdf 337  
 eqnarray 244  
 eqnarray\* 244  
 eqref 246  
 equation 223  
 expdlist 207  
 Expertensystem 29  
 explist 204  
 Exponenten 235

## F

Familien 38  
 fancyhdr 311  
 fancyvrb 217  
 Farbe 90  
 Farbmodelle 91  
 Fehler 39, 255  
 Flatterrand 70  
 fltpoint 110  
 Folgefehler 39  
 fontfamily 319  
 fontshape 319  
 fontsize 319  
 footmisc 312  
 Formattribut 320  
 Formeln 47  
 framebox 156  
 Fremdsprachen 84  
 Fußnoten 73  
 Fußsteg 21

Fußzeilen 309

## G

gcd 249  
 geometry 61  
 german 62  
 Gleitende Objekte 121  
 Gliederungstext 291  
 Glossar 308  
 Grafiken 153  
 graphicx 44  
 graphpap 168  
 graphpaper 168  
 Größenattribut 38, 319  
 gruende 315

## H

Harmonie 21  
 helvet 322  
 Hilfsprogramme 126  
 Hintergrundbilder 135  
 Horizontale Linie 102  
 Horizontaler Abstand 43  
 html2latex 350  
 hyperref 312, 353  
 hypersetup 353  
 hypertex 353

## I

ifpdf 355  
 ifthen 203  
 ifthenelse 269  
 include 280  
 include-Dateien 30  
 includeonly 280  
 includepdf 358  
 index 302  
 Index 301  
 indexentry 302  
 Indizes 235

inf 249  
 Inhaltsverzeichnisse 63  
 input 279  
 Installation 18, 363  
 Integrale 237  
 item 302  
 itemindent 204  
 itemize 189  
 itemize-Umgebung 45

## J

jurabook 313  
 juramisc 313  
 juraovw 315  
 juraurl 314

## K

Kapitälchen 69  
 Knuth, Donald 17  
 keyval 273  
 Kommandozeilenprogramme 19  
 Kommentar 33  
 Kompiliert 29  
 Konfigurationsdateien 125  
 Konstanten 227  
 Konvertierer 126  
 Koordinatensystem 49  
 Kopfsteg 21  
 Kopfzeile 309  
 Korollare 251  
 Kreise 153  
 Kurve 160

## L

label 192  
 label Parameter 46  
 labelwidth 204  
 Lamport, Leslie 17  
 Längen 265  
 Längeneinheiten 49

lascheck 262  
 LaTeX Compilers 23  
 LaTeX-Pakete 30  
 latexsym 325  
 left 241  
 leftmargin 204  
 Lemmata 251  
 leqno 249  
 Letter 328  
 liftoftables 296  
 Ligaturen 68  
 lim 249  
 liminf 249  
 limsup 249  
 line 154, 181  
 Linien 153  
 Linienformen 116  
 Linienstärke 161  
 Linienstile 116  
 list 199, 202  
 list-Umgebung 46  
 Listen 45, 189  
 listings 219  
 listoffigures 296  
 listparindent 204  
 Literaturverzeichnis 296  
 lmodern 323  
 LR Modus 37  
 lstlisting 220  
 LuaTex 27

## M

makebox 156, 160  
 makedx 302  
 makeindex 302  
 makelabel 200  
 MakeShortVerb 214  
 maketitle 281  
 manueller Seitenumbruch 100  
 manyfoot 313

marvosym 327  
 Maßeinheiten 43, 265  
 math 222  
 math-Umgebung 47  
 Mathematikmodus 37, 109, 221  
 mathptmx 322  
 Matrix 241  
 max 249  
 mdwlist 206  
 media 360  
 Mehrfachintegrale 238  
 Metafont 320  
 Metriken 321  
 microtype 356  
 MikTeX 363  
 mimetype 360  
 min 249  
 minipage 246  
 minitoc 294  
 mktexlsr 367  
 Modi 37  
 moreverb 216  
 mspatho 322  
 multiline 251  
 multilinegap 251  
 multiput 163  
 Musterzeile 98  
 myparbox 274

## N

newcommand 199  
 newcounter 267  
 newenvironment 202  
 newindex 308  
 newsavebox 167  
 ngerman 62  
 nicht-proportionale Schriften 64  
 nonumber 244  
 Normseite 94  
 Normzeile 94

numberline 291  
 numberwithin 245  
 Nummerierung 73

## O

OpenOffice 21, 343  
 Operatoren 231  
 oval 158  
 overleftarrow 240  
 overleftrightharpoon 240  
 overrightarrow 240

## P

pagestyle 309  
 Papierformate 57  
 Papiergröße 41  
 Paragraphen-Modus 37  
 paralist 205, 207  
 parbox 164, 273  
 parsep 204  
 part 288  
 PDF 24  
 pdfcprot 356  
 pdfcrop 338  
 pdfdraftcopy 360  
 pdfgpages 356  
 pdflatex 337  
 pdfpages 358  
 pdfpic 340  
 pdftex 126, 353  
 pdftricks 340  
 Personenverzeichnisse 307  
 pgf 358  
 Pickup 328  
 picture-Umgebung 49  
 pifont 322  
 Pixelraster 320  
 polygon 181  
 Polygonkurven 178  
 postscript 337

PostScript Tricks 24, 168  
 Pr 249  
 Präambel 30  
 preview 339  
 printindex 302  
 proportional 64  
 protect 289  
 providelcommand 271  
 ps 30  
 ps2pdf 337, 353  
 pscurve 181  
 pscurve 181  
 psdot 180  
 psdots 180  
 pscurve 181  
 psframe 176  
 psgrid 176, 181  
 psnfss 322  
 pspolygon 176, 181  
 pstricks 49, 168

## Q

qline 176  
 Quellcode 29  
 Querverweis 83

## R

raisetag 244  
 rccol 110  
 rechnerunabhängig 17  
 Rechtecke 156  
 ref 193, 267  
 refcounter 267  
 relsize 67  
 remreset 314  
 renewcommand 195  
 reqno 249  
 resetindent 313  
 right 241  
 rightharpoon 203

rmfamily 320  
rotating 113

## S

Sachregister 63  
Sätze 251  
Satzspiegel 21  
Satzspiegelkonstruktion 21  
Satztechnik 17  
Satztextsysteme 17  
savebox 167  
Schriftbild 64  
Schriftform 64, 319  
Schriftgröße 64  
Schriftsatzsystem 17  
Schriftstärke 64  
section 288  
see 306  
Seitenkopf 309  
Seitennummern 310  
Seitenumbruch 77  
Serienattribut 319  
Serifenlose Schrift 64  
Serifenschrift 64  
setcounter 268  
setkeys 273  
setlength 58, 268  
setspace 312  
settheme 315  
shoidx 307  
shortstack 165  
shortverb 214  
SI Einheiten 95  
Silbentrennung 63, 77  
Soul 69  
Spalten 43  
Spaltenform 101  
Spaltentypen 105, 106  
Spaltenzuordnung 109  
split 248, 251  
splitidx 307  
splitindex 307, 308  
Spracheigenschaften 62  
stackrel 241  
Standardcodierung 319  
Standarddezimaltrenner 110  
Standardseiten 94  
Standardserie 319  
Standardumgebungen 104  
Stege 21  
stepcounter 267  
Steuerdatei 321  
Stichwortverzeichnisse 307  
subitem 302  
subsubitem 302  
Summen 237  
sup 249  
sverb 216  
symbol 323  
Symbole 85, 229, 233

## T

tabbing 42, 97  
tabbing-Umgebung 97  
Tabellen 42  
Tabellenfunktionen 97  
Tabellenstil 103  
Tabellenumgebung 43, 100  
tableofcontents 286  
tabular-Umgebung 100  
Tabulator 42, 97  
Tabulatorstops 97  
tabwidth 216  
Teildokument 279  
tetex 26  
tex4ht 353  
texdoc 367  
TEXINPUTS 132  
textacutedbl 327  
textasciibreve 327

textbf 307  
 textcomp 327  
 textit 307  
 textlarger 67  
 textrm 320  
 textsmaller 67  
 Textumflossene Objekte 150  
 textures 353  
 Textverarbeitungssysteme 21  
 thebibliography 296  
 theequation 245  
 theindex 302  
 thicklines 161  
 thinlines 161  
 thispagestyle 309  
 title 284  
 titlepage 281, 284  
 titlesec 284  
 tocdepth 288  
 tocindent 290  
 tocleft 290  
 tocloft 296  
 Tools 19, 301  
 Treiber 126  
 trivlist 204  
 turn-Umgebung 113  
 typearea 58  
 typein 369  
 typeout 369

## U

Umgebungen 37  
 Umgebungsvariablen 132  
 Umlaute 46  
 underleftarrow 240  
 underline 307  
 underrightarrow 240  
 usebox 167

## V

Variablen 227, 229, 233  
 vebbdef 215  
 vector 154  
 Vektorform 321  
 verb 212  
 verbatim 211  
 verbatiminput 215  
 verbatimtab 216  
 verbwrite 217  
 vertikale Position 101  
 Viewer 26, 29  
 vtx 353

## W

Wasserzeichen 360  
 wasysm 326  
 whiledo 271  
 widthof 203  
 Word 21, 343  
 writer2latex 343

## X

XeTeX 27  
 xindy 308  
 xkeyval 273  
 xleftarrow 240  
 xrightarrow 240  
 xspace 275

## Z

Zähler 267  
 Zapf Dingbats 324  
 Zeichensätze 38, 319  
 Zeilenabstand 79  
 Zeilennummern 94  
 Zeilenumbruch 76

# LaTeX

## Das Praxisbuch

*Keine Angst vor LaTeX! Das freie Textsatzsystem ist erste Wahl, wenn es um Diplomarbeiten und Dissertationen sowie um wissenschaftliche Fachartikel und Bücher geht. Autor Alexander Schunk macht es Ihnen leicht, sich in LaTeX einzuarbeiten, und führt Sie zielsicher von einfachen Dokumenten zu perfekt gesetzten Abschlussarbeiten und Büchern. Aber auch Alltagstexte wie Geschäftsbriefe kommen in diesem Buch nicht zu kurz.*

### ► LaTeX im Griff

Die Bedienung von LaTeX gilt als komplex, aber schon nach kurzer Zeit gelangen Ihnen überzeugende Dokumente. Ob einfacher Text, Tabellen, Grafiken, Register oder Verzeichnisse – dieses Buch demonstriert die komplette Bandbreite der Gestaltungsmöglichkeiten in LaTeX und zeigt Ihnen, wie Sie diese effektiv einsetzen.

### ► Eleganter Formelsatz

Vor allem für wissenschaftlich-technische Dokumente wie Master- und Diplomarbeiten sowie Dissertationen ist LaTeX ideal. Autor Alexander Schunk führt Sie in den mathematischen Formelsatz ein und zeigt, wie Sie mathematische Ausdrücke nicht nur fachlich korrekt, sondern auch elegant setzen. Darüber hinaus erfahren Sie, wie Sie Schaubilder, Stichwortverzeichnisse und Bibliografien erstellen, Programm Listings in Ihre Publikation einbinden und Beiträge anderer Autoren in Ihre Veröffentlichung integrieren.

### ► Print, PostScript oder PDF – alles ist möglich

LaTeX ermöglicht Ihnen größtmögliche Flexibilität für Ihre Publikationen: Egal, ob Sie Ihr Dokument einfach nur ausdrucken wollen, mit PostScript arbeiten oder das Dokument als PDF online publizieren – es gibt für alles eine Lösung. Sie erfahren zudem, wie Sie Texte aus Fremdformaten wie OpenOffice und Rich Text Format (RTF) konvertieren, damit Sie sie problemlos in LaTeX verwenden können.

### ► Fehler erkennen und vermeiden

Da LaTeX im Kern eine Programmiersprache ist, können Fehler und unbeabsichtigte Effekte auftreten. Autor Alexander Schunk gibt Ihnen wertvolle und zeitsparende Tipps, wie Sie Probleme vermeiden, und stellt Werkzeuge vor, die Ihnen bei der Fehlerbeseitigung helfen.

## Aus dem Inhalt:

- Erste Schritte mit LaTeX
- Die grundlegenden Befehle
- Einfache Texte formatieren
- PDF- und PostScript-Dokumente erstellen
- Dokumente mit Tabellen und Bildern verfassen
- Aufzählungen, Listen und Inhaltsverzeichnisse realisieren
- Mathematischer Formelsatz
- Bücher und wissenschaftliche Dokumente erstellen
- Grafiken und Zeichnungen mit LaTeX
- Fehler finden und vermeiden
- Teildokumente einbinden
- Verweise und Register
- LaTeX-Befehle modifizieren und eigene Kommandos definieren
- Fremdformate in LaTeX umwandeln

## Über den Autor:

Alexander Schunk studierte Informatik, Englisch und Französisch. Er ist als IT-Trainer sowie als Fachübersetzer im Bereich der Software-Lokalisierung tätig. Zudem verfasste er Fachbücher zu den Themen Java und C++. Er lebt in der Nähe von Köln.

## Auf [www.buch.cd](http://www.buch.cd):

- MiKTeX 2.7 (LaTeX-Basispaket für Windows-Rechner)
- TeXMaker (grafische LaTeX-Umgebung für Windows, Mac OS und Linux)
- Alle Beispieldateien



9 783772 367304

30, EUR [D]

ISBN 978-3-7723-6730-4

Besuchen Sie unsere Website  
[www.franzis.de](http://www.franzis.de)