Einführung in LaTeX 2ε

mit Berücksichtigung von KOMA-Script

16.–20. März 2015

von Marei Peischl: TEX@mareipeischl.de
nach der Vorgängerversion von Florian Rödl

Quelle der Bilddatei: https://www.ctan.org/lion/files

Inhaltsverzeichnis

Ergänzende Literatur

Die folgenden Literaturempfehlungen knüpfen an die Inhalte dieses Kurses an und behandeln einen Großteil des Stoffes vertiefend. Ein wenig praktische Übung im Vorfeld zur Lektüre (z. B. mit diesem Kurs) ist sehr hilfreich.

1. Herbert Voß. *Einführung in LaTeX 2ε. Unter Berücksichtigung von pdfLaTeX, XeLaTeX und LuaLaTeX*. 1. Aufl. Lehmanns Media, 2012. ISBN: 978-3-86541-462-5
   In der Lehrbuchsammlung unter 17/ST 351 L35 V969 zu finden.
2. Helmut Kopka. *Einführung*. 1. Aufl., [unveränd. Nachdr.] Bd. 1. LaTeX. Bonn [u.a.]: Addison-Wesley, 1994. XIX, 428 S. ISBN: 3-89319-664-1
   Bestellbar aus dem Magazin oder von der OTH Regensburg.
3. Frank Mittelbach und Michel Goossens. *Der LaTeX-Begleiter*. 2., überarb. und erw. Aufl., korr. Nachdr. München [u.a.]: Pearson Studium, 2007. XXIX, 1137 S. ISBN: 386894088X
   Das ultimative Nachschlagewerk zu LaTeX, bestellbar von der OTH Regensburg.

Darüber hinaus findet sich sehr viel Hilfe und Dokumentation im Internet:

1. Comprehensive TeX Archive Network: `http://ctan.org`
   (Anleitungen zu sämtlichen Ergänzungspaketen, die in TeX Live integriert sind, sowie weitere Pakete und Lösungen)
2. Deutsche TeX-FAQ der Dante e. V.: `http://projekte.dante.de/DanteFAQ`
   (Hinweise zur Typografie sowie sehr viele ergänzende Hinweise für Anwender)

0 Was ist LaTeX?

LaTeX ist eine Sammlung von Makros für das Textsatzprogramm TeX, welches für den Satz qualitativ hochwertiger Dokumente verwendet wird. Die meiste Verbreitung findet es im technischen oder wissenschaftlichen Bereich, allerdings ist es zum Satz sämtlicher Textdokumente bestens geeignet.

0.1 Word vs. LaTeX – Die Philosopie

Textverarbeitungsprogramme gliedern sich in zwei verschiedene Sparten:

- Wortprozessoren (bestes Beispiel: MS Word): Eine Formatierungseigenschaft des Textes wird ausgewählt und der formatierte Text erscheint in eben dieser Weise auf dem Bildschirm.
- Satzprogramme wie TeX: Man bearbeitet einen Quelltext, welcher sich aus Formatierungsbefehlen und dem eigentlichen Text zusammensetzt. Dieser wird anschließend durch das Satzprogramm übersetzt, wobei das Dokument als Ausgabefile erstellt wird.

Beide Konzepte haben ihre eigenen Vorteile und die Wahl der Software liegt letzten Endes immer beim Anwender. Der wesentlichste Unterschied jedoch liegt darin, dass bei einem Wortprozessor der Inhalt des Dokumentes eine Einheit mit der Formatierung bildet. LaTeX hingegen trennt diese beiden Teile voneinander. Somit ermöglicht es dem Autor wissenschaftlicher Veröffentlichungen sowohl den inhaltlichen Ansprüchen gerecht zu werden, als auch die Herausforderung zu meistern das Dokument angenehm und ermüdungsfrei lesbar zu machen.

Zudem eröffnet LaTeX die Möglichkeit, ein Dokument relativ einfach an ein vorgefertigtes Layout anzupassen, so ist es beispielsweise möglich ein und demselben Dokument lediglich durch das Einbinden einer entsprechenden Formatvorlage zwei vollkommen unterschiedliche Layouts zu verpassen. Die Philisophie von LaTeX beruht darauf dem Text eine logische Struktur zu geben und diese von den visuellen Konzepten zu trennen. So werden beispielsweise *Hervorhebungen* durch das Makro gemacht. Normalerweise setzt es den Text kursiv, jedoch kann man mit einer kurzen Zeile Code das Dokument insofern verändern, dass sämtliche Hervorhebungen fett gedruckt werden. Diese Flexibilität macht LaTeX so unschlagbar praktisch.

LaTeX liefert dem Benutzer einige Basislayouts für die geläufigsten Textformen (Berichte, Bücher, Artikel usw.), welche bereits sämtliche Ansprüche für Text-Dokumente erfüllen. Außerdem sind kompliziertere Arbeiten, wie der Satz mathematischer Formalismen, die Positionierung von Bildern, sowie Formatierungsänderung, selbst für sehr lange Werke, durch die Benutzung eines Satzprogrammes mit weitaus weniger Aufwand zu realisieren, als dies für Wortprozessoren der Fall ist.

0.2 TeX, METAFONT, LaTeX 2_ε, . . .

TeX („Tau Epsilon Chi“, gesprochen Tech) und METAFONT wurden ab 1977 von Donald E. Knuth (Professor an der Stanford University) „zum Satz schöner Bücher und insbesondere Bücher, die viel Mathematik enthalten“[8] und der Erzeugung besonderer Zeichensätze entwickelt. Der direkte Umgang mit dem sogenannten plain TeX ist jedoch relativ kompliziert und erfordert einige Erfahrung.

Leslie Lamport hat aus diesem Grund in den 1980er Jahren das Programmpaket LaTeX (LaTeX für LamportTeX) entwickelt. Es greift mithilfe von Makros auf die ursprünglichen Deklarationen in TeX zurück und stellt somit eine benutzerfreundlichere Möglichkeit da, TeX zu benutzen.

Die Entwicklung der nächsten Jahrzehnte lief in viele unterschiedliche Richtungen, in deren Rahmen Programme wie NFSS, SLiTeX, $\mathcal{AMS}$-LaTeX und Weitere entstanden. Ab 1993 entwickelte das LaTeX3 Project Team die jetzt „aktuelle“ Version LaTeX 2_ε und schuf somit einen neuen Standard [16].

Heutzutage ist die am häufigsten verwendete Version pdfLaTeX, welches auf LaTeX 2_ε und andere Zusatzprogramme zurückgreift. Es erzeugt im Gegensatz zu Standard LaTeX keine DVI- sondern eine PDF-Ausgabe, die sich häufig als praktischer erweist. Andere Programme, die teilweise als aktueller angesehen werden, sind z. B. XeLaTeX und LuaLaTeX (implementiert zusätzlich die Programmiersprache Lua), welche sich jedoch in der Grundanwendungsweise und Dokumentenstruktur nicht wesentlich von pdfLaTeX unterscheiden.

Im Rahmen dieses Kurses werden wir uns somit auf pdfLaTeX beschränken, für Interessierte sei jedoch auf die Literaturangaben und -empfehlungen verwiesen.

Teil I

TEXnische Grundlagen

1 Die grundlegende Funktionsweise

1.1 Komponenten eines LaTeX-Systems

Um mit LaTeX arbeiten zu können braucht man (prinzipiell) nicht ein Programm, sondern *drei*:

Editor Hier wird der Text mit den zugehörigen Formatierungsbefehlen eingegeben. Es genügt ein einfacher Texteditor, z. B. Editor, Emacs, Notepad.

Allerdings ist es gerade für Anfänger oft einfacher einen speziellen TeX-Editor zu benutzen, da er unter Anderem auch bei der Fehlersuche hilft.
Beispiele für solche TeX-Editoren: TeXstudio, TeXshop, TeXmaker, Kile.

Distribution Sie enthält das eigentliche LaTeX-Programm. Es wandelt den Text unter Berücksichtigung der Formatierungsbefehle in ein druckbares Dokument, z. B. in eine PDF-Datei um.
Beispiele: LaTeX, pdfLaTeX, XeLaTeX, LuaLaTeX.

Diese Programme sind allerdings alleine genauso nutzlos wie ein `C`-Compiler ohne Bibliotheken. LaTeX benötigt hier verschiedene Makro-Pakete. Eine Distribution umfasst all dies, also das Satzprogramm, die Makro-Pakete, sonstige Bibliotheken und auch Zusatzprogramme.
Beispiele für diese Distributionen sind: TeX Live, MacTeX (TeX Live für Mac), sowie MikTeX (nur für Windows).
Wegen der Systemunabhängigkeit empfiehlt es sich hier TeX Live oder MacTeX zu verwenden, da sich die vorhandenen Pakete von Distribution zu Distribution unterscheiden.

Viewer/Reader Wird benötigt um das erzeugte Dokument anzusehen bzw. um es auszudrucken. Bei PDFs wäre das z. B. der Adobe Reader. Bei manchen Editoren, z. B. TeXstudio oder TeXmaker ist bereits ein PDF-Viewer integriert.

1.2 Vom Quellcode zum Dokument

Da beim TeXen nicht nur eine Datei entsteht (wie etwa bei Word), sollte man für jedes Dokument einen neuen Ordner anlegen. Anschließend öffnet man den Editor und erstellt die `.tex`-Datei. Der einfachste Quellcode hierfür lautet in etwa so:

```
\documentclass{scrartcl}
\begin{document}
Mein erstes Dokument!
\end{document}
```

Ist der Code gespeichert, übergibt man ihn an das Satzprogramm, welches den Quellcode interpretiert. Es ruft gegebenenfalls auch weitere externe Programme auf, die zum Beispiel zur Erstellung einer Bibliografie oder eines Indexes notwendig sind. Anschließend erstellt es die Ausgabe.

Selbst für den Fall, dass man ohne eine grafische Entwicklungsumgebung (GUI) für LaTeX arbeitet, ist das Vorgehen denkbar einfach. Für einfache Texte gelten folgende Schritte:

1. Texte im Editor schreiben und abspeichern: dokument.tex

2. Dokument übersetzen mit pdfLaTeX. In der Konsole ausführen: pdflatex dokument.tex
   Dabei wird das PDF-Dokument und einige zusätzliche Hilfsdateien erzeugt. Siehe auch Tabelle 1.1 für genauere Erläuterungen zu den Dateitypen.
3. Erneutes Übersetzen des Dokuments durch Ausführen von pdflatex document.tex
   Dies ist nötig, damit Verzeichnisse, wie das Inhaltsverzeichnis, erstellt und Querverweise gesetzt werden (siehe Abschnitt 3.7)
4. PDF-Dokument ansehen.

In der Praxis benutzt man in der Regel nicht mehr den Weg über die Kommandozeile, sondern arbeitet mit einem fortschrittlicheren Editor. Ein Grund dafür ist, dass bei diesem der Quellcode dank Syntaxhighlighting wesentlich übersichtlicher wird. Zudem wird das Speichern des Quellcodes, die Übergabe an das Satzprogramm und das anschließende Öffnen der PDF-Datei meist mit einem einzigen Tastendruck erledigt.

Je nach Komplexität des Dokumentes, werden beim Kompilieren noch andere Dateien erstellt. Tabelle 1.1 zeigt die Ursache und Bedeutung der Files mit der entsprechenden Dateiendung.

Tabelle 1.1: Bedeutung der Dateiendungen der wichtigsten Hilfsdateien [vgl. 23, S.13f.]

Endung	Erläuterung
`.tex`	LaTeX oder TeX Input-Format. Kann kompiliert werden.
`.sty`	LaTeX-Style-File: Ergänzungspaket. Kann mit `\usepackage` geladen werden, siehe auch Abschnitt 3.2.
`.cls`	LaTeX-Class-File. Dokumentenklasse. Kann mit `\documentclass` geladen werden, siehe auch Abschnitt 3.1.
`.dvi`	Device-Independent-File. Das Standard-Ausgabeformat von LaTeX. Der Inhalt kann mit dem in TeXlive enthaltenen Programm DVIOUT angesehen werden.
`.log`	Detaillierte Informationen über den letzten LaTeX-Durchlauf. Enthält Warnungen und Fehlermeldungen.
`.toc`	Speichert alle Abschnittsüberschriften für den Eintrag in das Inhaltsverzeichnis.
`.lof`	Analog `.toc` für Abbildungsverzeichnis.
`.lot`	Analog `.toc` für Tabellenverzeichnis.
`.aux`	Hilfsdatei, welche die zugehörigen Informationen zu Querverweisen enthält.

1.3 Die Struktur der Makros

Bei LaTeX gibt es jede Menge unterschiedliche Makros. Jedoch lassen sich sämtliche Formen auf drei grundlegende Typen zurückführen:

- Sonderzeichen; Die Zeichen `\`, `#`, `$`, `&`, `~`, `_`, `^`, `%`, `"`, `{` und `}` haben besondere Bedeutungen und fungieren direkt als Befehle, siehe Tabelle 1.2.
- Einzelne Sonderzeichen hinter einem Backslash, z. B. `\$` ($), siehe ebenfalls 1.2.
- Die klassischen Makros bestehend aus Backslash und einer Buchstabenfolge. Das erste folgende Sonderzeichen wird als Befehlsende interpretiert. Die genaue Struktur und Verwendung wird nun im Folgenden genauer erläutert:

  `\Befehl`

Dies ist die einfachste Form von Befehlen, sie werden mit \ eingeleitet und benötigen keine weiteren Argumente.
Ein Beispiel hierfür ist das LaTeX-Logo: `\LaTeX`→\LaTeX
Anmerkung: Sämtliche LaTeX Makros sind case sensitive, die Groß- und Kleinschreibung ist somit von Bedeutung, z. B. `\LaTeX` ≠ `\latex`

`\Befehl{`*Argument*`}`

Dies ist die Art von Befehl, die entweder verschiedene Einstellungsmöglichkeiten haben oder sich nur auf einen bestimmten Textabschnitt beziehen. Das Argument in geschweiften Klammern ist notwendig, da sonst der Befehl keinen Sinn macht. (Das Beispiel benötigt das Paket color oder xcolor.)

```
\color{gray}
Dieser Text ist grau.\\
Dieser Text ist schwarz.
```

Dieser Text ist grau.
Dieser Text ist schwarz.

Ohne eine Angabe der Farbe grau würde der Befehl `\color` keinen Sinn ergeben und nicht funktionieren. Außerdem erkennt man, dass der Befehl ein sogenannter Schalter ist. Er bewirkt, dass alles, was danach steht, grau geschrieben wird. Um dies einzuschränken, kann man wiederum Klammern setzen:

```
{\color{gray}
Dieser Text ist grau.}\\
Dieser Text ist schwarz.
```

Dieser Text ist grau.
Dieser Text ist schwarz.

Befehle wirken somit nur innerhalb des Bereiches, indem sie ausgeführt werden. Nach einem zum Beispiel durch Klammern abgegrenzten Bereich ist alles wieder wie vorher.

`\Befehl[`*optionales Argument*`]{`*Argument*`}`

Außerdem gibt es Befehle mit optionalen Argumenten. Diese funktionieren wie einfache Befehle mit Argumenten, allerdings ist hier ein Standardwert hinterlegt für den Fall, dass kein Argument angegeben wird. Ein Beispiel hierfür sind Wurzeln im Mathemodus[1]:

```
$\sqrt{2}$ eine Wurzel ohne Argument\\
$\sqrt[3]{2}$ eine Wurzel mit Argument
```

$\sqrt{2}$ eine Wurzel ohne Argument
$\sqrt[3]{2}$ eine Wurzel mit Argument

`$\sqrt{2}$` liefert hierbei das gleiche wie `$\sqrt[]{2}$` man kann somit in der Regel eckige Klammern bei Befehlen einfach weglassen, wenn sie nichts enthalten.

Umgebungen

`\begin{`*Umgebungsname*`}`...`\end{`*Umgebungsname*`}`

Umgebungen funktionieren wie Befehle. Sie können also ebenfalls notwendige, sowie optionale Argumente besitzen. Umgebungen werden meistens zum Umschalten von Textmodi benutzt (Abschnitt 9.1) oder um Befehle einzugrenzen, wie es bei Schaltern mit Hilfe von Klammern gemacht wird. So liefern

```
{\color{gray}
Dieser Text ist grau.}
Dieser Text ist schwarz.
```

```
\begin{color}{gray}
   Dieser Text ist grau.
\end{color}
Dieser Text ist schwarz.
```

[1] Die Dollarzeichen vor und nach dem Ausdruck, dienen lediglich dazu in den Mathemodus zu wechseln, vgl. Kapitel 15

das gleiche Ergebnis:

Dieser Text ist grau. Dieser Text ist schwarz.

Es ist zudem also auch möglich, die Wirkung von Makros durch ein Klammernpaar (`{...}`) einzugrenzen. Dies funktioniert jedoch nicht bei globalen Makros: (Die Bedeutung dieser Makros wird später besprochen.)

```
\newcounter     \pagenumbering  \newlength
\setcounter     \thispagestyle  \newsavebox
\addtocounter
```

Diese Art von Deklarationen bleibt bestehen, bis sie von einer weiteren Deklaration desselben Typus überschrieben wird.

1.4 Die Eigenheiten von Satzprogrammen

Manche Eingaben werden in Satzprogrammen, also auch bei TEX und LATEX grundlegend anders interpretiert, als man es von herkömmlichen Wortprozessoren kennt:

Leerzeichen und Zeilenwechsel LATEX interpretiert Leerzeichen und Zeilenwechsel im Quellcode als ein Wort- oder Befehlsende. Treten mehrere Leerzeichen auf, werden sie als ein Einziges angesehen. Die Wortabstände werden von LATEX intern zugewiesen und sind entsprechend dem Blocksatz in einem gewissen Maß variabel. Sie werden so gewählt, dass der Text links und rechtsbündig schließt.
Leerzeichen zu Beginn einer Codezeile werden ignoriert.
Leerzeichen, die ein Makro beenden, werden nicht gedruckt, dies kann bei Nichtbeachtung zu fehlenden Leerzeichen führen.

```
Dieser Satz wurde mit \LaTeX gesetzt.
```

erzeugt beispielsweise die Ausgabe

Dieser Satz wurde mit LATEXgesetzt.

Abhilfe kann man dadurch schaffen, dass man entweder das Makro durch eine leere Gruppe (`{}`) beendet oder das Leerzeichen explizit eingibt (`\␣`).

```
Dieser Satz wurde mit \LaTeX{} gesetzt.
Dieser Satz wurde mit \LaTeX\␣gesetzt.
```

Leerzeilen LATEX interpretiert Leerzeilen im Quellcode nicht nur als Wort-, sondern auch als Absatzende. Treten mehrere Leerzeilen auf, so werden sie analog zu den Leerzeichen als eine Einzige angesehen. Entsprechend werden die Absatzabstände von LATEX intern verwaltet und sind variabel.

Sonderzeichen Sie stellen spezielle Befehle dar und müssen, wenn sie als Text ausgegeben werden sollen, über andere Befehle angesprochen werden. Tabelle 1.2 zeigt die Sonderzeichen, welche als Befehl interpretiert werden, samt ihrer Bedeutung und den zugehörigen Befehlen für die Textausgabe.

[2] Prinzipiell könnte man diese Zeichen auch als Akzente über leere Zeichen setzen, z. B. `\~{}`→˜ beziehungsweise `\^{}`→ˆ, allerdings ist dies formal nicht korrekt und sollte somit vermieden werden.
[3] siehe Abschnitt 4.3.2

Tabelle 1.2: Bedeutung der Sonderzeichen in LaTeX und Eingabebefehle.

Zeichen	*Bedeutung des Sonderzeichens*	*Eingabe*
\	Makro-/Befehlsbeginn	\textbackslash
{	Beginn einer Gruppe	\{
}	Ende einer Gruppe	\}
#	Parameter	\#
$	Mathe-Zeilenmodus	\$
&	Trennzeichen bei Matrizen und Tabellen	\&
_	Subscript im Mathemodus (Index)	_
^	Superscript im Mathemodus (Exponent)	\textasciicircum[2]
~	Geschütztes Leerzeichen[3]	\textasciitilde[2]
%	Leitet Kommentare ein	\%

Kommentare Da das Konzept von LaTeX sich damit befasst, dem Text eine logische Struktur zu geben, kann es häufig notwendig sein, sich Notizen zu neuen Kommentaren oder eingebundenen Paketen zu machen. Auf diese Weise kann man vermeiden, dass zu Beginn des Dokumentes erst einmal hundert unnötige Definitionen gemacht werden. das Prozentzeichen (%) ist das sogenannte Kommentarzeichen bei LaTeX. Das Programm ignoriert alles was in einer Zeile hinter einem Prozentzeichen steht. Somit kann man Beschriftungen vornehmen, die nicht in der Ausgabe erscheinen sollen.

1.5 Die Struktur des Quellcodes

Der grundsätzliche Aufbau eines Dokumentes teilt sich formal in zwei Teile:

```
\documentclass[Optionen]{Name}[Version]   } Präambel
...
\begin{document}                          }
...                                       } Textkörper
\end{document}                            }
```

Die Präambel – oft auch Header genannt – enthält prinzipiell alles, was für das gesamte Dokument gültig sein soll, also globale Definitionen und Einstellungen. Dabei können unter anderem auch Definitionen und Befehle überschrieben, beziehungsweise neu definiert werden, die bereits in der Dokumentenklasse (siehe Abschnitt 3.1) festgelegt wurden. Der Teil zwischen `\begin{document}` und `\end{document}` heißt Text-, oder Dokumentenkörper, wird allerdings oft auch nur als Body bezeichnet. Er enthält den eigentlichen Text mit lokalen Formatierungsbefehlen. Mit dem Befehl `\end{document}` wird das Compiler-Programm beendet. Dies bedeutet, dass alles, was danach steht, von LaTeX nicht mehr beachtet wird.

2 Das erste LaTeX-Dokument

Nun geht es zum praktischen Teil: Dem ersten eigenen Dokument. Beispiel 1 zeigt, wie ein einfaches Dokument aussieht. Um nun genau zu verstehen welcher Befehl wofür wichtig ist, zeigt es außerdem eine Untergliederung des Quellcodes nach seinen verschiedenen Funktionen. Im Folgenden werden nun die Funktionen und Eigenschaften der einzelnen Makros betrachtet.

Beispiel 1: Ein einfaches Beispieldokument

Textsatz mit LaTeX

Max Mustermann

4. September 2012

1 Einführung

Ein einfaches Dokument. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

1

```
%Ein kleines Beispiel                          – Kommentar

\documentclass[paper=a5, ngerman,              } Dokumentenklasse;
fontsize=10pt]{scrartcl}                         Abschnitt 3.1

\usepackage[utf8]{inputenc}                    }
                                                 System-&
\usepackage{babel}                               Sprachanpassung;
\usepackage[T1]{fontenc}                         Abschnitt 3.3
\usepackage{lmodern}
\usepackage{microtype}                         }

\usepackage{blindtext}                         – für Blindtexte[1]
\usepackage{geometry}

\title{Textsatz mit \LaTeX}                    } Definitionen für
\author{Max Mustermann}                          die Titelei;
\date{\today}                                    Abschnitt 3.4

\begin{document}                               – Beginn des Textkörpers

\maketitle                                     – Titelerzeugung
\section{Einführung}                           – Gliederungsebene;
                                                 Abschnitt 3.6
Ein einfaches Dokument.                        } Einfacher Text
\blindtext[1]

\end{document}                                 – Ende des Dokumentes
```

[1] Dieses Paket hat lediglich praktische Gründe, da es an der Stelle des Befehls `\blindtext` einen Text zum Testen von Dokumenten ausgibt, sonst jedoch keinerlei Bedeutung.

3 Der Aufbau von LaTeX-Dokumenten

3.1 Die Dokumentenklasse

Zu Beginn eines Dokuments muss zunächst die Art des Dokuments gewählt werden:

`\documentclass[`*Option1*, *Option2*,...`]{`*Klasse*`}`

Dies legt auch gleich Standardlayout und -formatierung des Dokuments fest. LaTeX liest diese aus dem entsprechenden `*.cls`-File der ausgewählten Dokumentklasse aus.

Die wichtigsten Dokumentenklassen sind scrbook, scrartcl, scrreprt, scrlettr2 und die äquivalenten Standardklassen book, article, report und letter. Die sogenannten Standardklassen richten sich jedoch nach US-amerikanischen Konventionen für Typografie und Papierformat, wohingegen die KOMA-Klassen (scr...) europäische und somit insbesondere auch deutschen Gepflogenheiten der Typografie berücksichtigen. Außerdem bieten die KOMA-Klassen eine weitaus größere Menge an Optionen und Zusatzfunktionen, die insbesondere für die Bedürfnisse europäischer Anwender entwickelt wurden. Die Benutzung der KOMA-Klassen ist außerdem aufgrund der typografischen Feinanpassungen und deutlichen Erweiterung der Auszeichnungssprache LaTeX sinnvoll. Sie sind bei weitem flexibler und bieten deutlich mehr Bedienkomfort als die Standardklassen. Die KOMA-Script-Klassen sind sehr gut dokumentiert, sodass ein Blick in die Anleitung [10] oft hilfreich ist!

Um für ein Dokument die richtige Dokumentenklasse wählen zu können, empfiehlt es sich einen Blick auf die grundlegenden Eigenschaften zu werfen:

scrbook (book)
Für große Schriftwerke im Stil von Vorlesungsmitschriften, Büchern, Bachelor-/Diplom-/Masterarbeiten sowie Dissertationen.

- ▷ Titel auf eigener Seite
- ▷ Seitenzählung mit römischen Ziffern (Vorspann, Nachspann) und arabischen Ziffern (Hauptteil), siehe auch Abschnitt 3.5
- ▷ Ebenen[1] `\part`, `\chapter`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, `\subparagraph`
- ▷ Nummerierung von Abbildungen, Tabellen und Gleichungen nach Kapiteln, z. B. 1.1
- ▷ Nummerierung der Fußnoten wird jedes Kapitel neu begonnen

scrartcl (article)
Für kleinere Werke im Stil von Artikeln, Kurzberichten oder Referaten.

- ▷ Titel auf keiner eigenen Seite
- ▷ Seitenzählung mit arabischen Ziffern
- ▷ Ebenen[1] `\part`, `\section`, `\subsection`, `\subsubsection`, `\paragraph`, `\subparagraph`
- ▷ Fortlaufende Nummerierung der Abbildungen, Tabellen, Fußnoten und Gleichungen

[1] Die einzelnen Ebenen und die dazugehörigen Befehle `\section, \part` usw. werden in Abschnitt 3.6 erläutert.

Tabelle 3.1: Die wichtigsten Dokumentenklassenoptionen für KOMA-Script

Bedeutung	`Option`=*Wert* (mögliche Werte)	**Beispiel & Erklärung**
Papierformat	`paper`=`letter`, `legal`, `executive`, `aX`, `bX`, `cX`, `dX`, `landscape`, `seascape`, `portrait`	`paper=a4`[2] `X` ist durch 0, 1,. . . ,8 zu ersetzen. Es werden sämtliche ISO-Formate unterstützt. Zusätzlich kann die Ausrichtung angegeben werden: `landscape` entspricht Querformat, `seascape` dem um 180° gedrehtem Querformat und `portrait` dem Hochformat.
Bindekorrektur	`BCOR`=*Länge*	`BCOR=0mm`[2] Wert muss absolut und mit Einheit angegeben werden. Relative Werte wie 1em sind nicht erlaubt.
Teilungsverhältnis	`DIV`=`4`, `5`,. . . , `calc`, `classic`, `areaset`, `last` oder `default`	`DIV=11` Gibt an in wie viele Streifen der Satzspiegel eingeteilt werden soll.
Kopfzeilen	`headlines`=*Anzahl der Zeilen*	`headlines=1.25`[2] Anzahl der Zeilen, die für die Kopfzeile frei gehalten werden sollen. Der Standard von 1,25 Zeilen bietet in der Regel genug Platz für einzeilige Kopfzeilen und Trennlinien.
einseitiger Druck	`oneside=true` oder `false`	`oneside=true` Hier werden die Ränder nur nach links und rechts unterschieden.
doppelseitiger Druck	`twoside=true` oder `false`	`twoside=true` Buchdrucklayout. Die Ränder unterscheiden sich hierbei nach äußerem und innerem Rand. Ungerade Seitenzahlen werden den rechten Seiten zugeordnet.
einspaltiger Textsatz	`onecolumn=true` oder `false`	`onecolumn=true`[2]
zweispaltiger Textsatz	`twocolumn=true` oder `false`	`twocolumn=false`[2]
Kapitelanfang	`open`=`any`, `right` oder `left`	`open=right` Nur bei `twoside=true` sinnvoll. Neue Kapitel beginnen nur auf den ungeraden, also rechten Seiten.
Größe der Überschriften	`headings=big, normal` oder `small`	`headings=normal`[2]

[2]Wert entspricht dem Standardwert

Fortsetzung auf der nächsten Seite

(Fortsetzung)

Bedeutung	`Option`=*Wert* (mögliche Werte)	**Beispiel & Erklärung**
Formatierung der Überschriften	`headings=onelineappendix, onelinechapter, openany, openleft, openright, twolineappendix, twolinechapter`	`headings=onelineappendix` Ein-/Zweizeilige Kapitel-/Anhangsüberschriften sowie Beginn der Kapitel auf linker/rechter/beliebiger Seite
Seitenvorschub	`cleardoublepage`=*Seitenstil*	`cleardoublepage=empty`[2] Bestimmt den Seitenstil von Vakatseiten, also den Seiten, die beim Satz absichtlich leer bleiben.
Titelseite	`titlepage=true` oder `false`	`titlepage=false` Schaltet bei Verwendung von `\maketitle` zwischen Titelkopf und Titelseiten um.
Absatzabstand/-Einzug	`parskip=full`, `half*`, `false`, `never`,...	`parskip=full` Wählt die Methode mit der ein Absatzumbruch gekennzeichnet werden soll. Alle möglichen Werte werden später in Tabelle 4.3 (Seite 43) genauer erklärt.
Kapitelpräfix	`chapterprefix=true` oder `false`	`chapterprefix=false` Kapitelnummer mit („Kapitel 1“) oder ohne („1“) Präfix „Kapitel“
Anhangspräfix	`appendixprefix=true` oder `false`	`appendixprefix=true` Kapitelnummern im Anhang „Anhang A“) oder nur „A“.
`caption`	`captions=bottombeside, above,...`	`captions=centeredbeside` Positionierung und Formatierung der `captions`. Die Werte werden später in Tabelle 13.2 genauer erläutert.
Schriftgröße	`fontsize=10pt, 11pt, 12pt`	`fontsize=11pt`[2] Es sind auch andere Angaben der Schriftgröße möglich, allerdings muss der Anwender in diesem Fall eine Datei *.clo-Datei zur Verfügung stellen, in der er LaTeX erklärt wie die angegebene Größe zu verwenden ist (Zusatzpakete).

[2] Wert entspricht dem Standardwert

Fortsetzung auf der nächsten Seite

(Fortsetzung)

Bedeutung	`Option`=*Wert* (mögliche Werte)	**Beispiel & Erklärung**
Zusammenfassung	`abstract`=`true` oder `false`	`abstract=true` Zusammenfassung mit oder ohne Titel
Nummerierung	`numbers`=`auto`, `enddot` oder `noenddot`	`numbers=noenddot` Überschriftennummerierung mit (4. oder 4.1.) oder ohne (4 oder 4.1) Punkt am Ende. Diese Option bezieht sich auch alle nummerierten Überschriften.
Entwurfsmodus	`draft`=`true` oder `false`	`draft=false`[2] Bei überlangen Zeilen werden am Zeilenende kleine, schwarze Kästchen ausgegeben. Dies erleichtert dem ungeübten Auge zu sehen, wo eine manuelle Nachbearbeitung nötig ist. Diese Option beeinflusst auch viele Ergänzungspakete, beispielsweise deaktiviert sie sämtliche Funktionen von hyperref.
Kopflinie	`headsepline`=`true` oder `false`	`headsepline=false` Trennt, falls gewünscht, die Kopfzeile vom Textkörper durch eine Linie ab.
Fußlinie	`footsepline`=`true` oder `false`	`footsepline=true` Trennt, falls gewünscht, die Fußzeile vom Textkörper durch eine Linie ab.
Linienlängen	`ilines, clines, olines`	`ilines` Dies Trennlinie zwischen Text und Kopf, bzw. zwischen Text und Fuß wird bündig innen (`ilines`), zentriert (`clines`) oder bündig außen (`olines`) gesetzt.

[2]Wert entspricht dem Standardwert

scrreprt (report)
Für mittlere Schriftwerke im Stil von Berichten, Hausarbeiten und Praktikumsprotokollen.

▷ Wie scrartcl, jedoch mit `\chapter`[1]

scrlettr2 (letter)
Für Briefe. Diese Dokumentenklasse hat keinerlei Ebenen, allerdings spezielle Briefelemente, wie zum Beispiel: Absender, Anschrift, Betreff, Anlagen,...

Außerdem bietet KOMA-Script eine riesige Menge an Dokumentenklassenoptionen zur leichten Anpassung von Dokumenten. Der wichtigste Teil dieser Optionen findet sich in Tabelle 3.1 (auf den Seiten 16–18). Einige darin auftauchende Begriffe werden jedoch erst im weiteren Verlauf genauer besprochen.

3.2 Ergänzungspakete

Die große Zahl an Nutzern mit unterschiedlichen Bedürfnissen, denen LaTeX gerecht werden möchte resultiert in einer riesigen Menge an Ergänzungspaketen. Sie liefern Befehle und/oder verändern das Dokumentenlayout auf verschiedenste Weise. Normalerweise installiert man zusammen mit der Distribution automatisch die am weitesten verbreiteten Pakete. Sie müssen somit lediglich geladen werden:

`\usepackage[`*Option1*`,`*Option2*`,...]{`*Paket*`}`

Dieser Befehl muss innerhalb der Präambel, also zwischen `\documentclass` und `\begin{document}` stehen.

Im Laufe dieses Kurses werden wir einige wichtigere Pakete genauer betrachten. Tabelle 3.2 zeigt die diejenigen, die hier behandelt werden mitsamt einer kleinen Beschreibung.

Globale und lokale Optionen

Zusätzlich zu den Dokumentenklassenoptionen kann man bei der Angabe der Dokumentenklasse auch globale Optionen festlegen. Die dort gewählten Optionen gelten für *alle* Packages, die geladen werden. Man kann und sollte Optionen, die von mehreren Paketen verarbeitet werden können, als Dokumentenklassenoption angeben. Insbesondere gilt dies für die Optionen `ngerman` und `draft`.

Paketanleitungen finden

Für speziellere Anwendungen oder zum nachschlagen bestimmter Makros empfiehlt sich häufig ein Blick in die Paketanleitung. Die richtige Paketanleitung zur installierten Version findet man über die Eingabe von

`texdoc` *Paketname*

und anschließendem `Enter` in der Kommandozeile[3]. Dann öffnet sich die Anleitung automatisch.

[1] Die einzelnen Ebenen und die dazugehörigen Befehle `\section, \part` usw. werden in Abschnitt 3.6 erläutert.
[3] Unter Windows heißt das Programm hierfür `cmd` und es lässt sich über die Suchfunktion im Startmenü finden.

Tabelle 3.2: Die Ergänzungspakete, die in diesem Kurs besprochen werden

`amsmath`	Verbesserter Formelsatz; Kapitel 15
`amssymb`	für $\mathcal{AMS}$-Mathe-Symbole; Kapitel 15
`array`	Weitere Spaltenformatierungen für Tabellen; Abschnitt 12.1
`babel`	Sprachunterstützung, Trennregeln,. . . ; Abschnitt 3.3.2
`bm`	Fettdruck in mathematischen Formalismen; Abschnitt 15.2
`booktabs`	Verbessertes Spacing und Linien bei Tabellen; Abschnitt 12.4
`fontenc`	Ausgabekodierung und Vektorschrift; Abschnitt 3.3.3
`color`	bringt Farbe ins Spiel; Abschnitt 4.5
`csquotes`	Kontextsensitive Anführungszeichen; Abschnitt 3.3.4
`enumitem`	Erweiterte Möglichkeiten im Umgang mit Aufzählungen; Abschnitt 8.4
`geometry`	für individuelle Seitenformatierung; Abschnitt 6.2
`graphicx`	Einbinden von Bildern; Kapitel 10
`hyperref`	Hyperlinks und erweiterte `PDF`-Fuktionalitäten; Abschnitt 5.3
`inputenc`	Eingabekodierung; Abschnitt 3.3.1
`lmodern`	verbesserte Fassung der Schriftart Computer Modern; Abschnitt 3.3.3
`longtable`	mehrseitige Tabellen;Abschnitt 12.5
`mhchem`	Chemische Formeln; Abschnitt 15.18
`microtype`	verbesserte Trennregeln durch Berücksichtigung der Mikrotypografie; Abschnitt 3.3.2
`multicol`	Mehrspaltiger Textsatz;Abschnitt 6.3
`multirow`	für den multirow-Befehl bei Tabellen; Kapitel 12
`placeins`	für den FloatBarrier-Befehl bei Gleitobjekten; Abschnitt 13.1
`ragged2e`	Modifizierter Flattersatz mit Worttrennungen; Tabelle 4.4
`scrlayer-scrpage`	Modifikationen des Seitenstils; Abschnitt 7.1
`setspace`	Zeilenabstand ändern; Abschnitt 4.2.4
`siunitx`	Zahlen und Einheiten typographisch korrekt setzen; Abschnitt 15.17
`tabularx`	Tabellen mit einer dehnbaren `X`-Spalte; Abschnitt 12.2
`tabulary`	Tabellen die sich entsprechend des Textinhaltes der Spalten auf eine bestimmte Breite dehnen lassen (`L`-, `C`-, `R`-, `J`-Spaltentypen); Abschnitt 12.3
`typearea`	professionelle Satzspiegelkonstruktion; Abschnitt 6.1
`wrapfig`	Objekte von Text umfließen lassen;Abschnitt 13.3

3.3 Notwendige Anpassungen an System & Sprache

Ein riesiger Vorteil von LaTeX ist seine Unabhängigkeit von sowohl Betriebssystem, als auch Nutzungssprache. Jedoch bringt das mit sich, dass man dem Programm mitteilen muss, mit welchem der vielen möglichen Systeme und somit auch Zeichenkodierungen, beziehungsweise mit welcher Sprache man arbeiten möchte. Dies geschieht mithilfe von Zusatzpaketen, die LaTeX erklären, wie es den Binärcode, den es vom System zum Verarbeiten bekommt, interpretieren soll. Hier ein Beispiel für ein kurzes Dokument mit allen nötigen Anpassungen.

```
\documentclass[ngerman]{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage{babel}
\usepackage[T1]{fontenc}
\usepackage{lmodern}
\begin{document}
Hier steht der Dokumenteninhalt!
\end{document}
```

3.3.1 Eingabekodierung (Das inputenc-Paket)

LaTeX versteht zwar von sich aus den normalen Buchstabensatz, jedoch gibt es jede Menge verschiedener Kodierungen für die Sonderzeichen. So wird man gerade als deutscher Nutzer sehr schnell Probleme beim Satz von Umlauten bekommen. Hierzu benutzt man für gewöhnlich das inputenc-Paket. Es lässt den Anwender aus einer Reihe von Zeichensätzen wählen und fungiert dann als Übersetzer. Da es mit Unicode gelungen ist eine einheitliche Kodierung für alle Systeme zur Verfügung zu stellen, empfiehlt es sich, um die Systemunabhängigkeit von Dokumenten zu gewährleisten, diese zu nutzen. UTF-8 kann somit als derzeitiger Standard angesehen werden und sollte nach Möglichkeit benutzt werden. Dafür lädt man das Paket mit der Option `utf8`:

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

3.3.2 Sprachanpassung (Das babel-Paket)

Um den Satz eines Absatzes möglichst perfekt zu gestalten, benutzt TeX einen sehr effektiven Algorithmus um die Wörter möglichst gleichmäßig zu verteilen und somit unschöne Löcher im Blocksatz zu verhindern. Ein Teil dieses Vorgangs beschäftigt sich auch mit der Trennung von Wörtern am Zeilenende. Jedoch gelten in den unterschiedlichen Sprachen unterschiedliche Regeln. Man muss also einstellen, welche Regeln benutzt werden sollen.

Außerdem ermöglicht eine Sprachanpassung, dass die Verzeichnisse richtig beschriftet werden, also dass zum Beispiel über dem Inhaltsverzeichnis wirklich „Inhaltsverzeichnis“ steht und nicht mehr „Table of Contents“.

Die Einstellung der Sprache geschieht bei pdfLaTeX mit dem babel-Paket. Es kennt jede Menge Sprachen und erlaubt es sogar innerhalb des Dokumentes die Sprache zu wechseln. Die Sprache wird, wie auch die Kodierung, mithilfe einer Paketoption ausgewählt:

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

`ngerman` steht hierbei für „neue deutsche Rechtschreibung“. Es gibt jedoch auch weitere Pakete, die eine Sprachanpassung zulassen, vor allem Pakete, die mit den Verzeichnissen, insbesondere dem

Literaturverzeichnis oder dem Index arbeiten. Somit ist es sinnvoll, die Option `ngerman` nicht als Paketoption zu übergeben, sondern als globale Dokumentenoption. Somit kann jedes Paket, dass diese Option verarbeiten kann, sie sich aus der Dokumentenoption nehmen und man muss keine Rücksicht darauf nehmen, welches Paket eine Sprachanpassung benötigt und welches nicht. Somit schreibt man besser:

```
\documentclass[ngerman, Option2, Option3,...]{Dokumentenklasse}
\usepackage{babel}
```

Für anders- oder mehrsprachige Dokumente gibt es äquivalente Optionen. Die zuletzt geladene Option entspricht dabei der Hauptsprache. Ein Dokument auf deutsch mit englischen Passagen benötigt somit das Makro

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

Als Dokumentenklassenoption wird in diesem Fall nur die Hauptsprache gesetzt. Damit diese jedoch nicht durch die Paketoptionen überschrieben wird, benötigt man dort beide Angaben. Innerhalb des Dokumentes kann dann die Sprache global

```
\selectlanguage{Sprache}
```

oder lokal

```
\foreignlanguage{Sprache}{Text}
```

geändert werden.

Leerzeichen nach einem Punkt

Neben den Trennregeln unterscheiden sich die beiden Sprachen Deutsch und Englisch auch durch die Leerzeichen nach einem Satzendepunkt. Mit Standardeinstellungen ist bei LaTeX der Zwischenraum nach einem Punkt größer als ein normaler Wortzwischenraum. Im deutschen Sprachraum sollte dieser Abstand jedoch den übrigen Wortzwischenräumen entsprechen (sogennantes `\frenchspacing`). Das `babel`-Paket setzt auch diese Einstellungen automatisch für die gewählte Sprache.

Soll dieser zusätzliche Zwischenraum nur in einzelnen Fällen unterdrückt werden, zum Beispiel bei Abkürzungen, kann dies durch das einfügen eines normalen Leerzeichens bewerkstelligt werden:

```
Otto Müller \& Co.\␣erhöhten ihre Kapazitäten auf das doppelte.
```

Punkte die auf Großbuchstaben folgen, werden als Abkürzungen interpretiert. Hier folgt in beiden Fällen ein normaler Wortzwischenraum. Um LaTeX im Fall von `\nofrenchspacing` (z. B.: englischer Text) zu sagen, dass es sich dennoch um einen Satzendepunkt handelt, muss man diesem die Zeichenkombination `\@` voranstellen. Zum Beispiel:

```
Zitronen enthalten Vitamin C\@.
```

Der zusätzliche Zwischenraum bei `\nofrenchspacing` wird auch bei anderen Satzzeichen, dem Fragezeichen (?), Ausrufezeichen (!) und dem Doppelpunkt (:) eingefügt. Hier kann das jedoch auf die selbe Weise wie bei einem Punkt verhindert (`\␣`) oder ermöglicht (`\@`)

Verbessertes Trennverhalten mit microtype

Um ein noch besseres Trennverhalten der Wörter und zusätzliches Kerning zu erhalten, empfiehlt es sich außerdem, vor allem bei längeren und wichtigeren Arbeiten zusätzlich das Paket microtype über

```
\usepackage{microtype}
```

einzubinden. Es verbessert den Trennalgorithmus um die Möglichkeiten der Mikrotypografie (optischer Randausgleicher, Wort- und Zeichendehnung) und vermeidet somit viele unschöne Zeilenumbrüche und überlange Zeilen.

Anführungszeichen und Umlaute

Eine weitere Eigenschaft des babel-Paketes ist der Satz von Anführungszeichen und Umlauten. Ein Überbleibsel aus der Zeit vor UTF-8, also als man Umlaute noch nicht direkt eingeben konnte, ist dabei, dass Zeichenfolgen wie „`"U`“ als „Ü“ ausgegeben werden. Für den korrekten Satz von Anführungszeichen siehe Abschnitt 3.3.4.

3.3.3 Schriftkodierung (Die Pakete fontenc und lmodern)

Bei der Ausgabe hat LaTeX wieder das gleiche Problem wie am Anfang: Die Übersetzung von Ausgabe in den gewünschten Zeichensatz beziehungsweise die gewünschte Schriftart. Standardmäßig verwendet pdfLaTeX hierbei die Schriftsippe Computer Modern. Allerdings existiert sie gerade bei älteren Distributionen nur in der Bitmap-Variante, was zu dem Problem führen kann, dass sie bei zu großer Vergrößerung pixelig wird. Um dies zu vermeiden, sollte man Vektorschriften benutzen, die skalierbar sind.
Außerdem ist der Standard Zeichensatz sehr begrenzt. Somit werden Umlaute wie „ä“ nicht als einzelnes Zeichen, sondern als „a“ mit dem Akzent „¨“ gesetzt. Dies hat jedoch Nachteile, wenn man die Vorteile eines digitalen Dokumentes, wie zum Beispiel die Suchfunktion, nutzen will: Das Wort „ändern“ wird trotz Vorhandensein nicht gefunden, da man ja ein „ä“ sucht und kein „a“ mit zwei Punkten.
Um diese Probleme zu vermeiden und zusätzlich eine Worttrennung an Umlauten zu ermöglichen, benötigt man das fontenc-Paket und eine Vektorschrift. Hierfür bietet sich Latin Modern an, da sie mittlerweile bei allen Distributionen vorhanden ist.

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

3.3.4 Anführungszeichen – Zitate

Anführungszeichen stehen prinzipiell vor und hinter [vgl. 17, S.29]:

- einer wörtlich wiedergegebenen Äußerung (direkte Rede),
- einer wörtlich angeführten Textstelle (Zitat)[4]
- zitierten Überschriften, Titeln von Büchern, Filmen, Gedichten, Namen von Zeitungen und ähnlichem,

[4] Bei Zitaten ist zudem zu beachten, dass sie immer wörtlich wiederzugeben sind. Sie dürfen weder im Wortlaut noch in Rechtschreibung und Interpunktion vom Original abweichen. Eigene Korrekturen und Ergänzungen zum Zitat sind duch eckige Klammern zu kennzeichnen

- Wortschöpfungen und Worten, die im übertragenen Sinne gemeint sind (Metaphern),
- einzelnen Wortteilen, Wörtern oder Textteilen, die hervorgehoben werden sollen.

„Lange Zitate werden nich in Anführungszeichen eingeschlossen, sondern eingerückt mit einer `quote`- oder `quotation`-Umgebung gesetzt“[17, S.30], siehe Abschnitt 4.4.

Zitate aus Fremdsprachen werden ebenfalls in Anführungszeichen der Dokumentensprache gesetzt. Lediglich in dem Fall, dass das Zitat selbst fremdsprachliche Anführungszeichen enthält, sind diese zu übernehmen

Zudem bleibt zu sagen, dass bei allen Schriften in OT1-Kodierung das Kerning zwischen Anführungszeichen und nachfolgenden bzw. vorangehenden Zeichen fehlt. Dies spricht wieder für die Wahl der T1-Kodierung (Abschnitt 3.3.3) und für erweitertes Kerning auch für das Paket microtype (Abschnitt 3.3.2).

Eingabe von Anführungszeichen

Deutsche Anführungszeichen:
Gänsefüßchen:

`\glqq` oder `"‘`	„
`\glq`	‚
`\grqq` oder `"’`	“
`\grq`	‘

Spitze Form:

`">`, `>>` oder `\flqq`	»
`\frq`	›
`"<`, `<<` oder `\frqq`	«
`\flq`	‹

Englische Anführungszeichen:

‘‘	“
,,	‘
‘	”
,	’

Das Paket csquotes

Das Paket csquotes vereinfacht die Eingabe der Anführungszeichen erheblich. Es ermöglicht unter anderem mit dem Makro

`\enquote{`*Zitatinhalt*`}`

kontextsensitive und durch babel an die Sprache angepasste Zitate. Folgendes Beispiel zeigt die Unterschiede:

> Die Zeitung schrieb: „Die Bahn hat bereits im Frühjahr erklärt: ‚Wir haben die feste Absicht, die Strecke stillzulegen‘ und sie hat das auf Anfrage gestern noch einmal bestätigt.“

Man kann diesen Satz auf zwei verschiedene Arten setzen:
Zunächst nur mit dem babel-Paket:

```
Die Zeitung schrieb: \glqq{}Die Bahn hat bereits im Frühjahr erklärt:
\glq{}Wir haben die feste Absicht, die Strecke stillzulegen\grq{}, und
sie hat das auf Anfrage gestern noch einmal bestätigt.\grqq{}
```

Hier mit csquotes:

```
Die Zeitung schrieb: \enquote{Die Bahn hat bereits im Frühjahr erklärt:
\enquote{Wir haben die feste Absicht, die Strecke stillzulegen} und sie
hat das auf Anfrage gestern noch einmal bestätigt.}
```

3.3.5 Binde- & Gedankenstrich

Auch wenn der Unterschied relativ gering ist, sollte man beim schreiben wichtiger Dokumente darauf achten den Unterschied zwischen Binde- und Gedankenstrich richtig umzusetzen. Im Englischen ist das ganze noch eine Stufe schwieriger, dort unterscheidet sich der „von-bis-Strich“ auch noch vom Gedankenstrich, im Deutschen sind diese beiden identisch. Deswegen stellt auch LaTeX drei verschiedene Stricharten zur Verfügung.

-	- Bindestrich
--	– „von-bis-Strich“ und deutscher Gedankenstrich
---	— englischer Gedankenstrich

Zusätzlich ist es wichtig, dass der deutsche Gedankenstrich im Gegensatz zum englischen durch beidseitige Leerzeichen abgesetzt wird.

3.3.6 Akzente und Sonderzeichen

Die deutsche Sprache gehört zu denen, die spezielle Zeichen benötigen. Dank UTF-8 ist es mittlerweile möglich alle dafür benötigten Zeichen und auch die meisten Sonderzeichen von Fremdsprachen direkt über die Tastatur einzugeben. Dennoch gibt es einige Zeichen, über die unsere Tastatur nicht verfügt. Die Tabellen 3.3 und 3.4 zeigen die möglichen Sonderzeichen und Akzente in der klassischen LaTeX-Notation.

Tabelle 3.3: Akzente in der klassischen LaTeX-Notation am Beispiel des Buchstaben „o“. Die Befehle sind jedoch auf alle Buchstaben anwendbar.

ò	\‘{o}	õ	\~{o}	ǒ	\v{o}	o̧	\c{o}
ó	\’{o}	ō	\={o}	ő	\H{o}	ọ	\d{o}
ô	\^{o}	ȯ	\.{o}	o͡o	\t{oo}	o̲	\b{o}
ö	\"{o}	ŏ	\u{o}				

Tabelle 3.4: Sprachspezifische Symbole

œ	\oe	å	\aa	ł	\l	¿	?‘
Œ	\OE	Å	\AA	Ł	\L	¡	!‘
æ	\ae	ø	\o	ß	\ß		
Æ	\AE	Œ	\O				

3.3.7 Auslassungszeichen

Das Auslassungszeichen (. . .) wird bei LaTeX mit dem Befehl

```
\ldots
```

gesetzt. Beim Satz drei einfacher Punkte (...) stimmen die Abstände nicht. Bei Aufzählungen ist es oft nötig einen kleinen Abstand zwischen die Auslassungspunkte und ein folgendes Komma zu setzen:

$a_1 < b_1$ für $i = 1, 2, \ldots, n$.

erzeugt mit

```
$a_1<b_1$ für $i=1$,~2, \ldots\,,~$n$.
```

Hierfür bietet sich das Makro `\dots` and, welches eine Abkürzung für `\ldots\,` darstellt.

3.4 Titelei

Bei Dokumenten wird zwischen zwei verschiedenen Arten von Titeln unterschieden: Es gibt entweder ganze Titelseiten oder lediglich einen Titelkopf.
Titelseiten zeigen den Dokumententitel zusammen mit weiteren Informationen, wie beispielsweise Autor auf einer eigenen Seite. Neben der Haupttitelseite gibt es, insbesondere bei Büchern, noch weitere Titelseiten mit Verlagsdaten, Widmung oder ähnlichen Informationen. Beim Titelkopf erscheint der Titel samt Autor lediglich am Anfang der ersten Seite. Nach einem kleinen Abstand wird hier entweder eine kurze Zusammenfassung oder direkt der erste Abschnitt gesetzt. Das Umschalten hierfür übernimmt die Dokumentenklassenoption `titlepage` (siehe auch Tabelle 3.1, Seite 16).

Für die Erzeugung der Titelei gibt es zwei verschiedene Möglichkeiten (Es sollte immer *nur eine* davon benutzt werden, auch wenn theoretisch Kombinationen möglich sind.):

Klassische Titeleierzeugung mit `\maketitle`

Man übergibt hier mit Hilfe von anderen Makros, wie zum Beispiel `\author{`*Autor*`}` die Informationen an LaTeX und lässt es dann mithilfe des Befehls `\maketitle` die Titelseite automatisch erzeugen. Die Standardklassen setzen auf der Titelseite lediglich die Informationen Autor, Titel und Jahr. Die KOMA-Klassen hingegen bieten eine weitaus größere Menge an Makros zur Erzeugung von Titelseiten, wie sie bei jedem Buch, insbesondere in Fachbüchern gefunden werden. Die Titelerzeugung kann dabei folgendermaßen aussehen.

. . . Präambel mit Dokumentenklasse und Paketen . . .
`\begin{document}`

```
\titlehead{Titelkopf (frei gestaltbar)}
\title{Titel}
\subtitle{Untertitel}
\subject{Typisierung}
\author{Autor 1 \and Autor 2}
\date{Datum}
```

```
\maketitle[Seitennummer der ersten Titelseite]
```

. . . Inhalt des Dokumentes . . .
`\end{document}`

Optisch sieht das ganze wie beim Beispieldokument (Beispiel 1, S. 14) aus. Wobei sich der Titel, abhängig von Dokumentenklasse und Optionen, entweder auf eigenen Seiten befindet oder wie im Beispiel lediglich als Titelkopf auftritt.

KOMA-Script bietet zudem weitere Makros zum Erzeugen von Widmungen, Schmutztitel[5], Verlagsinformationen,...

`\extratitle{`*Schmutztitel*`}`
`\publishers{`*Verlag*`}`
`\uppertitleback{`*Titelrückseitenkopf*`}`
`\lowertitleback{`*Titelrückseitenfuß*`}`
`\dedication{`*Widmung*`}`
`\thanks{`*Fußnote*`}` mit `fnsymbol` gekennzeichnet, siehe Abschnitt 3.10

Frei gestaltbare Titelseiten

`\begin{titlepage}...\end{titlepage}`

Diese Variante bietet mehr Gestaltungsfreiraum, ist aber auch aufwendiger. Es wird lediglich der Seitenstil auf `empty` gesetzt (siehe Kapitel 7, Seite 60), sodass die Titelseite keine Kopf- und Fußzeile besitzt. Ansonsten kann die komplette Titelseite vollkommen frei mithilfe von Textformatierungen, Abständen, sowie Bildern gestaltet werden.

Die Dokumentenklassenoption `titlepage` hat hier keinen Einfluss. Es werden immer komplette Titelseiten erstellt. Die Erstellung eines Titelkopfes ist hiermit nicht möglich.

3.5 Vorspann, Hauptteil & Nachspann

Sehr lange Dokumente, hauptsächlich Bücher werden häufig noch in Vorspann, Hauptteil und Nachspann unterteilt. LaTeX bietet hierfür in der Dokumentenklasse book und somit auch der darauf basierenden KOMA-Klasse scrbook Schalterbefehle um den jeweiligen Buch-Teil einzuleiten:

`\frontmatter` Vorspann: Seitennummern mit kleinen römischen Zahlen (`roman`), Kapitelüberschriften nicht nummeriert – Feinere Untergliederungen in Abschnitte nicht sinnvoll. Dieser Abschnitt eignet sich für die Titelei, diverse Verzeichnisse und ein Vorwort.

`\mainmatter` Hauptteil, arabische Seitenzahlen beginnend bei 1.

`\backmatter` Nachspann, Untergliederung wie beim Vorspann, Seitennummerierung wird aus dem Hauptteil fortgesetzt. Möglicher Inhalt wäre ein Literaturverzeichnis, ein Stichwortverzeichnis und/oder ein Anhang.

[5] Der Schmutztitel ist die erste rechte Seite im Buch, auf der lediglich der Haupttitel, teilweise sogar nur eine Kurzform steht.

3.6 Gliederungsebenen

Zur Einteilung des Dokumentes in verschiedene Ebenen stellt LaTeX jede Menge Befehle zur Verfügung. Diese Makros setzen nicht nur die entsprechenden Überschriften, sondern zeichnen sie auch dem logischem Markup entsprechend aus. Die Hierarchie ist hierbei abhängig von der Dokumentenklasse, siehe auch Tabelle 3.5. Die Syntax zum Einleiten eines neuen Gliederungspunktes ist immer dieselbe.

Tabelle 3.5: Die klassische Hierarchie der Gliederungsebenen in LaTeX

`\part`	Ebene -1 bei scrbook, scrreprt o. Ä. Ebene 0 bei scrartcl o. Ä. immer auf eigener Seite, bei scrbook immer auf ungerader (rechter) Seite
`\chapter`	Ebene 0; nur bei scrbook, srcreprt o. Ä. bei scrreprt auf neuer Seite, bei scrbook auf nächster ungeraden (rechten) Seite
`\section`	Ebene 1
`\subsection`	Ebene 2; Letzte nummerierte Ebene in `\scrbook`, `\scrreprt` o. Ä.
`\subsubsection`	Ebene 3; Letzte nummerierte Ebene in `\scrartcl`
`\paragraph`	Ebene 4; Kein direkter Zeilenumbruch nach der Überschrift
`\subparagraph`	Ebene 5; Optisch nicht von `\paragraph` unterscheidbar

Für ein Kapitel sieht das zum Beispiel so aus:

```
\chapter[Kurzform]{Kapitelname}
\chapter*{Kapitelname}
```

Diese Makros erledigen nicht nur die automatische Nummerierung und Formatierung der Überschriften, sondern tragen zusätzlich den Text der Überschrift (oder falls angegeben die Kurzform) ins Inhaltsverzeichnis und in die Kolumnentitel[6] ein. Die gesternte Version unterdrückt diesen Vorgang und setzt lediglich eine nicht nummerierte Überschrift in der entsprechenden Formatierung. Für weitere einfache Möglichkeiten die Darstellung der Kapitelüberschriften zu ändern, ist ein Blick in Tabelle 3.1 hilfreich.

Normalerweise wird die Abschnittsnummerierung von der zweithöchsten Ebene an mit angezeigt. So erhält zum Beispiel die zweite `section` im dritten `chapter` die Nummer 3.2. Diese Zählung kann jedoch mit Hilfe der zugehörigen Counter (siehe `secnumdepth` und `tocdepth` in Abschnitt 3.10) manipuiert werden.

```
\minisec{Überschrift}
```

Zusätzlich zu den klassischen Gliederungsebenen bietet KOMA-Script den weiteren Überschriftentypus der `\minisec`. Diese Art der Überschrift hat im Gegensatz zum Paragraph einen Zeilenumbruch nach dem Titel und kleinere Abstände. Dieses Makro setzt lediglich eine Überschrift. Es entspricht keiner Gliederungsebene und erhält somit weder eine Nummer, noch einen Eintrag in das Inhaltsverzeichnis.

[6] Kolumnentitel sind die Überschriften einzelner Buchseiten, zum Beispiel die Angabe der Kapitel-/Abschnittsüberschrift in der Kopfzeile.

Zusammenfassung

Zusätzlich zu den Gliederungsebenen existiert in den Dokumentenklassen scrartcl und scrreprt eine Umgebung für Zusammenfassungen.

```
\begin{abstract}  Text  \end{abstract}
```

Die Zusammenfassung folgt normalerweise direkt der Titelei und soll einen kurzen Überblick über das Dokument geben. Die Zusammenfassung erscheint beidseitig eingerückt und ohne Überschrift (es sei denn die Dokumentenoption `abstract=true` ist gesetzt). In der Dokumentenklasse scrartcl folgt sie direkt dem Titelkopf. Bei scrreprt erscheint sie auf einer eigenen Seite vertikal zentriert. In der Dokumentenklasse scrbook existiert kein `abstract`.

Anhang

Im Gegensatz zur Grobaufteilung (Abschnitt 3.5) ist ein Anhang bei jedem Dokumententyp zu finden. Er wird analog mit einem Schalterbefehl eingeleitet:

```
\appendix
```

Der Anhang entspricht einem `\part`. Die höchste Gliederungsebene (`\chapter` bei scrbook – `\section` bei scrartcl) wird hier jedoch mit Großbuchstaben nummeriert. Kleinere Ebenen haben entsprechend die Form „A.1“.

3.7 Das Inhaltsverzeichnis

LaTeX kann durch sein logisches Markup automatisch ein Inhaltsverzeichnis anlegen, in welches die Überschriften mit der zugehörigen Seitennummer eingetragen werden. Hierfür verwendet das Programm eine Hilfsdatei mit der Endung `*.toc`. Der Übersichtlichkeit halber werden jedoch nur die obersten Gliederungsebenen in das Inhaltsverzeichnis eingetragen (siehe auch Tabelle 3.7 in Abschnitt 3.10).

```
\tableofcontents
```

Dieser Befehl erzeugt an der entsprechenden Stelle im Dokument das Inhaltsverzeichnis. Wenn das aktuelle Dokumente ein Verzeichnis (gilt auch für Abbildungs- und Tabellenverzeichnis) enthält, so muss das Dokument mindestens zweimal kompiliert werden. Im ersten Durchlauf wird die Hilfsdatei (in diesem Fall `*.toc`) erstellt. Zu Beginn des zweiten Laufes wird die Datei geladen und ihr Inhalt entsprechend formatiert.

Einfache Formatierungsänderungen geschehen mithilfe der Dokumentenklassenoption `toc`. Tabelle 3.6 zeigt die möglichen Werte mitsamt einer Erläuterung. Für weitere Informationen zur Formatierung wird auf Kapitel 14 verwiesen.

Tabelle 3.6: Werte für die Dokumentklassenoption `toc`

`bib`	Das Literaturverzeichnis erscheint im Inhaltsverzeichnis ohne Nummerierung
`bibnumbered`	Das Literaturverzeichnis erscheint im Inhaltsverzeichnis mit Nummerierung
`flat` oder `left`	Das Inhaltsverzeichnis ist tabellarisch. In der ersten Spalte stehen die Gliederungsnummern, in der Zweiten die Überschriften und in der Letzten die Seitenzahlen.
`graduated`	Das Inhaltsverzeichnis ist hierarchisch aufgebaut mit begrenztem Platz für die Gliederungsnummern.
`index`	Das Stichwortverzeichnis erscheint im Inhaltsverzeichnis ohne Nummerierung
`listof`	Abbildungs- und Tabellenverzeichnis erscheinen im Inhaltsverzeichnis ohne Nummerierung
`listofnumbered`	Abbildungs- und Tabellenverzeichnis erscheinen im Inhaltsverzeichnis mit Nummerierung
`nobib`	Kein Literaturverzeichnis im Inhaltsverzeichnis
`noidx`	Kein Stichwortverzeichnis im Inhaltsverzeichnis
`nolistof`	Kein Abbildungs- und kein Tabellenverzeichnis im Inhaltsverzeichznis

3.8 Lange Dokumente aufteilen

Bei recht langen Dokumenten (50 Seiten oder mehr) empfiehlt es sich aus mehreren Gründen, das Dokument in einzelne `*.tex`-Dateien aufzuteilen. Dies macht die Dokumente übersichtlicher und spart insbesondere auch Zeit beim kompilieren, wenn jeweils nur die aktuell bearbeitete Datei neu kompiliert wird, anstatt jedes mal das gesamte Dokument zu übersetzen.

`\input{`*Dateiname*`}`

Das `\input`-Makro fügt dabei den Inhalt der `.tex`-Datei ohne Modifikationen an der Position des Befehles ein. Die Benutzung ist auch innerhalb der Präambel möglich. Es eignet sich somit auch bestens dafür sämtliche persönliche Anpassungen in einer selbst erstellten Präambel-Datei auszulagern und diese zu Beginn jedes Dokumentes des gleichen Typs einzubinden.

Eine Dateiendung muss hierbei lediglich dann angegeben werden, falls sie sich von `*.tex` unterscheidet.

Zudem ist es möglich `\input` zu schachteln, dies bedeutet, dass auch Dateien, die mit `\input` geladen werden, ebenso das Makro `\input` enthalten dürfen.

Innerhalb des Textkörpers empfiehlt es sich für jedes Kapitel eine eigene Datei anzulegen. Hierfür ist das `\include`-Makro zu bevorzugen.

`\include{`*Dateiname*`}`

Der Befehl bindet den Code aus der Datei so ein, dass der Dateiinhalt auf einer neuen Seite beginnt und mit `\clearpage` abgeschlossen wird. Es entspricht somit der Makrofolge

```
\clearpage\input{Dateiname}\clearpage
```

`\includeonly{`*Dateienliste*`}`

`\includeonly` bewirkt, dass lediglich bestimmte Dateien kompiliert werden, die mit `include` eingefügt wurden. Dieser Befehl muss in der Präambel stehen.

Wenn man zuvor das gesamte Dokument fertig kompiliert hat und sich dann erst der Bearbeitung einer Teildatei widmet, so bleibt die Nummerierung nach dem Einschub von `\includeonly` erhalten. z. B.:

```
...
\includeonly{kapitel-1,kapitel-4}
...% enthält \begin{document}
\include{kapitel-1}
...
\include{kapitel-n}
...
```

Wenn hier die Datei schon einmal ohne `\includeonly` kompiliert wurde, so bleibt Kapitel 1 auch nach dem Einschub Kapitel 1 und Kapitel 4 erhält weiterhin die Nummer 4, auch wenn 2 und 3 im aktuellen Ausgabedokument gar nicht auftauchen. Dies bleibt solange erhalten, solange man an der Struktur der Datei keine Änderungen vornimmt.

3.9 Eigene Befehle

LaTeX gestattet die Deklaration eigener Befehle bzw. die Umdefinition bereits vorhandener Befehle (sehr nützlich bei sich wiederholenden Ausdrücken). Die Syntax hierfür lautet:

`\newcommand*{\`*Befehlsnahme*`}[`*Anzahl zu übergebender Argumente*`]{`*Definition*`}`
`\newcommand{\`*Befehlsnahme*`}[`*Anzahl zu übergebender Argumente*`]{`*Definition*`}`
`\renewcommand{\`*Befehlsnahme*`}[`*Anzahl zu übergebender Argumente*`]{`*Definition*`}`

Die Sternchenversion ist immer dann zu bevorzugen, wenn das Makro entweder keine Argumente erhält oder diese Argumente keine Absatzumbrüche enthalten sollen. Sie erlaubt keine Absatzumbrüche innerhalb der Argumente.

Die Argumente werden der Reihe nach mittels #1,#2,... abgefragt

`\renewcommand` erlaubt es, bereits vorhandene Befehle zu überschreiben. Dies sollte daher nur verwendet werden, wenn man genau weiß, wofür das Makro, welches man überschreiben möchte verwendet wird.

Hierfür ein paar Beispiele:

```
\newcommand*{\doubleint}[0]{\int \hspace{-1ex}\int}
\newcommand*{\blau}[1]{\textcolor{blue}{#1}}
\renewcommand{\thefootnote}{\roman{footnote}}
```

somit ergibt sich:

$\iint$ (`$\doubleint$`);
Blaue Schrift (`\blau{Blaue Schrift}`);
Fußnoten werden mit kleinen römischen Ziffern nummeriert;

Im weiteren Verlauf dieses Kurses werden wir sehr häufig die Syntax von `\renewcommand` benötigen und somit auch die Verwendung dieser Makros weiter vertiefen.

3.10 Zähler

In einem Dokument wird vieles nummeriert, wie z.B. die Seiten, Abbildungen oder Überschriften. Hinter jeder solchen Nummerierung steckt ein Counter, der beliebig verändert werden kann. Die Namen der wichtigsten Counter findet man in Tabelle 3.7.

Tabelle 3.7: Wichtige Counter

part, chapter,. . . , subparagraph	Zähler für part, chapter,. . .
enumi, enumii, enumiii, enumiv	Aufzählungszähler für die Ebenen 1 bis 4
page	Seitennummer
footnote	Fußnotenzähler
equation	Formel-Zähler
figure	Bilder-Zähler
table	Tabellen-Zähler
secnumdepth	Nummerierungstiefe bei Überschriften
tocdepth	Nummerierungstiefe im Inhaltsverzeichnis

Der Wert eines Zählers kann mit folgenden Befehlen manipuliert werden:

`\setcounter{`*Counter*`}{`*neuer Wert*`}`	Weist dem Counter den neuen Wert zu
`\addtocounter{`*Counter*`}{`*Zahl*`}`	Addiert die Zahl (auch negative Zahl möglich) zum Wert des Counters
`\stepcounter{`*Counter*`}`	Inkrement 1

Das Auslesen oder die Abfrage der verschiedenen Counter sowie die Auswahl des Nummerierungsstils geschieht mittels folgender Makros:

`\value{`*Counter*`}`	Wert; kann überall dort als Argument angegeben werden, wo LaTeX eine Wertangabe erwartet. Erzeugt keine Ausgabe.
`\arabic{`*Counter*`}`	arabische Ziffer
`\Roman{`*Counter*`}`	große römische Ziffern
`\roman{`*Counter*`}`	kleine Römische Ziffern
`\alph{`*Counter*`}`	Kleinbuchstaben a-z
`\Alph{`*Counter*`}`	Großbuchstaben A-Z
`\fnsymbol{`*Counter*`}`	spezielle 9 Symbole, gedacht für die Fußnoten in der Titelei

erfolgen. Die dokumentenklasseninterne Abfrage, z. B. zur Ausgabe der Seitenzahl oder der Kapitelnummerierung geschieht jedoch mit

`\the`*Countername*

zum Beispiel also

```
\thepage
```

Über diese Befehle kann man auch die Art der Nummerierung ändern. Möchte man zum Beispiel bei einer Aufzählung Buchstaben statt Zahlen verwenden, so kann man dies durch Umdefinition des Befehles (Abschnitt 3.9) `\theenumi` erreichen:

```
\renewcommand{\theenumi}{\alph{enumi}}
```

Bei den Seitennummerierungen kann der Stil entweder direkt mit dem Seitenstil definiert sein oder durch

`\pagenumbering{`*Nummernstil*`}`

gesetzt werden. Die Nummerierungsstile sind wieder `arabic`, `roman`, `Roman`, `alph`, `Alph` oder `fnsymbol`. Die Seitenzahl kann jederzeit mit dem `\setcounter`-Befehl neu gesetzt werden und mit `\thepage` abgefragt werden.

Eigene Counter definieren

Ein beliebiger neuer Counter wird mit

`\newcounter{`*Countername*`}`

definiert. Er kann auch als eine sogenannte Parkvariable bei Umdeklaration von anderen Countern dienen. Zusätzlich kann man optional einen weiteren Counter als Reset-Referenz angeben. Wird dieser Referenzcounter verändert, so wird der neu definierte Counter zurückgesetzt.

```
\newcounter{mycounter}[section]
```

Das Beispiel definiert einen neuen Counter namens `mycounter`, der zu Beginn jedes neuen Abschnittes zurückgesetzt wird.

3.11 Längen

LaTeX verwaltet mit der Prozessierung eines Dokuments auch einige Längen wie z. B. die Breite des Textes. Einige Beispiele für wichtige *Längen* sind in Tabelle 3.8 zu finden. Die Manipulation einer Länge funktioniert dabei relativ ähnlich zu den Countern und kann mit einem der folgenden Befehle durchgeführt werden.

`\setlength{`*Länge*`}{`*neues Maß*`}`	Länge auf neues Maß setzen
`\setlength{`*Länge*`}{`*Maß* `plus`*Maß1* `minus`*Maß2*`}`	elastische Definition einer Länge
`\addtolength{`*Länge*`}{`*Maß*`}`	Addition des Maßes zur Länge

Zum Beispiel

```
\setlength{\textwidth}{15cm}
\setlength{\textwidth}{\paperwidth}
\setlength{\parskip}{1ex plus0,5ex minus 0,2ex}
```

Jedoch sollte man mit der direkten Änderung von Längen sehr vorsichtig sein. Es empfiehlt sich, Änderungen an Maßen die den Satzspiegel beeinflussen, weitestgehend zu unterlassen beziehungsweise schnellstmöglich wieder rückgängig zu machen.

Bei manchen Längen empfiehlt es sich, LaTeX ein gewisses Spektrum zu geben, aus dem es die Länge wählen kann. So kann ein Abstand elastisch von LaTeX aus diesem Spektrum gewählt werden.

Tabelle 3.8: Wichtige Längen

`\arraycolsep`	halbe Breite des Spaltenzwischenraums für `array`s
`\arrayrulewidth`	Dicke der Linien in Tabellen
`\baselineskip`	Abstand zwischen zwei Zeilen innerhalb eines Absatzes
`\doublerulesep`	Abstand von Doppellinien
`\evensidemargin`	linker Rand für gerade Seiten
`\footskip`	Abstand Unterkante Rumpf bis Unterkante Fußzeile
`\headsep`	Abstand Unterkante Kopf bis Oberkante Rumpf
`\oddsidemargin`	linker Rand allgemein oder für ungerade Seiten
`\paperheight`	Seitenhöhe
`\paperwidth`	Seitenbreite
`\parindent`	Einrückabstand der ersten Zeile eines Absatzes
`\parskip`	Absatzabstand
`\tabcolsep`	halbe Breite des Spaltenzwischenraums für `tabular`s
`\textheight`	Texthöhe
`\textwidth`	Textbreite
`\topmargin`	Abstand oberer Rand bis Oberkante Kopfzeile
`\topskip`	Abstand Oberkante Rumpf bis Grundlinie der ersten Zeile

```
\setlength{\baselineskip}{1.2em plus2em}
```

Alle Längenangaben erfordern natürlich immer auch eine Einheit. Eindeutig ist z. B. der Bruchteil einer anderen Länge (z. B. 0.5`\textwidth`) oder eine Einheit an die Zahl angehängt, wie:

`cm`	Zentimeter	`bp`	big point (1 in = 72 bp)
`mm`	Millimeter	`dd`	Didot (1157 dd = 1238 pt)
`in`	Inches (2,54 cm)	`cc`	Cicero (1 cc = 12 dd)
`pt`	Punkte (1 in = 72,27 pt)	`sp`	scaled point (1 pt = 65536 sp)
`pc`	Picas (1 pc = 12 pt)		
`em`	Buchstabenbreite von M[7]		
`ex`	Buchstabenhöhe von x		

Bei manchen Befehlen wird vom Autor eine eigene Längenangabe gefordert. Manchmal ist es aber auch wünschenswert die Längenangabe gerade so zu wählen, wie ein einzugebender Text breit oder hoch ist. Hierfür sind folgende zwei Befehle sehr nützlich, die eine beliebige Textbreite oder Texthöhe in einer Länge ablegen:

```
\settowidth{Länge}{Text}
\settoheight{Länge}{Text}
```

So ermöglichen diese Makros zum Beispiel Rahmenboxen mit einer festen Breite anzufertigen.

```
\newlength{\templength}
\settowidth{\templength}{eine Zeile Text}
\fbox{\parbox{\templength}{kurze Zeile\\eine Zeile Text\\
   eine Lange Zeile Text}}
```

Das Ergebnis sieht wie folgt aus:

[7] 1 em entspricht in etwa einem Geviert, der typischen typografischen Maßeinheit, die noch aus der Zeit des Bleisatzes mit beweglichen Lettern stammt.

kurze Zeile
eine Zeile Text
eine lange Zeile
Text

Die `fbox` passt sich genau der Größe der innen liegenden `parbox` an. Die `parbox` ist so breit wie der Text „eine Zeile Text“, die lange Zeile wird entsprechend umgebrochen.

Eigene Längen können analog zu Zählern definiert werden mit:

`\newlength{`*neue Länge*`}`

definiert werden (sehr nützlich zum Abspeichern von Maßen).

Somit kann man zum Beispiel temporär den Absatzabstand auf 1 cm setzen.

Ein kleines bisschen Text mit nachfolgendem normalen Ansatzumbruch.

Ein kleines bisschen Text mit nachfolgendem Absatzabstand von 1cm.

Ein kleines bisschen Text mit nachfolgendem normalen Absatzumbruch.

Ein kleines bisschen Text.

Erzeugt mit dem Code:

```
Ein kleines bisschen Text mit nachfolgendem normalen
   Ansatzumbruch.\par
Ein kleines bisschen Text mit nachfolgendem Absatzabstand
   von 1cm.
\newlength{\parskipsaved}
\setlength{\parskipsaved}{\parskip}
\setlength{\parskip}{1cm}\par
Ein kleines bisschen Text mit nachfolgendem normalen
   Absatzumbruch.
\setlength{\parskip}{\parskipsaved}\par
Ein kleines bisschen Text.
```

3.12 Zwischenräume

Beliebige horizontale bzw. vertikale Zwischenräume sind mit

`\hspace{`*Maß*`}`
`\hspace*{`*Maß*`}`
`\vspace{`*Maß*`}`
`\vspace*{`*Maß*`}`

möglich, z. B.: dies ist eine horizontale Lücke von 1 cm. Der optionale Stern erzeugt den Zwischenraum auch wenn Zeilen- oder Seitenumbrüche involviert sind. Steht zusätzlich vor oder nach dem Befehl ein Leerzeichen, so wird dieses zum Abstand hinzugefügt:

```
Das ist\hspace{1cm}1\,cm.
Das ist \hspace{1cm}1\,cm.
Das ist \hspace{1cm} 1\,cm.
```

Das ist 1 cm.
Das ist 1 cm.
Das ist 1 cm.

Ein weiterer sehr nützlicher Befehl, bei dem LaTeX das Maß selber elastisch vorgibt ist

`\hfill` bzw. `\hspace{\fill}`
`\vfill` bzw. `\vspace{\fill}`

Dabei wird soviel Zwischenraum eingefügt, dass die laufende Zeile (Seite) links- und rechtsbündig (oben und unten bündig) abschließt. Ein mehrfaches Anwenden führt zusätzlich zu gleichen Abständen, ein Beispiel:

Herr Müller Brief Regensburg, den 10. März 2015

```
Herr Müller \hfill Brief \hfill Regensburg, den \today
```

Ähnliche Zwischenraumbefehle:

`\quad`	Zwischenraum der Breite der aktuellen Schriftgröße, also 10 pt bei 10 pt
`\qquad`	Zweimal `\quad` als Zwischenraum
`\dotfill`	Zwischenraum mit Punkten füllen
`\hrulefill`	Zwischenraum mit Strichen füllen
`\,`	3/18 em Zwischenraum
`\:`	4/18 em Zwischenraum
`\;`	5/18 em Zwischenraum
`\!`	-3/18 em Zwischenraum
`\␣`[8]	ein gewöhnliches Leerzeichen

Diese Makros können auch innerhalb von Mathe-Umgebungen benutzt werden (siehe Seite 102). Auch eine Kombination der `fill`-Befehle ist möglich.

Abfahrt .... 8:30 Ankunft _____ 11:45

```
Abfahrt\dotfill\dotfill\dotfill\ 8:30 \hfill\hfill
Ankunft \hrulefill\ 11:45\\
```

3.13 Vertikale Abstände

Analog zu den horizontalen Zwischenräumen, existieren die Makros

`\vspace{`*Maß*`}`
`\vspace*{`*Maß*`}`
`\vfill` bzw. `\vspace{\fill}`

[8] Backslash gefolgt von einem Leerzeichen

Sauber verwendet werden diese Makros bei Absatzumbrüchen, da LaTeX an anderen Positionen keine Notwendigkeit für vertikale Abstände sieht. Zum Beispiel ist der folgende Absatzumbruch 1 cm groß.

Erzeugt mit

```
\par\vspace{1cm}
```

Der Abstand folgt üblicherweise dem Absatzumbruch. Dies gehört schlicht und einfach zum „guten Ton“.

Weitere Makros um den Abstand zwischen zwei Absätzen zu vergrößern, sind die schriftgrößenabhängigen Befehle

`\bigskip`	`12 plus 4pt minus 4pt`
`\medskip`	`6pt plus 2pt minus 2pt`
`\smallskip`	`3pt plus 1pt minus 1pt`

4 Textformatierungen

4.1 Schriftarten und Textauszeichnung

Schriftarten werden nach Sippe, Familie, Form und Serie klassifiziert. Die Standard-Schriftsippe in LaTeX ist Computer Modern. Wie jedoch bereits in Abschnitt 3.3.3 erwähnt, sollte sie durch die erweiterte Version Latin Modern ersetzt werden. Falls gewünscht, ist auch das Einbinden anderer Schriftarten möglich. Dies funktioniert vollkommen analog zu Latin Modern. Beim Ändern der Schriftsippe ist jedoch immer zu beachten, dass sie alle benötigten Zeichen auch tatsächlich enthält. So bietet zum Beispiel nicht jede Schriftsippe mathematische Symbole oder alle Familien. Eine sehr gute Übersicht über die möglichen Schriftarten und deren Verwendung bietet „The LaTeX Font Catalogue“ `http://www.tug.dk/FontCatalogue/`.

Hier wird als Beispiel lediglich die Sippe Latin Modern genauer betrachtet um das Prinzip zu verstehen, wie die Änderung der Schriftart oder auch Schriftattribute in LaTeX funktionieren. Alle Befehle funktionieren für andere Schriftarten genauso gut, solange der Zeichensatz dies unterstützt.

4.1.1 Die Schriftsippe Latin Modern

Die Sippe Latin Modern gliedert sich in drei Familien

`\rmfamily` bzw. `\textrm{`*Text*`}`	Latin Modern Roman	mit Serifen
`\sffamily` bzw. `\textsf{`*Text*`}`	Latin Modern Sans	ohne Serifen
`\ttfamily` bzw. `\texttt{`*Text*`}`	`Latin Modern Typewriter`	Monofont

Jede dieser Familien Unterteilt sich wieder in verschiedene Formen und Serien. Tabelle 4.1 zeigt die vefügbaren Kombinationen für Latin Modern.

Da die aufrechte Form `\upshape` und die Serie Medium `\mdseries` Standard ist machen diese Befehle nur Sinn, wenn man die Form und Serie zuvor geändert hat.
Möchte man wieder zum Standardfont des Dokumentes wechseln, benötigt man lediglich einen der Befehle

```
\normalfont
\textnormal{Text}
```

Dies erlaubt es auch einen bestimmten Schrifttyp zu wählen, unabhängig davon, welcher Schrifttyp zuvor aktiviert war.

Tabelle 4.1: Übersicht über die verfügbaren Kombinationen aus Familie, Form und Serie für die Latin Modern Schriftsippe

Familie	*Form*	*Serie*	*Schalter*	*Befehl*	
Roman			`\rmfamily`	`\textrm{`*Text*`}`	Beispieltext
	aufrecht		`\upshape`	`\textup{`*Text*`}`	Beispieltext
		medium	`\mdseries`	`\textmd{`*Text*`}`	Beispieltext
		fett	`\bfseries`	`\textbf{`*Text*`}`	**Beispieltext**
	kursiv		`\itshape`	`\textit{`*Text*`}`	*Beispieltext*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*Beispieltext*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***Beispieltext***
	geneigt		`\slshape`	`\textsl{`*Text*`}`	*Beispieltext*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*Beispieltext*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***Beispieltext***
	Kapitälchen		`\scshape`	`\textsc{`*Text*`}`	BEISPIELTEXT
Sans Serif			`\sffamily`	`\textsf{`*Text*`}`	Beispieltext
	aufrecht		`\upshape`	`\textup{`*Text*`}`	Beispieltext
		medium	`\mdseries`	`\textmd{`*Text*`}`	Beispieltext
		fett	`\bfseries`	`\textbf{`*Text*`}`	**Beispieltext**
	geneigt		`\slshape`	`\textsl{`*Text*`}`	*Beispieltext*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*Beispieltext*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***Beispieltext***
`Typewriter`			`\ttfamily`	`\texttt{`*Text*`}`	`Beispieltext`
	aufrecht		`\upshape`	`\textup{`*Text*`}`	`Beispieltext`
		medium	`\mdseries`	`\textmd{`*Text*`}`	`Beispieltext`
		fett	`\bfseries`	`\textbf{`*Text*`}`	**`Beispieltext`**
	kursiv		`\itshape`	`\textit{`*Text*`}`	*`Beispieltext`*
	geneigt		`\slshape`	`\textsl{`*Text*`}`	*`Beispieltext`*
		medium	`\mdseries`	`\textmd{`*Text*`}`	*`Beispieltext`*
		fett	`\bfseries`	`\textbf{`*Text*`}`	***`Beispieltext`***
	Kapitälchen		`\scshape`	`\textsc{`*Text*`}`	BEISPIELTEXT

Kursivkorrektur

Wenn ein aufrechtes Zeichen einem kursiven folgt, so kann – je nach verwendeter Schriftart – der Abstand zwischen beiden zu klein werden, sodass beide Zeichen überlappen. Dies kann man mit der sogenannten Kursivkorrektur beheben:

`\/`

Das Makro `\textit` erledigt dies in so gut wie allen Fällen automatisch. Bei Benutzung des Schalterbefehles `\itshape` muss die Korrektur jedoch manuell gesetzt werden. [Beispiel nach 27, S. 97]

[Das *Schiff*]	`[Das {\itshape Schiff}] \\\relax`
[Das *Schiff*]	`[Das \textit{Schiff}] \\\relax`
[Das *Schiff*]	`[Das {\itshape Schiff\/}]`

Ligaturen

Ligaturen sind die Zusammenfassung mehrerer Buchstaben zu einem einzigen Zeichen. Sie stammen aus dem klassischen Bleisatz und wurden damals aus Stabilitätsgründen als einzelnes Zeichen gedruckt [27, S.97]. Dieses Verhalten wurde im Digitaldruck beibehalten. Somit werden heutzutage hauptsächlich die Buchstabenfolgen `fl`,`ff`,`fi`,`ffi` und `ffl` in Ligaturen gewandelt. Je nach Schriftart gibt es auch mehr oder teilweise auch gar keine Ligaturen.

Liegt die Ligatur an einer Trennstelle, so sollte sie aufgebrochen werden. Mit Verwendung des babel-Paketes geht dies mithilfe der Zeichenfolge

`"|`

zum Beispiel [nach 27, S. 98]

stofflich stofflich Tiefflieger Tiefflieger Auflage Auflage

erzeugt mit

```
stofflich stoff"|lich Tiefflieger Tief"|flieger Auflage Auf"|lage
```

4.1.2 Schriftgrößen

Die Schriftgrößen skalieren immer mit der im Befehl `\documentclass` festgelegten Standardgröße für die Schrift. So wird mit `\normalsize` die Standardschriftgröße abgefragt. Alle zur Auswahl stehenden Schriftgrößen mitsamt Beispiel findet man in Tabelle 4.2.

4.1.3 Auszeichnung von Text

Es gibt verschiedene Möglichkeiten Text hervorzuheben. Aus typografischen Gründen empfiehlt sich der Befehl:

`\emph{`*Text*`}`

Er setzt den Text kursiv, erlaubt jedoch im Gegensatz zu `\textit` Schachtelungen:

```
\emph{Erste Hervorhebungsebene \emph{zweite Ebene} Ende der ersten}
```

Erste Hervorhebungsebene zweite Ebene *Ende der ersten*

Tabelle 4.2: Skalierung der Schriften bei Hauptschrift 10pt und Angaben der tatsächlichen Schriftgrößen bei unterschiedlichen Hauptschriftgrößen

Schalter	*Beispiel*	*Tatsächliche Größen:*	10pt	11pt	12pt
`\tiny`	tiny		5pt	6pt	6pt
`\scriptsize`	scriptsize		7pt	8pt	8pt
`\footnotesize`	footnotesize		8pt	9pt	10pt
`\small`	small		9pt	10pt	11pt
`\normalsize`	normalsize		10pt	11pt	12pt
`\large`	large		12pt	12pt	14pt
`\Large`	Large		14pt	14pt	17pt
`\LARGE`	LARGE		17pt	17pt	20pt
`\huge`	huge		20pt	20pt	25pt
`\Huge`	Huge		25pt	25pt	25pt

Selbstverständlich kann man Hervorhebungen auch durch Änderung der Schriftattribute setzen, allerdings sollte man dabei immer darauf achten, den Textfluss nicht durch zu häufige oder zu auffällige Änderungen zu unterbrechen. Als weitere Möglichkeit der Auszeichnung ist auch eine Unterstreichung mit

`\underline{`*Text*`}`

möglich. Sie sollte jedoch aus typografischen Gründen ausschließlich dort verwendet werden, wo auf andere Möglichkeiten nicht zurückgegriffen werden kann.

4.1.4 KOMA-Spezielle Schriftänderungen

Wenn man dauerhaft die Schrifteinstellungen für verschiedene spezielle Boxen, wie die Kopf- und Fußzeile (siehe Kapitel 7), verändern will, kann man dies mit einem der folgenden Befehle erreichen:

`\setkomafont{`*Element*`}{`*Schriftformatierungsbefehle*`}`
`\addtokomafont{`*Element*`}{`*Schriftformatierungsbefehle*`}`

`setkomafont` definiert die Formatierung des Elements völlig neu, während `addtokomafont` die existierende Definition erweitert. Mithilfe des Befehls

`\usekomafont{`*Element*`}`

kann man direkt auf die Schriftart von einzelnen Elementen zugreifen bzw. umschalten.

Als Elemente stehen unter anderem die Folgenden zur Verfügung [siehe auch 10, Tabelle 3.2]:

`caption` Gleitumgebungsbeschreibung (Abschnitt 13.2)
`captionlabel` Gleitumgebungslabel (Abschnitt 13.2)
`chapter` Kapitelüberschriften (Abschnitt 3.6)
`chapterentry` Inhaltsverzeichniseintrag von Kapiteln (Abschnitt 3.7)
`chapterentrypagenumber` Seitenzahl des Inhaltsverzeichniseintrages von Kapiteln (Abschnitt 3.7)

chapterprefix Präfix bei Einstellung `chapterprefix=true` beziehungsweise `appendixprefix=true` (Tabelle 3.1)

descriptionlabel Label einer `description`-Umgebung (Kapitel 8)

footnote Marke und Text von Fußnoten (Abschnitt 5.2)

footnotelabel Marke einer Fußnote, überschreibt das Element `footnote` (Abschnitt 5.2)

footnotereference Referenzierung dr Fußnotenmarke (Abschnitt 5.2)

footnoterule Linie über den Fußnoten (Abschnitt 5.2)

labelinglabel Label der `labeling`-Umgebung (Abschnitt 8.3)

labelingseparator Trennzeichen einer `labeling`-Umgebung (Abschnitt 8.3)

minisec Überschrift einer `\minisec` (Tabelle 3.6)

pagefoot Seitenfuß bei Verwendung von scrlayer-scrpage (Abschnitt 7.1)

pageheadfoot Seitenkopf und -fuß (Kapitel 7)

pagenumber Seitenzahl, die mit `\pagemark` gesetzt wird (Kapitel 7)

paragrah Überschrift von Paragrafen (Abschnitt 3.6)

part Überschrift der Ebene `\part` (Abschnitt 3.6)

partentry Inhaltsverzeichniseintrag von `\part` (Abschnitt 3.7)

partentrypagenumber Seitenzahl des Inhaltsverzeichniseintrages von `\part` (Abschnitt 3.7)

partnumber Zeile mit Nummer von `\part` in der Überschrift (Abschnitt 3.6)

section Abschnittsüberschrift (Abschnitt 3.6)

sectionentry Inhaltsverzeichniseintrag eines Abschnittes (nur bei scrartcl verfügbar, Abschnitt 3.7)

sectionentrypagenumber Seitenzahl des Inhaltsverzeichniseintrages eines Abschnittes (nur bei scrartcl verfügbar, Abschnitt 3.7)

sectioning alle Gliederungsüberschriften, sowie die Überschrift der Zusammenfassung (Abschnitt 3.6)

subject Typisierung des Dokumentes (Abschnitt 3.4)

subparagraph Überschrift von `\subparagraph` (Abschnitt 3.6)

subsection Überschrift von `\subsection` (Abschnitt 3.6)

subsubsection Überschrift von `\subsubsection` (Abschnitt 3.6)

Beispiel: Soll `captionlabel` genauso wie `descriptionlabel` aussehen, so ginge das mit

```
\setkomafont{captionlabel}{\usekomafont{descriptionlabel}}
```

4.2 Umbrüche

4.2.1 Absatzumbruch

LaTeX orientiert sich bei der Formatierung des Textes nicht an Zeilen, sondern an Absätzen. Es wird somit zunächst der gesamte Inhalt eines Absatzes analysiert und dann die beste Möglichkeit des Zeilenumbruchs automatisch gewählt. Zeilenumbrüche im Code werden somit einfach ignoriert. Der Umbruch von Absätzen erfolgt hingegen einfach durch eine Leerzeile, oder mit dem Befehl

```
\par
```

Die Art und Weise, wie Absatzümbrüche gekennzeichnet werden sollen, wird über die Dokumentenklassenoption

`parskip=`*Methode*

festgelegt. Die Unterschiedlichen Methoden für den Umbruch finden sich in Tabelle 4.3. Hierzu bleibt zu sagen, dass man im englischsprachigen Raum üblicherweise die erste Zeile eines Absatzes einrückt und keinen Abstand beim Absatzumbruch sondern lediglich am Ende eines Abschnittes lässt (`parskip=false`, Standard). Im deutschsprachigen Raum hingegen werden Absätze durch einen vertikalen Abstand von einer halben oder ganzen Zeile gekennzeichnet (`parskip=full` oder `half`). Am Ende eines Abschnittes lässt man mehr Platz.

Tabelle 4.3: Werte für die Dokumentklassenoption `parskip`

`false`	Einzug von 1 em in der ersten Zeile, wobei der erste Absatz eines Abschnitts nicht eingezogen wird
`full`	Vertikaler Abstand von einer Zeile, Absatzenden haben Leerraum von mindestens 1 em
`full-`	Vertikaler Abstand von einer Zeile
`full+`	Vertikaler Abstand von einer Zeile, Absatzenden haben Leerraum von mind. 1/4 einer normalen Zeile
`full*`	Vertikaler Abstand von einer Zeile, Absatzenden haben Leerraum von mind. 1/3 einer normalen Zeile
`half`	Vertikaler Abstand von 1/2 Zeile, Absatzenden haben Leerraum von mindestens 1 em
`half-`	Vertikaler Abstand von 1/2 Zeile
`half+`	Vertikaler Abstand von 1/2 Zeile, Absatzenden haben Leerraum von mind. 1/4 einer normalen Zeile
`half*`	Vertikaler Abstand von 1/2 Zeile, Absatzenden haben Leerraum von mind. 1/3 einer normalen Zeile

Möchte man den Einzug der ersten Zeile lediglich lokal verhindern, oder einen Einzug an einer Stelle, wo normalerweise keiner stattfinden würde setzen, so benutzt man die Makros

```
\noindent
\indent
```

Zusätzlich kann man die Gestaltung der Absatzumbrüche auch über folgende Parameter beeinflussen:

`\parskip` Der Abstand zwischen zwei Absätzen, diese Länge sollte dehnbar definiert werden. Ihr Standardwert lautet `4.0pt plus 2.0pt minus 1.0pt`.

`\parindent` Die Länge des Einzuges am Absatzanfang, Standardwert ist 1 em.

4.2.2 Zeilenumbruch

`\\`	Zeilenumbruch (einfach)
`\\[`*Abstand*`]`	Zeilenumbruch mit optionalen Abstand[1] zur nächsten Zeile
`\\*[`*Abstand*`]`	Wie \\, jedoch kann kein Seitenumbruch vor der nächsten Zeile stattfinden.
`\newline`	Zeilenumbruch, wie \\
`\linebreak[`*Priorität*`]`	Zeilenumbruch, Zahl entspricht dabei der Priorität (0=niedrig bis 4=zwingend).

Vorsicht: Im LR-Modus (vgl. Abschnitt 9.1) von LaTeX (z.B. `\mbox{...}`) ist kein Zeilenumbruch erlaubt, ein Befehl für einen Zeilenumbruch wird ignoriert und erzeugt eine Warnung.

Um den Unterschied zwischen den verschiedenen Makros zu verdeutlichen hier ein kleines Beispiel:

> Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den Unterschied zwischen `\linebreak` und `\\` zu demonstrieren. (ohne manuellen Umbruch)
>
> Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den Unterschied zwischen `\linebreak` und `\\` zu demonstrieren. (mit `\linebreak`)
>
> Dies ist eine sehr lange Zeile Text bis zu einem Umbruch, um den
> Unterschied zwischen `\linebreak` und `\\` zu demonstrieren. (mit `\\`)

`\linebreak` hält somit den Blocksatz trotz allem ein. Die Priorität gibt dabei an inwieweit die Vorgaben für den Wortabstand berücksichtigt werden müssen. Ohne Angabe der Priorität entspricht das Makro dem Wert `4` und erzwingt somit einen Umbruch.

4.2.3 Zeilenumbruch verhindern

`...~...`	sog. „geschütztes Leerzeichen“, kein Zeilenumbruch zwischen Wörtern, Beispiel `Stufe~5`
`\nolinebreak[`*Priorität*`]`	kein Zeilenumbruch, mit Angabe der Priorität (0=niedrig bis 4=zwingend)

4.2.4 Zeilenabstand ändern

Für das Ändern des Zeilenabstands bieten sich zwei Möglichkeiten an: Möchte man den Zeilenabstand auf das eineinhalbfache oder das doppelte umschalten, so bindet man das Paket setspace ein

```
\usepackage{setspace}
```

und verwendet die Befehle

```
\onehalfspacing
\doublespacing
```

Möchte man zum einfachen Zeilenabstand zurückkehren, so verwendet man den Befehl:

```
\singlespacing
```

Alternativ kann man auch die Paketoptionen *onehalfspacing* und *doublespacing* benutzen.

Wenn man die Schriftart gewechselt hat und deshalb nur eine kleine Korrektur am Zeilenabstand nötig ist, genügt folgender Befehl (setspace-Paket nicht benötigt):

```
\linespread{Faktor}\selectfont
```

Der Standarddurchschuss beträgt 1.2, somit muss für einen eineinhalbfachen Zeilenabstand der *Faktor* 1.25 gewählt werden. Wichtig ist, dass der Befehle `\linespread` allein keinerlei Änderung bewirkt, die Änderung wird erst durch `\selectfont`, also durch neues Laden der Zeichentabelle aktiviert.

[1] Der Abstand muss mit Einheit angegeben werden, allerdings sind relative Einheiten, wie 1 ex erlaubt.

4.2.5 Seitenumbruch

`\newpage`	Seitenumbruch, Gleitobjekte(vgl. Kapitel 13) können hier jedoch noch positioniert werden
`\pagebreak[`*Priorität*`]`	Seitenumbruch mit Angabe der Priorität (0=niedrig bis 4=zwingend)
`\clearpage`	beendet die aktuelle Seite sofort; Übrige Gleitobjekte werden direkt im Anschluss ausgegeben (vgl. Kapitel 13)
`\cleardoublepage`	wie `\clearpage`, allerdings beginnt der Text mit der nächsten ungeraden (rechten) Seite

Der Unterschied zwischen einem Seitenumbruch mit `\newpage` und `\pagebreak` ist quasi derselbe, wie zwischen `\newline` und `\linebreak`: Bei `\pagebreak` wird der Blocksatz eingehalten. Jedoch wird hier nicht der Wortabstand entsprechend vergrößert, sondern der Text bis zum Zeilenende ausgeschrieben, bevor ein Seitenumbruch erfolgt. `\pagebreak` innerhalb eines Absatzes erzwingt somit lediglich einen Seitenumbruch am Zeilenende, wohingegen `\newpage` sofort die Seite beendet und im Anschluss an eventuell ausgegebene Gleitobjekte, siehe Kapitel 13, auf einer neuen Seite fortsetzt.

`\clearpage` beendet ebenfalls die Seite sofort wie `\newpage` und setzt im Anschluss auf einer oder mehreren Seiten sämtliche noch zu setzende Gleitobjekte, siehe ebenfalls Kapitel 13.

Zweispaltiger Textsatz

Wenn die Dokumentenklassenoption `twocolumn` gesetzt ist, oder das Makro `\twocolumn` aufgerufen wurde, dann beenden `\pagebreak` und `\newpage` lediglich die aktuelle Spalte und beginnen eine neue. Möchte man die komplette Seite beenden, was möglicherweise eine leere Rechte Spalte zur Folge hat, benötigt man eines der Makros `\clearpage` oder `\cleardoublepage`

4.2.6 Seitenumbruch verhindern

`\nopagebreak[`*Priorität*`]`	kein Seitenumbruch, mit Angabe der Priorität (0=niedrig bis 4=zwingend)
`\begin{samepage}...\end{samepage}`	Seitenumbruch nur zwischen Absätzen, außer er wird erzwungen

`\nopagebreak` zwischen zwei Absätzen verhindert dort einen Seitenumbruch. Innerhalb eines Absatzes verhindert es einen Seitenumbruch nach der aktuellen Zeile.

4.2.7 Unsaubere Seitenumbrüche – Seitengröße in Sonderfällen anpassen

In Sonderfällen ist es manchmal erforderlich den Textbereich einer einzelnen Seite zu vergrößern, da beispielsweise die nächste Seite lediglich eine Textzeile zeigt. Für diesen Fall gibt es das Makro

`\enlargethispage{`*Länge*`}` `\enlargethispage*{`*Länge*`}`

Der Stern sorgt dafür, dass die relevanten elastischen Maße (z. B. der Zeilenabstand) auf den Minimalwert gesetzt werden, um die Abweichung gegenüber den anderen Seiten möglichst gering zu halten. Beispielsweise sieht eine Vergrößerung des Textbereiches um eine einzelne Zeile so aus:

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

4.3 Trennung und Bindestrich

4.3.1 Worttrennung

LaTeX trennt Wörter, wenn der Zeilenumbruch rechtsbündig zwischen Wörtern nicht gelingt. Allerdings kennt LaTeX den Sinn der Wörter nicht und trennt deswegen manche zusammengesetzten Wörter nicht korrekt oder sinnvoll, z. B.: Urinstinkt.

`\-`	Trennmöglichkeit, die andere Trennungen ausschließt (`Ur\-instinkt`, `Stau\-becken`)
`"-`	Trennmöglichkeit, die andere Trennungen nicht ausschließt (wenn LaTeX eine Trennstelle einfach nicht kennt)
`""`	Trennmöglichkeit, bei der kein Trennstrich benötigt wird
`"|`	Auflösen einer Ligatur und Schaffung einer Trennmöglichkeit (`Auf"|forderung` schreibt Auforderung statt Aufforderung)

Beispiele für die verschiedenen Trennmöglichkeiten und Bindestriche finden sich in Beispiel 2.

Kommt ein Wort öfters vor, sollte es in die Trennnungsliste mit aufgenommen werden, mittels

```
\hyphenation{Stau-be-cken}
```

Im obigen Beispiel wird das Wort „Staubecken“ im ganzen Dokument nur an den vorgegebenen Stellen getrennt. Sollen Trennstellen für mehrere Wörter angegeben werden, so werden sie als Liste innerhalb des `\hyphenation`-Makros durch ein Leerzeichen getrennt angegeben, z. B.

```
\hyphenation{Ma-nu-skript Com-pu-ter Stau-be-cken}
```

4.3.2 Geschützte Leerzeichen

Wie bereits in Abschnitt 4.2.6 erwähnt kann man den Zeilenumbruch zwischen zwei Wörtern verhindern, indem man sie durch ein geschütztes Leerzeichen trennt. Dies ist insbesondere dann nötig, wenn eine Trennung die Lesbarkeit verschlechtern würde.

`~`	geschütztes Leerzeichen mit normalen Abstand. Beispiel: `Dr.~Mustermann`
`\,`	geschütztes Leerzeichen mit kleineren Abstand. Beispiel: `z.\,B.` oder `50\,kg`

4.3.3 Bindestrich

Beispiele für die verschiedenen Trennmöglichkeiten und Bindestriche finden sich in Beispiel 2.

`-`	Bindestrich, der andere Trennungen unterdrückt
`"=`	Bindestrich, der andere Trennungen erlaubt
`"~`	Bindestrich, an dem nicht getrennt werden darf

Beispiel 2: Unterschiedliche Arten der Worttrennung (vgl. [27, Bsp. 03-07-07])

1 Jetzt kommt hier eine ganz normale Trennstelle
2 Jetzt kommt hier eine ganz normale Trennstelle
3 Jetzt kommt hier eine Bindestrich-Trennstelle
4 Jetzt kommt hier eine echte Binde-strich-Trennstelle
5 Jetzt kommt hier eine echte Binde-strich-Trennstelle
6 Jetzt kommt hier eine echte Bin-de-strich-Trennstelle
7 Jetzt kommt hier eine echte Linie/Kurve-Trennstelle
8 Jetzt kommt hier eine echte Linie/Kurve-Trennstelle
9 Jetzt kommt hier eine echte Linie/Kurve-Trennstelle

```
\begin{enumerate}[nosep]
\item[1] Jetzt kommt hier eine
   ganz normale Trennstelle
\item[2] Jetzt kommt hier eine
   ganz nor\-male Trennstelle
\item[3] Jetzt kommt hier eine
   Bindestrich-Trennstelle
\item[4] Jetzt kommt hier eine
   echte Binde"=strich-Trennstelle
\item[5] Jetzt kommt hier eine
   echte Binde"~strich-Trennstelle
\item[6] Jetzt kommt hier eine
   echte Bin\-de"~strich Trennstelle
\item[7] Jetzt kommt hier eine
   echte Linie/Kurve-Trennstelle
\item[8] Jetzt kommt hier eine
   echte Linie/""Kurve-Trennstelle
\item[9] Jetzt kommt hier eine
   echte Linie\slash{}Kurve-Trennstelle
\end{enumerate}
```

4.4 Textausrichtung

Standardmäßig setzt LaTeX jeden Text im Blocksatz, also links- und rechtsseitig bündig. Es gibt daher keinen Befehl für Blocksatz, man erhält ihn indem man eine andere Form der Ausrichtung beendet. Um Text im Flattersatz zu setzen gibt es die in Tabelle 4.4 dargestellten Befehle und Umgebungen.

Zusätzlich existiert noch das TeX-Makro

`\centerline{`*Text*`}`

es dient dazu eine einzelne Zeile zu zentrieren. Zusätzlich zu dieser Ausrichtung ist auch eine Absatzeinrückung möglich. Standardmäßig wird diese mit Hilfe der beiden Umgebungen

```
\begin{quote}...\end{quote}
\begin{quotation}...\end{quotation}
```

gesetzt. Bei beiden wird der Text abgesetzt und beidseitig eingerückt. Sie unterscheiden sich lediglich dadurch, dass bei `quote` der Beginn eines neuen Absatzes durch Abstand und bei `quotation` durch Einrückung gekennzeichnet wird.

KOMA-Script fügt zudem einen Befehl hinzu der es erlaubt einzelne Absätze beliebig weit einzurücken:

```
\begin{addmargin}[linker Einzug]{rechter Einzug} ... \end{addmargin}
\begin{addmargin*}[innerer Einzug]{äußerer Einzug} ... \end{addmargin*}
```

Die Stern-Variante ist für zweiseitigen Textsatz gedacht und addiert den Einzug anstatt von links und rechts, auf der Innen- bzw. Außenseite. Wird lediglich das notwendige Argument für den Einzug angegeben, so wird dieser für beide Seiten verwendet.

```
\addmargin[2cm]{3cm}
```

setzt somit auf der linken Seite einen Einzug von 2 cm und rechts 3 cm.

```
\addmargin{1cm}
```

setzt auf beiden Seiten einen Einzug von 1 cm.

Tabelle 4.4: Gegenüberstellung von Umgebungen und Schalterbefehlen für die Ausrichtung von Text aus Standard-LaTeX und ragged2e

Ausrichtung	Standard-LaTeX	ragged2e	Standard-LaTeX	ragged2e
	Umgebungen		*Befehle*	
zentriert	`center`	`Center`	`\centering`	`\Centering`
linksbündig	`flushleft`	`FlushLeft`	`\raggedright`	`\RaggedRight`
rechtsbündig	`flushright`	`FlushRight`	`\raggedleft`	`\RaggedLeft`

Verbesserte Ausrichtung mit ragged2e – Worttrennung bei Flattersatz

Bei Benutzung der normalen Makros für den Flattersatz, wird die Worttrennung deaktiviert. Dies kann insbesondere bei schmalen Textspalten, beispielsweise innerhalb von Tabellen, zu sehr unschönen Ergebnissen führen. Das Paket ragged2e bietet einen Ersatz für die herkömmlichen Umgebungen und Makros für den Flattersatz, welche die Worttrennung nicht deaktivieren, siehe Tabelle 4.4. Zusätzlich lassen Sie sich in ihrem verhalten durch eine Vielzahl von Parametern beeinflussen. Diese können der Paketdokumentation [14] entnommen werden. Zusätzlich gibt es die Möglichkeit die normalen Makros durch die des Paketes ersetzen zu lassen. In diesem Fall muss das Paket mit der Option `newcommands` geladen werden.

4.5 Farben - Das color-Paket

Mit dem Paket color und einem der folgenden Befehle kann die Farbe des Textes verändert werden.

```
\color{Farbe}
\textcolor{Farbe}{Text}
```

Vodefinierte Farben sind white, black, red, green, blue, cyan, magenta, yellow. Der Text „white“ wurde zur besseren Darstellung in eine schwarze `colorbox` eingebunden.

Übrige Farben können in der Form

```
[Farbmodell]{Farbdefinition}
```

definiert werden, wobei das Farbmodell eines aus `rgb` (red,green,blue), `cmyk` (cyan, magenta, yellow, black), `gray` oder `named` ist. Die Farbdefinition ist eine Komma-getrennte Liste mit Werten von 0 bis 1, wobei der Wert den Anteil der Farbkomponenten des Modells widerspiegelt. Die Kodierung `[rgb]{1,0,0}` definiert somit Rot und `[cmyk]{0,1,0,0}` definiert Magenta.

Eigene Farben können mittels

```
\definecolor{Farbname}{Farbmodell}{Farbdefinition}
```

zusammengemischt werden.

Beispiel: Schreibt man

```
Dies ist
\definecolor{eisvogelblau}{cmyk}{0.9,0,0.3,0}
\textcolor{eisvogelblau}{Eisvogelblau}
```

So erscheint die Ausgabe: Dies ist Eisvogelblau.

Mit einem der folgenden Befehle wird die Seitenhintergrundfarbe verändert:

```
\pagecolor{Farbe}
\pagecolor[Farbmodell]{Farbdefinition}
```

Es sind auch mit Farbe gefüllte LR-Boxen möglich (siehe 9.2), wie diese hier ;o). Kreiert werden sie mit

```
\colorbox[Farbmodell]{Farbe}{Text}
\fcolorbox[Farbmodell]{Rahmenfarbe}{Hintergrundfarbe}{Text}
```

Um letzten Endes wieder zur Ausgangsfarbe zurückzukehren, existiert das Makro

```
\normalcolor
```

Es wechselt zurück zu der Farbe, die am Ende der Präambel gewählt war.

4.6 Code „wörtlich“ ausdrucken

Oft, zum Beispiel zum schreiben eines Skriptes über LaTeX ist es nötig Code wörtlich im Dokument abdrucken zu können. Hierfür existieren die Beiden Umgebungen

```
\begin{verbatim} Text \end{verbatim}
\begin{verbatim*} Text \end{verbatim*}
```

Die Sternchenform unterscheidet sich von der ungesternten dadurch, dass in ihr Leerzeichen als ␣ sichtbar gemacht werden:

```
\documentclass[ngerman]{scrartcl}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{babel}
\begin{document}
␣␣␣Ein␣bisschen␣Text\dots
\end{document}
```

Zusätzlich gibt es noch zwei Makros

```
\verb|Code|
\verb*|Code|
```

Das Zeichen | dient hierbei als Gruppierungszeichen. Es sind hierfür sämtliche nicht-Befehlsartigen Sonderzeichen zulässig (z. B. auch +)

5 Querverweise

5.1 Einfache Querverweise

An jeder Stelle im Text kann mit

```
\label{Markername}
```

ein Marker gesetzt werden, auf den man sich mit

```
\ref{Markername}
\pageref{Markername}
```

beziehen kann. Der Inhalt von `\ref{Markername}` ist je nach Ort des `\label`-Befehls die Nummer eines Abschnitts, eines Bildes, einer Tabelle oder einer Formel, während `\pageref{markername}` die Seitenzahl des Markers ausgibt. Ein Beispiel: Die Counter werden auf Seite 32 behandelt. (`\pageref{sec:counter}`)

5.2 Fußnoten und Randnotizen

Das ist eine Randnotiz.

Eine Randnotiz bzw. Fußnote[1], kann mit

```
\marginpar{Text}
\footnote{Text}
```

erstellt werden.

Fußnoten

Mit der Fußnote läuft auch ein Fußnotenzähler bzw. Counter namens `footnote` mit, damit LaTeX weiß, wie viele Fußnoten zu handeln sind. Er kann jederzeit mit dem `setcounter`-Befehl verändert werden. Des Weiteren kann der Bezifferungsstil der Fußnote z. B. auf `fnsymbol` umgestellt werden (vgl. Abschnitt 3.10):

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

Der Befehl darf nur im normalen Textmodus Verwendung finden, ansonsten wird die Fußnote im Ausgabefile nicht erzeugt. Somit ist es normalerweise nicht möglich Fußnoten in Tabellen, im Mathe-Modus sowie im LR-Modus zu setzen.Ist eine Fußnote erforderlich, so muss sie manuell eingefügt werden:

```
\footnotemark[Nummer]
\footnotetext[Nummer]{Fußnote}
```

[1] Dies ist eine Fußnote

Die Nummer wird dabei im laufenden Text mit dem ersten Befehl gesetzt. Der zweite Befehl muss im normalen Textmodus aufgerufen werden und erzeugt die Fußnote. Zum Beispiel funktioniert es damit Fußnoten innerhalb von LR-Boxen, Tabellen und im Mathemodus zu erzeugen. Hier nun ein Beispiel mit einer `\mbox`: `\mbox`-Umgebung ...jetzt[2] ... (`\mbox{jetzt\footnotemark[2]}\footnotetext[2]{Geht doch!}`) siehe unten. Für weitere Formatierungsänderungen der Fuß- bzw. Randnotizen sei auf [27, 11] verwiesen.

Noch viel mehr Formatierungen bietet das footmisc-Ergänzungspaket, siehe auch [21], mit folgenden Optionen

`perpage`	Fußnotenzähler wird bei jeder Seite zurückgesetzt
`marginal`	Kombiniert mit der Länge `\footnotemargin`[3] kann der Abstand der Fußnote zum linken Rand der Seite gesetzt werden
`flushmargin`	Das Fußnotensymbol erscheint bündig mit dem Text der Fußnote
`multiple`	bei mehreren Fußnoten an derselben Stelle werden Kommata als Separator eingesetzt

Referenzen auf Fußnoten setzen

Mit KOMA-Script ist es zudem möglich, Fußnoten mit einem `\label` zu versehen und anschließend mit

`\footref{`*Markername*`}`

zu referenzieren. Dies macht es erheblich einfacher Fußnoten öfter als ein einziges mal aufzuführen.

Fußnoten in `minipages`

Wie bereits erwähnt sind Fußnoten innerhalb der `minipage`-Umgebung erlaubt. Jedoch erscheint die Fußnote in diesem Fall unter der `minipage` und nicht am Ende der Seite.

Randnotizen

Für Randnotizen existiert noch eine weitere Variante des Erzeugungsmakros:

`\marginpar[`*linker Text*`]{`*rechter Text*`}`

Somit ist es möglich Randnotizen zu erzeugen, die unterschiedlich sind, je nachdem ob Sie auf dem rechten oder linken Rand gedruckt werden. Dies ist insbesondere im zweiseitigen Satz wichtig, da hier die Fußnoten immer außen gedruckt werden, also auf linken Seiten links und auf rechten Seiten rechts. Zum Beispiel wurden die beiden Pfeile als Randnotizen auf dieser und der folgenden Seite mit ein und demselben Makro erzeugt: ←

```
\marginpar[\hfill$\longrightarrow$]{$\longleftarrow$}
```

[2] Geht doch!
[3] Kann mit `\setlength` gesetzt werden

5.3 Hyperlinks - Das hyperref-Paket

Diese Randnotiz gehört zu einem Beispiel von Abschnitt 5.2

Das hyperref-Paket von Sebastian Rahtz und Heiko Oberdiek erweitert die Möglichkeiten im Umgang mit Textmarken und -referenzen um ein Vielfaches. Am meisten verbreitet ist die Einbindung des Paketes, um Hyperlinks zu erzeugen, das bedeutet, wenn man auf einen Link im PDF-Dokument klickt, springt das Dokument sofort an die verwiesene Stelle. Da das Paket hierzu viele Standard-LaTeX-interne Makros ändert, sollte immer darauf geachtet werden, dass dieses Paket als letztes geladen wird.

Zusätzlich zum setzten der Links, kann man auch die Art der Markierung von Hyperlinks anpassen. Standardmäßig werden diese mit roten Rahmen gekennzeichnet, jedoch sieht farbiger Text oft eleganter aus. Diese Änderung geschieht mit Hilfe von Paketoptionen:

```
\usepackage[colorlinks=true, linkcolor=Farbe]{hyperref}
```

`colorlinks=true` färbt dabei alle Hyperlinks ein, anstatt Rahmen zu erzeugen und `linkcolor=`*Farbe* wählt zusätzlich noch die Farbe für dokumenteninterne Verweise (nicht Literaturverweise). Die übrigen Arten von Referenzen (Literaturangaben, URLs, ...) bleiben jedoch in der Ihnen zugewiesenen Standardfarbe. Tabelle 5.1 zeigt die Möglichkeiten zur Farbänderung auf. Die Änderung der Farbe, welcher Art auch immer, gehört zum Text, dies bedeutet, wenn man blaue Links wählt und später die entsprechende Seite druckt, so werden die Links auch blau ausgedruckt. Man sollte sich somit zuvor Gedanken machen, zu welchem Zweck man das Dokument samt der Links erstellt.

Tabelle 5.1: hyperref-Optionen zum Setzen der Farbwerte einzelner Linktypen

Option	*Standardwert*	*Beschreibung*
`linkcolor`	red	Farbe für einfache, dokumenteninterne Verweise
`anchorcolor`	black	Farbe für Linktext
`citecolor`	green	Farbe für Literaturveweise
`filecolor`	cyan	Farbe für Links, die Dateien öffnen
`menucolor`	red	Farbe für Acrobat Menüeinträge
`runcolor`	filecolor	Farbe für die Links, die Anmerkungen aufdecken
`urlcolor`	magenta	Farbe für URLs
`allcolors`		Weist allen Farbwerten den gleichen Wert zu

Zusätzlich bietet das hyperref-Paket die Möglichkeit Dokumenteneigenschaften, wie z. B. Autor und Titel, zu den Eigenschaften der PDF-Datei hinzuzufügen. Gerade bei Dokumenten, die digital publiziert werden sollen, sollte man diese Funktionen nutzen. Tabelle 5.2 zeigt die entsprechenden Funktionen. Man sollte hierbei *immer* mindestens die Felder `pdfauthor` und `pfftitle` setzen. Will man zusätzlich noch schwarze Links, wie in diesem Dokument, sehen die Einstellungen wie folgt aus:

```
\usepackage[
     pdftitle={Kursskript:"Einführung in LaTeX"},
     pdfauthor={Marei Peischl},
     colorlinks=true,
     allcolors=black
    ]{hyperref}
```

Tabelle 5.2: Die wichtigsten Paketoptionen von hyperref

Option	*Beschreibung*
`baseurl=`*URL*	Setzt den Basis-Link des Dokumentes
`pdftitle=`*Titel*	Setzt das Feld „Titel“ in den PDF-Eigenschaften
`pdfauthor=`*Author*	Setzt das Feld „Autor“ in den PDF-Eigenschaften
`pdfsubject=`*Thema*	Setzt das Feld „Thema“ in den PDF-Eigenschaften
`pdfkeywords=`*Schlüsselworte*	Setzt das Feld „Schlüsselworte“ in den PDF-Eigenschaften
`colorlinks=`*Wahrheitswert*	Kennzeichnet Links mit farbigem Text statt farbiger Rahmen

Neben den bereits genannten Funktionen, bietet hyperref eine Reihe von Makros für verschiedenste Referenzen:

`\autoref{`*label*`}`

Eine alternative zum normalen `\ref` der Link wird um den Typnamen des Elements auf welches verlinkt wird ergänzt, z. B.:

`\autoref{sec:hyperref}` liefert Abschnitt 5.3

Die Namen werden über das babel-System übersetzt. Ihre Makros samt der Standarddefinitionen im Englischen und Deutschen finden sich in Tabelle 5.3.

Tabelle 5.3: `\autoref`-Namen für hyperref samt der deutschen Übersetzungen mit dem babel-System

Makro	*Deutsch*	*Englisch*
Abbildung	Abbildung	Figure
Tabelle	Tabelle	Table
Teil	Teil	Part
Anhang	Anhang	Appendix
Gleichung	Gleichung	Equation
Punkt	Punkt	item
Kapitel	Kapitel	chapter
Abschnitt	Abschnitt	section
Abschnitt	Unterabschnitt	subsection
Abschnitt	Unterunterabschnitt	subsubsection
Teil	Absatz	paragraph
Unterabsatz	Unterabsatz	subparagraph
Fußnote	Fußnote	footnote
Gleichung	Gleichung	Equation
Theorem	Theorem	Theorem
Seite	Seite	page

`\href{`*URL*`}{`*Text*`}`

Der *Text* wird als Hyperlink zur *URL* gesetzt. Zum Beispiel findet sich hier ein Link zur LaTeX-Kurs-Website.

```
\href{http://www.physik.uni-regensburg.de/studium/edverg/latex/}{hier}
```

Teil II

Das Seitenlayout

6 Der Satzspiegel

> Der Satzspiegel ist wie der Rahmen eines Bildes. Ein echter Rembrandt in einem schiefen, bunten, neonfarbenen PVC-Rahmen wird immer wie eine billige Kopie wirken. Ebenso wird ein inhaltlich perfektes Dokument mit verkorkstem Satzspiegel nicht die Geltung erfahren, die es verdient.
>
> *Markus Kohm & Jens-Uwe Morawski* [10, S.485]

6.1 Professionelle Satzspiegelkonstruktion mit typearea

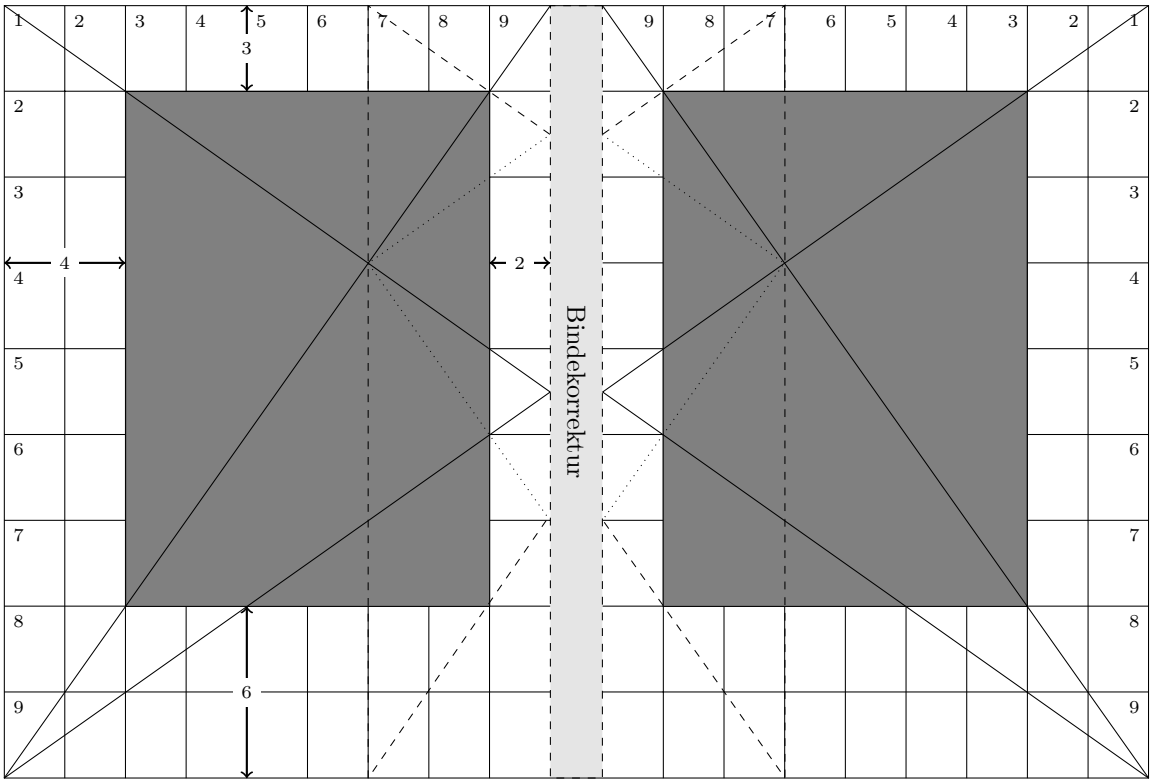


Abbildung 6.1: Doppelseite mit Satzspiegelkonstruktion in klassischer Neunerteilung (entspricht `DIV=9`) [nach 10, S.32]

Die Satzspiegelkonstruktion übernimmt bei KOMA-Klassen das in KOMA-Script integrierte typearea-Paket. Gleiche Ergebnisse in Nicht-KOMA-Klassen können durch zusätzliches laden dieses Paketes erfolgen. Im Folgenden wird nun zunächst das Prinzip der Satzspiegelkonstruktion durch Teilung betrachtet, mit dem man relativ einfach einen harmonischen Satzspiegel erzeugen kann.

Das Prinzip basiert auf der klassischen Neunerteilung (siehe Abbildung 6.1), bei der die Seite sowohl horizontal, als auch vertikal in neun Streifen geteilt wird. Über Berechnungen mit Hilfe des gerundeten ganzzahligen „Goldenen Schnittes“ ergibt sich das Ränderverhältnis innen:oben:außen:unten von 2:3:4:6. Die Bindekorrektur (`BCOR`) bleibt davon als unsichtbarer Teil der Seite vollkommen unberücksichtigt.

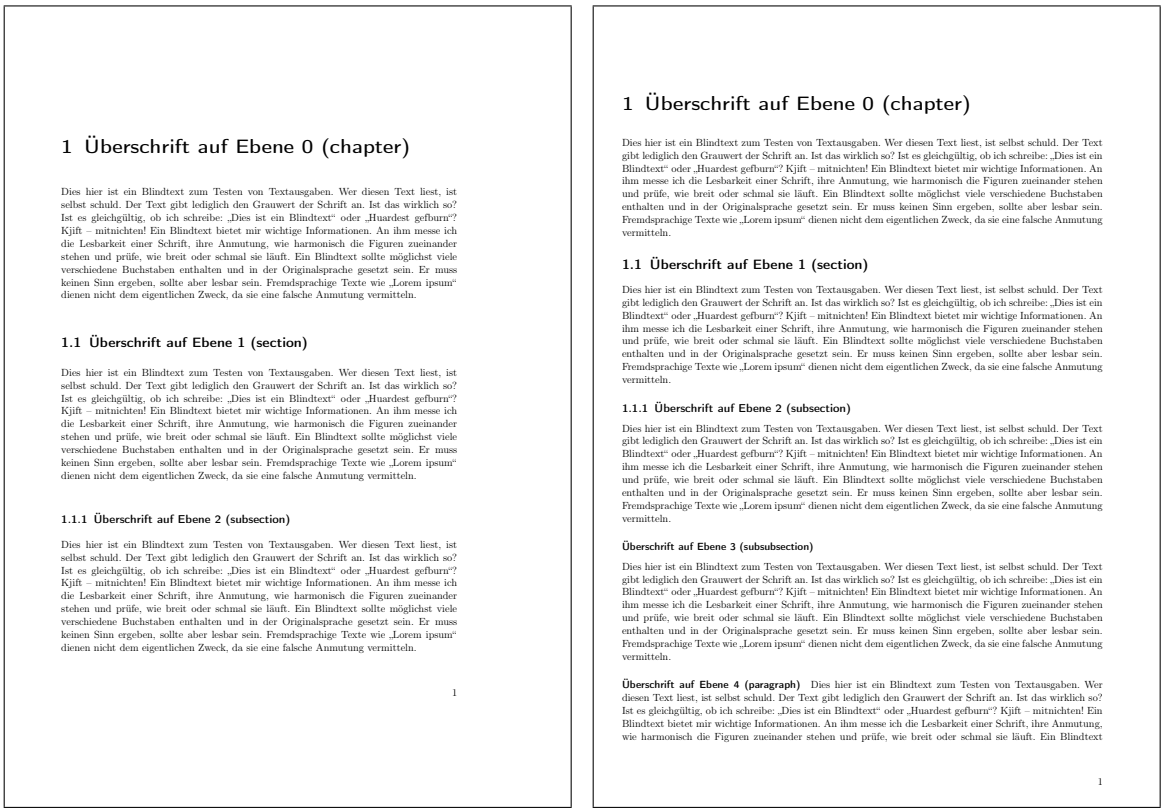
Die Anzahl der Streifen kann über die KOMA-Klassenoption (oder typearea-Paketoption)

```
\documentclass[DIV=Anzahl, Optionen]{scr...}
\usepackage[DIV=Anzahl, Optionen]{typearea}
```

nur bei Nicht-KOMA-Klassen nötig

nach persönlichen Wünschen modifiziert werden. Das Teilungsverhältnis von 2:3:4:6 bleibt jedoch erhalten. Somit wird klar, dass bei einem größeren `DIV`-Wert auch ein größerer Teil der Seite beschrieben wird. Standardmäßig verwendet typearea `DIV=8` für die Grundschrift 10 pt, `DIV=10` für 11 pt und `DIV=12` für 12 pt Grundschriftgröße. Beispiel 3 den Unterschied zwischen `DIV=10` und `DIV=20` für die Grundschriftgröße 11 pt. Zusätzlich gibt es Paketoptionenswerte für eine automatisierte Berechnung:

Beispiel 3: Vergleich der `DIV`-Werte 10 und 20 bei der Grundschriftgröße von 11 pt. `DIV=10` wirkt deutlich übersichtlicher und professioneller, wohingegen `DIV=20` lediglich Papier einspart.



`DIV=calc` Berechnung des optimalen Satzspiegels (empfohlen für Papierformate ungleich A4)

`DIV=classic` Bestimmung des DIV-Wertes, welcher dem spätmittelalterlichen Buchseitenformat (Satzspiegelkonstruktion durch Kreisschlagen) am nächsten kommt.

`DIV=current` Neuberechnung des Satzspiegels mit aktuellem `DIV`-Wert

`DIV=last` Neuberechnung des Satzspiegels aufgrund des zuletzt gewählten `DIV`-Wertes

`DIV=default` Neuberechnung des Satzspiegels für das aktuelle Papierformat und die aktuelle Grundschrift

Bindet man eine neue Schriftart ein, sollte der Satzspiegel unter Berücksichtigung der Schrift neu berechnet werden. Normalerweise wird der Satzspiegel zu Beginn des Dokumenten (direkt nach `\begin{document}`) berechnet. Nimmt man anschließend noch Änderungen an Schriftsippe, Basisschriftgröße,... vor, so ist eine Neuberechnung erforderlich. Diese funktioniert auch innerhalb des Dokumentes mit den Befehlen:

```
\typearea[BCOR]{DIV}
\areaset[BCOR]{Breite}{Höhe}
\recalctypearea
```

6.2 Das Seitenlayout manuell einstellen mit geometry

Manchmal ist es notwendig bestimmte Größen im Seitenlayout manuell festzusetzen. Das geometry-Paket von Hideo Umeki bietet eine sehr benutzerfreundliche Möglichkeit sowohl Papierformat als auch Satzspiegel manuell anzupassen. Diese Variante ist jedoch nicht sonderlich elegant, weswegen man auf diese Variante nur dann zurückgreifen sollte, wenn es absolut keine andere Möglichkeit gibt.

```
\usepackage[Option1, Option2, ...]{geometry}
```

Die Basis-Einstellung werden in der Regel über *Optionen* (siehe Tabelle 6.1) vorgenommen. Zusätzlich existieren Befehle, um das Layout innerhalb des Dokumentes zu ändern:

`\geometry{`*Option1,...*`}`	ändert das Layout entsprechend der *Optionen* Die Optionen werden zusätzlich zu vorherigen verwendet. Sollte nur in der Präambel verwendet werden.
`\newgeometry{`*Option1,...*`}`	Überschreibt die bisherigen Optionen. Papiergröße ist damit nicht einstellbar.
`\restoregeometry`	Aktiviert wieder die ursprünglichen Einstellungen aus dem Header
`\savegeometry{`*Name*`}`	speichert die aktuellen Einstellungen unter *Name* ab
`\loadgeometry{`*Name*`}`	aktiviert die unter *Name* gespeicherten Einstellungen

Zusätzlich zu den genannten Optionen ist es auch möglich, den Kopf-, bzw. den Fußbereich mit zum Satzspiegel zu zählen, siehe Abbildung 6.2. Hierfür existieren die Optionen

```
includehead
includefoot
includeheadfoot
```

Tabelle 6.1: Übersicht über die wichtigsten Paketoptionen von geometry

Option	*mögliche Werte*	*Erläuterung*
`paper`	`a0paper`, `a1paper`,..., `a6paper`, `b0paper`, `b1paper`,..., `b6paper`, `c0paper`, `c1paper`,..., `c6paper`, `b0j`, `b1j`,..., `b6j`, `letterpaper`, `executivepaper`, `legalpaper`,...	Papierformat
`screen`		225 mm auf 180 mm für Präsentationen
`paperwidth`	*Länge*	
`paperheight`	*Länge*	
`papersize`	{*Breite*,*Höhe*} oder *Länge*	manuelle Angabe der Seitengröße, eine einzige Längenangabe erzeugt ein quadratisches Format
`landscape`		Querformat
`portrait`		Hochformat
`left/inner`	*Länge*	linker/innerer Rand
`right/outer`	*Länge*	rechter/äußerer Rand
`top`	*Länge*	oberer Rand
`bottom`	*Länge*	unterer Rand
`hmarginratio`	*links (innen)*:*rechts (außen)*	Größenverhältnis der Seitenränder. Zulässig sind nur ganzzahlige Werte bis 100. Standardwert ist 1:1 für einseitigen und 2:3 für zweiseitigen Satz.
`vmarginratio`	*oben*:*unten*	Größenverhältnis des oberen zum unteren Rand. Standardwert ist 2:3
`bindingoffset`	*Länge*	Bindekorrektur

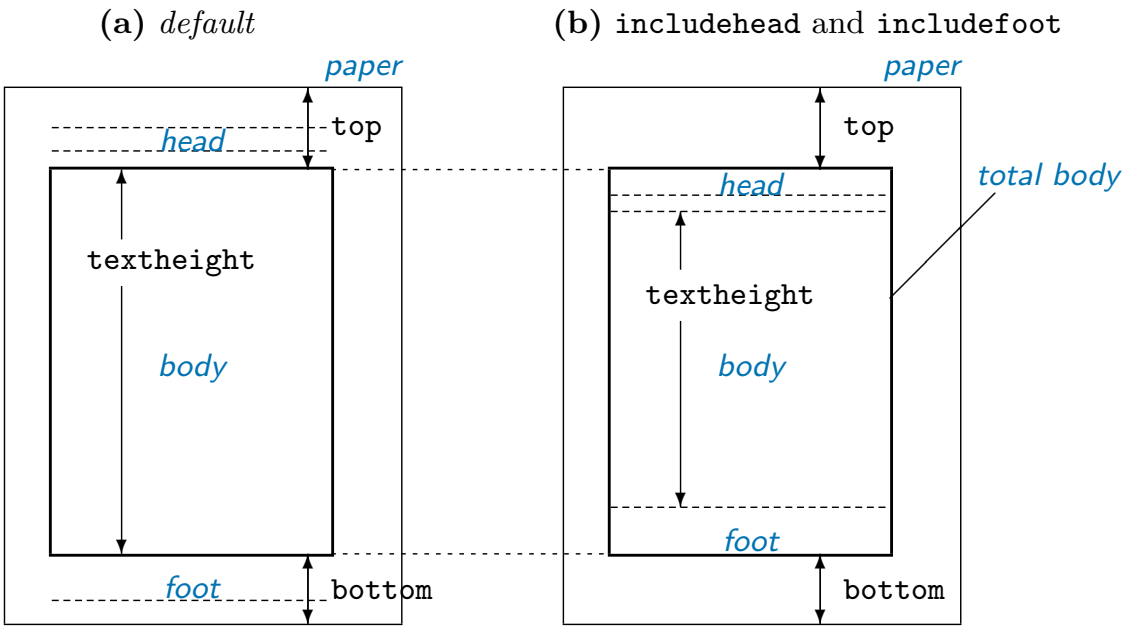


Abbildung 6.2: Bedeutung der geometry-Optionen `includehead`, `includefoot` und `includeheadfoot` [6, S.3]

6.3 Mehrspaltiger Textsatz

Zweispaltiger Text kann mithilfe der Dokumentenklassenoption `twocolumn=true` erreicht werden. Zusätzlich existieren in Standard-LaTeX die Makros

`\twocolumn[`*Überschrift*`]`	Beendet die aktuelle Seite. Die folgende Seite wird zweispaltig gesetzt. Die optionale Überschrift wird auf der neuen Seite über die Breite der gesamten Seite gesetzt.
`\onecolumn`	Beendet die aktuelle zweispaltige Seite und beginnt eine neue einspaltige.

Das multicol-Paket

multicol Das multicol-Paket behebt einige Unschönheiten von Standard-LaTeX. Beispielsweise füllt es auf jeder Seite beide Spalten gleichmäßig auf und nicht zuerst die linke Spalte vollständig. Zudem erlaubt es mehrspaltigen Textsatz bis hin zu zehnspaltigem Satz. Mit der Umgebung

`\begin{multicols}{`*Spaltenanzahl*`}[`*Überschrift*`]` *Text* `\end{multicols}`

kann man mitten auf der Seite in mehrspaltigen Text wechseln. Die Überschrift wird über alle Spalten gesetzt, bevor der mehrspaltige Text beginnt.

7 Der Seitenstil

Die Gepflogenheiten der Typografie erfordern es häufig, verschiedene Seitenstile innerhalb eines Dokumentes zu verwenden. So wird die Titelseite beispielsweise nicht nummeriert und die Anfangsseite eines neuen Kapitels besitzt häufig keine Kopfzeile. Dieses Umschalten funktioniert bei LaTeX oft automatisch über die Dokumentenklasse, allerdings kann es erforderlich sein, manuell einzugreifen, zum Beispiel um ein Bild zu positionieren, das die Kopfzeile teilweise überdeckt. Das manuelle Umschalten kann über die Befehle

```
\pagestyle{Stil}
\thispagestyle{lokaler Stil}
```

erfolgen. Dabei ist `\pagestyle` ein Schalter, der den Seitenstil global umstellt, wohingegen sich das Makro `\thispagestyle` lediglich auf die aktuelle Seite bezieht. Außerdem bietet KOMA-Script mit den Befehlen `\titlepagestyle, \partpagestyle, \chapterpagestyle` und `\indexpagestyle` den Stil der Titelseiten der jeweiligen Ebene zu ändern.

Tabelle 7.1: Übersicht über die vordefinierten Seitenstile

Seitenstil	*Erklärung*
`empty`	Kopf und Fußzeilen bleiben vollständig leer.
`headings`	Dies ist der Seitenstil für sogenannte lebende Kolumnentitel. Hier werden die Überschriften automatisch in den Seitenkopf übernommen. In den Klasse scrbook und scrreprt werden, im doppelseitigen Layout die Überschriften der Kapitel und Abschnitte außen in der Kopfzeile wiederholt. Die Seitenzahl findet sich im Fuß außen. Im einseitigen Layout werden nur die Überschriften der Kapitel verwendet und wie die Seitennummer zentriert ausgegeben.
`myheadings`	Es entspricht dem Seitenstil headings. Allerdings werden die Kolumnentitel nicht automatisch erzeugt. Hierzu werden die Anweisungen `\markboth` und `\markright` verwendet.
`plain`	Hierbei werden keinerlei Kolumnentitel verwendet, sondern lediglich die Seitenzahl ausgegeben. Im doppelseitigen Layout außen, sonst zentriert.

Die vordefinierten Seitenstile finden sich in Tabelle 7.1. Beim Seitenstil myheadings wird der Kolumnentitel nicht automatisch ausgegeben. Hier muss den Inhalt manuell setzen, hat aber dadurch die Wahl, was genau in der Kopfzeile ausgegeben werden soll. Hierfür gibt es die Befehle

```
\markboth{linke Marke}{rechte Marke}
\markright{rechte Marke}
```

Die rechte Marke wird im Kopf rechter (ungerader) Seiten verwendet, die linke Marke analog links. Bei einseitigem Satz gibt es nur die rechte Marke. Für weitere Anpassungen und den Befehl `\markleft` empfiehlt es sich das scrlayer-scrpage-Paket zu verwenden (siehe Abschnitt 7.1).

7.1 Kopf- und Fußzeilen mit scrlayer-scrpage

Mit dem Paket scrlayer-scrpage lassen sich recht einfach komplexere Kopf- und Fußzeilen erzeugen. Es basiert auf dem Paket scrlayer, welches ein Ebenenmodell, wie es von Bildbearbeitungsprogrammen bekannt sein könnte, sowie ein darauf basierendes Seitenstil-Modell anbietet. Zu den grundlegenden Funktionen gehören zwei individuell konfigurierbare Seitenstile: `scrheadings` und `plain.scrheadings`. Diese Seitenstile können, wie in Kapitel 7 beschrieben, mithilfe von

```
\pagestyle{scrheadings}
```

aktiviert werden. Damit wird der der Seitenstil `plain` automatisch durch `plain.scrheadings` ersetzt. Man kann jedoch auch `plain.scrheadings` direkt aktivieren.

Für automatische Kolumnentitel sollte die Option

```
\usepackage[automark]{scrlayer-scrpage}
```

gesetzt werden. Standardmäßig werden somit die Marken so gesetzt, dass in Klassen mit `\chapter` die Kapitelüberschrift links und die Abschnittsüberschrift rechts gesetzt wird. In den übrigen Klassen ist dieses Verhalten eine Ebene nach unten verschoben.

7.1.1 Höhe von Kopf und Fuß

Normalerweise ist die Fußzeile bei LaTeX-Dokumenten immer einzeilig. scrlayer führt nun eine Höhe `\footheight` analog zu `\headheight` ein. scrlayer-scrpage interpretiert dann denn Abstand `\footskip` so, „dass es den Abstand der letzten Grundlinie des Texbereichs von der ersten Standard-Grundlinie des Fußes darstellt“ [9, S.264].

7.1.2 Seitenstile modifizieren

Die Modifiationen der Seitenstile funktionieren nun ähnlich zu `myheadings` mit Befehlen der Form:

`\`*Feld*`[`*plain.scrheadings-Inhalt*`]{`*scrheadings-Inhalt*`}`

Die Benennung der Felder kann Abbildung 7.1 entnommen werden. Will man nun zum Beispiel erreichen, dass die Seitenzahl, sowohl bei scrheadings, als auch bei plain.scrheadings auf den rechten Seiten, außen in der Kopfzeile steht, benutzt man einfach

```
\rohead[\pagemark]{\pagemark}
```

Sollen zusätzlich bei scrheadings die Kolumnentitel auf allen Seiten im Kopf innen stehen, setzt man zusätzlich zum obigen Makro

```
\ihead{\headmark}
```

Die Kopf- und Fußzeile sind jedoch vor den Änderungen meistens nicht leer. Um zu vermeiden, dass zusätzliche Einträge in den Kopf- beziehungsweise Fußzeilen auftreten, die noch von der Dokumentenklasse stammen, benutzt man am besten

```
\clearmainofpairofpagestyles
\clearplainofpairofpagestyles
\clearpairofpagestyles
```

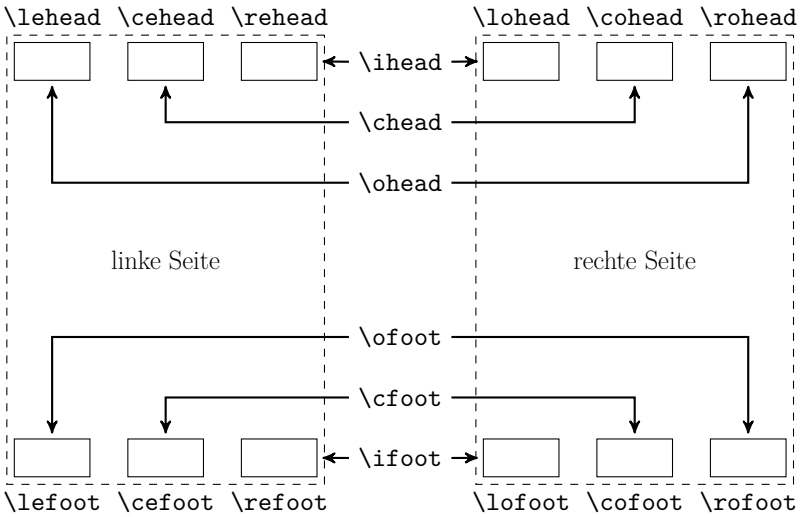


Abbildung 7.1: Zuordnung der Befehle für scrlayer-scrpage [9, S.270/273]

`\clearmainofpairofpagestyles` leert dabei lediglich alle Felder des scrheadings-Seitenstils, wobei `\clearplainofpairofpagestyles` dasselbe für plain.scrheadings erfüllt. `\clearpairofpagestyles` löscht sämtliche Kopf und Fußzeilen beider Seitenstile. Dies erspart beispielsweise viel Schreibarbeit, wenn man nur zwei der sechs möglichen Felder benutzen will. Zum Beispiel bewirken die beiden folgenden Varianten dasselbe:

```
\clearscrheadfoot
\ohead{\headmark}
\ofoot[\pagemark]{\pagemark}
```

```
\ihead[]{}
\chead[]{}
\ohead{\headmark}
\ifoot[]{}
\cfoot[]{}
\ofoot[\pagemark]{\pagemark}
```

In den vorangehenden Beispielen wurden Befehle benutzt, die noch gar nicht besprochen wurden. Dies wird sofort nachgeholt.
Um die Kolumnentitel möglichst einfach zu setzen, gibt es vordefinierte Befehle, die es vereinfachen, sogar lebende Kolumnentitel[1] zu erzeugen.

`\leftmark, \rightmark` Diese Befehle greifen auf die Kolumnentitel zurück, die für die linke bzw. für die rechte Seite gedacht sind. (Standard: links Kapitelnummer mit -name; rechts Abschnittsnummer mit -name)

`\headmark` Dieser Befehl ermöglicht es ebenfalls auf den Inhalt der Kolumnentitel zurückzugreifen. Jedoch muss man hierbei nicht auf die richtige Zuordnung von linker und rechter Seite achten.

`\pagemark` Entspricht der Seitenzahl, der Stil kann mit `\pagenumbering{`*Stil*`}` geändert werden, vgl. Tabelle 3.10.

`\manualmark` Wie der Name schon sagt, deaktiviert dieser Befehl die automatische Aktualisierung der Kolumnentitel. Man kann nun mit den Befehlen `\markboth` bzw. `\markright` die Kolumnentitel manuell ändern.

[1] Lebende Kolumnentitel sind Seitenüberschriften, die sich im Verlauf des Buches ändern. Zum Beispiel die Überschrift der Seite durch den jeweils aktuellen Kapitelnamen zählt zu den lebenden Kolumnentiteln. Tote Kolumnentitel ändern dagegen sich im Verlauf des Werkes nicht.

`\automark*[`*rechte Seite*`]{`*linke Seite*`}` Dieses Makro aktiviert die automatische Aktualisierung des Kolumnentitels. Für die Parameter kann man einfach Gliederungsebenen (Abschnitt 3.6) einsetzen. Möchte man erreichen, dass auf den linken Seiten der Titel des Kapitels und auf den rechten Seiten der des Abschnittes steht, benötigt man den Befehl

```
\automark[section]{chapter}
```

`\leftmark` und `\rightmark` werden dementsprechend modifiziert. Die Sternchenversion (nicht verfügbar in scrpage2) unterscheidet sich dadurch, dass `\automark` alle vorherigen Vorgaben aufhebt und `\automark*` lediglich die Aktionen für die angegebenen Gliederungsebenen ändert.

7.1.3 Formatierung der Kopf- und Fußzeilen

Eine Änderung der Schriftart innerhalb der Kopf und Fußzeilen funktioniert vollkommen analog zum Ändern der Überschriftenschriftart (siehe Abschnitt 4.1.4), z. B.:

```
\setkomafont{pagehead}{\normalfont\sffamily\bfseries}
\setkomafont{pagefoot}{\normalfont\sffamily}
\setkomafont{pagenumber}{\normalfont\rmfamily\slshape}
```

`headwidth=`*Breite*`:` *Verschiebung*
`footwidth=`*Breite*`:` *Verschiebung*

„Normalerweise entsprechen die Breiten von Kopf- und Fußzeile der Breite des Textbereichs.“ [10, Seite 264] Manchmal ist es jedoch nötig diese Breiten entsprechend anzupassen. Für Standardfälle gibt es symbolische Werte für das Argument *Breitenoption* (Die Namen sind selbsterklärend.): `page`, `text`, `textwithmarginpar`, `head`, `foot`.

Neben der Schriftart und der Breite existieren verschiedene Trennlinien, um den Seitenkopf und -fuß zu formatieren.

`headtopline=`*Länge*`:` *Dicke*	Linie über dem Seitenkopf
`headsepline=`*Länge*`:` *Dicke*	Linie zwischen Kopf und Textkörper
`footsepline=`*Länge*`:` *Dicke*	Linie zwischen Text und Fuß
`footbotline=`*Länge*`:` *Dicke*	Linie unter dem Fuß

Standardmäßig werden diese Linien zentriert, um die Ausrichtung zu ändern, gibt es drei verschiedene Paketoptionen für scrlayer-scrpage:

`ilines`	innen bündige Linien
`clines`	zentrierte Linien
`olines`	außen bündige Linien

Um neben der Länge und der Dicke auch andere Eigenschaften, wie beispielsweise die Farbe zu ändern, existieren entsprechende Komafonts. Möchte man z. B. die `headtopline` blau einfärben, so schreibt man (nachdem das color-Paket geladen wurde):

```
\addtokomafont{headtopline}{\color{blue}}
```

Um die Einstellung der Trennlinien auch für `plain.scrheadings` zu übernehmen gibt es zusätzliche Optionen, denen man einen Wahrheitswert zuweisen kann:

`plainheadtopline=` *Wahrheitswert*
`plainheadsepline=` *Wahrheitswert*
`plainfootsepline=` *Wahrheitswert*
`plainfootbotline=` *Wahrheitswert*

7.1.4 Ältere KOMA-Script-Versionen: Das Paket scrpage2

Das Paket scrlayer-scrpage ersetzt das Paket scrpage2 aus älteren KOMA-Script Versionen. Eine Kompatibilität ist jedoch nach wie vor gegeben. So genügt es bei älteren Dokumenten meistens, lediglich das zu ladende Paket zu ändern.

Problematisch ist jedoch, dass häufig (z. B. im Linux-CIP-Pool) noch ältere Versionen installiert sind. Es ist dort somit nicht möglich scrlayer-scrpage zu verwenden. Die meisten Features funktionieren auch unter scrpage2. Ein grundlegender Unterschied ist jedoch, dass der Seitenstil für die plain-Seiten dort `scrplain` anstatt `plain.scrheadings` heißt. Zudem sind die Makros wie `\clearpairofpagestyles` dort unter anderem Namen verfügbar: `\clearscrheadings`, `clearscrplain`, `clearscrheadfoot`. Diese Makros sind zwar in scrlayer-scrpage ebenfalls enthalten, jedoch nur aus Gründen der Kompatibilität und sollten somit nicht mehr verwendet werden. Zusätzlich funktioniert auch die Breiteneinstellung von Kopf- und Fußzeile sowie die Liniensetzung bei scrpage2 unterschiedlich (siehe Anleitung zu scrpage2 oder [10]).

Teil III

Weitere wichtige Elemente

8 Aufzählungen

Hierfür existieren drei Umgebungen:

```
\begin{itemize} \item ...\item ... usw.\end{itemize}
\begin{enumerate} \item ... \item ... usw. \end{enumerate}
\begin{description} \item[Name1] ... usw. \end{description}
```

Bei der `itemize`-Umgebung werden die Punkte durch ein Aufzählungszeichen getrennt und die einzelnen Texte voneinander abgesetzt. Bei der `enumerate`-Umgebung wird fortlaufend nummeriert und bei der `description`-Umgebung erscheint der optionale Name, als zu erläuternder Begriff fett gedruckt.

8.1 Verschachtelte Aufzählungen

Die Aufzählungsumgebungen können bis zu viermal ineinander verschachtelt werden und je nach Verschachtelungstiefe ändert sich das Markierungszeichen und zusätzlich der vertikale Abstand der Aufzählungstiefen zueinander, siehe Beispiel 4.

Beispiel 4: Eine verschachtelte Aufzählung

- erste Stufe erste Zeile
  - zweite Stufe erste Zeile
    - dritte Stufe erste Zeile
      - vierte Stufe erste Zeile
      - vierte Stufe zweite Zeile
    - dritte Stufe zweite Zeile
  - zweite Stufe zweite Zeile
- erste Stufe zweite Zeile

```
\begin{itemize}
\item erste Stufe erste Zeile
\begin{itemize}
\item zweite Stufe erste Zeile
\begin{itemize}
\item dritte Stufe erste Zeile
\begin{itemize}
\item vierte Stufe erste Zeile
\item vierte Stufe zweite Zeile
\end{itemize}
\item dritte Stufe zweite Zeile
\end{itemize}
\item zweite Stufe zweite Zeile
\end{itemize}
\item erste Stufe zweite Zeile
\end{itemize}
```

8.2 Änderung der Markierungen

Mit einem optionalen Parameter beim Befehl

`\item[`*Marke*`]`

wird das Markierungszeichen an diesem Punkt geändert.
Beispiel: Mit `\item[$\clubsuit$]` und `\item[$\diamondsuit$]`:

- ♣ erste Stufe erste Zeile
  - ♢ zweite Stufe erste Zeile
  - ♢ zweite Stufe zweite Zeile
- ♣ erste Stufe zweite Zeile

Die Aufzählungszeichen können auch dauerhaft verändert werden mittels

`\renewcommand{`*Zeichenname*`}{`*neues Zeichen*`}`

Der Zeichenname lautet dabei entsprechend der Verschachtelungstiefe

```
\labelitemi   \labelitemii   \labelitemiii   \labelitemiv
\labelenumi   \labelenumii   \labelenumiii   \labelenumiv
```

Es sei noch erwähnt, daß LaTeX bei `enumerate` mit den Aufzählungsebenen vier Counter (vgl. Abschnitt 3.10), `enumi, enumii, enumiii` und `enumiv` betreibt. Das Beispiel liefert in der zweiten Ebene die Nummerierung mit Abschnittsnummer und Zähler der zweiten Stufe.

```
\renewcommand{\labelenumii}{\textit{\thesection-\arabic{enumii}.)}}
```

1. erste Stufe erste Zeile
2. erste Stufe zweite Zeile
   8.2-1.) zweite Stufe erste Zeile
   8.2-2.) zweite Stufe zweite Zeile
   8.2-3.) zweite Stufe dritte Zeile
3. erste Stufe dritte Zeile

8.3 KOMA-Auflistung

KOMA-Script liefert eine weitere Auflistungsumgebung neben den Standardlisten von LaTeX, und zwar

`\begin{labeling}[`*Trennzeichen*`]{`*Muster*`}` ... `\end{labeling}`

Das Muster gibt die Einrücktiefe bei den Stichpunkten an und das Trennzeichen ist beliebig. Genau wie bei den Standardaufzählugen beginnt jeder Punkt mit einem `\item`. Beispiel 5 zeigt die Funktionsweise, für genauere Informationen und weitere Beispiele sei auf [10] verwiesen.

Beispiel 5: Beispiel für die Verwendung der `labeling`-Umgebung [nach 10, S. 132]

`itemize` – Listen mit Aufzählungspunkten
`enumerate` – Nummerierte Listen
`description` – Begriffserläuterungen
`labeling` – Personalisierte Listen

```
\setkomafont{labelinglabel}{\ttfamily}
\setkomafont{labelingseparator}{\normalfont}
\begin{labeling}[~--]{description}
\item[itemize] Listen mit Aufzählungspunkten
\item[enumerate] Nummerierte Listen
\item[description] Begriffserläuterungen
\item[labeling] Personalisierte Listen
\end{labeling}
```

8.4 Weitere Optionen mit dem enumitem-Paket

```
\usepackage{enumitem}
```

definiert zu den drei Umgebungen ein zusätzliches optionales Argument, worüber viele Anpassungen gemacht werden können. Hier sollen nur wenige Möglichkeiten gezeigt werden, alle Befehle und Optionen finden sich in der Paketdokumentation [7].

Eine mögliche Option ist `label`. Damit lässt sich die Beschriftung der Listenpunkte für die jeweilige Ebene ändern. Die Änderung wirkt aber nur innerhalb der Umgebung, wo die Option gesetzt wurde. Für die `itemize`-Umgebung kann man beliebige Symbole benutzen wie z.B das Pfeilsymbol → (`$\rightarrow$`), für die `enumerate`-Umgebung gibt es `\alph*`, `\Alph*`, `\arabic*`, `roman*` und `Roman*`, wobei statt dem Stern auch ein Countername in geschweiften Klammern stehen kann (ein Beispiel steht weiter unten).

Sind zwei Listen durch Text getrennt und die zweite Liste soll die erste fortführen (und nicht wieder mit der Zählung von vorne beginnen), so gibt man bei den Optionen `resume` mit an (was natürlich auch über `\setcounter` machbar wäre, siehe Kapitel 3.10). Sollen zusätzlich die Formatdefinitionen aus der ersten Liste übernommen werden (falls in der Ebene vorhanden) so benutzt man `resume*` (vgl. Beispiel 6).

Möchte man die Beschriftung dauerhaft ändern so kann man die bestehenden Listen (`itemize`, `enumerate`, `description`) neu definieren oder komplett neue erstellen.

Neue Listen werden folgendermaßen erstellt:

```
\newlist{Listenname}{Typ}{max. Tiefe}
\renewlist{Listenname}{Typ}{max. Tiefe}
```

Für *Typ* kann man `enumerate`, `itemize` oder `description` einsetzen. *Listenname* ist ein beliebiger Name für die neue Liste, und *max. Tiefe* gibt die Anzahl der Ebenen an, die die Aufzählung maximal haben darf. Möchte man die Standardlisten neu definieren so benutzt man stattdessen `\renewlist`. Als nächstes müssen die Eigenschaften der Liste festgelegt werden. Dies geschieht mit

```
\setlist[Listenname, Ebene]{Optionen}
```

Wenn man bei der Definition mehrere Ebenen zulässt, so muss für jede Ebene eine Beschriftung definiert sein. Wenn man *Ebene* weglässt, gilt die *Option* für alle Ebenen, kann aber für einzelne Ebenen wieder überschrieben werden (Beispiel 7).

Zusätzlich zur Option, die das Label verändert gibt es noch eine Reihe von Optionen um die Formatierung der Liste an sich zu ändern. Im Folgenden finden sich einige Beispiele:

Beispiel 6: Ein Beispiel für die Benutzung von enumitem mit einer Liste der Form a), b), c), usw.

```
\begin{enumerate}[label=\alph*)]
\item Punkt 1
\item Punkt 2
\begin{enumerate}[label=\roman*)]
\item Unterpunkt 1
\item Unterpunkt 2
\end{enumerate}
\item Punkt 3
\end{enumerate}
Hier steht Text, der beide
Listen voneinander trennt.
\begin{enumerate}[resume*]
\item Punkt 1
\item Punkt 2
\begin{enumerate}
\item Unterpunkt 1
\item Unterpunkt 2
\end{enumerate}
\item Punkt 3
\end{enumerate}
```

a) Punkt 1

b) Punkt 2

   i) Unterpunkt 1

   ii) Unterpunkt 2

c) Punkt 3

Hier steht Text, der beide Listen voneinander trennt.

d) Punkt 1

e) Punkt 2

   a) Unterpunkt 1

   b) Unterpunkt 2

f) Punkt 3

Beispiel 7: Aufzählung mit vier zulässigen Ebenen, bei der die oberste Ebene in großen römischen und die restlichen mit arabischen Zahlen beschriftet sind.

```
\newlist{steps}{enumerate}{4}
\setlist[steps]{label=(\arabic*)}
\setlist[steps, 1]{label=\Roman*)}
\begin{steps}
\item Punkt 1
\item Punkt 2
\begin{steps}
\item Unterpunkt 1
\item Unterpunkt 2
\end{steps}
\item Punkt 3
\end{steps}
```

I) Punkt 1

II) Punkt 2

   (1) Unterpunkt 1

   (2) Unterpunkt 2

III) Punkt 3

`leftmargin=`*Länge*	Rückt den Beschriftungstext der Labelst um *Länge ein*, die Labels beginnen somit bei `leftmargin−labelwidth`.
`labelwidth=`*Länge*	Setzt die Breite des Platzes der für die Labels reserviert ist auf *Länge*
`align=left/right`	Richtet die Labels entweder links- oder rechtsbündig aus
`itemsep=`*Länge*	Setzt den Abstand zwischen den einzelnen Items auf *Länge*.
`parsep=`*Länge*	Setzt den Abstand zur Absatzkennzeichnung innerhalb der Liste auf *Länge*.
`start=`*Nummer*	Setzt den Startwert der Liste auf *Nummer*.

9 Das Prinzip der Boxen

TeX berechnet viele Elemente, wie z. B. Absatzumbrüche, nicht am Text sondern an einer Reihe von Boxen ohne Inhalt, die lediglich als Platzhalter fungieren.

Wenn zwischen zwei Zeichen keine Trennung möglich sein soll, so werden sie zu einer großen Box zusammengefasst:

Eine gute Vertrautheit mit diesen Prinzipien ist wichtig, da sie von TEXbeim Satz von Tabulatoren und Tabellen verwendet werden.

9.1 Die drei Bearbeitungs-Modi

Innerhalb des Textkörpers kennt LATEX drei unterschiedliche Modi:

1. **Paragraph Mode:** normaler Textmodus
   z. B. `\begin{document}...\end{document}`, `\parbox{2cm}{...}`
2. **Mathematical Mode:** mathematischer Modus
   z. B. `\begin{equation}...\end{equation}` (Kapitel 15)
3. **LR Mode:** (Left–Right)-Mode; Wie Paragraph Mode, jedoch mit dem Verbot des Zeilenumbruchs. Der Text wird ohne Umbruch von links nach rechts gesetzt.
   z. B. `\mbox{}, \fbox{}` (Abschnitt 9.2)

Innerhalb einer bestimmten Box kann immer nur ein Modus aktiv sein. Eine gerade zu Beginn recht häufige Fehlerquelle ist somit, dass man sich im falschen Modus befindet. So funktionieren mathematische Befehle nun einmal nur im Mathemodus. Häufig kann man diese Probleme, wie zum Beispiel den verbotenen Zeilenumbruch im LR-Mode dadurch umgehen, dass man innerhalb „des Modus“ eine Box im Paragraph Mode erstellt.

9.2 Rahmenboxen

Die einfachste Möglichkeit Code zu einer großen Box zusammenzufassen bieten die Boxen im LR-Mode, die sogenannten `mboxen`.

`\makebox[`*Breite*`][`*Position*`]{`*Text*`}`
`\mbox{`*Text*`}`

Die ausgeschriebene Variante erlaubt es weitere Parameter festzusetzen.

Diese Boxen sind jedoch unsichtbar, weshalb sich ihre Funktionsweise leichter mit den sichtbaren Rahmenboxen verdeutlichen lässt.

`\frame{`*Text*`}`

Das `\frame`-Makro verhält sich identisch zur `\mbox` und setzt somit sein Argument in den LR-Mode. Der Rahmen ist hierbei jedoch sichtbar:

ein bisschen Text in einem `\frame`

Die Abstände sind jedoch, wie man sieht, für Text vollkommen ungeeignet. Frames sollten somit lediglich zum Umrahmen von Objekten, wie z. B. Abbildungen verwendet werden. Um einen einfachen Rahmen um Text zu ziehen gibt es die `\fbox`. Die ausgeschriebene Variante dieses Makros ermöglicht es weitere Parameter anzugeben:

`\framebox[`*Breite*`][`*Position*`]{`*Text*`}`
`\fbox{`*Text*`}`

Der Text wird entsprechend dem LR-Modus nicht umgebrochen und ragt über die Box hinaus, falls diese für den Text zu klein ist. Als Position zur laufenden Zeile gelten die Optionen l, r, c, s für left, right, center und stretched (gleichmäßig verteilt). Beispiel:

Dies ist Text innerhalb einer 12 cm breiten `framebox` mit Ausrichtung rechts.

Die Dicke der Rahmen, sowie der Abstand des Rahmens vom Inhalt, kann über die beiden Längen

`\fboxrule`	bestimmt die Dicke der Rahmenlinien
`\fboxsep`	bestimmt den Abstand zwischen Rahmen und Inhalt

`\fboxrule``\fboxsep` gesteuert werden, vgl. Abschnitt 3.11

Positionierung von LR-Boxen

`\raisebox{`*Offset*`}[`*Oberlänge*`][`*Unterlänge*`]{`*Text*`}`

`\raisebox` erzeugt ebenfalls eine `\mbox`. Jedoch können die darin enthaltenen Boxen (bzw. Zeichen) mit einem beliebigen vertikalen Offset zur laufenden Zeile positioniert werden. Ober- und Unterlänge sind die obere und untere Grenze der Box relativ zur Grundlinie und definieren den Abstand zur nächsten und vorherigen Zeile. Die Wirkungsweise wird wohl am deutlichsten an einem Beispiel, bei dem die Boxen sichtbar gemacht werden mit dem `\fbox`-Kommando. Im zweiten Beispiel wurde die Ober- und Unterlänge mit 5 ex bzw. 3 ex korrigiert (Beispiel 8).

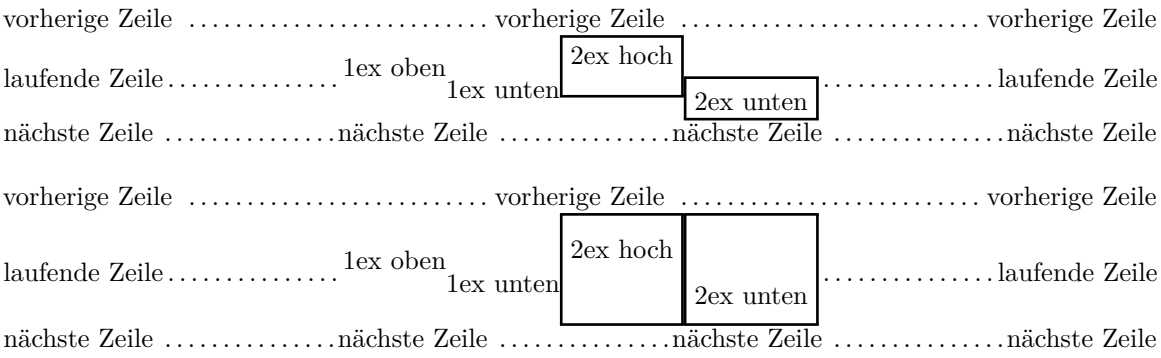
9.3 Absatzboxen

Es stellt sich auch die Notwendigkeit nach Boxen mit Zeilenumbruch. Hierfür gibt es die folgenden zwei Möglichkeiten, wobei die `minipage`-Variante die mächtigere darstellt und wieder alle Möglichkeiten des Textmodus eröffnet:

`\parbox[`*Position*`][`*Höhe*`][`*vertikal Pos.*`]{`*Breite*`}{`*Text*`}`

oder

Beispiel 8: Unterschiedliche Möglichkeiten zur Größe und Positionierung von Boxen.

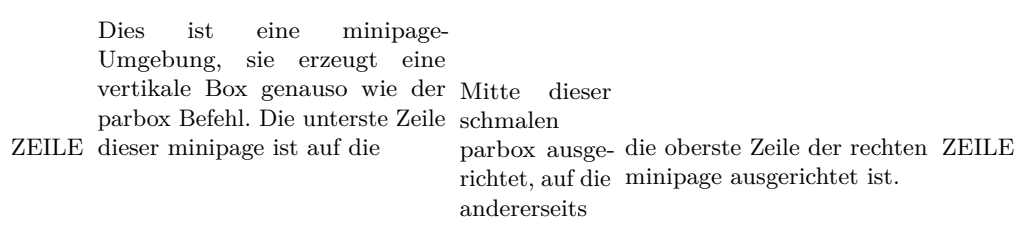



```
laufende Zeile\dotfill\raisebox{1ex}{1ex oben}\raisebox{-1ex}{1ex unten
      }\fbox{\raisebox{2ex}{2ex hoch}}\fbox{\raisebox{-2ex}{2ex unten}}
      \dotfill laufende Zeile\\

laufende Zeile\dotfill\raisebox{1ex}{1ex oben}\raisebox{-1ex}{1ex unten
      }\fbox{\raisebox{2ex}[5ex][3ex]{2ex hoch}}\fbox{\raisebox{-2ex}
      [5ex][3ex]{2ex unten}}\dotfill laufende Zeile\\
```

```
\begin{minipage}[Position][Höhe][vertikale Pos.]{Breite}
Text
\end{minipage}
```

Als Position zur laufenden Zeile können die Optionen `t`, `b`, `c` für top, bottom und center und als vertikale Positionen innerhalb der box auch `t`, `b`, `c` für top, bottom und center Verwendung finden. Beispiel:



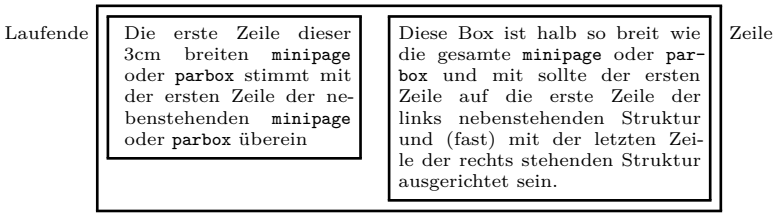
Der Quellcode hierfür gliedert sich in

```
\begin{minipage}[b]{4.6cm}
...
\end{minipage}\hfill
\parbox{3cm}{...}\hfill
\begin{minipage}[t]{4cm}
...
\end{minipage}
```

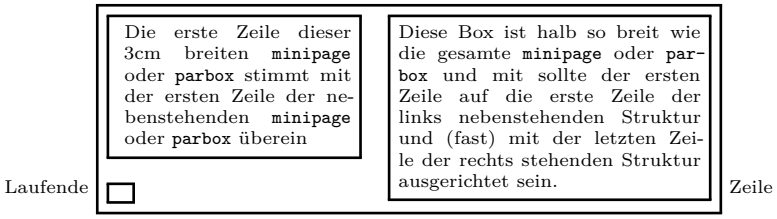
Innerhalb einer Minipage wird eine Parbox als eine Bearbeitungseinheit angesehen, also wie ein einzelnes Zeichen, was gerne zu überraschenden Ergebnissen führt. So bringt die Kombination

```
\begin{minipage}[b]{..}
\parbox[t]{..}{..}                    \parbox[t]{..}{..} \end{minipage}
```

Nicht das gewünschte Ergebnis von zwei oben gleich ausgerichteten Boxen deren unterste Zeile zur laufenden Zeile ausgerichtet ist:



Behoben werden kann das ganze mit einer sogenannten Nullzeile. Dadurch wird die `minipage` zweizeilig und kann somit an der untersten Zeile ausgerichtet werden. Die folgende `\mbox` dient lediglich dazu, dass diese Nullzeile nicht vollkommen leer ist, was zu Warnungen führt.



Erzeugt mit dem Code:

```
\begin{minipage}[b]{..}
\parbox[t]{..}{..} \hfill \parbox[t]{..}{..}\\[-\baselineskip]\mbox{}
\end{minipage}
```

9.4 Balkenboxen

Ein einfacher Balken wird mit

`\rule[`*vert. Offset*`]{`*Breite*`}{`*Höhe*`}`

gebildet, z. B.  (`\rule[1em]{1cm}{0.2cm}`) Er kann mit einer Höhen- oder Breitenangabe von 0 pt unsichtbar gemacht werden. Dies ist zum Beispiel für vertikalen/horizontalen Positionierungen anstelle von `\vspace`/`\hspace` sinnvoll.

9.5 Boxen speichern

Manchmal kann es vorkommen, dass man eine bestimmte Box, also ein Element mehrfach benötigt. Hierfür bietet LaTeX die Möglichkeit, ganze Boxen abzuspeichern um Sie mehrfach verwenden zu können. Hierzu muss zunächst eine neue Speicherbox definiert werden:

`\newsavebox{\`*Boxname*`}`

Um unter dem Makro `\Boxname` dann etwas zu speichern gibt es drei verschiedene Möglichkeiten:

`\sbox{\`*Boxname*`}{`*Code*`}`
`\savebox{\`*Boxname*`}[`*Breite*`][`*Position*`]{`*Code*`}`
`\begin{lrbox}{\`*Boxname*`}` *Code* `\end{lrbox}`

Möchte man somit beispielsweise eine 2 cm breite Box, in deren Inneren sich eine rote Rahmenbox mit dem Inhalt „Text“ rechts ausrichtet, so funktioniert dies mit:

```
\newsavebox{\mybox}
\savebox{\mybox}[2cm][r]{\fcolorbox{red}{white}{Text}}
```

Benutzt werden kann diese Box anschließend mit

`\usebox{`*Boxname*`}`

Im laufenden Text sieht das dann folgendermaßen aus:

laufende Zeile Text laufende Zeile

9.6 Drehung von Boxen

Um Boxen zu drehen benutzt man das rotating-Paket ([20]).

```
\begin{sideways} Text \end{sideways}
\begin{rotate}{Winkel} Text \end{rotate}
\begin{turn}{Winkel} Text \end{turn}
\turnbox{Winkel}{Text}
```

Die `sideways`-Umgebung dreht den Inhalt um 90°. Die Umgebung `rotate` und das Makro `\turnbox` sind äquivalent und drehen den Inhalt um den angegebenen Winkel in Grad. `turn` vergrößert zudem die umrahmende Box entsprechend, sodass es keine Überschneidungen mit dem Umgebenden Text gibt, siehe Beispiel 9.

Beispiel 9: Drehung eines Absatzes mit Hilfe des Paketes rotating

```
\fbox{
\begin{turn}{30}
\begin{minipage}{10em}
Dieser Text hier steht in einer
Minipage, die 10em breit und
um 30 Grad gedreht ist.
Dies erreicht man mit der
\texttt{turn}-Umgebung.
\end{minipage}
\end{turn}}
```


```
\fbox{
\begin{rotate}{30}
\begin{minipage}{10em}
Dieser Text hier steht in einer
Minipage, die 10em breit und
um 30 Grad gedreht ist.
Dies erreicht man mit der
\texttt{turn}-Umgebung.
\end{minipage}
\end{rotate}}
```

10 Grafiken

Bilder können mit dem Paket graphicx ganz einfach eingebunden werden. Das Makro hierfür lautet:

`\includegraphics[`*Optionen*`]{`*Bildpfad*`}`

Es positioniert die unter *Bildpfad* gespeicherte Datei an der Position des Makros. Optional kann die Skalierung geändert werden:

```
\includegraphics[width=Breite]{Bildpfad}
\includegraphics[height=Höhe]{Bildpfad}
\includegraphics[scale=Faktor]{Bildpfad}
\includegraphics[angle=Winkel]{Bildpfad}
```

Es ist auch möglich zu kombinieren. Jedoch sollte dabei darauf geachtet werden, dass die Kombinationen sinnig sind. Eine Angabe von Breite und Höhe, zum Beispiel, verzerrt das Bild entsprechend.

Die möglichen Bildformate hängen hierbei vom verwendeten Compiler ab. Wird `pdflatex` verwendet, können `*.pdf`, `*.png` und `*.jpg`-Dateien eingebunden werden, wird `latex` dagegen verwendet ist man auf postscript-Dateien, wie `*.ps` und `*.eps`, eingeschränkt (Außerdem ist zur Ausgabe im *.dvi-file ghostscript nötig). Sollen die Formate gemischt werden, lässt man am besten die Endung weg und LaTeX sucht sich den zugehörigen Treiber selbst. Prinzipiell sind jedoch (falls verfügbar) die Formate *.eps, *.ps oder *.pdf vorzuziehen, da es sich bei Ihnen um Vektorgrafiken handelt und sie somit frei skalierbar sind. Neuere Editoren wandeln zudem eingebundene `*.eps`-Grafiken bei Verwendung von `pdflatex` automatisch in ein `*.pdf` um, sodass auch Postscript-Dateien ohne Probleme eingebunden werden können.

Die Bilddatei sollte, damit LaTeX sie findet, im selben Ordner, wie das zu kompilierende Dokument oder in einem Unterordner liegen. Bei der Pfadangabe müssen dann jedoch die \ durch / ersetzt werden. Im folgenden Beispiel wird eine Bilddatei „Grafik“ so eingebunden, dass sie genauso breit ist wie eine Textzeile. Sie liegt in einem Unterordner namens „Bilder“:

```
\includegraphics[width=\linewidth]{Bilder/Grafik}
```

Für Einträge in ein Abbildungsverzeichnis wird der Leser auf die `figure`-Umgebung hingewiesen, siehe Kapitel 13.

11 Tabulatoren

Zur Verwendung von Tabulatoren wird die `tabbing`-Umgebung herangezogen:

```
\begin{tabbing}
Code \= Code \= Code \kill oder \\
Text4 \> Text5 \> Text6 \\
Text7 \> Text8 \> Text9 \\
...
\end{tabbing}
```

Innerhalb der `tabbing`-Umgebung werden Tabulatorstops mit `\=` gesetzt und die Zeilen mit `\\` umbrochen. In den nachfolgenden Zeilen kann dann mit `\>` auf die Tabulatorstopps umgesprungen werden. Die Zeile zur Definition der Tabulatorstops kann eine beliebige Musterzeile sein, deren Ausgabe mit dem Befehl `\kill` unterdrückt werden kann. Es können so viele Musterzeilen wie nötig eingefügt werden und an beliebiger Stelle stehen. Des Weiteren sind folgende Befehle auch sehr nützlich (siehe auch Beispiel 10):

`\pushtabs`	Löschen des Tabstopps
`\poptabs`	mit `\pushtab` gelöschten Tabstopp wieder einsetzen
`\+`	dauerhaftes Hochschalten um einen Tab
`\-`	dauerhaftes Runterschalten um einen Tab

Beispiel 10: Die Verwendung der `tabbing`-Umgebung

```
\begin{tabbing}
\hspace{2cm}\=\hspace{3cm}\=\hspace{3cm}\=\hspace{3cm}\=\kill
Buslinie \> 1 \> 2 \> 6 \> 11\\
Ziel \> Pommernstr. \> Schwabenstr. \> Klinikum \> Burgweinting\\
Abfahrt \> 9:00 \> 9:00 \> 9:00 \> 9:00\+\+\\
10:00 \> 10:00 \> 11:00\+\+\\
12:00\\
13:00\-\-\-\\
14:00 \> 16:00 \> 18:00 \> 20:00\\
\end{tabbing}
```

Buslinie	1	2	6	11
Ziel	Pommernstr.	Schwabenstr.	Klinikum	Burgweinting
Abfahrt	9:00	9:00	9:00	9:00
		10:00	10:00	11:00
				12:00
				13:00
	14:00	16:00	18:00	20:00

12 Tabellen

Mit den `box`- und `tabbing`-Befehlen können beliebige tabellenartige Strukturen erzeugt werden, jedoch würden sich dabei viele Arbeitsschritte wiederholen. Sinnvoller ist es daher, folgende Umgebungen zu verwenden:

```
\begin{array}[Position]{Spaltenformatierung}...\end{array}
\begin{tabular}[Position]{Spaltenformatierung}...\end{tabular}
\begin{tabular*}{Breite}[Position]{Spaltenformatierung}...\end{tabular*}
```

Im Gegensatz zu den letzten beiden darf die `array`-Umgebung nur im Mathe-Modus gesetzt werden, vgl. Kapitel 15. Diese drei Umgebungen entsprechen einer Absatzbox, wie die `minipage`. Sie erzeugen jeweils eine Tabelle an der Stelle ihres Auftretens. Die optionale Position ist der vertikale Positionsparameter (t top, b bottom, c center), der die Tabelle mit der obersten, untersten oder mittleren Zeile relativ zur laufenden Zeile ausrichtet. Für die Spaltenformatierung gibt es mehrere Möglichkeiten:

`l, r, c`	Spalte links, rechts und zentriert
`p{`*Breite*`}`	Der Text dieser Spalte wird in eine Absatzbox gesetzt (Blocksatz, mit automatischen Zeilenumbruch). Die oberste Zeile ist dabei auf die laufende Tabellenzeile ausgerichtet.
`*{`*Anzahl*`}{`*Spaltenf.*`}`	Die Spaltenformatierung wird *Anzahl*-mal reproduziert. `*{3}{|c|}` ist somit dasselbe, wie `|c|c|c|`
`|, ||`	ein, zwei vertikale Striche
`@{...}`	Dieser Ausdruck definiert den Zwischenraum zwischen Spalten oder vor der ersten bzw. nach der letzten Spalte und kann beispielsweise mit `\hspace` verändert werden. `...` steht dabei für einen beliebigen Textmodus-Befehl. Wird das Argument leer gelassen, so wird kein Zwischenraum eingefügt. Dies ist nützlich, um überstehende Linien zu vermeiden.

Weitere sinnvolle Befehle innerhalb der Tabelle:

`\\[`*Abstand*`]`	beendet die gesamte Tabellenzeile
`\tabularnewline[`*Abstand*`]`	liefert unabhängig vom Spaltentyp einen Zeilenumbruch innerhalb der Tabelle.
`\newline`	kommt innerhalb einer `p,m,b`-Spalte (siehe auch Abschnitt 12.1) für eine neue Zeile zum Einsatz.
`&`	liefert einen Spaltenwechsel innerhalb der Tabellenumgebung.
`\hline`	darf nur vor der ersten Zeile oder nach einem Zeilenumbruch stehen und erzeugt eine horizontale Linie der Breite der Tabelle. Entsprechend erzeugt `\hline\hline` eine Doppellinie.

`\cline{`*n*`-`*m*`}`	darf nur vor der ersten Zeile oder nach einem Zeilenumbruch stehen und erzeugt eine horizontale Linie vom linken Rand der *n*.ten bis zum rechten Rand der *m*.ten Spalte.
`\vline`	erzeugt einen vertikalen Strich über die Zeilenhöhe des Auftretens, auch innerhalb einer Spalte.
`\multicolumn{`*Spaltenanzahl*`}` `{`*Spaltenformatierung*`}{`*Text*`}`	erzeugt eine gemeinsame Spalte aus den folgenden Spalten, samt Zwischenräume.
`\multirow{`*Zeilenanzahl*`}` `{`*Breite*`}{`Text`}`	aus dem multirow-Paket fasst Zeilen zusammen und liefert eine bequeme Möglichkeit vertikal zu zentrieren. Als Breite kann einfach ein * eingegeben werden, damit die Breite auf die tatsächliche Textbreite optimiert wird.
`@{}`	Der Zwischenraum zwischen den Spalten wird mit diesem Spaltenformatierungsbefehl unterdrückt.
`tabular*`-Umgebung	Hier übergibt der Autor LaTeX die gewünschte Breite der Tabelle. Irgendwo in den Spaltendefinitionen muss ein `@{\extracolsep{\fill}}` stehen, damit in allen Folgespalten so viel Zwischenraum eingefügt wird, sodass die Tabellenbreite erreicht wird.

Ein Beispiel für eine komplexere Tabelle, aus [11]:

Tag	9.00-12.00		13.00-15.00		15.30-17.30	
	Fach	Lehrer	Fach	Lehrer	Fach	Lehrer
		Raum		Raum		Raum
Mo	UNIX	Dr. Schmidt	Fortran	Frau Schulz	Num.Math.	Herr Meier
		Rechenraum		Hörsaal		Hörsaal
Fr.	LaTeX	Frl. Müller	C-Praxis	Fr. Schulz	entfällt	
		Praktikum		Praktikum		

generiert mit diesem Quellcode

```
\begin{tabular}{|c||c|l|c|l|c|l|}
  \hline
  & \multicolumn{2}{c|}{9.00-12.00} & \multicolumn{2}{c|}{13.00-15.00}
      & \multicolumn{2}{c|}{15.30-17.30}
  \\\cline{2-7}
  \multirow{2}{*}{Tag}& \multirow{2}{*}{Fach}& Lehrer &\multirow{2}{*}{Fach}& Lehrer
      &\multirow{2}{*}{Fach}& Lehrer \\ \cline{3-3}\cline{5-5}\cline{7-7}
  && Raum && Raum && Raum \\ \hline\hline
  \multirow{2}{*}{Mo}&\multirow{2}{*}{UNIX}& Dr. Schmidt &\multirow{2}{*}{Fortran}&Frau Schulz&
      \multirow{2}{*}{Num.Math.} & Herr Meier \\ \cline{3-3}\cline{5-5}\cline{7-7}
  && Rechenraum && Hörsaal && Hörsaal \\ \hline
  \multirow{2}{*}{Fr.}&\multirow{2}{*}{\LaTeX{}}&Frl. Müller &\multirow{2}{*}{C-Praxis}&Fr. Schulz&
      \multicolumn{2}{c|}{\multirow{2}{*}{entfällt}}\\ \cline{3-3}\cline{5-5}
  && Praktikum && Praktikum & \multicolumn{2}{c|}{}\\ \hline
\end{tabular}
```

Und hier noch ein Beispiel für die `tabular*`-Umgebung:

```
\begin{tabular*}{.97\linewidth}{|p{1.2cm}|
p{4.3cm}|@{\extracolsep{\fill}}p{2cm}|p{2cm}|}
...
\end{tabular*}
```

Alle Zellen haben definierte Breiten	zwischen der zweiten und dritten Spalte wurde das `@{\extraccolsep{\fill}}` eingefügt	das hat auch Auswirkungen auf den Raum zwischen	der dritten und vierten Spalte wie man leicht erkennen kann
Wie man sieht,	wird bei `tabular*`	nur der Spaltenzwischenraum	gedehnt!

12.1 Weitere Spaltenformatierungen – Das array-Paket

Dieses Paket erweitert die `tabular`- und `array`-Umgebung unter anderem um drei weitere Spaltenformatierungen:

`m{`*Breite*`}`	erzeugt wie `p` eine Spalte der Breite „`Breite`“ allerdings sind die Zellen hier vertikal zentriert ausgerichtet.
`b{`*Breite*`}`	analog `m`, wobei die Zellen am Boden ausgerichtet sind
`>{`*Code*`}` bzw. `<{`*Code*`}`	setzt man vor bzw. nach den Spaltenformatierung l, r, c, p, m, b, und fügt den *Code* zu Beginn, bzw. am Ende jeder Zelle dieser Spalte ein.

Besonders die letzte Erweiterung ist sehr nützlich, da viele sich wiederholende Befehle nicht mehr geschrieben werden müssen. Ein Beispiel: Wenn der Inhalt einer ganzen Spalte mathematisch ist, dann müsste man normalerweise in jede dieser Zellen eine Matheumgebung einfügen (siehe Abschnitt 15). Mit dem array-Paket formatiert man besser die ganze Spalte (hier tabular-Umgebung, Mathe-Spalte zentriert, zweite Spalte linksbündig):

```
\begin{tabular}{>{$}c<{$}l} ... \end{tabular}
```

Wenn man öfter Spalten wie oben formatiert, dann ist es noch eleganter wenn man sich einen neuen Spaltentyp definiert:

`\newcolumntype{`*Name*`}{>{`*Code*`}{`*Spaltendefinition*`}<{`*Code*`}}`

Für das Beispiel oben könnte man also definieren `\newcolumntype{C}{>{$}{c}<{$}}` und es mit `\begin{tabular}{Cl} ... \end{tabular}` benutzen.

Man kann dem Befehl `\newcolumntype` auch Argumente übergeben.

```
\newcolumntype{V}[1]{>{\hspace{#1}}c}
```

Man übergibt hier also *ein* Argument (deshalb `[1]`) und dieses soll den zusätzlichen horizontalen Abstand zur `c`-Spalte festlegen. In der Tabelle kann das dann so aussehen:

```
\begin{tabular}{lV{2cm}} ... \end{tabular}
```

12.2 Variable Spaltenbreite – Das tabularx-Paket

Wird dieses Paket benutzt, so wird automatisch auch das array-Paket geladen (es stehen also auch alle Befehle des array-Pakets bereit, siehe Abschnitt 12.1). Deshalb muss das array-Paket nicht zusätzlich mit `\usepackage` geladen werden. Das tabularx-Paket ist eines der wenigen, das nach dem hyperref-Paket geladen werden muss.

Mit dem tabularx-Paket ist es möglich Tabellen mit einer bestimmten Breite zu erstellen. Im Gegensatz zur `tabular*`-Umgebung wird hier nicht der Spaltenzwischenraum, sondern die Spaltenbreite selbst

vergrößert. Man muss also nicht die Breite der Spalten mit `p{Breite}` selbst festlegen (was natürlich immer noch möglich ist), sondern man gibt die Breite der Tabelle vor und alle `X`-Spalten werden entsprechend gedehnt. Ist der Text innerhalb der `X`-Spalte breiter als die Spalte, dann wird automatisch im Blocksatz umgebrochen. Werden mehrere `X`-Spalten verwendet, so werden sie gleichmäßig gedehnt. Die Syntax lautet:

`\begin{tabularx}{`*Breite*`}{`*Spaltenformatierungen*`} ... \end{tabularx}`

Beispiel 11: Die `tabularx`-Umgebung

```
\begin{tabularx}{\linewidth}{|l|X|X|}
...
\end{tabularx}
```

Hier etwas linksbündiger Text	Die erste `X`-Spalte. Wie man sieht, steht dieser Text im Blocksatz.	Die zweite `X`-Spalte ist genauso breit wie die erste
Im Gegensatz zu	der `tabular*`-Umgebung wird	hier die Spaltenbreite und nicht der Zwischenraum vergrößert.

12.3 Variable Spaltenbreite mit Ausrichtung – Das tabulary-Paket

Das tabulary-Paket liefert ähnlich dem tabularx-Paket (siehe Abschnitt 12.2) eine Tabellenumgebung mit der man die Breite der Tabelle festlegen kann. Auch hier wird das array-Paket automatisch geladen.

Der Spalteninhalt wird automatisch an die Breite der Tabelle angepasst. Als „dehnbare“ Spalten stehen zur Verfügung:

- `R` Text rechtsbündig
- `C` Text zentriert
- `L` Text linksbündig
- `J` Text im Blocksatz

Die Stärke dieses Pakets liegt in der Verwendung mehrerer „dehnbarer“ Spalten. Im Gegensatz zum tabularx-Paket, bei dem alle X-Spalten gleichmäßig gedehnt werden, wird hier relativ zum Spalteninhalt gedehnt, sodass der Inhalt der (dehnbaren) Spalten möglichst bündig abschließt. Hier ist die Syntax:

`\begin{tabulary}{`*Breite*`}{`*Spaltenformatierungen*`} ... \end{tabulary}`

12.4 Verbessertes Spacing – Das booktabs-Paket

Macht man die horizontalen Linien mit dem Befehl `\hline`, so stellt man fest, dass der Abstand zum oberen Text nicht der gleiche ist wie zum unteren. Außerdem erscheinen beide Abstände zu klein (siehe z.B. die Tabelle auf Seite 80). Um diese Unschönheiten zu korrigieren, kann man das booktabs-Paket (oder das ctable-Paket) verwenden. Mit diesem Paket wurde der `\hline`-Befehl ergänzt durch folgende drei Befehle:

Beispiel 12: Die `tabulary`-Umgebung

```
\begin{tabulary}{\linewidth}{|l|C|L|}
...
\end{tabulary}
```

Hier etwas Text.	Die erste dehnbare Spalte. Wie man sieht, steht dieser Text zentriert.	Die zweite dehnbare-Spalte ist linksbündig und breiter wie die erste, da hier mehr Text steht.
Der Text	der einzelnen Zellen	schließen relativ bündig ab, zumindest bei den dehnbaren Spalten

`\toprule[`*Dicke*`]`
`\midrule[`*Dicke*`]`
`\bottomrule[`*Dicke*`]`

Der erste und der dritte Befehl sind gedacht für die oberste bzw. unterste horizontale Linie einer Tabelle. Sie sind dicker als bei `\hline` und haben nach unten bzw. oben mehr Abstand. Zwischen den einzelnen Zeilen benutzt man `\midrule`. Dies ist eine dünne Linie mit gleichem Abstand nach oben und unten. Anstelle des Befehls `\cline{n-m}` kann man nun

`\cmidrule[`*Dicke*`](`*Zuschnitt*`){`*n*`-`*m*`}`

verwenden, der ebenfalls eine dünne Linie mit gleichen Abständen nach oben und unten liefert. Die optionalen Parameter erlauben es jeweils für diese eine Linie eine Dicke zu wählen. Ebenfalls optional ist der Parameter *Zuschnitt*. Dafür kann entweder `l`, `r`, `l{`*Länge*`}`, `r{`*Länge*`}` oder eine Kombination von diesen verwendet werden und hat den Effekt, dass die Linie links und/oder rechts verkürzt wird. Wird die *Länge* nicht mit angegeben, so wird eine Standardlänge verwendet. Möchte man die Dicken global ändern, so geht das über den bekannten `\setlength`-Befehl (siehe Seite 33) mit den Längen

`\heavyrulewidth`
`\lightrulewidth`

Die erste Länge manipuliert die Dicke von `\toprule` und `\bottomrule`, die zweite die von `\midrule`. Arbeitet man mit mehreren Dicken, so ist es angenehmer sich mit `\newcommand` (siehe Seite 31) eine neue Linie zu definieren, z.B. liefert

```
\newcommand{otoprule}{\midrule[\heavyrulewidth]}
```

eine Linie mit der Dicke von `\toprule` aber mit gleichen Abständen nach oben und unten. Noch ein wichtiger Hinweis: Da die neuen Befehle andere Abstände vor und nach den Linien haben, schließen die vertikalen Linien nicht mehr bündig mit den horizontalen ab[1].

Als Beispiele für diese horizontalen Linien dienen die Tabellen in diesem Skript.

12.5 Mehrseitige Tabellen – Das longtable-Paket

Das longtable-Paket von David Carlisle bietet die Möglichkeit mehrseitige Tabellen zu setzen. Die Syntax entspricht weitestgehend der einer normalen `tabular`-Umgebung:

[1] Allerdings sollte man die vertikalen Linien aus ästhetischen Gründen sowieso nicht benutzen

```
\begin{longtable}[horizontale Pos.]{Spaltendefinitionen}
...&...&
\end{longtable}
```

Für die optionale horizontale Positionierung können hierbei die Werte `l`, `c` oder `r` (`c` ist Standard) verwendet werden.

Ein Seitenumbruch ist nur nach Abschluss einer Tabellenzeile und nicht innerhalb einzelner Zellen möglich. Der Umbruch kann wie üblich auch duch `\\*` oder `\nopagebreak` verhindert beziehungsweise mit `\newpage` erzwungen werden.

Weitere Formatierungsmöglichkeiten für die `longtable`-Umgebung kann der Paketdokumententation [3] entnommen werden.

13 Gleitobjekte – Positionierung von Bildern und Tabellen

Abbildung 13.1: Der TEX-Löwe

Große Objekte, wie lange Tabellen oder Bilder, sollten in ihrer Position gleitend definiert werden, da sonst der Seitenumbruch nicht immer besonders platzsparend gelingt. LaTeX liefert hier dem Autor die Möglichkeit: Ist genug Platz für das Objekt vorhanden dann setze es bitte an den Ort seiner Erscheinung entsprechend dem Quellcode, ansonsten führe den Text weiter und positioniere das Objekt an geeigneter Stelle neu. Dies alles geschieht mit

```
\begin{table}[Position]
TABELLE
\caption[optionaler Verzeichniseintrag]{Tabellenunterschrift}
\label{Marker}
\end{table}
```

oder

```
\begin{figure}[Position]
ABBILDUNG
\caption[optionaler Verzeichniseintrag]{Bildunterschrift}
\label{Marker}
\end{figure}
```

mit geeigneter Wahl der Position. Bei der Verwendung einer zweispaltigen Formatierung ist es sinnvoll einen `*` nach `table` oder `figure` einzufügen, damit wird ein über 2 Spalten reichender Platz für das Objekt reserviert im Gegensatz zur ungesternten Form. Die optional gewünschte Position des Autors kann als Liste aus `t` top, `b` bottom, `h` here oder `p` page (extra Seite am Ende des Abschnitts) gesetzt werden. In der Stern-Variante sind `b` und `h` nicht möglich. Der erste Wert der Liste hat die höchste Priorität und wird wenn möglich verwirklicht. Der Standard ist `tbp`, kein `h`! Ein `!` vor der Positionsangaben verstärkt den Positionierungswunsch des Autors und versucht die gewünschte Position zu erzwingen. Um Warnungen vorzubeugen, sollte in jeder Liste, und wenn auch am letzten Punkt der Parameter `p` stehen.

Mit dem `caption`-Befehl wird eine Bild- oder Tabellenbeschreibung eingefügt, die standardmäßig auch im Abbildungs- bzw. Tabellenverzeichnis steht (ein optionaler Text erlaubt dagegen andere Einträge in die entsprechenden Verzeichnisse). Das `\label`-Makro funktioniert vollkommen analog zu anderen Bezügen (Abschnitt 5.1). Ein einfaches Beispiel für ein Bild ist Abbildung 13.1. Oft möchte man jedoch, die Bildbeschreibung neben dem Bild positionieren, hierfür gibt es die beiden Möglichkeiten in Abbildung 13.2 und Abschnitt 13.2.1.

Regeln nach denen Gleitobjekte positioniert werden

Bei der Positionierung der Gleitobjekte befolgt LaTeX strenge Regeln, in ihrem Rahmen lässt sich jedoch sagen, dass jedes Gleitobjekt so früh wie möglich ausgegeben wird:

- Gleitobjekte werden niemals auf einer früheren Seite ausgegeben, als auf der, welcher sie definiert wurden.
- Gleitobjekte werden in der Reihenfolge in der sie definiert wurden ausgegeben.
- Gleitobjekte werden nur an einer nach den Positionsparametern erlaubten Position ausgegeben.
- Wenn kein `!` unter den Positionsparametern ist, so hält sich LaTeX and die Vorgaben der Stilparameter. Einige davon sind in Tabelle 13.1 angegeben, eine komplette Liste findet sich in [12, S.171ff.].
- Im Fall der Kombination `ht` ist `h` von höherer Priorität. Das Gleitobjekt wird am Punkt der Definition eingefügt, auch wenn oben auf einer Seite genug Platz wäre.
- Gleitobjekte, die beim Auftreten eines `\clearpage`-Befehls noch nicht positioniert wurden, werden unabhängig von der Wahl der Positionierungsparameter auf einer eigenen Seite oder Spalte ausgegeben.

Möchte man noch weitere Positionseinschränkungen vornehmen, dann finden sich ein paar Tricks in Abschnitt 13.1 und Tabelle 13.1.

13.1 Floats einschränken - Das placeins-Paket

Mit

```
\usepackage{placeins}
```

kann der Autor mittels

```
\FloatBarrier
```

Barrieren setzen, die Gleitumgebungen nicht überwinden können. Sollen Gleitumgebungen pauschal nicht über eine `section` hinausgleiten, so erreicht man dies mit

Abbildung 13.2: Noch einmal der Löwe nur diesmal zwei minipages innerhalb der `figure`-Umgebung eine mit dem Bild, die andere mit der `caption`, beide vertikal zentriert relativ zueinander. Darunter befindet sich der entsprechende Code

```
\begin{figure}[htbp]
   \begin{minipage}[c]{.49\linewidth}
       \centering
       \includegraphics[width=5cm]{Bilder/ctanlion}
   \end{minipage}
   \hfill
   \begin{minipage}[c]{.5\linewidth}
      \caption[Löwe 2]{Noch einmal...}
   \end{minipage}
\end{figure}
```

`\usepackage[section]{placeins}`

Hier wird der `\section`-Befehl einfach um ein `\FloatBarrier` erweitert.

`\FloatBarrier` geht sehr strikt vor. Man kann dies soweit lockern, dass Gleitobjekte die Barriere insoweit noch überwinden können, damit sie noch auf der selben Seite eingebunden werden. Für das Überwinden nach „oben“ nimmt man die Paketoption `above`, nach unten `below`.

13.2 KOMA-Spezielle `caption`-Einstellungen

13.2.1 Position der `caption`

LaTeX geht beim Befehl `caption` immer von einer *Unter*schrift der Gleitumgebungen aus. Große Tabellen sollten aber eine *Über*schrift aufweisen, damit der Leser zu Beginn weiß worum es sich auch in der nachfolgenden Tabelle handelt (eine Überschrift hat eine andere Formatierung wie eine Unterschrift, man beachte Abstand Tabelle und Über- bzw. Unterschrift).

`\documentclass[captions=`*Option*`]{scr...}`

Zu beachten ist, dass dies nur die Abstände korrigiert, die `caption` taucht immer nur da auf, wo sie auch definiert wurde. Soll bei einzelnen Tabellen gewechselt werden so geschieht dies mit

`\captionabove[`*Verzeichniseintrag*`]{`*Text*`}`	`caption` oben
`\captionbelow[`*Verzeichniseintrag*`]{`*Text*`}`	`caption` unten

Für Beschreibungen neben dem Gleitobjekt kann man anstelle zweier Parboxen jetzt auch die folgende Umgebung verwenden

Tabelle 13.1: Tricks für die Positionierung von Gleitobjekten

`\setcounter{topnumber}{`*Zahl*`}`	Maximale Anzahl der Gleitobjekte bei top-Positionierung pro Seite (default=2)
`\setcounter{bottomnumber}{`*Zahl*`}`	Maximale Anzahl der Gleitobjekte bei bottom-Positionierung pro Seite (bei zweispaltiger Formatierung heißt der Counter `dbltotalnumber`) (default=1)
`\setcounter{totalnumber}{`*Anzahl*`}`	Maximale Anzahl der Gleitobjekte pro Seite (default=3)
`\renewcommand{topfraction}{`*Wert <1*`}`	Seitenbruchteil für top-Platzierungen
`\renewcommand{bottomfraction}{`*Wert <1*`}`	Seitenbruchteil für bottom-Platzierungen
`\renewcommand{textfraction}{`*Wert <1*`}`	Seitenbruchteil für Text
`\renewcommand{\floatpagefraction}{`*Wert <1*`}`	Der Bruchteil einer Seite, welcher für float-Objekte reserviert ist (Standard 0.5)
`\supressfloats[`*Pos*`]`	unterbindet das Auftreten nachfolgend erklärter Gleitobjekte für die laufende Seite. Pos. (optional): t oder b verhindert Ausgabe am Anfang oder Ende der Seite

```
\begin{captionbeside}[Verzeichnistitel]{Titel}[Anordnung][Breite][Offset]
...
\end{captionsbeside}
```

Diese Umgebung steht innerhalb der Gleitumgebung und umschließt selbst das Gleitobjekt. Der *Titel* ist der, der `caption`, als *Anordnung* gibt es `l`(links), `r`(rechts), `i`(innen) oder `o`(außen). Wird eine *Breite* angegeben so wird die genutzte *Breite* bezüglich dem Textkörper zentriert. Ein positiver Wert von *Offset* entspricht einer Verschiebung nach rechts. Wird hinter dem optionalen Parameter *Offset* noch ein Stern gesetzt, so stellt *Offset* im doppelseitigen Druck auf linken Seiten eine Verschiebung relativ zum rechten Rand dar. Ein *Offset* von 0 pt wäre somit bündig zum inneren Rand. In Tabelle 13.2 sind unter anderem Werte für die Dokumentklassenoption `captions` angegeben, die für die Umgebung `captionbeside` wichtig sind.

13.2.2 Formatierung der `caption`

Unterscheidung zwischen einzeilig und mehrzeilig Mit dem Wert `nooneline` für die Dokumentklassenoption `captions` wird die automatische Zentrierung bei einzeiligen Über- bzw. Unterschriften abgeschaltet.

Schriftattribute Man kann die Schrifteinstellungen des Gleitumgebungslabels („Abbildung“, „Tabelle“) und der Gleitumgebungsbeschreibung mit dem `\setkomafont`-Befehl bearbeiten, welcher in Abschnitt 4.1.4 bereits erklärt wurde.

Tabelle 13.2: Werte für die Dokumentklassenoption `captions`

`heading`	Abstände von `\caption` immer wie `\captionabove`
`signature`	Abstände immer wie bei Bildunterschrift (Standard)
`figureheading`	Wie `heading`, jedoch nur für `figure`
`figuresignature`	Wie `signature`, jedoch nur für `figure`
`tableheading`	Wie `heading`, jedoch nur für `table`
`tablesignature`	Wie `signature`, jedoch nur für `table`
`bottombeside`	Titel für `captionsbeside`-Umgebung auf unterster Linie des Gleitobjekts
`topbeside`	Titel für `captionsbeside`-Umgebung auf oberster Linie des Gleitobjekts
`centeredbeside`	Titel für `captionsbeside`-Umgebung vertikal zentriert rel. zum Gleitobjekt
`innerbeside`	Titel für `captionsbeside`-Umgebung bei zweiseitigen Dokumenten innen
`leftbeside`	Titel für `captionsbeside`-Umgebung links von Gleitobjekt
`rightbeside`	Titel für `captionsbeside`-Umgebung rechts von Gleitobjekt
`outerbeside`	Titel für `captionsbeside`-Umgebung bei zweiseitigen Dokumenten außen

Hängend oder eingerückt Bei den KOMA-Klassen „hängt“ standardmäßig die Beschreibung am Label, möchte man davon abweichen, so kann man dies mit

```
\setcapindent{Einzug}
\setcapindent*{XEinzug}
```

Einzug ist ein Maß wie weit die zweite Zeile eingerückt wird. Ist der Wert negativ, so bekommt man nach dem Label einen Zeilenumbruch und nur die erste Zeile wird eingerückt. Soll ein Zeilenumbruch nach dem Label erfolgen und alle Zeilen um *XEinzug* eingerückt werden, so nimmt man die gesternte Form.

Soll auf den Standard zurückgeschaltet werden, so geht dies mit dem Makro

```
\setcaphanging
```

Beispiel 13 zeigt die verschiedenen Möglichkeiten zum Einrücken der Gleitobjektbeschriftungen.

Breite und Ränder Die `Breite` der `caption` kann eingestellt werden über

```
\setcapwidth[Ausrichtung]{Breite}
```

wobei das optionale Argument die horizontale Ausrichtung festlegt. Man kann wählen zwischen `l` (linksbündig), `r` (rechtsbündig), `c` (zentriert), `i` (innen) und `o` (außen).

Einen Rand legt man fest mit

```
\setcapmargin[Rand links]{Rand rechts}
\setcapmargin*[Rand innen]{Rand außen}
```

Soll der linke Rand vom rechten Rand abweichen so ist das optionale Argumente nötig. Für den doppelseitigen Druck gibt es die gesternte Variante.

Beispiel 13: Beispiel für verschiedene Werte des Makros `\setcapindent`

Standardversion der `caption`:

Gleitobjekt 1: Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcapindent{2cm}` – *Einzug* = 2 cm:

2 cm

Gleitobjekt 2: Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcapindent*{1cm}` – gesternte Version mit *XEinzug* = 1 cm:

1 cm

Gleitobjekt 3:
Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcapindent{-2cm}` – negativer Wert bei `\setcapindent`:

2 cm

Gleitobjekt 4:
Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

`\setcaphanging` – Zurückschalten zur Standardversion:

Gleitobjekt 5: Dies ist eine sehr lange `caption`, die über viele, wenn nicht gar sehr viele Zeilen geht. Sie ist dafür da zu verdeutlichen, die man die `caption` setzen kann, hängend oder eingerückt.

Zusammensetzung von Label und Trenner Das Label ist standardmäßig aufgebaut aus dem Namen des Gleitobjektes („Abbildung“, „Tabelle“) dem zugehörigen Zähler und eventuell einem Gliederungspunkt. Möchte man an diesem Aufbau was ändern oder hinzufügen so geht das mit `\renewcommand`. Standardmäßig ist das Label für Abbildungen so definiert: [1]

```
\newcommand*{\figureformat}{\figurename~\thefigure\autodot}
```

Bei Tabellen nimmt man statt `\figureformat` `\tableformat`.

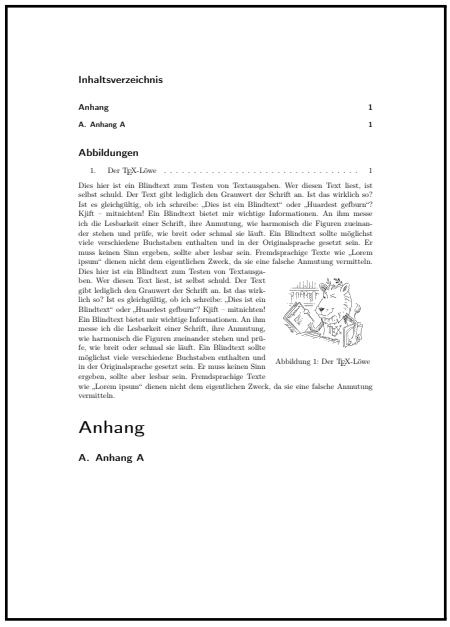
Der Trenner ist normalerweise ein Doppelpukt und ein Leerzeichen. Möchte man stattdessen einen Gedankenstrich einschließlich der notwendigen Leezeichen, so ginge dies mit:

```
\renewcommand*{\captionformat}{\ --\ }
```

13.3 Objekte von Text umfließen lassen – Das wrapfig-Paket

Grafiken und Tabellen sollten im Allgemeinen vom Text abgehoben dargestellt werden. Eine Ausnahme hierbei sind besonders schmale Objekte. Hier kann es durchaus sinnvoll sein, diese vom Text umfließen zu lassen, um Platz einzusparen. Das Paket wrapfig von Donald Arseneau definiert zwei Umgebungen, die es ermöglichen ein Objekt vom Text *eines* Absatzes umfließen zu lassen.

```
\begin{wrapfigure}[Zeilen]{Position}[Randüberhang]{Breite}
...
\end{wrapfigure}
\begin{wraptable}[Zeilen]{Position}[Randüberhang]{Breite}
...
\end{wraptable}
```


Inhaltsverzeichnis

Anhang 1

A. Anhang A 1

Abbildungen

1. Der TEX-Löwe 1

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburr“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.
Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburr“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Abbildung 1: Der TEX-Löwe

Anhang

A. Anhang A

```
\usepackage{wrapfig,graphicx}
\usepackage{blindtext}

\begin{wrapfigure}{r}{5cm}
\centering
\includegraphics[width=4cm]{ctanlion}
\caption{Der \TeX-Löwe}
\end{wrapfigure}
\blindtext
```

[1] `\autodot` setzt an seiner Stelle, das entsprechende Trennzeiche. Im Fall der `caption` das, welches in `captionformat` hinterlegt ist. Normalerweise handelt es sich dabei um einen Doppelpunkt gefolgt von einem Leerzeichen.

14 Verzeichnisse

14.1 Abbildungs- und Tabellenverzeichnis

Analog zum Inhaltsverzeichnis werden Abbildungs- und Tabellenverzeichnis mit den Makros

```
\listoffigures
\listoftables
```

erstellt. Die zugehörigen Hilfsdateien haben die Endung `*.lof` (Abbildungen) und `*.lot` (Tabellen). Ebenfalls wie beim Inhaltsverzeichnis wird somit wieder zweifaches Kompilieren benötigt.

14.2 Verzeichnisse manipulieren

Zusätzlich zu den automatischen Einträgen, ist es möglich manuelle Eintragungen in die Verzeichnisse zu erstellen:

`\addcontentsline{`*Verzeichniskürzel*`}{`*Gliederungstyp*`}{`*Verzeichniseintrag*`}`

Das *Verzeichniskürzel* ist in der Regel entweder `toc`, `lof` oder `lot`. Der *Gliederungstyp* ist z. B. `section` im Inhaltsverzeichnis, so dass der Eintragtext wie eine weitere `section` dargestellt wird. Als Seitenzahl wird wie bei den Überschriften, die aktuelle automatisch eingetragen. Die Überschriften der einzelnen Verzeichnisse und auch die Ausgabe zu Beginn einer Bildunterschrift werden oft durch eingebundene sprachbezogene Ergänzungspakete bestimmt (babel). Sie können jedoch ebenfalls geändert werden:

`\renewcommand{`*Befehl der Überschrift/Bezeichnung*`}{`*neuer Verzeichnistitel*`}`

Die möglichen Befehle für Überschriften und Bezeichner lautet:

`\bibname`	„Literaturverzeichnis“ bei scrbook und scrreprt[1]
`\refname`	„Literatur“ bei scrartcl
`\listtablename`	„Tabellenverzeichnis“
`\listfigurename`	„Abbildungsverzeichnis“
`\tablename`	„Tabelle“
`\figurename`	„Abbildung“

[1] (mit biblatex-Paket nur „Literatur“)

14.3 Literaturverzeichnis

14.3.1 Manuelle Erstellung des Literaturverzeichnisses

Ein Literaturverzeichnis kann manuell mit der Umgebung

```
\begin{thebibliography}{Mustermarke}
\bibitem[Marke]{Bezug1} Text1
\bibitem[Marke]{Bezug2} Text2
\bibitem[Marke]{Bezug3} Text3
...
\end{thebibliography}
```

erzeugt werden. Die Mustermarke ist dabei eine Reihe von beliebigen Zeichen, die LaTeX die Größe der Marke mitteilt. Sie sollte demnach mindestens so groß sein wie die längste `Marke`. Mit Hilfe des optionalen Parameters `Marke` kann eine beliebige Markierung für jeden einzelnen Literaturverweis erstellt werden (Standard: laufende Zahl in eckigen Klammern). Bezug ist ein zwingender Parameter bzw. ein Wort mit dem die Zuordnung stattfindet (darf keine Kommata enthalten). An der Stelle im Text an der man den Literaturverweis haben möchte schreibt man entsprechend

```
\cite[Informationen]{Bezug1,Bezug2,....}
```

Beispiel: Dieses Skript baut in Teilen auf dem Vorgängerskript von [5] (`\cite{roedl}`) und dem Buch von Helmut Kopka [11, 3. Auflage] (`\cite[3. Auflage]{kopka}`) auf. Eine so erstellte Bibliographie ist allerdings an die Reihenfolge der Einträge im Literaturverzeichnis geknüpft und nummeriert entsprechend durch. Mit den optionalen Informationen kann noch beliebiges weiteres Material wie Seitenzahl oder Kapitel zur Verfügung gestellt werden.

Oft werden beim Erstellen eines Dokuments entsprechende Literaturverweise ergänzt aber auch wieder entfernt. So stellt sich die Notwendigkeit nach einem Literaturverzeichnis, das nur die im Dokument verwendeten Einträge ins Ausgabefile übernimmt und auch die Nummerierung der Querverweise aktualisiert.

14.3.2 Automatisches Literaturverzeichnis mit Biber und dem biblatex-Paket

LaTeX bietet eine bequeme Möglichkeit die verwendeten Literaturzitate mit einer Literaturdatenbank zu synchronisieren, die der Autor LaTeX in einem `*.bib`-File mitteilt. Hierfür kommt die TeX-Variante BibTeX zum Einsatz. Sie liest nur die Zitate aus der Literaturdatenbank aus, die im Dokument wirklich Verwendung finden und sortiert sie entsprechend. Diese Variante des Literaturverzeichnisses hat den Vorteil, dass man, zum Beispiel bei Google Books, direkt BibTeX-Dateien der betreffenden Bücher herunterladen kann. Außerdem unterstützt das Literaturverwaltungssystem „Citavi“, dass an der Universität Regensburg als Campus-Lizenz zur Verfügung steht, den Export der *kompletten Citavi-internen Datenbank* in ein BibTeX-file.

Allerdings ist es relativ schwierig das Literaturverzeichnis mit BibTeX zu formatieren. Hier hilft uns das Paket biblatex, mit dem man auch mit einfachen LaTeX-Befehlen die Einträge formatieren kann, einem aber auch viele Stile für die Zitate und für das Literaturverzeichnis anbietet. Praktischerweise greift biblatex auf das `babel`-Paket zurück, man muss sich also kein Gedanken über eine deutsche Anpassung machen. Zusätzlich zu den hier vorgestellten Funktionen, ist es mit diesem Paket auch möglich eigene Stile zu erstellen, mehrere Literaturverzeichnisse anzulegen (beispielsweise aufgeteilt nach Kapitel oder Thema) oder die Einträge in einen Index mit aufzunehmen. Genaue Erläuterungen dazu finden sich in der zugehörigen Paketdokumentation [19].

Um mit BibTEX und dem Paket biblatex zu arbeiten legt man eine oder auch mehrere Dateien mit der Erweiterung *.bib an. Diese enthalten die Literaturangaben. Zunächst werden jedoch die Befehle innerhalb des eigentlichen Dokumentes erläutert:

```
\usepackage[backend=biber, Optionen]{biblatex}
\bibliography{*.bib-File}
```

Das Laden des Paketes funktioniert wie üblich, allerdings ist bei älteren Systemen (bis TeXLive 2012) die Option `backend=biber` nötig. Dies liegt daran, dass biblatex mittlerweile nicht mehr Standardmäßig BibTEX, sondern Biber benutzt. Biber ist im Endeffekt das gleiche wie BibTEX, jedoch mit Unicode-Support. Arbeitet man überwiegend auf Rechnern mit einer etwas älteren Distribution, so kann alternativ auch `backend=bibtex` verwendet werden, jedoch dürfen die `*.bib`-Dateien in diesem Fall keine Unicode-kodierten Sonderzeichen enthalten.

Zusätzlich zum Paket selbst muss jedoch auch noch die Literaturdatenbank geladen werden, was durch Angabe des Dateipfades (falls sich die Datei im selben Ordner befindet genügt der Name) mit dem `\bibliography` geschieht. Die Dateiendung .bib muss dabei nicht angegeben werden. Allerdings ist zu beachten, dass für den Fall, dass es sich um mehrere Dateien handelt, die Namen durch Kommata, jedoch ohne nachfolgendes Leerzeichen getrennt werden.

Die Paketoptionen bestimmen letzten Endes die Formatierung der Zitate und des Literaturverzeichnisses fest.

```
\printbibliography
```

Dieser Befehl wird an der Stelle im Dokument positioniert, an der das Literaturverzeichnis erscheinen soll. Dort werden jedoch nur die Literaturangaben abgedruckt, auf die im Text entweder mit einem gedrucktem oder unterdrücktem Bezug verwiesen wird.

```
\cite[vorher][nachher]{Bezug}
\nocite{Bezug}
```

Das Makro `\cite` setzt dabei einen Bezug zum betreffenden Eintrag des Literaturverzeichnisses. *Vorher* und *nachhher* sind dabei Textbausteine, die z. B. noch zusätzlich auf einen bestimmten Teil der Literatur, also eine Seite oder ein Kapitel verweisen. Beispiel: [vgl. 27, S. 345] `\cite[vgl.][S. 345]{einfuehrung}`

`\nocite` setzt dahingegen einen stummen Bezug, also einen Bezug der gar nicht erscheint. Dies ermöglicht es Einträge im Literaturverzeichnis auszugeben, auf welche man gar nicht verwiesen hat. Möchte man alle Einträge ausgegeben haben, welche die `*.bib`-Datei enthält, so funktioniert dies durch einen stummen Bezug auf das gesamte Verzeichnis:

```
\nocite{*}
```

Das Ausführen von Biber beginnt ganz normal mit dem Starten des LATEX-Interpreten (`F6`), damit das `*.aux`-file aktualisiert wird; dann wird der Biber-Interpretr (`F11`) starten (erzeugt ein `*.bbl`-file) und zum Schluss noch zweimal den LATEX-Interpreten starten für die Verzeichniseinträge und danach für die Erstellung der Zitatnummern im laufenden Text. Je nach Editor kann dies auch automatisch erfolgen.

In der `*.bib`-Datei stehen alle Informationen über das zitierte Werk. Die Einträge einer solchen Datei sehen wie folgt aus:

```
@Typ{Bezug1,
AUTHOR = {Autor1 and Autor2 and Autor3...},
TITLE = {Titel},
JOURNAL = {Journal oder Verlag},
YEAR = {Erscheinungsjahr},
VOLUME = {Auflage},
NUMBER = {issue bei papers},
PAGES = {entsprechender Ausschnitt von–bis},
MONTH = {Erscheinungsmonat}
}
@Typ{Bezug2,
AUTHOR = {...
```

Die Angabe der Einträge ist dabei nicht zwingend, jedoch erwartet LaTeX je nach *Typ* bestimmte Angaben. Die wichtigsten Typen samt der erwarteten Angaben, finden sich in Tabelle 14.3, eine vollständige Liste gibt es in der Paketdokumentation von `biblatex`. Die Groß- und Kleinschreibung spielt bei den Argumenten AUTHOR, TITLE usw. keine Rolle. Als Beispiel dient wieder [27].

```
@book{einfuehrung,
author = {Voß, Herbert},
year = {2012},
title = {Einführung in \LaTeX2e: Unter Berücksichtigung von pdf\LaTeX,
Xe\LaTeX und Lua\LaTeX},
keywords = {Empfehlung},
edition = {1},
publisher = {Lehmanns Media},
isbn = {978-3-86541-462-5}
}
```

Man kann in den Paketoptionen zudem einen Stil festlegen. Man kann sowohl für die Zitate als auch für das Verzeichnis den gleichen Stil, muss man aber nicht.

`style=`*Stil*	überschreibt den Stil für das Zitieren und fürs Literaturverzeichnis
`citestyle=`*Stil*	überschreibt den Stil für das Zitieren
`bibstyle=`*Stil*	überschreibt den Stil für das Literaturverzeichnis

Eine Auswahl an möglichen Stilen findet man in Tabelle 14.1. Die Stile unterstützen unterschiedliche Varianten, die alle in der Dokumentation gut verständlich erklärt sind. Hier soll nur die `-comp` Variante vorgestellt werden:

```
\usepackage[backend=biber,style=numeric-comp]{biblatex}
```

Hier bekommt man wegen dem Style `numeric` eine Zahl in eckigen Klammern. Die zusätzliche (optionale) Variante `-comp` sorgt dafür dass bei mehreren Angaben die Zahlen sortiert und zusammengefasst werden. Statt [8,3,1,7,2] bekommt man also [1-3,7,8].

Man kann festlegen wie die Einträge im Literaturverzeichnis sortiert werden. Es gibt vordefinierte Schemata, zu finden in Tabelle 14.2, man kann sich aber auch sein eigenes Schema basteln. Hierbei sei wieder auf die Paketdokumentation verwiesen.

```
sorting = Option
```

In Tabelle 14.2 findet man die möglichen *Sortieroptionen*.

Mit dem `biblatex`-Paket gibt es mehrere Zitierbefehle, welche immer die Struktur

Tabelle 14.1: Stile für das Paket `biblatex`

`numeric`	geeignet für In-Text-Zitate, Varianten: -comp, -verb [5], [5, S.59], [siehe 5, S. 59–63]
`authoryear`	geeignet für In-Text-Zitate, Varianten: -verb Goossens, Mittelbach, and Samarin 1994, S. 59–63
`alphabetic`	geeignet für In-Text-Zitate und Fußnoten, Varianten: -comp, -ibid, -icomp [GMS94], [GMS94, S. 59], [siehe GMS94, S. 59–63]
`authortitle`	geeignet für Fußnoten Varianten: -comp, -ibid, -icomp, -terse, -tcomp, -ticomp siehe Aristotle, *Rhetoric*, S. 59–63
`verbose`	geeignet für Fußnoten Varianten: -ibid, -note, -inote, -trad1, -trad2, -trad3 Immanuel Kant. „Kritik der praktischen Vernunft“ . In: *Kants Werke. Akademie Textausgabe*. Bd. 5: *Kritik der praktischen Vernunft. Kritik der Urtheilskraft*. Berlin: Walter de Gruyter, 1968, S. 1–163 (im Folgenden zit. als KpV).
`draft`	Entwurfszitierstil der den Zitatschlüssel ausgibt Varianten: bertram; companion; aristotle:anima

Tabelle 14.2: Sortieroptionen für das Paket `biblatex`

`none`	nicht sortiert, Auflistung nach Zitierreihenfolge
`nty`	Name, Titel, Jahr
`nyt`	Name, Jahr, Titel
`nyvt`	Name, Jahr, Volume, Titel
`anyt`	alphabetischem Label, Name Jahr, Titel
`anyvt`	alphabetischem Label, Name Jahr, Volume, Titel
`ynt`	Jahr, Name, Titel
`ydnt`	absteigendem Jahr, Name, Titel

Tabelle 14.3: Auflistungstypen für BibTEX und Biber

Typ	*erwartet*	*optional*
article	author, title, journaltitle, year/date	translator, annotator, commentator, subtitle, titleaddon, editor, editora, editorb, editorc, journalsubtitle, issuetitle, issuesubtitle, language, origlanguage, series, volume, number, eid, issue, month, pages, version, note, issn, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
book	author, title, year/date	editor, editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
booklet	author/editor, title, year/date	subtitle, titleaddon, language, howpublished, type, note, location, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
collection	editor, title, year/date	editora, editorb, editorc, translator, annotator, commentator, introduction, foreword, afterword, subtitle, titleaddon, maintitle, mainsubtitle, maintitleaddon, language, origlanguage, volume, part, edition, volumes, series, number, note, publisher, location, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate
misc	author/editor, title, year/date	subtitle, language, howpublished, type, version, organization, doi, eprint, url, urldate
online	author/editor, title, year/date, url	subtitle, titleaddon, language, version, note, organization, date, month, year, addendum, pubstate, urldate
thesis	author, title, type, institution, year/date	subtitle, titleaddon, language, note, location, month, isbn, chapter, pages, pagetotal, addendum, pubstate, doi, eprint, eprintclass, eprinttype, url, urldate

\befehl[vorher][nachher]{Bezug}

haben. *Vorher* steht immer vor dem Zitat (z.B. *siehe*), *nachher* immer nach dem Zitat (oft Seitenangabe). Bei nur einer Eingabe wird diese als *nachher* behandelt. Möchte man nur *vorher*, so muss man ein leeres zweites Argument benutzen. Mögliche Zitierbefehle siehe Tabelle 14.4.

Tabelle 14.4: Zitierbefehle für das Paket biblatex

\cite[*vorher*][*nachher*]{*Bezug*}	Standard Zitierbefehl zitiert nach dem gesetzten Zitierstil
\parencite[*vorher*][*nachher*]{*Bezug*}	umklammertes Zitat (rund bzw. eckig)
\footcite[*vorher*][*nachher*]{*Bezug*}	Erzeugt Literaturangabe und Fußnote
\autocite[*vorher*][*nachher*]{*Bezug*}	je nach Zitierstil unterschiedliche Ausgabe, siehe Paketoption autocite
\textcite[*vorher*][*nachher*]{*Bezug*}	in allen nicht-verbose Stilen verfügbar, gibt Autoren gefolgt vom Zitierschlüssel aus
\supercite{*Bezug*}	in numerischen Stilen verfügbar, ergibt hochgestellte Zitatnummer ohne Klammern
\nocite{*Bezug*/*}	fügt key/alle Einträge der Datenbank zu Bibliographie ohne Zitat hinzu
\fullcite[*vorher*][*nachher*]{*Bezug*}	fügt kompletten Literaturverweis wie aus Bibliographie ein
\citeauthor[*vorher*][*nachher*]{*Bezug*}	Ausgabe Autoren, Editoren oder Übersetzer
\citetitle[*vorher*][*nachher*]{*Bezug*}	Ausgabe wenn vorhanden shorttitle ansonsten title
\citeyear[*vorher*][*nachher*]{*Bezug*}	Ausgabe Erscheinungsjahr
\citedate[*vorher*][*nachher*]{*Bezug*}	Ausgabe volles Datum
\citeurl[*vorher*][*nachher*]{*Bezug*}	Ausgabe url-Feld

Man kann auch auf mehere Werke auf einmal verweisen:

\cites[*vorher*][*nachher*]{*Bezug*}[*vorher*][*nachher*]{*Bezug*}...

Weitere nützliche Paketoptionen findet man in Tabelle 14.5.

Tabelle 14.5: Weitere Paketoptionen für das Paket biblatex

autocite	plain	verhält sich wie \cite
	inline	verhält sich wie \parencite
	footnote	verhält sich wie \footcite
	superscript	verhält sich wie \supercite
		Standardwert hängt vom Zitierstil ab
backref	true/false	fügt Seitenzahlen der Zitate an die Bibliographieeinträge an
backend	biber	unterstützt von ASCII bis UTF-8, on-the-fly rencoding, Sortierungsregeln über sortlocale, sortcase, sortupper einstellbar
	bibtex	traditionelles BibTeX, unterstützt nur ASCII, Sortieren ist immer abhängig von Groß/Kleinschreibung
	bibtex8	8-bit Implementierung von BibTeX, zusätzlich 8-bit encodings wie Latin 1
	bibtexu	BibTeX mit Unicode Unterstützung, nicht aktiv entwickelt als Backend
heading	label	eigene Bibliographieüberschriften einsetzen, definieren mit \defbibheading{label}{code}

Man kann auch auf mehrere Datenbanken verweisen:

\addbibresource{*Dateiname*.bib}

```
\addbibresource[location=remote]{http://www.bibtex.org/bib/6}
\addbibresource[location=remote,label=lan]{ftp://bib/file.bib}
```

Wie man kleine Formatierungen der Zitate und der Einträge im Literaturverzeichnis vornimmt soll anhand eines kleinen Beispiels gezeigt werden:

```
\usepackage[style=authortitle]{biblatex}
\DeclareFieldFormat[book]{title}{$\clubsuit$ #1 $\spadesuit$}
\DeclareFieldFormat[book]{citetitle}{$\ast$ \textsc{#1} $\ast$}
\DeclareFieldFormat[article]{citetitle}{[ #1 ]}
\renewcommand*{\multinamedelim}{ + }
\renewcommand*{\finalnamedelim}{ und manchmal }
\begin{document}
\cite{companion} \cite{murray} \textcite{companion}
\printbibliography
```

Hier werden die Einträge im Literaturverzeichnis primär nach Autorennamen sortiert. Vor jedem Buchtitel steht das `clubsuit`-Symbol, danach immer ein `spadesuit`-Symbol. Bei den Zitaten im Text werden die Buchtitel mit Sternchen umschlossen. Die Titel der Artikel stehen in den Zitaten in eckigen Klammern. Gibt es mehrere Autoren werden sie mit einem Pluszeichen getrennt, außer vor dem letzten wo hier „und manchmal“ steht.

Eine komplette Liste der formatierbaren Attribute findet man in der Paketdokumentation.

14.4 Index

Die Indexerstellung bei LaTeX funktioniert analog zum Literaturverzeichnis. Es gibt eine manuelle Variante, die jedoch sehr aufwendig ist, und die automatische Indexerstellung über das Zusatzprogramm `makeindex`.

`\makeindex`	Nur einmal in der Präambel ausführbar.
`\index{`*Argument*`}`	

Das Makro `\makeindex` aktiviert die Indexerstellung und schreibt die zugehörigen Befehle in die `*.idx`-Datei. Das `\index`-Makro wird überall da positioniert, wohin die Einträge verweisen sollen.

Zum Ausdrucken und zur Formatierung des Index benutzt man üblicherweise das Standard-LaTeX 2_ε-Paket makeidx. Dies liefert das Makro

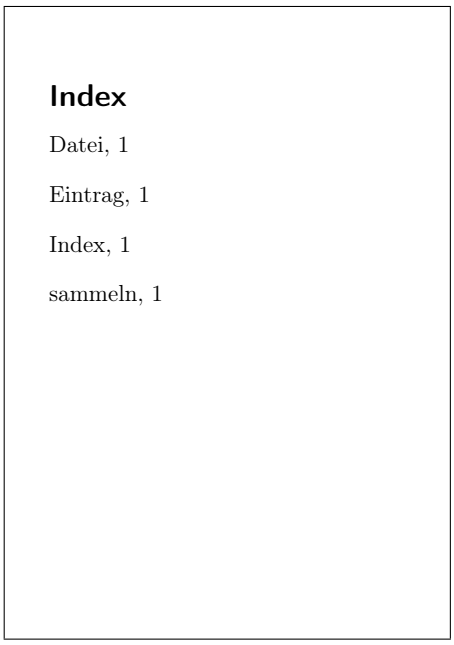
`\printindex`

und ermöglicht so eine einfache Ausgabe, siehe Beispiel 14. Analog zur Bibliografie muss hierbei zunächst einmal mit `pdflatex` (F6) kompiliert werden, anschließend lässt man `makeindex` (F12) laufen, bevor wiederum zwei `pdflatex`-Läufe nötig sind.

Für Querverweise im Index, z. B. „Index, siehe Verzeichnis“ benötigt das Paket makeidx interne Makros, die auch von babel entsprechend übersetzt werden. Zusätzlich können sie mit `\renewcommand` verändert werden.

`\seename`	Standard: „siehe“
`\alsoname`	Standard: „siehe auch“

Beispiel 14: Einfache Indexerstellung mit makeidx. In Anlehnung an [27, S. 10-7-02]



Index

Datei, 1

Eintrag, 1

Index, 1

sammeln, 1

```
...
\usepackage{makeidx}
\makeindex
...
\begin{document}
Ein Index\index{Index} wird erstellt.
Die Datei\index{Datei} sammelt die Einträge.
\index{Eintrag}\index{sammeln}
...
\printindex
...
\end{document}
```

Die Verwendung in den Indexeinträgen selbst hat eine etwas gewöhnungsbedürftige Syntax, siehe Tabelle 14.6 [vgl. 27, S. 512].

Tabelle 14.6: Syntaxerläuterungen für den Satz von Indexeinträgen mit `makeindex`.

Syntax	*Bedeutung*
`!`	Es folgt ein Unterverzeichniseintrag, der dem vorangehenden untergeordnet wird. Im Index wird er entsprechend eingerückt. Es sind maximal 2 Ebenen an Untereinträgen möglich.
`@`	Die Angabe vor `@` entspricht dem Sortierkriterium, der Eintrag dahinter stellt den tatsächlichen Indexeintrag dar.
`|`	Das Encap-Zeichen interpretiert den folgenden Namen als Makro, z. B. `|bfseries` oder `|see{`*Eintrag*`}`.
`|(`	Die Seitenzahlen des Indexeintrages werden bis zu einem Eintrag mit `|)` zusammengefasst.
`|)`	Gegenstück zu `|(`

Im Zusammenhang mit den `\index`-Makro stellen die folgenden Makros Beispiele für die Verwendung der Syntax da. Die Zahlenwerte neben den Makros beziehen sich auf die Nummerierung der Einträge in Beispiel 15.

`\index{`*Eintrag*`}`	Einfacher Indexeintrag, siehe 1)
`\index{`*Haupteintrag*`!`*Untereintrag*`}`	Untereintrag, siehe 9),10),. . .
`\index{`*Eintrag*`|(}`	Anfang eines „von–bis“-Verweises, siehe 2)
`\index{`*Eintrag*`|)}`	Ende des Verweises, siehe 7)
`\index{`*Sortiereintrag*`@`*Eintrag*`}`	unterschiedliche Sortierung, siehe 15)
`\index{`*Eintrag*`|see{`*Eintrag*`}}`	Verweis auf anderen Eintrag, siehe 5)
`\index{`*Eintrag*`|seealso{`*Eintrag*`}}`	

Beispiel 15: Die verschiedenen Indexmakros.

Index


```
\usepackage{makeidx}\makeindex

\printindex          \newpage
1) \index{Index}     \newpage
2) \index{Eintrag|(}\newpage
3) \index{idx-Datei@\texttt{idx}-Datei}
4) \index{Zirkus!Roncalli}   \newpage
5) \index{Manege|see{Zirkus}}\newpage
6) \index{Eintrag}\index{ABC-Schütze}
7) \index{Eintrag|)}
8) \index{Show|seealso{Zirkus}}
9) \index{Gemüse!Erbsen}     \newpage
10)\index{Gemüse!Bohnen}
11)\index{shutdown@\texttt{shutdown}}
12)\index{Wichtig@\textbf{Wichtig}}
13)\index{Mathematik}
14)\index{Clown!Pic}\newpage
15)\index{alpha@$\alpha$}
16)\index{$\beta$}  \newpage
17)\index{123}      \newpage
18)\index{\&}       \newpage
19)\index{Zirkus!Pic@\textsc{Pic}|see{Clown}}
```

15 Formeln und Einheiten

LaTeX wurde ursprünglich entwickelt, um unter anderem einen ordentlichen Formelsatz zu ermöglichen. Somit ist es selbstverständlich, dass es eine riesige Menge an Anpassungsmöglichkeiten für Formeln gilt.

15.1 Typografische Regeln

Variablen oder physikalische Größen werden durch einzelne lateinische oder griechische Buchstaben dargestellt. Nach internationalen Konventionen werden diese grundsätzlich kursiv gesetzt:

- Einfache Variablen x, y, z
- Mathematische Funktionen $z = f(x, y), \psi(t) = \int_t \psi \, \mathrm{d}t$
- Physikalische Konstanten ϵ_0, μ_0
- Indizes, die Variablen oder physikalischen Konstanten entsprechen $a_{i,j}, c_v$

Die folgenden Größen werden jedoch aufrecht gesetzt:

- Alle Ziffern $123, \alpha_{25}$
- Mathematische Operatoren $\mathbf{A}^T = \mathbf{B}$
- Mathematische Funktionen, die einem bestimmten Typ angehören $a = \arccos(x)$, $\Gamma(x)$
- Einheiten samt Vorsatz $\lambda = 0.4\,\mu\mathrm{m}$, $12\,\mathrm{kg}$
- Indizes die zur Kennzeichnung nach etwas benannt sind $x_{\max}, \mu_\mathrm{B}$
- Chemische Summenformeln $\mathrm{H_2O}$

Physikalische Größen bestehen immer aus Zahl und Einheit, wobei

- Zahl und Einheit werden mit `\,` voneinander getrennt: `15\,km` (15 km) oder `39\,\textcelsius` (39 °C)
- lediglich bei Winkelangaben in Grad entfällt dieser Abstand 18°21′4″
- Einheiten können ausgeklammert werden: $P = 50\,\mathrm{kW} \pm 4\,\mathrm{kW} = (50 \pm 4)\,\mathrm{kW}$

Außerdem ist zu beachten:

- Prozentangaben werden mit `\thinspace` (`\,`) gesetzt: `13\,\%` (13 %) oder 1,4 ‰
- Zahlenkolonnen können zur besseren Lesbarkeit von rechts an in Dreiergruppen unterteilt werden. Dies geschieht mit ebenfalls `\thinspace` (`\,`): 1 234 567 890 statt 1234567890
- Da LaTeX amerikanischen Standards unterliegt, ist ein Punkt als Dezimaltrennzeichen definiert. Schreibt man anstelle des Punktes einfach ein Komma stimmen jedoch die Abstände nicht mehr (1, 11 statt 1.11). Dies kann man korrigieren indem man das Komma einklammert 1,11 (`$1{,}11$`) oder Komma und Punkt in der Präambel entsprechend umdeklariert:

```
\DeclareMathSymbol{,}{\mathord}{letters}{"3B}
\DeclareMathSymbol{.}{\mathpunkt}{letter}{"3A}
```

- ▷ Der Buchstabe „d“ bei Differentialoperatoren und Integralen ist in deutschsprachigen Texten aufrecht zu setzen, zudem wird er bei Integralen generell durch einen Abstand (`\,`) vom Integranden getrennt:

$$\frac{\mathrm{d}}{\mathrm{d}x}f(x) \text{ anstatt } \frac{d}{dx}f(x); \qquad \int_0^y f(x)\,\mathrm{dx} \text{ anstatt } \int_0^y f(x)\mathrm{dx}$$

- ▷ Der „Doppelpunkt“ bei der Definition von Funktionen ist für richtige Abstände mit dem Makro `\colon` zu setzen: $f\colon D \to Z$ anstatt $f : D \to Z$
- ▷ Folgt auf den Doppelpunkt ein Gleichheitszeichen, so kann man das mit dem Symbol $\coloneqq$ (`\coloneqq`) aus dem mathttools-Paket setzen. Bei der Zeichenfolge `:=` ist der Doppelpunkt nicht ganz zur Achse des Gleichheitszeichens zentriert. Benutzt man das mathtools-Paket, kann man allerdings auch einfach den Doppelpunkt durch das Makro `\mathtoolsset{centercolon}` generell zentrieren lassen und die Zeichenfolde `:=` benutzen.

15.2 Schriftattribute

Standardmäßig geht LaTeX davon aus, dass es sich bei Buchstaben um Variablen handelt, sie werden somit automatisch kursiv gesetzt. Zahlen erscheinen jedoch aufrecht. Um die Darstellung der Schrift zu verändern gibt es folgende Befehle:

`\mathrm{`*Ausdruck*`}`	Roman ABCabc
`\mathtt{`*Ausdruck*`}`	Typewriter `ABCabc`
`\mathsf{`*Ausdruck*`}`	Sans Serif ABCabc
`\mathit{`*Ausdruck*`}`	kursiv *ABCabc*
`\mathcal{`*Ausdruck*`}`	Kaligraphie $\mathcal{ABC}$
`\mathnormal{`*Ausdruck*`}`	Standardschrift
`\mathbf{`*Ausdruck*`}`	fett außer griechische Kleinbuchstaben und Symbole
`\mathbb{`*Ausdruck*`}`	benötigt `amssymb`-Paket, Mengensymbole $\mathbb{N}, \mathbb{C}$
`\boldsymbol{`*Ausdruck*`}`	benötigt amsmath-Paket; fett einschließlich der griechischen Buchstaben und Symbole, allerdings beeinflusst es die Abstände
`\boldmath`	Schalterbefehl. Muss vor der Matheumgebung angewendet werden - (fast) alle Zeichen fett
`\unboldmath`	Ausschalten von `\boldmath`
`\bm{`*Ausdruck*`}`	benötigt das `bm`-Paket; innerhalb der Matheumgebung; schreibt mathmatische Ausdrücke fett (beeinflusst jedoch nicht wie boldsymbol die Abstände)

Zu den mathematischen Schriftattributen bleibt zu sagen, dass sie im Gegensatz zu den Attributen im Textmodus über keine Ligaturen verfügen.

15.3 Schriftstile

Neben den Schriftattributen kennt LaTeX vier mathematische Schriftstile, siehe Tabelle 15.1. Man nennt dies Befehle bewusst „styles“ und nicht „Größen“, da sie dynamischen Charakter haben, d.h. ihre wirkliche Größe ist abhängig von der Umgebung. Beispielsweise ist in Textformeln die Größe von Zähler und Nenner kleiner als wenn sie nicht auf bzw. unter dem Bruchstrich stünden, da dies für den Zeilenabstand günstiger ist. Den Unterschied zwischen Größe und Style sieht man auch an den Beispielen, da die Symbolgrößen nicht gleichstark skaliert werden.

Tabelle 15.1: Die verschiedenen mathematischen Schriftstile. `\displaystyle` ist Standard im abgesetzten Modus und `\textstyle` die Voreinstellung im Zeilenmodus.

Stil	*Zeile*
`\displaystyle`	$\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$
`\scriptstyle`	$\mathrm{FT}(f)=\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$
`\scriptscriptstyle`	$\mathrm{FT}(f)=\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$
`\textstyle`	$\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$

15.4 Mathematische Akzente

Die in Abschnitt 3.3.6 beschrieben Akzentbefehle sind für den Gebrauch in normalem Text bestimmt. Ihre Verwendung ist im Mathemodus nicht zulässig. Die Akzente für den Mathemodus sind in **??** gezeigt. Zu Demonstrationszwecken dient hier der Buchstabe „a“, sie können jedoch auf allen Buchstaben plaziert werden.

Tabelle 15.2: Mathematische Akzente

$\hat{a}$	`\hat{a}`	$\acute{a}$	`\acute{a}`	$\bar{a}$	`\bar{a}`	$\dot{a}$	`\dot{a}`
$\check{a}$	`\check{a}`	$\grave{a}$	`\grave{a}`	$\vec{a}$	`\vec{a}`	$\ddot{a}$	`\ddot{a}`
$\breve{a}$	`\breve{a}`	$\tilde{a}$	`\tilde{a}`				

Zusätzlich zu den einfachen Befehlen `\hat` und `\tilde` gibt es auch breitere Versionen, die über mehrere Buchstaben gesetzt werden können. So produziert zum Beispiel das Makro `\widehat{1-x}` die Zeichenkombination $\widehat{1-x}$. `\widetilde` funktioniert analog.

15.5 Zeilenmodus vs. abgesetzter Modus

Bei LaTeX wird zwischen zwei grundlegend verschiedenen Arten des Formelsatzes unterschieden: Dem mathematischen Zeilenmodus (inline math mode) und dem abgesetzten Modus.

Der Zeilenmodus wird dazu verwendet mathematische Elemente in laufende Zeilen, wie zum Beispiel $\mathrm{FT}(f) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}\,\mathrm{d}t$, zu setzen. Theoretisch gibt es keinerlei Beschränkung von Größe und Inhalt, allerdings werden die beiden äußeren Zeilen so weit auseinandergezogen, dass der Inhalt dazwischen passt. Dies kann dazu führen, dass ein sehr unschönes Layout entsteht. Ordnet man zum Beispiel eine Matrix in einer Zeile an lässt sich dies eigentlich nicht vermeiden: $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$

Allerdings gibt es Befehle die die Umgebungen kleiner erscheinen lassen und somit das Schriftbild weniger stören. Für den Fall der Matrix bietet sich die `smallmatrix`-Umgebung aus dem amsmath-Paket an $A = \left(\begin{smallmatrix} a & b & c \\ d & e & f \\ g & h & i \end{smallmatrix}\right)$. Jedoch empfiehlt es sich solche Formeln eher im abgesetzten Modus (Abschnitt 15.5.2) zu setzen um das Layout nicht zu stören.

15.5.1 Der Zeilenmodus

Der Zeilenmodus kann durch drei verschiedene Umgebungen aktiviert werden:

`\(`*Formelcode*`\)`	`\(` ist kein robustes Makro, d.h. es kann nicht innerhalb anderer Makros verwendet werden
`$`*Formelcode*`$`	Diese Sequenz ist zwar robust, kann jedoch in manchen (weniger häufigen) Umgebungen, wie `alltt` nicht verwendet werden
`\begin{math}`*Formelcode*`\end{math}`	entspricht der ersten Variante, ist somit ebenfalls nicht robust

Somit empfiehlt es sich `$...$` Anzuwenden. Für den Fall, dass man tatsächlich eine `alltt`-Umgebung verwenden will, kann man innerhalb auf `\(...\)` zurückgreifen.

15.5.2 Der abgesetzte Modus

Einzeilige Umgebungen:

`\[`*Formelcode*`\]`	nicht nummeriert
`\begin{displaymath}`*Formelcode*`\end{displaymath}`	wie `\[...\]`
`\begin{equation}`*Formelcode*`\end{equation}`	(rechtsbündig) nummeriert (siehe auch `leqno` in Tabelle 3.1)
`\begin{equation*}`*Formelcode*`\end{equation*}`	nur mit amsmath-Paket verfügbar; nicht nummeriert

Zusätzlich existiert noch eine für TEX gültige Makrokombination `$$...$$`*. Dies Syntax ist jedoch aus internen Gründen zu vermeiden!*

Mehrzeilige Umgebungen:
Standard-LaTeXleifert lediglich die `eqnarray`-Umgebung. Sie sollte jedoch aufgrund erheblicher Mängel nicht mehr verwendet werden. Mehrzeilige Umgebungen erfordern somit das amsmath-Paket.

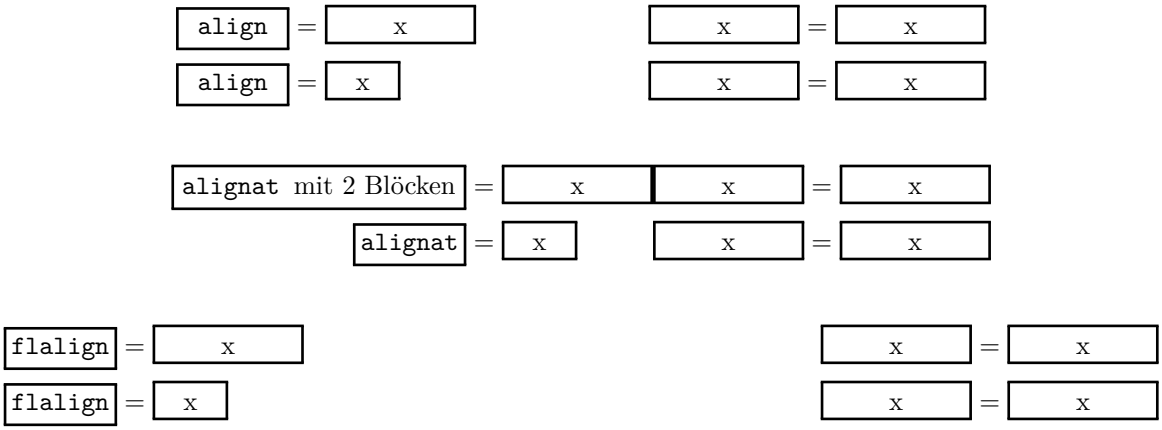
```
\begin{align}Formelcode\end{align}
\begin{align*}Formelcode\end{align*}
\begin{flalign}Formelcode\end{flalign}
\begin{flalign*}Formelcode\end{flalign*}
\begin{alignat}{Anzahl der Blöcke}Formelcode\end{alignat}
\begin{alignat*}{Anzahl der Blöcke}Formelcode\end{alignat*}
```

Es gibt somit drei verschiedene Arten von `align`-Umgebungen. Die Sternchenformen unterdrücken jeweils wie üblich die Nummerierung. Will man die Nummer lediglich für einzelne Zeilen unterdrücken, so setzt man vor den Zeilenumbruch den Befehl

```
\nonumber
```

er unterdrückt schlicht und einfach die Nummerierung der aktuellen Formelzeile.

Die Darstellung der `align`-Umgebungen entspricht folgendem Prinzip:



Der Code zum obigen Beispiel ohne die Rahmenboxen und Schriftattribute lautet:

```
\begin{align*}
   align&=x&x&=x\\
   align&=x&x&=x
\end{align*}

\begin{alignat*}{2}
   alignat\ mit\ 2\ Blöcken&=x&x&=x\\
   alignat&=x&x&=x
\end{alignat*}

\begin{flalign*}
   flalign&=x&x&=x
   flalign&=x&x&=x
\end{flalign*}
```

Das `&`-Zeichen sollte grundsätzlich *vor* dem Gleichheitszeichen, bzw. dem Symbol für eine Relation gesetzt werden. Somit erreicht man, dass die Ordnungssymbole alle direkt untereinander gesetzt werden. Prinzipiell haben alle `align`-Umgebungen eine Spaltenstruktur von Rechts-Links-Anordnungen (`{rlrlrl...}`), und unterscheiden sich lediglich durch die Positionierung (s.o.).

`\begin{aligned}`*Formelcode*`\end{aligned}`

Die `aligned`-Umgebung ermöglicht prinzipiell dasselbe wie die `align`-Umgebung. Allerdings kann sie geschachtelt werden und somit innerhalb einer beliebigen anderen Matheumgebung angewendet werden. Man kann mit ihrer Hilfe erreichen, dass zum Beispiel drei Gleichungen nur eine einzelne, gemeinsame Nummer erhalten, siehe Beispiel 16.

`\begin{gather}`*Formelcode*`\end{gather}`
`\begin{gather*}`*Formelcode*`\end{gather*}`
`\begin{gathered}`*Formelcode*`\end{gathered}`

Die `gather`-Umgebung zentriert Formeln einfach nur horizontal. Es existiert hierfür ebenfalls eine Sternchenform, die - wie üblich - die Nummerierung unterdrückt.

Beispiel 16: Die `aligned`-Umgebung. Auf die Struktur der Klammern sowie der Makros `\left` und `\right` wird in Abschnitt 15.12 genau eingegangen.

$$\left.\begin{aligned} 2x+3 &= 7 & 2x+5-5 &= 7-3 \\ 2x &= 4 & \frac{2x}{2} &= \frac{4}{2} \\ x &= 2 \end{aligned}\right\} \quad (5.1)$$

```
\begin{align}
\left.
\begin{aligned}
2x+3 &=7  &2x+5-5          &=7-3\\
2x     &=4  &\frac{2x}{2}  &=\frac42\\
x       &=2
\end{aligned}
\right\}
\end{align}
```

Die `gathered`-Umgebung hat die gleiche Verwendungsweise wie `aligned`, jedoch ohne Positionierung. Hier wird wie wie bei `gather` einfach nur zentriert.

`\begin{multline}`*Formelcode*`\end{multline}`

Die `multline`-Umgebung ist ebenfalls eine mehrzeilige Umgebung, jedoch wird sie insbesondere für sehr lange Gleichungen verwendet:

$$\begin{aligned} &\text{links} \\ &\qquad\text{zentriert} \\ &\qquad\text{zentriert} \\ &\qquad\text{zentriert} \\ &\qquad\qquad\text{rechts} \end{aligned} \quad (5.2)$$

Hierbei sind keine `&`-Zeichen nötig. Die erste Zeile wird automatisch linksbündig, die mittleren zentriert und die letzte Zeile rechtsbündig gesetzt.

Anmerkung: Um einfacher einen ordentlichen Formelsatz zu erreichen existiert der Befehl

`\phantom{`*Phantomtext*`}`

Er setzt eine Box in der Größe des „Phantomtextes“, damit kann man eine bessere Ausrichtung erreichen, z. B.:

$$|a| = \begin{cases} a; & a \geq 0 \\ -a; & a < 0 \end{cases}$$

anstatt

$$|a| = \begin{cases} a; & a \geq 0 \\ -a; & a < 0 \end{cases}$$

```
$
\vert{}a\vert =
\begin{cases}
    \phantom{-}a;&a\geq 0\\
    -a;&a<0
\end{cases}
$
```

15.6 Referenz und Bezug

Querverweise innerhalb von Formeln kann, wie auch im laufenden Text mit `\label` und `\ref` gesetzt werden. Zusätzlich liefert jedoch das amsmath-Paket den Befehl

`\eqref{`*Markername*`}`

dieser setzt Klammern (vgl. (1.1) statt vgl. 1.1) um die Zahl und kennzeichnet den Bezug somit unverwechselbar als Formel. Zusätzlich bietet sich noch die Möglichkeit die Beschriftung mit dem Befehl

`\tag{`*Beschriftung*`}`

manuell zu generieren. Zudem existiert eine Sternchenform `\tag*`, bei der die Klammern um die *Beschriftung* entfallen.

```
\begin{equation}
      a^2+b^2=c^2 \tag{Satz des Pythagoras}\label{pythagoras}
\end{equation}
Der \ref{pythagoras} ist einer der fundamentalen Sätze der euklidischen
Geometrie.
```

$$a^2 + b^2 = c^2 \qquad \text{(Satz des Pythagoras)}$$

Der Satz des Pythagoras ist einer der fundamentalen Sätze der euklidischen Geometrie.

15.7 Exponenten und Indizes

Hochstellen geschieht mittels `^` und Tiefstellen mit `_` innerhalb der Matheumgebungen. Auch mehrfache Indizes mit beliebiger Tiefe sind möglich. Sollen mehr als ein Zeichen hochgestellt werden, muss die Zeichenfolge in geschweifte Klammern gesetzt werden, sonst ist die Reihenfolge des Hoch- und Tiefstellens mehrdeutig definiert. Beispiele:

`x^2`	x^2	`a_n`	a_n	`x_i^n`	x_i^n
`x^{2n}`	x^{2n}	`x_{2y}`	x_{2y}	`A_{i,j,k}^{-n+2}`	$A_{i,j,k}^{-n+2}$
`x^{y^{z^2}}`	$x^{y^{z^2}}$	`x^{y_1}}`	x^{y_1}	`A^{x_i^2}_{j^{2n}_{n,m}}`	$A^{x_i^2}_{j^{2n}_{n,m}}$

Hat man in einem Ausdruck verschiedene Indizes und Exponenten, so muss bei den Variablen ohne Exponent ein leerer Exponent gesetzt werden, damit alle Indizes sich auf gleicher Höhe befinden: $a_l^2 b_m c_n^2$ (`$a_l^2 b_m^{} c_n^2$`) statt $a_l^2 b_m c_n^2$

Zur Darstellung von Isotopen werden in manchen Schreibweisen auf Indizes auf der Linken Seite benötigt. Dies ist jedoch ohne Zusatzpaket nur sehr kompliziert möglich. Am besten eignet sich hierfür dann das Paket mhchem*, siehe Abschnitt 15.18*

15.8 Brüche und Binomialkoeffizienten

Für Brüche wird folgender Befehl genutzt.

`\frac{`*Zähler*`}{`*Nenner*`}`

Der Befehl setzt den Bruch und erzeugt einen Bruchstrich der Länge des Zählers oder Nenners, je nachdem, welcher breiter ist. Beispiele:

```
\frac{1}{x+y}
```
$\frac{1}{x+y}$

```
\frac{a^2+b^2}{a+b}
```
$\frac{a^2+b^2}{a+b}$

```
\frac{\frac{a}{x-y}+\frac{b}{x+y}}
{1+\frac{a-b}{a+\frac{b}{a^2}}}
```
$\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$

Das amsmath-Paket liefert das Makro `\genfrac` („generalized fraction“). Es benötigt sechs Parameter, die teilweise leer bleiben können, allerdings sind sie *nicht* optional, und dürfen nicht wegfallen.

`\genfrac{`*Links*`}{`*Rechts*`}{`*Liniendicke*`}{`*Mathe-Stil*`}{`*Divident*`}{`*Divisor*`}`

Links/*Rechts* Linkes bzw. rechtes Begrenzer-Symbol. Bei einem Bruch ist dieses Argument leer, bei einem Binomialkoeffizienten setzt man hier Runde Klammern.

Liniendicke Dicke des Bruchstriches. Ein leerer Parameter liefert hier die Standardstärke. Für einen Binomialkoeffizienten *muss* dieser Parameter auf `0pt` gesetzt werden.

Mathe-Stil Die mathematischen Schriftstile werden hier durch eine Ziffer kodiert ausgewählt.

0 `\displaystyle`

1 `\textstyle`

2 `\scriptstyle`

3 `\scriptscriptstyle`

Bleibt dieses Feld leer, erzwingt das eine Beachtung des aktuelle mathematischen Stils. Bei `\scriptstyle` wird ebenfalls `scriptstyle` gewählt.

Dividend Der Zähler des Bruches bzw. der obere Teil des Binomialkoeffizienten.

Divisor Nenner des Bruches bzw. der untere Teil des Binomialkoeffizienten.

amsmath definiert intern noch ein paar Kurzformen für die alltägliche Anwendung von `\genfrac`:

`\binom{`*oberer Teil*`}{`*unterer Teil*`}`

Setzt einfache Binomialkoeffizienten.

`\dfrac{`*Zählrt*`}{`*Nenner*`}`

Dieser Befehl setet Brüche automatisch im Displaystyle (Abschnitt 15.3) setzt. Im normalen Textsatz werden die Formeln dann größer dargestellt, was eine bessere Lesbarkeit mit sich bringt, jedoch die Zeilenabstände unschön beeinflusst. Jedoch bietet es eine Möglichkeit Doppelbrüche auch im Zeilenmodus lesbar darzustellen, wenn es denn sein muss.

Beispiel:

einfacher Zeilenmodus $\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$

äußerster Bruch im `\displaystyle` $\displaystyle\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$

gesamter Bruch im `\displaystyle`

$$\frac{\frac{a}{x-y}+\frac{b}{x+y}}{1+\frac{a-b}{a+\frac{b}{a^2}}}$$

15.9 Wurzeln

Wurzelausdrücke werden erzeugt mit

`\sqrt[`*Ordnung*`]{`*Formelcode*`}`

Beispiele:

`\sqrt{a}` $\sqrt{a}$

`\sqrt[3]{8}` $\sqrt[3]{8}$

```
\sqrt[n+3]{\frac{-
q+sqrt{q+p}}{1+q^3}}
```

$\sqrt[n+3]{\frac{-q+\sqrt{q+p}}{1+q^3}}$

15.10 Operatoren

Operatoren können sowohl durch einzelne Zeichen, als auch Namen ausgedrückt werden. Im allgemeinen ist die Wahl der Schreibweise willkürlich, jedoch werden grundsätzlich alle Operatoren aufrecht gesetzt. Zudem werden andere Abstände sowohl vor als auch nach dem Ausdruck gesetzt. LaTeX hat bereits einige Standardbefehle für die häufigsten Operatoren definiert. Dazu gehören unter anderem die Befehle für Summen, Produkte, Limites und Integrale:

```
\sum_{Laufindes=Startwert}^{Endwert}
\prod_{Laufindes=Startwert}^{Endwert}
\lim_{Variable \to Grenzwert}
\int_{untere Grenze}^{obere Grenze}
```

Für eine vollständige Auflistung der Standardmäßigen Symbol-Operatoren, siehe Tabelle 15.3. Diese

Tabelle 15.3: In Standard-LaTeX verfügbare Symbol-Operatoren mit `limits`

`\int`	$\int$	`\sum`	$\sum$	`\prod`	$\prod$	`\bigvee`	$\bigvee$	`\smallint`	$\smallint$
`\bigwedge`	$\bigwedge$	`\biguplus`	$\biguplus$	`\bigcap`	$\bigcap$	`\bigcup`	$\bigcup$	`\bigotimes`	$\bigotimes$
`\bigoplus`	$\bigoplus$	`\bigodot`	$\bigodot$	`\oint`	$\oint$	`\bigsqcup`	$\bigsqcup$		

Symbol-Operatoren erscheinen in drei Größen, je nachdem in welcher Schriftstil aktiv ist (vgl. Abschnitt 15.3). Mit `\limits` bzw. `\nolimits` kann zwischen den Positionen der Grenzen bzw. Indizes hin und hergeschaltet werden.

Beispiele:

$2\sum_{i=1}^{n} a_i \int_a^b f_i(x)\,dx$ abgesetzte Formel

$2 \sum_{i=1}^{n} a_i \int_a^b f_i(x)\,dx$ Textformel

$\sum_{i=1}^{n} a_i$ `\sum_{i=1}^n a_i`

$\sum\nolimits_{i=1}^{n} a_i$ `\sum\nolimits_{i=1}^na_i`

$\sum_{\substack{i=1\\j=1\\k=1}}^{\substack{\infty\\n\\m}}$ `\sum_{\substack{i=1\\j=1\\k=1}}^{\substack{\infty\\n\\m}}`

Bemerkung: Der Befehl `\substack` stammt dabei aus dem amsmath-Paket und erlaubt innerhalb seines Argumentes Zeilenumbrüche in Indizes.

`\substack{`*Code*`\\`*Code*`\\...}`

Zusätzlich zu den Symboloperatoren gibt es noch eine Reihe Operatornamen, z. B. die trigonometrischen Funktionen, siehe Tabelle 15.4. Viele von Ihnen, wie zum Beispiel `\sin` können das `\limits`-Makro nicht verarbeiten. Bei Ihnen werden immer Indizes und niemals Grenzen gesetzt.

Tabelle 15.4: Weitere in Standard-LaTeX definierte Operatoren, viele davon ohne `\limits`

`\log`	log	`\lg`	lg	`\ln`	ln	`\lim`	lim	`\limsup`	lim sup
`\liminf`	lim inf	`\sin`	sin	`\arcsin`	arcsin	`\sinh`	sinh	`\cos`	cos
`\arccos`	arccos	`\cosh`	cosh	`\tan`	tan	`\arctan`	arctan	`\tanh`	tanh
`\cot`	cot	`\coth`	coth	`\sec`	sec	`\csc`	csc	`\max`	max
`\min`	min	`\sup`	sup	`\inf`	inf	`\arg`	arg	`\ker`	ker
`\dim`	dim	`\hom`	hom	`\det`	det	`\bmod`	mod	`\Pr`	Pr
`\gcd`	gcd	`\deg`	deg	`\exp`	exp				

Eigene Operatoren definieren

Die Definition eigener Operatoren ist zwar auch mit Standard-LaTeX relativ einfach, jedoch wird sie durch das Paket amsopn von Michael Downes, welches automatisch mit amsmath geladen wird, noch weiter vereinfacht.

`\DeclareMathOperator*{`*Makro*`}{`*Operatorname*`}`
`\DeclareMathOperator{`*Makro*`}{`*Operatorname*`}`

Kann `\limits` benutzen.

Beide Makros dürfen lediglich in der Präambel verwendet werden. Ihre Verwendung sieht dann z. B. folgendermaßen aus:

$$\mathrm{foo}_1^2 = \mathrm{baz}_1^2 = \mathrm{foo}_1^2 = \overset{2}{\underset{1}{\mathrm{baz}}}$$

erzeugt mit

```
\DeclareMathOperator{\foo}{foo} %in der Präambel
\DeclareMathOperator*{\baz}{baz} %in der Präambel

\[\foo_1^2 = \baz\nolimits_1^2 = \foo\limits_1^2 = \baz_1^2\]
```

Möchte man einen bestimmten Operator lediglich in einem Einzelfall verwenden, so liefert ams-math zwei Makros zum Satz von Operatornamen. Die Unterscheidung ist hierbei dieselbe, wie bei `\DeclareMathOperator`:

`\operatorname{`*Operatorname*`}`
`\operatornamewithlimits{`*Operatorname*`}`

15.11 Spezielle Zeichen

Griechische und hebräische Zeichen können oft direkt entsprechend ihrem Wortlaut eingegeben werden, Kaligraphische mit `\mathcal{...}` und Frakturschrift mit `\mathfrak{...}`

`\alpha`	α	`\beta`	β	`\pi`	π
`\Pi`	Π	`\omega`	ω	`\theta`	θ
`\aleph`	$\aleph$	`\beth`	$\beth$	`\gimel`	$\gimel$
`\mathcal{A}`	$\mathcal{A}$				

Manche spezielle Zeichen und Strukturen seien hier noch erwähnt:

`\nabla`	∇	`\triangle`	$\triangle$
`\square`	$\square$	`\hbar`	$\hbar$
`\infty`	∞	`\oint`	$\oint$
`\vec{a}`	$\vec{a}$	`\dot{a}\,\ddot{a}`	$\dot{a}\,\ddot{a}$
`f''''`	f''''	`\stackrel{a\to b}{=}`	$\stackrel{a\to b}{=}$
`\mathbb{C}`	$\mathbb{C}$	`\pm\times`	$\pm\times$
`\equiv`	$\equiv$	`\sim`	$\sim$
`\approx`	$\approx$	`\Re\Im`	$\Re\Im$
`a \atop b`	$a \atop b$	`\cong`	$\cong$

Die riesige Anzahl an mathematischen Symbolen, macht es leider unmöglich alle hier aufzulisten. Viele Editoren bieten jedoch intern Symbolleisten oder Listen, die das fertige Symbol enthalten und bei Aktivierung den zugehörigen Quellcode an der Cursorposition schreiben. Um weitere vor allem auch unbekannte Symbole schnell finden zu können bietet sich einerseits das Tool „Detexify[2]“ (`http://detexify.kirelabs.org`) und zum anderen die „The Comprehensive LaTeX Symbol list“ (`http://www.tex.ac.uk/tex-archive/info/symbols/comprehensive/symbols-a4.pdf`).

15.12 Klammern

Die Größe mathematischer Klammern[1],

`()`	$()$	`[]`	$[]$
`\{\}`	$\{\}$	`\langle\rangle`	$\langle\ \rangle$
`/|\backslash`	$/\ \vert\ \backslash$	`\big\lmoustache\big\rmoustache`	$\big\lmoustache\ \big\rmoustache$
`\arrowvert\vert`	$\arrowvert\ \vert$	`\Vert\Arrowvert`	$\Vert\ \Arrowvert$
`\uparrow\downarrow\updownarrow`	$\uparrow\ \downarrow\ \updownarrow$	`\Uparrow\Downarrow\Updownarrow`	$\Uparrow\ \Downarrow\ \Updownarrow$
`\lceil\rceil`	$\lceil\ \rceil$	`\lfloor\rfloor`	$\lfloor\ \rfloor$
`\big\lgroup\big\rgroup`	$\big\lgroup\ \big\rgroup$	`\big\bracevert`	$\big\bracevert$

kann entweder manuell z.B. mit

`\big(\Big(\bigg(\Bigg(` $\big(\Big(\bigg(\Bigg($

oder automatisch an die Größe der nachfolgenden oder vorstehenden Zeichen angepasst werden

`\left( ... \right)`

Hierbei ist zu beachten, dass für die Festlegung der Größe diese Makros immer als Paar benötigt werden, sonst erhält man eine Fehlermeldung.

Zwei kleine Beispiele:

```
\left(\begin{array}{c} 9 \\ 2 \\ 4 \end{array}\right)
```

$$\left(\begin{array}{c} 9 \\ 2 \\ 4 \end{array}\right)$$

```
2\pi\hbar\left\{a+\frac{\frac{3}{2}
     \sum x_i^2}{m^*}\right\}
```

$$2\pi\hbar\left\{a+\frac{\frac{3}{2}\sum x_i^2}{m^*}\right\}$$

Die Darstellung der Klammer kann mit `\left.` oder`\right.` unterdrückt werden, z.B.

```
\[
y=\left\{
\begin{array}{r@{\quad:\quad}l}
-1 & x<0 \\
0 & x=0 \\
1 & x>0
\end{array}
\right.
\]
```

$$y=\left\{\begin{array}{r@{\quad:\quad}l} -1 & x<0 \\ 0 & x=0 \\ 1 & x>0 \end{array}\right.$$

Diese Struktur hätte man mit dem amsmath-Paket auch direkt erzeugen können mit:

[1] Die Klammernbefehle, welchen bereits ein `\big` vorangestellt wurde, existieren nicht in der kleinsten Größe.

```
\[
y=
\begin{cases}
-1 & : \quad x<0\\
0 & : \quad x=0\\
1 & : \quad x>0
\end{cases}
\]
```

$$y=\begin{cases}-1 & : \quad x<0\\ 0 & : \quad x=0\\ 1 & : \quad x>0\end{cases}$$

\overbrace und \underbrace

Neben den normalen vertikalen Klammern, existieren auch horizontale geschweifte Klammern um damit Formelsegmente beschriften zu können:

```
\overbrace{Formelsegment}^{Beschriftung}
\underbrace{Formelsegment}_{Beschriftung}
```

$$\overbrace{Formelsegment}^{\text{mit overbrace}} \qquad \underbrace{Formelsegment}_{\text{mit underbrace}}$$

15.13 Matrizen

Matrizen erzeugt man beispielsweise mit einer array-Umgebung, also beispielsweise mit

```
\[
  \left(\begin{array}{cccc}
      a_{11} & a_{12} & \cdots & a_{1n} \\
      \vdots & \vdots & \ddots & \vdots \\
      a_{n1} & a_{n2} & \cdots & a_{nn}
    \end{array}\right)
\]
```

$$\left(\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{array}\right)$$

Auch hierfür gibt es extra Matrix-Befehle aus dem amsmath-Paket. Das obige Beispiel sähe dann so aus:

```
\[
\begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix}
\]
```

Das amsmath-Paket hat noch weitere Befehle für tabellenartige Strukturen. Alle diese Befehle haben die gleiche Syntax wie `pmatrix`.

$$\texttt{Vmatrix}\begin{Vmatrix}a & b\\ c & d\end{Vmatrix} \quad \texttt{Bmatrix}\begin{Bmatrix}a & b\\ c & d\end{Bmatrix} \quad \texttt{matrix}\begin{matrix}a & b\\ c & d\end{matrix}$$

$$\texttt{vmatrix}\begin{vmatrix}a & b\\ c & d\end{vmatrix} \quad \texttt{bmatrix}\begin{bmatrix}a & b\\ c & d\end{bmatrix} \quad \texttt{pmatrix}\begin{pmatrix}a & b\\ c & d\end{pmatrix}$$

Für die Matrizendarstellung erweisen sich auch Fortsetzungspunkte als sehr nützlich:

`\ldots`	$\ldots$	`\cdots`	$\cdots$
`\vdots`	$\vdots$	`\ddots`	$\ddots$

15.14 Overset und Underset

`\overset{`*Ausdruck drüber*`}{`*Formeltext*`}`
`\underset{`*Ausdruck drunter*`}{`*Formeltext*`}`

amsmath bietet mit diesen Befehlen die Möglichkeit Objekte direkt über oder unter Formelfragmenten anzuordnen.

$$\overset{\textit{Ausdruck drüber}}{\textit{Formeltext}} \qquad \underset{\textit{Ausdruck drunter}}{\textit{Formeltext}}$$

Zum Beispiel wird dieser Befehl benutzt um Beschriftungen zu erzeugen oder chemische Formeln mit Oxidationszahlen zu versehen:

$$\overset{+\mathrm{VII}}{\mathrm{MnO_4}}^- + 8\,\mathrm{H_3O^+} + 5\,\underset{\substack{\uparrow\\ \text{freies Elektron}}}{\mathrm{e}^-} \longrightarrow \overset{+\mathrm{II}}{\mathrm{Mn}}^{2+} + 12\,\mathrm{H_2O}$$

Zusätzlich existiert das Makro

`\sideset{_{`*links unten*`}^{`*links oben*`}}{_{`*rechts unten*`}^{`*rechts oben*`}}{`*Operator*`}`

um Indizes auch auch linksseitig setzen zu können.

$$\sideset{_{\text{links unten}}^{\text{links oben}}}{_{\text{rechts unten}}^{\text{rechts oben}}}{\prod}_{\text{unten}}^{\text{oben}}$$

Dieser Befehl funktioniert jedoch lediglich bei Operatoren, wie z. B. $\prod$,$\sum$,...; Siehe auch Abschnitt 15.10.

15.15 Text im Mathemodus

Mit dem amsmath-Paket gibt es den Befehl

`\text{`*Text*`}`

Hier wird der Text entsprechend des mathematischen Schriftstiles gewählt und als richtige Textbox formatiert. Das Makro

`\mathrm{`*Text*`}`

setzt dahingegen keine Textbox, sondern lediglich Buchstaben aufrecht. Leerzeichen innerhalb des Argumentes werden (wie auch sonst im Mathe-Modus) ignoriert. Zudem enthält dieser Schrifttyp als Mathematikschrift keine Ligaturen. Das Makro `\mathrm` sollte somit nicht zum Satz von Worten, sondern lediglich zum Satz einzelner Zeichenfolgen dienen.

Soll Text zwischen die Zeilen einer mehrzeiligen Formel eingefügt werden, so geschieht dies mittels

`\intertext{`*Text*`}`

Der Text wird dann so gesetzt, als würde die Matheumgebung beendet und anschließend neu begonnen. Dies ermöglicht jedoch die Ausrichtung fortzusetzen, obwohl ein Text eingefügt wird.

Andere Möglichkeiten zum Setzen von Text im Mathemodus existieren zwar, sollten jedoch aufgrund der verminderten Anpassungsfähigkeiten im Gegensatz zu `\text` vermieden werden.

15.16 Theoreme – Eigene Strukturen

Der Autor kann auch eigene Strukturen bzw. Umgebungen mit eigenen Zähler deklarieren. So werden z.B. in den Naturwissenschaften manche Dokumente von einer Reihe von Axiomen oder Definitionen durchzogen. Der Autor kann somit die Formatierung für diese Strukturen einheitlich festlegen und damit leicht umsetzen. Dies geschieht mit

`\newtheorem{`*Strukturname*`}[`*Zählung*`]{`*Strukturbegriff*`}[`*Gliederung*`]`

Der *Strukturbegriff* (z.B. Definition) erscheint in Fettdruck gefolgt vom laufenden Zähler. Soll die neue Umgebung die gleiche Zählung verwenden wie eine andere Theoremumgebung so verwendet man *Zählung*. *Gliederung* ergänzt den Zähler um einen Gliederungszähler, wie z.B `section`.

Das amsthm-Paket ermöglicht es, den Theorem-Stil mit

`\theoremstyle{`*stylename*`}`

zu ändern. Es gibt drei vordefinierte Stile, siehe Tabelle 15.5

Tabelle 15.5: Von amsthm vordefinierte Theorem-Stile

stylename	*Beschreibung*	*Optik*
`plain`	Standard. Für Theoreme, Lemmata, Propositionen, usw.	**Theorem 1.** *Text*
`definition`	Für Definitionen und Beispiele	**Definition 2.** Text
`remark`	Für Bemerkungen	*Bemerkung 3.* Text

Beispiel:

> **Definition 15.16.1.** Fermionen (benannt nach Enrico Fermi) sind Teilchen, die einen halbzahligen Spin besitzen bzw. einen Drehimpuls von Vielfachen von $\hbar/2$.

erzeugt mit

```
\theoremstyle{definition}
\newtheorem{Def}{Definition}[section]
\begin{Def}
...
\end{Def}
```

Beweise

Zusätzlich definiert amsthm die proof-Umgebung für Beweise.

```
\begin{proof}[Beweistitel] ... \end{proof}
```

Beweisname. Hier steht ein Beweis. □

erzeugt mit

```
\begin{proof}[Beweisname]
Hier steht ein Beweis.
\end{proof}
```

15.17 Zahlen mit Einheiten – Das siunitx-Paket

Beim Tippen von Einheiten müssen mehrere Aspekte beachtet werden: Zum Beispiel werden Einheiten nicht kursiv geschrieben. Da sie aber oft in einer Matheumgebung gebraucht werden, bräuchte man zum Aufrichten extra einen Befehl (wie etwa `\mathrm`, siehe Abschnitt 15.2). Weiterhin soll bei einem Zeilenumbruch die Zahl nicht von der Einheit getrennt werden und der Abstand zwischen Zahl und Einheit kleiner als ein Leerzeichen sein (was man beides mit `\,` beheben könnte, siehe Abschnitt 4.3.2). Das Paket siunitx dient dazu Zahlen und Einheiten richtig zu formatieren, sodass deren Zusammengehörigkeit klar ersichtlich wird. Hier werden nur die wichtigsten Befehle vorgestellt, für eine detailliertere Beschreibung sei auf die Paketdokumentation verwiesen [29].

```
\usepackage[globale Optionen]{siunitx}
\sisetup{lokale Optionen}
```

Eine Auswahl der wichtigsten Optionen findet sich in Tabelle 15.6.

Tabelle 15.6: Wichtigste Optionen zu siunitx

Option=Wert	*Beispiel*	*Beschreibung*
`output-decimal-marker={,}`	1,23	`{,}` als Dezimaltrennzeichen anstatt des Punktes
`exponent-product=\cdot`	$5 \cdot 10^{20}$	Malpunkt anstatt des Kreuzes
`per-mode=fraction`	$\frac{\mathrm{m}}{\mathrm{s}}$	Echter Bruch anstatt negativer Potenzen
`per-mode=symbol`	m/s	Schrägstrich als Bruch anstatt negativer Potenzen

Zahlen:

```
\num[Optionen]{Zahl}
```

12 345	`\num{12345}`	lange Zahlen bekommen extra Zwischenräume
0.1234	`\num{0,1234}`	das Dezimalzeichen kann ein Komma oder ein Punkt sein
0.123 45	`\num{.12345}`	die führende Null wird ergänzt
3.45×10^{-4}	`\num{3.45e-4}`	Zehnerpotenzen bekommt man mit `e`, `E`, `d` und `D`

Winkelangaben:

`\ang[`*Optionen*`]{`*Grad*`;`*Minuten*`;`*Sekunden*`}`

$12.3°$	`\ang{12.3}`	das Dezimalzeichen kann ein Komma oder ein Punkt sein
$1°2′3″$	`\ang{1;2;3}`	auch die Schreibweise mit Minuten und Sekunden ist möglich
$-1″$	`\ang{;;-1}`	wenn man nur Sekunden anzugeben hat

Einheiten

`\si[`*Optionen*`]{`*Einheiten*`}`

$\mathrm{kg\,m/s^2}$	`\si{\kg.\m/\s^2}`
$\mathrm{g_{polymer}\,mol_{cat}\,s^{-1}}$	`\si{g_{polymer}~mol_{cat}.s^{-1}}`
$\mathrm{kg_{metal}}$	`\si{\kilogram\of{metal}}`

Man kann die Einheiten nicht nur auf direktem Weg eintippen, sondern auch wie man sie auf englisch spricht. Beachte auch den Unterschied zwischen `square` und `squared` sowie zwischen `tothe` und `raiseto`. Es wird nirgends eine Matheumgebung benötigt, jedoch ist es selbstverständlich möglich die folgenden Ausdrücke auch innerhalb einer Matheumgebung zu setzen.

$\mathrm{kg\,m\,s^{-2}}$	`\si{\kilo\gram\metre\per\square\second}`
$\mathrm{g\,cm^{-3}}$	`\si{\gram\per\cubic\centi\metre}`
$\mathrm{V^2\,lm^3\,F^{-1}}$	`\si{\square\volt\cubic\lumen\per\farad}`
$\mathrm{m^2\,Gy^{-1}\,lx^3}$	`\si{\metre\squared\per\gray\cubic\lux}`
$\mathrm{H\,s}$	`\si{\henry\second}`
$\mathrm{H^5}$	`\si{\henry\tothe{5}}`
$\mathrm{H^{-5}}$	`\si{\per\henry\tothe{5}}`
$\mathrm{rad^{4.5}}$	`\si{\raiseto{4.5}\radian}`

Werte mit Einheiten

`\SI[`*Optionen*`]{`*Wert*`}[`*Vor-Einheit*`]{`*Einheit*`}`

$1.23\,\mathrm{J\,mol^{-1}\,K^{-1}}$	`\SI[mode=text]{1.23}{J.mol^{-1}.K^{-1}}`
$0.23 \times 10^7\,\mathrm{cd}$	`\SI{.23e7}{\candela}`
$\$1.99/\mathrm{kg}$	`\SI[per-mode=symbol]{1,99}[\$]{\per\kilogram}`
$1.345\,\frac{\mathrm{C}}{\mathrm{mol}}$	`\SI[per-mode=fraction]{1,345}{\coulomb\per\mole}`

15.18 Chemische Formeln – Das mhchem-Paket

Das mhchem-Paket vereinfacht die nötige Befehlsstruktur um chemische Formeln zu setzen. Zudem ermöglicht es Indizes auf der linken Seite hinzuzufügen, was sich beim Setzen von Isotopen als besonders praktisch erweist.

`\ce{`*Summenformel*`}`

Dies ist der hauptsächliche Befehl. Im folgenden finden sich einige Möglichkeiten der Anwendung.

Summenformeln Einfache chemische Summenformeln werden durch Eingabe der enthaltenen Zeichen direkt hintereinander erzeugt.

Einige Beispiele:
H_2O (`\ce{H2O}`), Sb_{203} (`\ce{Sb203}`), H^+ (`\ce{H+}`), CrO_4^{2-} (`\ce{CrO4^2-}`), $[AgCl_2]^-$ (`\ce{[AgCl2]-}`), Y^{99+} (`\ce{Y^{99}+}` oder `\ce{Y^{99+}}`)

Mengen Mengen werden direkt vor der restlichen Formel geschrieben:
$2\,H_2O$ (`\ce{2H2O}`), $\frac{1}{2}\,H_2O$ (`\ce{1/2H2O}`)

Isotope Indizes auf der linken Seite werden wie normale Indizes, jedoch vor dem Elementsymbol:
${}^{227}_{90}Th^+$ (`\ce{^{227}_{90}Th+}`)

Bindungen A−B=C≡ (`\ce{A\sbond B\dbond C\tbond}`)

Formeln Beispiele für verschiedene Arten von Reaktionspfeilen (auch mit Super- und Subscript):

$CO_2 + C \longrightarrow 2\,CO$ (`\ce{CO2 + C -> 2CO}`)
$CO_2 + C \longleftarrow 2\,CO$ (`\ce{CO2 + C <- 2CO}`)
$CO_2 + C \rightleftharpoons 2\,CO$ (`\ce{CO2 + C <=> 2CO}`)
$H^+ + OH^- \rightleftharpoons H_2O$ (`\ce{H+ + OH- <=>> H2O}`)
$A \longleftrightarrow A'$ (`\ce{$A$ <-> $A'$}`)
$CO_2 + C \xrightarrow{\alpha} 2\,CO$ (`\ce{CO2 + C ->[\alpha] 2CO}`)
$CO_2 + C \xrightarrow[\beta]{\alpha} 2\,CO$ (`\ce{CO2 + C ->[\alpha][\beta] 2CO}`)

Im Mathemodus werden sämtliche Elementsymbole immer aufrecht gesetzt. Im Textmodus wird die aktuelle Schriftart auch für die Formeln übernommen.

Das Paket mhchem bietet noch mehr Möglichkeiten zur Formatierung chemischer Formeln. Alle Befehle findet man selbstverständlich in der Dokumentation.

Literaturverzeichnis

[1] Hendri Adriaens und Christopher Ellison. *The powerdot class*. URL: ftp://ftp.rrzn.uni-hannover.de/pub/mirror/tex-archive/macros/latex/contrib/powerdot/docs/powerdot.pdf.

[2] David Carlisle. *The tabularx package*. URL: http://mirror.physik-pool.tu-berlin.de/tex-archive/macros/latex/required/tools/tabularx.pdf.

[3] David Carlisle. *The longtable package*. URL: http://truefunny.com/m/ctan/macros/latex/required/tools/longtable.pdf.

[4] David Carlisle. *The tabulary package*. URL: ftp://ftp.fu-berlin.de/tex/CTAN/macros/latex/contrib/tabulary/tabulary.pdf.

[5] Florian Rödl. *Einführung in LaTeX 2ε*. 2012.

[6] Hideo Umeki. *The geometry Package*. URL: http://sunsite.informatik.rwth-aachen.de/ftp/pub/mirror/ctan/macros/latex/contrib/geometry/geometry.pdf.

[7] Javier Bezos. *Customizing lists with the enumitem package*. URL: http://mirror.informatik.uni-mannheim.de/pub/mirrors/tex-archive/macros/latex/contrib/enumitem/enumitem.pdf.

[8] Donald Ervin Knuth. *The TeXbook*. [Repr.] Bd. A. Computers and typesetting. Reading, Mass: Addison Wesley und Addison-Wesley, 2006. 483 // ix, 483. ISBN: 9780201134483.

[9] Markus Kohm. *KOMA-Script. Eine Sammlung von Klassen und Paketen für LATEX 2e; Anleitung zu Version 3.13*. 5., überarb. und erw. Aufl. für Koma-Script 3. Berlin: Lehmanns, 2014. 678 S. ISBN: 3865416136.

[10] Markus Kohm und Jens-Uwe Morawski. *KOMA-Script. Eine Sammlung von Klassen und Paketen für LaTeX 2ε*. 4., überarb. und erw. Auflage für KOMA-Script 3. Lehmann, 2012. ISBN: 978-3-86541-459-5.

[11] Helmut Kopka. *Einführung*. 1. Aufl., [unveränd. Nachdr.] Bd. 1. LaTeX. Bonn [u.a.]: Addison-Wesley, 1994. XIX, 428 S. ISBN: 3-89319-664-1.

[12] Helmut Kopka und Patrick W. Daly. *Guide to LaTeX. tools and techniques for computer typesetting*. 4. ed., 5. print. Addison-Wesley series on tools and techniques for computer typesetting. Boston, Mass. [u.a.]: Addison-Wesley, 2005. XII, 597 S. ISBN: 0-321-17385-6.

[13] Leslie Lamport. *Das LaTeX-Handbuch*. Bonn [u.a.]: Addison-Wesley, 2002. XVIII, 325 S. ISBN: 9783893198269.

[14] Martin Schröder. *The ragged2e-package*. URL: http://truefunny.com/m/ctan/macros/latex/contrib/ms/ragged2e.pdf.

[15] Frank Mittelbach und Michel Goossens. *Der LaTeX-Begleiter*. 2., überarb. und erw. Aufl., korr. Nachdr. München [u.a.]: Pearson Studium, 2007. XXIX, 1137 S. ISBN: 386894088X.

[16] Frank Mittelbach und Michel Goossens. *The LaTeX companion*. 2nd ed. Boston [etc.]: Addison-Wesley, op. 2004. XXVII, 1090. ISBN: 978-0-201-36299-2.

[17] Marion Neubauer. „Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil I“. In: *Die TeXnische Komödie* 1996 (4 1996), S. 23–40.

[18] Marion Neubauer. „Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil II“. In: *Die TEXnische Komödie* 1997 (1 1997), S. 25–44.

[19] Philipp Lehman. *The biblatex Package. Programmable Bibliographies and Citations.* URL: `http://ftp.fau.de/ctan/macros/latex/contrib/biblatex/doc/biblatex.pdf`.

[20] Robin Fairbairns. *A Package für rotated objects in LATEX.* URL: `ftp://ftp.tu-chemnitz.de/pub/tex/macros/latex/contrib/rotating/rotating.pdf`.

[21] Robin Fairbairns. *footmisc. a portmanteau package for customising footnotes in LATEX.* URL: `http://ftp.fau.de/ctan/macros/latex/contrib/footmisc/footmisc.pdf`.

[22] Till Tantau, Joseph Wright und Vedran Miletić. *The beamer class. User Guide for version 3.20.* URL: `http://mirror.physik-pool.tu-berlin.de/tex-archive/macros/latex/contrib/beamer/doc/beameruserguide.pdf`.

[23] Tobias Oetiker u. a. *The Not So Short Introduction to LATEX 2ε.* Version 5.01. URL: `http://sunsite.informatik.rwth-aachen.de/ftp/pub/mirror/ctan/info/lshort/english/lshort.pdf`.

[24] Herbert Voß. *Präsentationen mit LATEX.* 1. Aufl. Lehmanns Media, 2009. ISBN: 978-3-86541-353-6.

[25] Herbert Voß. *Tabellen mit LATEX.* 2. Aufl. Lehmanns Media, 2010. ISBN: 978-3-86541-370-3.

[26] Herbert Voß. *Bibliografien mit LATEX.* 1. Aufl. Lehmanns Media, 2011. ISBN: 978-3-86541-415-1.

[27] Herbert Voß. *Einführung in LATEX 2ε. Unter Berücksichtigung von pdfLATEX, XeLATEX und LuaLATEX.* 1. Aufl. Lehmanns Media, 2012. ISBN: 978-3-86541-462-5.

[28] Herbert Voß. *Mathematiksatz mit LATEX.* 2. Aufl. Lehmanns Media, 2012. ISBN: 978-3-86541-485-4.

[29] Joseph Wright. *siunitx. A comprehensive (SI) units package.* URL: `http://ctan.space-pro.be/tex-archive/macros/latex/contrib/siunitx/siunitx.pdf`.

Index