Fachschaft Physik

Handout zum LaTeX-Kurs

Januar/Februar 2007

Teil 1: LaTeX-Einführung

Dieses Skript dient als Handout zum LaTeX-Kurs der Fachschaft Physik [1] an der Universität Konstanz [2]. Der Kurs und das Skript sind über mehrere Generationen engagierter Fachschaftler entstanden und weiterentwickelt worden. Bisher haben folgende Personen zu diesem Werk beigetragen: Volker Dobler, Bernd Rinn, Frank Bickendorf, Tobias Müther, Jörg Werner, Roland Hackl, Jens Dorfmüller, Oliver Gräser, Daniel Träutlein, Timo Böhm, Claudius Riek, Franziska Maier, Marcel Wunram. Die Autoren erheben keinen Anspruch auf Vollständigkeit und Fehlerfreiheit. Lob, Kritik und Anregungen bitte per Mail an:

Fachschaft.Physik@uni-konstanz.de

 Viel Spaß mit LaTeX!

Inhaltsverzeichnis

1. The Name of the Game

The Name of the Game – so beginnt auch Donald E. Knuths [3] Buch „The TeXbook“ [4], in dem er sein Textsatzprogramm TeX vorstellt. Im ersten Kapitel (das allerdings auch nur eine Seite umfasst), beschreibt er zunächst, wie es zu dem Namen TeX kam. Damit halten wir uns jetzt nicht auf. Aber einen kurzen Abriss zur Geschichte von TeX wollen wir doch geben.

1.1. Von TeXzu LaTeX

Donald E. Knuth, ein bekannter Informatik-Professor an der Standford University, begann in den 60er Jahren ein Standardwerk zur Programmierung zu schreiben: „The Art of Computer Programming“. Dabei ärgerte er sich so über die damals verbreiteten Programme zum Setzen von Büchern, dass er kurzentschlossen ein eigenes Programm dafür entwickelte: TeX. Letztendlich dauerte die damit verbundene Arbeit zehn Jahre, eine erste Version erschien 1978. Eine abgeänderte und verbesserte Version erschien dann 1982.
TeX war von Anfang an für die Texteingabe über einen gewöhnlichen Texteditor konzipiert. Knuth hatte nie im Sinn, ein grafisches Programm zur Erstellung von Dokumenten zu entwickeln.[1] Vielmehr legte er Wert auf eine strukturierte Vorgehensweise beim Erstellen von Dokumenten und die Erweiterbarkeit durch eigene „Macros“. Außerdem war ihm wichtig, das einmal erstellte Dokumente auf allen Systemen identisch aussehen, dass also dieselben Schriften in denselben Schriftgrößen verwendet werden und der Text an denselben Stellen umgebrochen wird. Und insbesondere sollte TeX das Setzen mathematischer Formeln vereinfachen.
Knuths TeX kennt 300 elementare Befehle, so genannte „Primitives“. Aus diesen Primitives können alle für den Textsatz nötigen Funktionen zusammengestellt

[1] Das wäre mit den damals verbreiteten Computersystemen auch sehr schwer gewesen.

werden. Knuth erstellte auch eine Macrosammlung, die 600 weitere Befehle enthält, die jedoch alle wiederum aus Primitives zusammengesetzt sind. Diese Macrosammlung nannte er „plain TeX“. Sie ist ein erster Schritt, die Arbeit mit TeX zu vereinfachen.

Trotzdem bedarf es fundierter Kenntnisse im Textsatz, um mit plain TeX professionell aussehende Dokumente zu setzen. Da nicht jeder die Zeit und Lust hat, sich dieses Wissen anzueignen, wurde in der Folge das LaTeX-Macropaket für TeX geschaffen. Es vereinfacht die Erstellung von Dokumenten noch einmal erheblich, da es dem Benutzer praktisch alle Layoutaufgaben abnimmt. Die einem Dokument zugrunde liegende Textdatei muss nur noch eine bestimmte Strukturierung aufweisen, LaTeX erzeugt dann „mit dem Wissen eines Buchdruckers“ vollautomatisch ein korrekt formatiertes druckfertiges Layout. Die aktuelle Version des Macropaketes ist LaTeX 2_ε von 1994.

1.2. pdfLaTeX

Knuth verwendete für die von TeX gesetzten Dokumente ein eigenes „Device-Independent“-Format (Dateiendung: *'.dvi'*), also ein Format, das unabhängig vom für die Ausgabe des Dokumentes verwendeten Ausgabegerät (Bildschirm, Drucker, ...) ist und damit sicherstellen soll, dass das Dokument auf allen Ausgabegeräten identisch aussieht.
Das DVI-Format setzte sich allerdings nicht durch, vielmehr übernahm die Funktion eines plattformunabhängigen Formates für die Druckvorstufe das PostScript-Format (Endung: *'.ps'*), das 1984 von Adobe [5] vorgestellt wurde. In der Folge mussten DVI-Dateien meist mit einem Konverter in PostScript-Dateien umgewandelt werden, um sie drucken zu können.
1993 stellte Adobe dann ein auf PostScript bassierendes, ebenfalls plattformunabhängiges Format vor: Das „Portable Document Format“ (Endung: *'.pdf'*). Da das PDF-Format einige Vorteile gegenüber PostScript hatte und ein passendes Programm zur Anzeige der PDF-Dateien von Adobe kostenlos verteilt wurde (der *Acrobat Reader* [6]), löste PDF PostScript immer mehr als Format für die Druckvorstufe ab.

Will man nun aus TeX-Quelltextdateien PDF-Dokumente erzeugen, muss man zunächst die von TeX erstellte DVI-Datei in eine PostScript-Datei konvertieren und diese wiederum in eine PDF-Datei umwandeln. Dabei gibt es einige Hürden zu überwinden, insbesondere was die verwendeten Schriften anbetrifft. Außerdem lassen sich die Funktionen, die das PDF-Format gegenüber dem DVI-Format mehr bietet, nicht nutzen.
Deshalb lag es nahe, das TeX-Programm so anzupassen, dass es direkt PDF-Dokumente erstellen kann. Diese Aufgabe übernahm Hàn Thế Thành im Rahmen seiner Dissertation [7], die er im Jahr 2000 vorlegte. In erster Linie beschäfigte er sich darin mit der Verbesserung des Textsatzes von TeX. Nur als „Nebenprodukt“ entstand pdfTeX, ein Programm, das aus TeX-Quelldateien direkt PDF-Dokumente erzeugt.[2]

[2] Soll pdfTeX mit LaTeX verwendet werden, so heißt es pdfLaTeX und wird mit **pdflatex**

2. Das erste Dokument

2.1. Ein Beispieldokument

Dieses Skript ist selbst ein gutes Beispiel für ein LaTeX-Dokument. Allerdings wäre es etwas unangemessen, anhand dieses Dokumentes die Grundlagen von LaTeX zu erarbeiten. Deshalb wählen wir ein einfacheres Beispiel, das aber dennoch bereits viele Feinheiten enthält:

Kleines Beispieldokument zum LaTeX-Kurs

Franziska, Oliver, Claudius

14. Dezember 2005

Inhaltsverzeichnis

1 Einleitung

Wir wollen euch mit diesem Dokument zeigen, was LaTeX kann und eine Idee davon vermitteln, wie man LaTeX dazu bringt, das zu tun, was es kann. Natürlich kann ein solches Dokument nicht den ganzen Kurs ersetzen, aber es soll euch zeigen, was wir euch in den nächsten drei Einheiten alles beibringen wollen:

1. LaTeX-Einführung
2. Mathematischer Formelsatz
3. LaTeX-Vertiefung

2 Mathematischer Formelsatz

2.1 Es geht einfach ...

Vermeintlich einfache Formeln sind auch für andere Programme wie Word kein Problem: $E = mc^2$. Diese Programme bieten oft einen einfach zu bedienenden grafischen Formel-Editor – allerdings...

2.2 ...aber auch kompliziert

...werfen LaTeX auch komplizierte Konstrukte wie die **zukünftige Weltformel**[1] nicht aus der Bahn:

$$\vec{x} = -i \left[\frac{(\sin(\Omega\psi))^{e^{-\vec{k}\omega}}}{2\pi} + 1 \right] \tag{1}$$

Wir hoffen, dass ihr jetzt einen Eindruck davon bekommen habt,
warum LaTeX einfach gut ist!

[1] Ein bisschen müssen wir daran wohl noch arbeiten. Wenigstens sollte auch rechts ein Vektor stehen, wenn schon links einer steht.

1

2.2. Der Quelltext des Beispieldokumentes

Wie bereits erwähnt, erstellt man ein LaTeX-Dokument durch Eingabe des Textes und entsprechender Formatierungsbefehle in eine einfache Textdatei, die

aufgerufen. Mehr dazu in Anhang C.

meist auf *'.tex'* endet und als „Quelltext“ des Dokumentes bezeichnet wird. Für unser Beispieldokument sieht diese Quelltextdatei folgendermaßen aus:

```
\documentclass[12pt]{scrartcl}

\usepackage[latin1]{inputenc}
\usepackage{ngerman}
\usepackage[T1]{fontenc}
\usepackage{mathptmx,helvet,courier}

\begin{document}

\title{Kleines Beispieldokument zum \LaTeX-Kurs}
\author{Claudius, Oliver und Timo}
\date{19. Mai 2005}

\maketitle
\tableofcontents

\section{Einleitung}

Wir wollen euch mit diesem Dokument zeigen, was \LaTeX\ kann und [...]
Wochen alles beibringen wollen:
\begin{enumerate}
\item \LaTeX-Einführung [...]
\end{enumerate}

\section{Mathematischer Formelsatz}

\subsection{Es geht einfach \ldots}

\emph{Vermeintlich} einfache Formeln sind auch für andere Programme wie Word kein
Problem: $E = mc^2$. Diese Programme [...] Formel-Editor -- allerdings\ldots

\subsection{\ldots aber auch kompliziert}

\ldots werfen \LaTeX\ auch komplizierte Konstrukte wie die \textbf{zukünftige
Weltformel}\footnote{Ein bisschen müssen [...] steht.} nicht aus der Bahn:

\begin{displaymath}
\vec{x} = -i \left[ \frac{\left( \sin \left( \Omega \psi \right) \right)^{e^{-i \vec{k}
\omega}}}{2\pi} + 1\right]
\end{displaymath}

\vspace{1.2cm}

\begin{center}
{\Large Wir hoffen, dass ihr jetzt einen Eindruck davon bekommen habt, warum
\LaTeX\ einfach gut ist!}
\end{center}

\end{document}
```

Analysisieren wir den Quelltext im Einzelnen. Dabei wollen wir zunächst den so genannten „Dokumentenkopf“ (englisch „document header“) übergehen und mit dem Hauptteil des Dokumentes, in dem der Text steht, beginnen.
Wichtig zu wissen ist nur, dass im Dokumentenkopf die „Dokumentenklasse“ festgelegt wird, die das Aussehen des Dokumentes bestimmt. Wir haben im Beispieldokument die Klasse *scrartcl* gewählt.
Zunächst müssen wir allerdings etwas zu Befehlen in TEX sagen.

2.3. Kleiner Exkurs: Zeichen und Befehle

Vergleicht man das fertig gesetzte Dokument mit dem Quelltext, fällt gleich auf, dass die Formatierungsbefehle offenbar alle mit einem Backslash „\“ eingeleitet werden. Darüber hinaus fällt auf, dass häufig geschweifte Klammern „{}“ auftreten, die ebenfalls nicht im fertigen Dokument auftauchen.

2.3.1. Befehle in TeX

Diese Beobachtung ist ganz richtig. In TeX haben Befehle die folgende Form: `\Befehl[Optionen]{Parameter}`. Dabei kann ein Befehl Optionen haben, er kann Parameter erfordern, kann Optionen haben und Parameter erfordern oder auch weder Optionen haben noch Parameter erfordern.
Erfordert ein Befehl Parameter, so stehen diese immer in geschweiften Klammern. Die geschweiften Klammern müssen in aller Regel angegeben werden, können aber auch leer sein.
Optionen stehen in eckigen Klammern und sind – wie der Name schon sagt – optional. Das heißt, werden keine Optionen angegeben, kann der Befehl auch auf `\Befehl{Parameter}` verkürzt werden.
Bei Befehlen, die weder Parameter erfordern noch Optionen haben, entfallen einfach die Bereiche mit eckigen und geschweiften Klammern: `\Befehl`.
Noch ein Abschließender Hinweis: TeX unterscheidet bei Befehlen zwischen *Groß- und Kleinschreibung*.
Detaillierteres zu Befehlen findet sich in Abschnitt 5.5.

2.3.2. Leerzeichen

Stehen mehrere Leerzeichen im Quelltext nacheinander, werden sie zu einem Leerzeichen zusammengefasst. Eine einzeln stehende Zeilenschaltung (also eine, die keine Leerzeile erzeugt) gilt dabei auch als Leerzeichen. Möchte man dennoch mehrere Leerzeichen einfügen, muss Gruppierung wie in Abschnitt 5.5 beschrieben, verwendet werden.
Eine oder mehrere Leerzeilen im Quelltext stehen für das Ende eines Absatzes. TeX beginnt dann automatisch einen neuen. Mehr dazu in Abschnitt 5.2.

2.3.3. Reservierte Zeichen

In TeX haben einige Zeichen ein besondere Bedeutung. Deshalb können sie nicht direkt bei der Texteingabe eingegeben werden. Selbstverständlcih gibt es aber eine Möglichkeit, sie im Text zu verwenden:

reserviertes Zeichen	$	&	%	#	_	{	}
Eingabe in TeX	`\$`	`\&`	`\%`	`\#`	`\_`	`\{`	`\}`

Für drei reservierte Zeichen gibt es keine so einfache Möglichkeit wie das voranstellen eines Backslashs, um sie im Text zu verwenden:

res. Zeichen	~	^	\
in TeX	`\textasciitilde`	`\textasciicircum`	`\textbackslash`

2.4. Es geht los: Umgebungen

Jetzt kommen wir aber endlich zum eigentlichen Text. Dieser steht immer zwischen den Befehlen `\begin{document}` und `\end{document}`, die die so genannte *document*-Umgebung aufspannen. LaTeX kennt viele verschiedene Umgebungen, drei weitere sind bereits in unserem Beispieldokument enthalten: Die *enumerate*-Umgebung für Aufzählungen (siehe Abschnitt 2.7), die *displaymath*-Umgebung für mathematischen Formelsatz (wird im zweiten Teil dieses Kurses behandelt) und die *center*-Umgebung für zentrierten Text (3.2). *Wichtig:* Die Verarbeitung eines Quelltextes durch LaTeX endet immer beim Befehl `\end{document}`; er sollte deshalb in der letzten Zeile jedes Quelltextes stehen.

2.5. LaTeX nimmt uns Arbeit ab: Die Textstrukturierung

Beim ersten Blick auf das Beispieldokument fällt sicher auf, dass es ein Inhaltsverzeichnis[3] und nummerierte Abschnitte und Unterabschnitte enthält. Allerdings findet sich nirgendwo im Quelltext ein Bereich, der den Text des Inhaltsverzeichnisses enthält. Alles was wir bei LaTeX nämlich zur Erstellung dieses Verzeichnisses benötigen, ist der Befehl `\tableofcontents`. Damit der Befehl weiß, was Abschnitte und was Unterabschnitte sind, müssen wir diese mit den Befehlen `\section[`*Kurzform*`]{`*Überschrift*`}` und `\subsection[`*Kurzform*`]{`*Überschrift*`}` einleiten. Wird eine Kurzform für eine Überschrift angegeben, so wird diese beispielsweise im Inhaltsverzeichnis verwendet.

Auch einen ansprechend formatierten Titel erstellt uns LaTeX ohne weiteres. Dafür müssen wir nur mit den Befehlen `\title{`*Titel*`}`, `\author{`*Autor*`}` und `\date{`*Datum*`}` die entsprechenden Angaben machen und dann mit `\maketitle` die Titelseite ausgeben lassen. Wird kein Datum angegeben, wird übrigens das aktuelle verwendet.

2.6. Textformatierung

Da LaTeX – bei Verwendung der im vorigen Abschnitt beschriebenen Formatierungsbefehle – bereits Überschriften und Titel korrekt formatiert, muss nur selten direkt Hand angelegt werden. Ein wichtiger Punkt ist jedoch, Text im allgemeinen Textfluss hervorheben zu können. Dies geschieht mit dem Befehl `\emph{hervorzuhebender Text}`. Der Befehl passt sich der aktuell verwendeten Schriftart, der Schriftgröße und dem Schriftstil an (siehe Abschnitt 3.2). In normalem Text wird beispielsweise der so hervorgehobene Textteil *kursiv* dargestellt.

[3] Um das Inhaltsverzeichnis zu erstellen, muss TeX zweimal das Dokument erstellen (auch „übersetzen“ oder „compilieren“). Beim ersten Mal merkt es sich in einer separaten Datei, welche Nummern die Abschnitte bekommen haben und auf welchen Seiten sie sich befinden. Diese Informationen nutzt es dann beim zweiten Durchlauf zum Erstellen des Inhaltsverzeichnisses. Mehr dazu in Anhang C.

2.7. Aufzählungen

LaTeX bietet eine einfache Möglichkeit, nummerierte[4] Aufzählungen zu erstellen. Dazu wird die *enummerate*-Umgebung verwendet. Jeder neue Aufzählungspunkt wird durch den Befehl `\item` eingeleitet. Aufzählungen können beliebig ineinander geschachtelt werden.

2.8. Fußnoten

Automatisch nummerierte Fußnotentexte erstellt man durch den Befehl `\footnote{`*`Fußnotentext`*`}`. Die Fußnoten werden automatisch am unteren Seitenrand angeordnet.

2.9. Vertikale Abstände

Ist der Abstand zwischen zwei Absätzen, den LaTeX automatisch wählt, zu groß oder zu klein, so kann er mit dem Befehl `\vspace{`*`Abstand`*`}` angepasst werden. Der Abstand muss dabei in einer der von TeX benutzten Größeneinheiten angegeben werden. Mehr dazu in Anhang B.

2.10. Kleiner Ausblick: Mathematischer Formelsatz

In unserem Beispieldokument kommen bereits zwei Formeln vor: Einmal im Fließtext und einmal zentriert in einer eigenen Zeile. Das erste wird erreicht, indem durch das Zeichen „$“ der Mathematikmodus für Fließtext aktiviert wird. Der Mathematikmodus für abgesetzte Formeln kann z. B. mit `\begin{displaymath}` aktiviert werden. Mehr dazu und zu den Befehlen, die im Mathematikmodus verwendet werden können, folgt im zweiten Teil dieses Kurses.

3. Weiterführendes

3.1. Weitere Strukturbefehle

Die Dokumentenklasse *scrartcl* kennt zusätzlich zu den in Abschnitt 2.5 beschriebenen Strukturbefehlen noch folgende weitere:

`\part[`*`Kurzform`*`]{`*`Überschrift`*`}`	Teildokument
⋮	
Hier kommen die Befehle `\section` *und* `\subsection`	
⋮	
`\subsubsection[`*`Kurzform`*`]{`*`Überschrift`*`}`	„Unterunterabschnitt“
`\paragraph[`*`Kurzform`*`]{`*`Überschrift`*`}`	Absatz
`\subparagraph[`*`Kurzform`*`]{`*`Überschrift`*`}`	Unterabsatz

Die Klassen für Berichte und Bücher, die wir im dritten Teil des Kurses kennenlernen werden, verstehen zusätzlich den Strukturbefehl `\chapter[`*`Kurzform`*`]`

[4] Wie nicht nummerierte Aufzählungen erstellt werden, zeigen wir in Abschnitt 3.3.2.

`{`*`Überschrift`*`}`, der in der Gliederung direkt nach `\part` kommt und eine Unterteilung der Dokumente in Kapitel erlaubt.
Die mit den Befehlen `\paragraph` und `\subparagraph` erzeugten Strukturen werden nicht nummeriert.

Steht im Quelltext der Befehl `\appendix`, stellt LaTeX die Nummerierung der Abschnitte auf Buchstaben um, wie es im Anhang zu einem Dokument üblich ist.

3.2. Weitere Formatierungen

3.2.1. Schriftgrößen

Folgende voreingestellte Schriftgrößen, die immer abhängig von der gewählten Grundschriftgröße der Dokumentenklasse eingestellt werden, stehen zur Verfügung:

`\tiny`	Beispieltext	`\large`	Beispieltext
`\scriptsize`	Beispieltext	`\Large`	Beispieltext
`\footnotesize`	Beispieltext	`\LARGE`	Beispieltext
`\small`	Beispieltext	`\huge`	Beispieltext
`\normalsize`	Beispieltext	`\Huge`	Beispieltext

3.2.2. Schriftarten

In jeder Dokumentenklasse werden für unterschiedliche Schriftstile Schriftarten definiert. Es gibt:

`\textrm{...}`	Serifenschrift
`\textsf{...}`	serifenlose Schrift
`\texttt{...}`	`Schreibmaschinenschrift`
`\textbf{...}`	**Fettschrift**
`\textmd{...}`	Medium-Schrift
`\textit{...}`	*Kursivschrift*
`\textup{...}`	aufrechte Schrift
`\textsc{...}`	Kapitälchen-Schrift
`\textsl{...}`	*Schrägschrift*

Wie aus obiger Tabelle ersichtlich, kann zwei oder mehr Schriftstilen auch ein und dieselbe Schriftart zugeordnet sein. Über `\textnormal{...}` kann man immer auf die in der Dokumentenklasse voreingestellte Grundschrift zugreifen. Die Schreibmaschinenschrift ist meist eine Festbreitenschrift, d. h. eine Schrift bei der alle Zeichen gleich breit sind. Alle anderen Schriftstile sind in der Regel Proportionalschriften.

3.3. Weitere Umgebungen

3.3.1. Textausrichtung

Wie es im professionellen Textsatz üblich ist, setzt LaTeX Text in Blocksatz, das heißt, der linke und der rechte Rand sind bündig. Bei Bedarf kann man dieses Verhalten ändern und mit den Umgebungen *flushleft*, *center* oder *flushright* linksbündigen, zentrierten oder rechtsbündigen Textsatz wählen.

```
\begin{flushleft}
Dieser Text wird linksbuendig gesetzt. Das
kommt selbstverstaendlich erst zum Tragen,
wenn er laenger als eine Zeile ist.
\end{flushleft}
\begin{center}
Zentrierter Text.
\end{center}
\begin{flushright}
Rechtsbuendiger Text, das bedeutet, dass der
linke Rand ausgefranzt erscheint. Fuer
Fliesstext ist rechtsbuendiger Text im
Allgemeinen nicht geeignet.
\end{flushright}
```

Dieser Text wird linksbuendig gesetzt. Das kommt selbstverstaendlich erst zum Tragen, wenn er laenger als eine Zeile ist.

Zentrierter Text.

Rechtsbuendiger Text, das bedeutet, dass der linke Rand ausgefranzt erscheint. Fuer Fliesstext ist rechtsbuendiger Text im Allgemeinen nicht geeignet.

3.3.2. Aufzählungen

Nicht nummerierte Aufzählungen werden mit der *itemize*-Umgebung erstellt, die genau wie die *enumerate*-Umgebung aus Abschnitt 2.7 funktioniert.
Eine weitere Umgebung um im weitesten Sinne Aufzählungen zu erstellen, ist die *description*-Umgebung. Sie setzt den Aufzählungstext, der dem `\item[`*`Aufzählungstext`*`]`-Befehl übergeben wird, fett und abgehoben.

```
\begin{itemize}
\item Dies ist eine \emph{itemize}-Umgebung.
\begin{description}
\item[Achtung] Das ist eine
  \emph{description}-Umgebung innerhalb
  der \emph{itemize}-Umgebung
\end{description}
\item Noch ein Punkt der
  \emph{itemize}-Umgebung
\end{itemize}
```

- Dies ist eine *itemize*-Umgebung.

  Achtung Das ist eine *description*-Umgebung innerhalb der *itemize*-Umgebung

- Noch ein Punkt der *itemize*-Umgebung

3.3.3. Zitate und Gedichte

Speziell für Zitate ist die *quote*-Umgebung gedacht. Je nach Vorliebe kann auch die *quotation*-Umgebung benutzt werden, die Abschnitte zusätzlich einrückt.

```
Auszug aus Martin Luther Kings
beruehmter Rede:
\begin{quote}
I have a dream that one day on the red
hills of Georgia the sons of former slaves
and the sons of former slaveowners will
be able to sit down together at a table
of brotherhood.
\end{quote}
```

Auszug aus Martin Luther Kings beruehmter Rede:

> I have a dream that one day on the red hills of Georgia the sons of former slaves and the sons of former slaveowners will be able to sit down together at a table of brotherhood.

Für Gedichte bietet sich die *verse*-Umgebung an. Zeilenumbrüche werden durch einen doppelten Backslash „\\“ am Zeilenende angezeigt. Verse werden durch eine Leerzeile getrennt.

```
Timos grosses Werk:
\begin{verse}
Fuer dieses Beispielchen \\
brauche ich jetzt -- ach

ein kleines Gedichtchen \\
drum ich verzach.
\end{verse}
```

Timos grosses Werk:

> Fuer dieses Beispielchen
> brauche ich jetzt – ach
>
> ein kleines Gedichtchen
> drum ich verzach.

3.3.4. Langer Schreibmaschinentext

Längerer Text in Schreibmaschinenschrift, wie er in diesem Skript z. B. zur Darstellung des Quelltextes in Abschnitt 2.2 verwendet wird, wird durch Verwendung der *verbatim*-Umgebung erreicht.
Innerhalb der *verbatim*-Umgebung können alle Zeichen verwendet werden, auch die reservierten. Es werden keine Befehle ausgeführt und es findet kein automatischer Zeilenumbruch statt – eben wie beim Schreiben auf einer Schreibmaschine.

4. Der Dokumentenkopf

Bisher haben wir nichts zu den ersten Zeilen des Quelltextes gesagt, in denen nur Befehle stehen, die nicht direkt mit der Textdarstellung zu tun haben.
Moment! – So kann man das eigentlich nicht sagen. Vielmehr sind es gerade diese Befehle, die ganz elementar die Darstellung des gesamten Dokumentes beeinflussen. Allerdings ändern sie nichts an der Struktur des Textes und am Text an sich.

4.1. Die Dokumentenklasse

Ein LaTeX-Dokument muss immer mit der Spezifikation der so genannten „Dokumentenklasse“ beginnen. Dies geschieht durch Angabe des Befehls `\documentclass[`*`Optionen`*`]{`*`Dokumentenklasse`*`}`. Die Dokumentenklasse legt fest, welches spezifische Layout LaTeX für das Dokument wählt. Es gibt beispielsweise Klassen für Bücher, Briefe und Präsentationen. Wir verwenden hier die Klasse *scrartcl* aus dem Koma-Script-Paket, die einen einseitig formatierten (wissenschaftlichen) Artikel mit den europäischen Textsatzkonventionen erzeugt. Weitere Klassen werden im dritten Teil dieses Kurses vorgestellt.
Der Dokumentenklasse kann man in eckigen Klammern einige klassenabhängige Optionen übergeben. Meist sind dies das Papierformat, die Grundschriftgröße und ob der Text zweispaltig gesetzt werden soll. Bei unserer Klasse ist das DIN-A4-Papierformat als Standard voreingestellt. Wir wählen als Option nur eine etwas größere Schriftgröße (12 Punkte statt den voreingestellten 11 Punkten).

4.2. Erweiterungen – Packages

LaTeX bietet eine einfache Möglichkeit, den Befehlsumfang des Grundsystems zu erweitern: so genannte „Pakete“ (englisch „package“). Pakete sind Befehlssätze, die man zusätzlich laden kann und die dann weitere Funktionen zur Verfügung stellen.
Der Befehl, um Pakete zu laden, heißt `\usepackage[`*`Optionen`*`]{`*`Paket`*`}`. Wie Klassen kann man auch Paketen Optionen übergeben. Sollen meherere Pakete geladen werden, denen keine Optionen übergeben werden, so kann man dafür einen `\usepackage`-Befehl verwenden und eine durch Komma getrennte Liste der zu ladenden Pakete angeben.

Zu den im Beispieldokument verwendeten Paketen:

4.2.1. inputenc

Das Paket *inputenc* wird benötigt, um z. B. deutsche Umlaute und das „ß“ direkt über die Tastatur eingeben zu können. Als Option muss man *inputenc* die Kodierung übergeben, in der das Dokument erstellt wurde. Wir haben hier die in Westeuropa und Nordamerika weitverbreitete Kodierung Latin-1 gewählt. Selbstverständlich muss dann der Editor, der zur Erstellung der LaTeX-Datei verwendet wird, ebenfalls auf Latin-1 eingestellt sein.[5] Für Emacs erreichen wir das mit der Konfigurationsdatei aus dem Vorkurs. Im Anhang A.1 wird erklärt, dass eventuell Latin-9 die bessere Wahl wäre. Aus technischen Gesichtspunkten sollte sogar UTF-8 verwendet werden, das zum jetzigen Zeitpung aber im Zusammenspiel mit LaTeX nur schwer handhabbar ist. Mehr zu dem umfassenden Thema der Textkodierung in Anhang A.1.

[5] Latin-1 ist eine häufig gebrauchte Bezeichnung und wird deshalb auch in LaTeX verwendet. Besser wäre aber ISO 8859-1.

4.2.2. ngerman

TEX und LATEX sind ursprünglich auf die Verwendung mit englischen Texten ausgelegt. Eine Anpassung an die deutsche Sprache ist – bis auf das Problem der deutschen Umlaute und des „ß“, mehr dazu im nächsten Abschnitt – nicht besonders schwer. Diese Anpassung nimmt das Paket *ngerman* vor, das beispielsweise die Silbentrennung und die Texte für Inhaltsverzeichnis (der deutsche Begriff „Inhaltsverzeichnis“ wird statt dem englischen „Contents“ verwendet) usw. beeinflusst.

4.2.3. Einschub: Deutsche Umlaute und Sonderzeichen in LaTeX

Deutsche Umlaute waren wie Umlaute aus anderen Sprachen von jeher nicht zur direkten Eingabe in das englischsprachige TEX vorgesehen. Auch enhalten die von Donald E. Knuth für TEX kreierten METAFONT-Schriften die Umlaute nicht als selbstständige Zeichen. Vielmehr enthalten sie nur die benötigten Accents – so beipsielsweise für die deutschen Umlaute den horizontalen Doppelpunkt „¨“ – die dann mittels TEX-Befehlen über den Buchstaben aus der Schrift gesetzt werden.
Das deutsche „ö“ kann in plain TEX durch folgende Befehle erzeugt werden: `\"o`[6] oder `\accent4 o`.[7] Da diese Eingabevarianten sehr umständlich waren, hat das Paket *ngerman* (bzw. *german* für die alte deutsche Rechtschreibung) eine neue Abkürzung eingführt. Fortan konnte man deutsche Sonderzeichen durch voranstellen des doppelten Anführungszeichens „`"`“ bekommen. Ein kleiner Überblick über die wichtigsten Abkürzungen in *ngerman*:

Zeichen	Ä	ä	Ö	ö	Ü	ü	ß	„	“
Abkürzung	`"A`	`"a`	`"O`	`"o`	`"U`	`"u`	`"s`	`"‘`	`"’`

Verwendet man *inputenc* mit der Zeichkodierung Latin-1 oder Latin-9, sollten die Umlaute und das „ß“ direkt eingegeben werden. Nur die deutschen Anführungszeichen müssen noch über die Abkürzungen aus *ngerman* eingebunden werden.
Wird *ngerman* verwendet, können die gewöhnlichen doppelten Anführungszeichen „`"`“ auf der Tastatur nicht mehr direkt eingegeben werden. Um sie dennoch in einen Text einzufügen, kann der Befehl `\dq` verwendet werden.

Einen kurzen stichpunktartigen Leitfaden durch den Dschungel der Kodierungen, Umlaute und Sonderzeichen gibt es in Anhang A.5.

4.2.4. fontenc, mathptmx, helvet und courier

Wie schon im vorigen Abschnitt beschrieben, setzt TEX bei den METAFONT-Schriften die Umlaute aus Accents und den jeweilige Buchstaben ohne Ac-

[6] Dabei ist dieser Befehl nur eine Abkürzung für den folgenden. Der TEX-Befehl für das „ß“ ist `\ss`.

[7] Wird nicht die T1-Kodierung der Schriften verwendet, wie es im nächsten Abschnitt beschrieben wird, so würde der Befehl `\accent127 o` lauten. Die Zahl gibt einfach an, an welcher Stelle der jeweiligen Schrift-Kodierung das Accent-Zeichen zu finden ist.

cents zusammen, weil die Umlaute selbst nicht als Zeichen in den Schriftdateien enthalten sind. Ein Nachteil dieser Vorgehensweise ist jedoch, dass in PDF-Dokumenten nicht nach Umlauten gesucht werden kann. So genannte T1-Kodierte-Schriften enthalten die Zeichen für die Umlaute direkt und setzen sie nicht aus mehreren Zeichen zusammen. Deshalb schalten wir im Dokumentenkopf durch Verwendung des Paketes *fontenc* mit der Option *T1* auf diese Kodierung um. Außerdem ersetzen wir die standardmäßig verwendeten METAFONT-Schriften aus der Computer-Modern-Schriftfamilie, die nicht in einer passenden T1-Kodierung vorliegen, durch T1-Kodierte-Schriften.
Das Paket *mathptmx* lädt als Standardschrift mit Serifen eine Schrift aus der Times-Familie und die dazu passenden Mathematik-Schriften. *helvet* lädt als serifenlose Schrift die Schrift Helvetica, *courier* als Schrift mit konstanter Zeichenbreite die Schrift Courier.
Es empfiehlt sich, diesen Header einfach zu übernehmen. Sollten jedoch die gewählten Schriften nicht gefallen, so gibt es noch weitere T1-Kodierte-Schriften, die in Frage kommen.

5. So arbeitet TEX

5.1. Spezielle Zeichen

In einem gewöhnlichen Fließtext gibt es mehrere Verwendungen für den gewöhnlichen Querstrich „-“: Er kann als Binde-strich oder als – Gedankenstrich – verwendet werden. Abhängig davon wird der Querstrich unterschiedlich lang gesetzt:

	in TEX	Länge
Bindestrich/Minus	`-`	-
Gedankenstrich	`--`	–
angelsächsischer Gedankenstrich	`---`	—

Die angelsächsischen “Anführungszeichen” werden durch „`‘‘`“ (öffnende, doppelter Accent grave) und „`’’`“ (schließende, doppeltes Apostroph) eingefügt, einfache ‚deutsche‘ durch „`\glq`“ und „`\grq`“.[8]

5.2. Leerzeichen, Absätze, Zeilen und Seiten

5.2.1. Absätze

Absätze dienen neben Kapiteln und Abschnitten der gedanklichen Gliederung des Textes. Die erste Zeile eines Absatzes wird normalerweise ein wenig eingerückt, der Text wird in Blocksatz (also links und rechts bündig) gesetzt und nur die letzte Zeile endet in der Regel nicht bündig mit dem rechten Seitenrand.
Eine Leerzeile im Quelltext zeigt LATEX das Absatzende an.[9] Um in LATEX Text

[8] Für die einfachen deutschen Anführungszeichen muss das Paket *ngerman* geladen sein.
[9] Ein Absatz beginnt dort wo das erste Zeichen dieses Absatzes steht. Da ein Absatz nur beendet werden kann, wenn er irgendwo vorher begonnen wurde, bewirken mehrere aufeinander folgende Leerzeilen dasselbe wie eine einzelne. Ein Absatz kann übrigens nicht mit einem Leerzeichen beginnen.

in einzelne Absätze zu gliedern, muss man also einfach im Quelltext eine Leerzeile nach dem Text jedes Absatzes einfügen, man muss also zwei Zeilenschaltungen einfügen. Alternativ zeigt auch der Befehl `\par` das Ende eines Absatzes an.

5.2.2. Zeilen

Jeder Absatz ist wiederum in Zeilen unterteilt, die (bis auf die letzte Zeile) eine feste Länge haben. Normalerweise bestimmt TEX automatisch, wo eine Zeile endet und eine neue Zeile beginnt. Es sucht sich dafür entweder ein Leerzeichen im Quelltext und beginnt danach die neue Zeile oder trennt ein Wort wie im Abschnitt 5.3 beschrieben. Möchte man vermeiden, dass bei bestimmten Leerzeichen eine neue Zeile begonnen wird, so muss das Leerzeichen durch eine Tilde „`~`“ ersetzt werden (Beispiel: `Prof.~Dr.~Einstein`) oder der Ausdruck mit dem Befehl `\mbox{`*`Ausdruck`*`}` angegeben werden.
Möchte man selbst angeben, wo eine neue Zeile beginnen soll, verwendet man dafür die Befehle `\\` oder `\newline`. Sie wechseln in eine neue Zeile, ohne einen neuen Absatz zu beginnen. Analog funktioniert der Befehl `\linebreak`, der allerdings zusätzlich versucht, den Blocksatz aufrecht zu erhalten.
Um einen etwas größeren Abstand zwischen den getrennten Zeilen zu setzen, kann `\\[`*`Abstand`*`]` verwendet werden.

5.2.3. Seiten

Bevor die aktuelle Seite voll ist und deshalb automatisch eine neue begonnen wird, wechselt der Befehl `\newpage` manuell auf eine neue Seite.

5.3. Die Worttrennung

Bei der Berechnung des Textsatzes wendet TEX automatisch einen Algorithmus für die Silbentrennung an. Wird das Pakage *ngerman* geladen, werden deutsche Silbentrennmuster verwendet. Die Worttrennung lässt sich auch manuell beeinflussen.
Soll ein Wort nicht getrennt werden, so wird das Wort unter Verwendung des Befehls `\mbox{`*`Wort`*`}` im Fließtext angegeben.
Wird andererseits ein Wort nicht automatisch getrennt, können die möglichen Trennstellen im Fießtext durch Einfügen von „`\-`“ in das Wort angegeben werden. Beispiel: `Tren\-nung`. Muss dies für ein und dasselbe Wort mehrere Male geschehen, kann ein neues Silbentrennmuster für dieses Wort mit dem Befehl `\hyphenation{`*`Wortliste`*`}` definiert werden. Beispiel: `\hyphenation{Linux Tren-nung}` verhindert, dass „Linux“ getrennt wird und legt ein Trennmuster für „Trennung“ fest.

5.4. Kommentare

Sollen im Quelltext Kommentare stehen, die weder zum Text gehören noch irgendeinen Einfluss auf die Bearbeitung des Textes durch TEX nehmen sollen,

so muss man ihnen das Prozentzeichen „%“ voranstellen. TEX ignoriert dann ab dem Prozentzeichen den Rest der Zeile.
Dies ist eine einfache Möglichkeit, Wendungen, die eigentlich nicht über mehrere Zeilen gehen dürfen, im Quelltext in mehrere Zeilen aufzuteilen:

```
Satanarchaeolue% Ab hier wird ignoriert
        % (auch Zeilenumbrueche)
  genial% bis zum naechsten Zeichen, das
  kohoel% kein Leerzeichen ist
  lischer Wunschpunsch
```

Satanarchaeoluegenialkohoellischer Wunschpunsch

5.5. Noch etwas zu Befehlen

5.5.1. Control Sequences: Control Words und Control Symbols

Auszug aus dem TEXbook [4] von Donald E. Knuth:

> „Control Sequences come in two flavors. The first kind [...] is called a *control word*; it consists of an escape character followed by one or more *letters*, followed by a space or by something besides a letter. [...] In case you're wondering what a "letter" is, the answer is that TEX normally regards the 52 symbols `A...Z` and `a...z` as letters. The digits `0...9` are *not* considered to be letters, so they don't appear in control sequences of the first kind.
>
> A control sequence of the other kind, like `\``, is called a *control symbol*; it consists of the escape character followed by a single *nonletter*. In this case you don't need a space to separate the control sequence from a letter that follows, since control sequences of the second kind always have a exactly one symbol after the escape character.“

In TEX ist es also möglich, Befehle auf zwei Arten anzugeben:[10] Einmal als „Befehlswort“ und einmal als „Befehlssymbol“. Beiden Arten gemeinsam ist, dass sie immer durch den so genannten „escape character“ – im Normalfall ist das der Backslash „\“ – eingeleitet werden. Weiter muss man dann unterscheiden:

- Bei *Befehlsworten* folgt auf den Escape Character direkt ein oder mehrere Buchstaben. Buchstaben sind dabei nur die Zeichen `A...Z` and `a...z`.[11] Danach muss ein Leerzeichen oder ein anderes Zeichen folgen, das kein Buchstabe ist, damit TEX das Ende des Befehlswortes erkennen kann.
- Bei *Befehlssymbolen* folgt auf den Escape Character direkt ein Zeichen, das kein Buchstabe ist. Da TEX genau weiß, dass ein Befehlssymbol nur aus einem einzigen Zeichen besteht, muss danach kein Leerzeichen folgen.

[10] Bei der Verwendung des Paketes *ngerman* kommt noch die Möglichkeit mit dem führenden doppelten Anführungszeichen, wie in Abschnitt 4.2.3 beschrieben, infrage.
[11] Insbesondere sind die Umlaute in diesem Sinne keine Buchstaben.

5.5.2. Praktisches

Das Problem ist nun, dass TEX ja mehrere Leerzeichen, die direkt aufeinander folgen, als ein Leerzeichen interpretiert. Wenn also ein Leerzeichen das Ende eines Befehlswortes anzeigt, kann es nicht gleichzeitig für ein Leerzeichen im Textfluss stehen.
Aus dem Quelltext

```
Ich möchte wissen, wie \LaTeX  diese Zeile ausgibt.
```

macht LATEX:

Ich möchte wissen, wie LATEXdiese Zeile ausgibt.

Obwohl im Quelltext nach dem Befehlswort `\LaTeX`[12] *zwei* Leerzeichen stehen, wird im Dokument keines zwischen dem LATEX-Schriftzug und dem folgenden Wort eingefügt. Nach den obigen Ausführungen war das sicher jedem klar. Aber wie kann man nun nach einem Befehlswort ein Leerzeichen im Dokument erzeugen? Dafür gibt es zwei Möglichkeiten:

1. Es gibt ein Befehlsymbol, dass im Dokument immer ein Leerzeichen erzeugt. Das Befehlssymbol dafür ist „`\ `“, also der Escape Character gefolgt von einem Leerzeichen.

2. Eine zweite Möglichkeit ist das so genannte „Grouping“, also das Zusammenfassen von Teilen des Quelltextes zu einer Gruppe. Dazu dienen in TEX die geschweiften Klammern „`{`“ und „`}`“. Alles was in solchen Klammern steht, bildet eine Gruppe. Befehle die in einer Gruppe aufgerufen werden, gelten nur innerhalb dieser Gruppe. Darüber hinaus ist es auch möglich, eine leere Gruppe (also `{}`) zu verwenden. Leerzeichen werden nicht über Gruppengrenzen hinaus zu einem einzigen zusammengefasst.

Mit dem obigen Wissen können wir nun auf verschiedene Arten den gewünschten Satz

Ich möchte wissen, wie LATEX diese Zeile ausgibt.

schreiben. Nämlich so:

```
Ich möchte wissen, wie \LaTeX\ diese Zeile ausgibt.
Ich möchte wissen, wie \LaTeX \ diese Zeile ausgibt.
Ich möchte wissen, wie {\LaTeX} diese Zeile ausgibt.
Ich möchte wissen, wie \LaTeX{} diese Zeile ausgibt.
Ich möchte wissen, wie \LaTeX {} diese Zeile ausgibt.
```

[12] Das – es wird wohl jeder gemerkt haben – den LATEX-Schriftzug erzeugt.

A. Zeichenkodierung im Detail

A.1. Zur Zeichendarstellung im Computer

In der Frühzeit der Computer, als nur wenig Speicher verfügbar war, stellte man Zeichen durch sieben Stellen, die jeweils die Wertigkeit 0 oder 1 annehmen konnten, dar (sieben Bit). Insgesamt konnte man also $2^7 = 128$ Zeichen darstellen. Nach Abzug von 32 notwendigen Steuerzeichen bleiben noch 96 darstellbare Zeichen. Diese verteilen sich auf die 52 Klein- und Großbuchstaben des lateinischen Alphabets, die zehn arabischen Ziffernzeichen, das Leerzeichen, ein speziellens Steuerzeichen, das für Lochkartenausgabe benötigt wurde, und die Zeichen `!"#$%&'()*+,-./:;<=>?@[\]^_‘{|}~`. Diese Zeichendarstellung und die dahinter stehende Zuordnung von Zeichen zu Zahlen zwischen 0 und 127 wird als ASCII-Code bezeichnet und wurde 1968 standardisiert.
Mit diesem Zeichensatz waren offensichtlich keine deutschen Umlaute darstellbar – ganz zu schweigen von Zeichen aus Sprachen, die nicht das lateinische Alphabet verwenden sondern beispielsweise das kyrillische.
Für die deutschen Umlaute gab es bald eine Lösung: man erweiterte die Zeichendarstellung auf acht Bit, also 256 darstellbare Zeichen. Auch für viele andere Zeichensätze wurde dies getan. Meist blieben dabei die ersten 128 Zeichen unverändert, die weiteren 128 waren aber keineswegs einheitlich. Das ist bis heute der Grund, warum man beim Öffnen einer Textdatei auf einem Computer immer wissen muss, welche Zuordnung von Zeichen zu Bitmuster (so genannte „Code-Page“) beim Abspeichern verwendet wurde.
Ein wenig verzwickter macht die Sache noch, dass verschiedene Betriebssysteme in der Standardeinstellung jeweils eigene, nicht kompatible Code-Pages verwendeten und teilweise bis heute noch verwenden. Es ist prinzipiell unmöglich, technisch die bei der Erstellung einer reinen Textdatei verwendete Code-Page festzustellen.[13]
Erst nach und nach etablierte die ISO einen allgemeinen Standard, der als ISO 8859 bezeichnet wird. Bis heute wurden sechzehn Code-Pages standardisiert (ISO 8859-1 bis ISO 8859-16), die jeweils durch eine 8-Bit-Darstellung dargestellt werden können und verschiedenen Sprachregionen zugeordnet sind:

[13] Es gibt in neuerer Zeit Versuche, dies über eine statistische Analyse des Inhalts zu bewerkstelligen.

Standard	weiterer Name	inputenc	Alphabet, Sprachregion
ISO 8859-1	Latin-1	latin1	lateinisch, Westeuropa
ISO 8859-2	Latin-2	latin2	lateinisch, Osteuropa
ISO 8859-3	Latin-3	latin3	lateinisch, Südeuropa
ISO 8859-4	Latin-4	latin4	lateinisch, Baltikum
ISO 8859-5			kyrillisch
ISO 8859-6			arbisch
ISO 8859-7			grichisch
ISO 8859-8		8859-8	hebräisch
ISO 8859-9	Latin-5	latin5	lateinisch, Türkei
ISO 8859-10	Latin-6		lateinisch, Nordeuropa
ISO 8859-11			Thai
ISO 8859-13	Latin-7		lateinisch, Baltikum
ISO 8859-14	Latin-8		lateinisch, keltische Sprachen
ISO 8859-15	Latin-9	latin9	lateinisch, Westeuropa
ISO 8859-16	Latin-10		lateinisch, Südosteuropa

Die Code-Pages Latin-1 bis Latin-10 enthalten alle die deutschen Umlaute und das „ß“, bieten sich also alle für deutsche Texte an. Aus praktischen Gründen empfehlen sich aber insbesondere Latin-1 und Latin-9:

A.2. Latin-1 und Latin-9

Latin-9 unterscheidet sich nur in acht Zeichen von Latin-1:

Latin-1	¤	¦	¨	´	¸	¼	½	¾
Latin-9	€	Š	š	Ž	ž	Œ	œ	Ÿ

Latin-1 ist heute die am weitesten verbreitete Code-Page der ISO-Familie. Da allerdings die Zeichen, die beim Übergang zu Latin-9 entfallen, im Gegensatz zu den neu hinzukommenden Zeichen wohl kaum genutzt werden, sollte Latin-9 verwendet werden.
Dabei gibt es im Zusammenspiel mit LaTeX und dem *inputenc*-Paket allerdings noch ein kleines Problem: Zur Darstellung vieler Zeichen, die in den über die ersten 128 Zeichen hinaus erweiterten Zeichensätzen enthalten sind, wird das Paket *textcomp* benötigt. Dieses Paket stellt das Euro-Symbol als Kombination des großen „C“s und des Gleichheitszeichens dar: „€“. Dies Darstellung sieht – gelinde gesagt – beschissen aus. Deshalb ist die Verwendung des Paketes *marvosym* dringend angeraten. Dort sieht das Eurozeichen, das man über den Befehl `\EUR` bekommt, so aus: „€“. Möchte man dieses Eurozeichen direkt über die Taste auf der Tastatur eingeben können, muss in den Dokumentenkopf folgender Befehl aufgenommen werden: `\renewcommand{\texteuro}{\EUR}`.

A.3. Der große Wurf – Unicode

Selbst die Zeichendarstellung über standardisierte Code-Pages war auf Dauer nicht befriedigend, denn sie hatte einige gravierende Nachteile: Hat man sich in einem Dokument einmal für eine Code-Page entschieden, so ist es nicht ohne

weiteres möglich, im Verlauf des Dokumentes zu einer anderen Code-page zu wechseln. Deutscher (mit Umlauten) und arabischer Text zusammen in einer Datei beispielsweise geht nicht.
Zusätzlich sind viele Sprachen bisher nicht in den ISO-Code-Pages berücksichtigt. Es fehlen z. B. afrikanische, japanische und chinesische Zeichen. (Für diese Sprachen gibt es teilweise andere Code-Pages, allerdings benötigen sie normalerweise mehr als 256 darstellbare Zeichen, also mehr als acht Bit.)
Als neuer Ansatz zur Zeichendarstellung in Computern wurde 1991 die Version 1.0.0 des Unicode-Standards vorgestellt. Unicode [8] erweiterte die Zahl der möglichen kodierbaren Zeichen erheblich von 256 im ISO-Standard auf zunächst 65536 (2 Byte), mit Version 2.0.0 dann auf 1.114.112 (dargestellt durch 4 Byte).[14]

Falls jedes Zeichen in einer Unicode-Textdatei durch vier Byte dargestellt würde, würde sie vier Mal so groß sein als eine Datei gleichen Inhalts, die mit Latin-9 o. ä. kodiert wäre. Deshalb wurde eine Kodierung für Unicode entwickelt, die sich dem jeweiligen Zeichen anpasst – das „8-Bit Unicode Transformation Format“, kurz *UTF-8*.[15] Jedes Zeichen wird mit UTF-8 abhängig von dem Bereich, in dem es in Unicode vorkommt, kodiert. Kommt es aus dem Zeichenbereich der ASCII-Kodierung (wird es also durch eine Zahl zwischen 0 und 127 kodiert), wird es durch ein Byte dargestellt. Die nächsten 1920 Zeichen werden durch zwei Byte dargestellt usw.

A.4. LaTeX und Unicode

Wie mit Latin-1 oder Latin-9 so kann *inputenc* auch mit der UTF-8-Darstellung des Unicode-Zeichensatzes umgehen. Allerdings gilt es dabei noch einige Hürden zu überwinden. Insbesondere was die Darstellung und Verarbeitung von UTF-8 in Editoren anbelangt, gibt es noch große Probleme. Deshalb haben wir zum jetzigen Zeitpunkt darauf verzichtet, in diesem Kurs die UTF-8-Kodierung zu verwenden. Die Umstellung ist aber geplant, denn LaTeX verträgt sich in jedem Fall mit Unicode, so ist z. B. dieses Dokument bereits in UTF-8-Kodierung erstellt. Für alle Interessierten: UTF-8 nutzt man prinzipiell mittels `\usepackage[utf8]{inputenc}`.[16]

A.5. Zusammenfassung

Wir wollen abschließend eine kurze stichpunktartige Zusammenfassung dessen, was in den letzten Abschnitten zur Kodierung angeführt wurde, geben. Diese Zusammenfassung kann auch als Leitfaden zum Umgang mit deutschen Umlauten und Sonderzeichen dienen.

[14] In Unicode Version 4.0.0 werden von den möglichen 1.114.112 Kodierungen tatsächlich 96.382, also etwas weniger als 9%, verwendet.
[15] UTF-8 wird am häufigsten verwendet und ist die einzige UTF-Kodierung, die LaTeX kennt. Es gibt aber auch noch UTF-7, UTF-16 und UTF-32.
[16] Leider sind die von *inputenc* für UTF-8 benötigten Dateien (noch) nicht auf den Rechnern in Phyma und Mittelerde installiert.

- Wird im Dokumentenkopf das Paket *inputenc* geladen, können deutsche Umlaute und Sonderzeichen direkt mit den dafür vorgesehenen Tasten der Tastatur eingegeben werden.
- Wird das Paket *inputenc* geladen, so muss ihm die bei der Erstellung der Quelltextdatei vom Texteditor verwendete Kodierung als Option übergeben werden.
- Für deutschen Text empfiehlt es sich, die Kodierung Latin-9 (`\usepackage[latin9]{inputenc}`) zu verwenden. Eine Lösung für das Problem bei der direkten Eingabe des Eurozeichens ist in Anhang A.2 beschrieben.
- Wird das Paket *inputenc* nicht verwendet, jedoch das Paket *ngerman* (oder *german*), können die in Abschnitt 4.2.3 vorgestellten Abkürzungen für deutsche Umlaute und Sonderzeichen verwendet werden. Andernfalls müssen die im selben Abschnitt vorgestellten TEX-Befehle benutzt werden.[17]
- Experimentierfreudige Mitmenschen sollten über den Umstieg auf Unicode und UTF-8 nachdenken. Allerdings sind so erstellte Quelltextdateien (noch) *nicht auf allen TEX-Installationen* (insbesondere nicht in Phyma und Mittelerde) compilierbar.
- Für Umlaute und Sonderzeichen anderer Sprachen gilt prinzipiell dasselbe. Auf spezielle Probleme können wir hier jedoch nicht näher eingehen.

B. Größenangaben in TEX

Größen können in TEX als positive oder negative Zahlen in Verbindung mit einer Größeneinheit angegeben werden. Die wichtigsten Größeneinheiten sind im Folgenden aufgeführt. Sollen Zahlen mit Nachkommastellen angegeben werden, so muss statt einem Komma ein Punkt als Dezimaltrennzeichen verwendet werden.

B.1. Absolute Einheiten

Einheit	in TEX	Bemerkungen
Millimeter	`mm`	
Zentimeter	`cm`	natürlich gilt 1 cm = 10 mm
Zoll (Inch)	`in`	1 in = 25.4 mm
Punkte (Point)	`pt`	1 pt = $\frac{1}{72{,}27}$ in $\approx 0,35$ mm

[17] Viele Editoren lassen sich so konfigurieren, dass sie beim Druck auf eine der Tasten für die deutschen Umlaute und Sonderzeichen direkt die Abkürzung des Paketes *ngerman* für das jeweilige Zeichen in den Quelltext einfügen. Wie diese Konfiguration für Emacs geht, ist im Anhang des Skriptes zum Vorkurs beschrieben. Wir empfehlen jedoch dringend die Verwendung von *inputenc* und die direkte Eingabe der Zeichen.

B.2. Relative Einheiten

Relative Größeneinheiten beziehen sich immer auf die gerade gewählte Schriftgröße.

in TeX	Bemerkungen
`em`	Ungefähr die Breite des „M“
`ex`	Ungeführ die Höhe des „x“

C. pdfLaTeX aufrufen und steuern

pdfLaTeX wird auf den Rechnern in Phyma und Mittelerde über den Befehl **pdflatex [Quelltextdatei]** aufgerufen. Standardmäßig speichert es alle Dateien, die es erstellt, immer in das aktuelle Arbeitsverzeichnis des Benutzers.
Neben der eigentlichen Dokument-Datei, die auf *'.pdf'* endet, erstellt pdfLaTeX weitere Dateien, die denselben Namen wie die Quelltextdatei haben, allerdings ohne die Endung *'.tex'*:

- Eine Datei, die auf *'.toc'* endet und die Informationen enthält, die zur Erstellung des Inhaltsverzeichnisses benötigt werden.
- Dieselbe Funktion für Referenzen, die im Text vorkommen (z. B. Verweise auf frühere oder spätere Abschnitte, Literaturangaben), übernimmt die Datei, die auf *.aux* endet.
- In die Datei, die auf *'.log'* endet, speichert pdfLaTeX ausführlich, welche Bearbeitungsschritte es vornimmt.
- (Andere Dokumentenklassen können selbstständig weitere benötigte Dateien anlegen.)

Stößt pdfLaTeX bei der Verarbeitung einer Quelldatei auf einen Fehler, unterbricht es und gibt eine Meldung aus. Trifft es z. B. auf einen unbekannten Befehl, kann die Warnung folgendermaßen aussehen:

```
! Undefined control sequence.
l.1097 speichert pdf\LaTe
                          ausführlich, welche Bearbeitungsschritte es vorni...

?
```

Das Ausrufezeichen zeigt an, dass ein Fehler aufgetreten ist. Danach folgt eine fehlerspezifische Meldung, die je nach Fehler mehr oder weniger hilfreich ist. In der folgenden Zeile gibt pdfLaTeX die Zeilennummer der Zeile in der Quelltextdatei an, in der der Fehler aufgetreten ist, dann folgt ein Ausschnitt aus der Quelltextdatei bis zu der Stelle, an der der Fehler auftrat. In der nächsten Zeile kommt ein Ausschnitt aus der Quelltextdatei nach der Stelle, an der der Fehler auftrat.
Das Fragezeichen in der letzten Zeile zeigt an, das pdfLaTeX eine Eingabe des Benutzers erwartet. Folgende Eingaben sind möglich:

? gibt einen kurzen Überblick über die möglichen Eingaben.

<*Return*> geht über den Fehler hinweg und fährt mit der Verarbeitung fort. (In den allermeisten Fällen wird das so erstellte Dokument nicht aussehen, wie es sollte.)

S wie <*Return*>, nur dass pdfLaTeX nicht mehr unterbricht, wenn es weitere Fehler findet.

I um Text einzugeben, den pdfLaTeX anstelle des fehlerhaften Textes verarbeitet.

eine Zahl zwischen 0 und 100 pdfLaTeX überspringt die angegebene Anzahl an Zeichen in der Datei, bevor es mit der Verarbeitung fortfährt.

X bricht die Verarbeitung der Datei ab.

H pdfLaTeX gibt einen ausführlicheren Hilfetext aus und fragt dann erneut, was es als nächstes tun soll.

Zeigt pdfLaTeX penetrant einen Fehler an, der nicht nachvollziehbar ist, hilft es häufig, die von pdfLaTeX angelegten *'.aux'*- und *'.toc'*-Dateien zu löschen.

Die Verweise im Text beziehen sich auf die unter „Quellen aus dem Text“ angegebenen Quellen.

Die Homepage zum Kurs

[1] Die Homepage zum LaTeX-Kurs der Fachschaft Physik mit einer Downloadmöglichkeit für die Skripte zu den Kursteilen: http://fachschaft.physik.uni-konstanz.de/fachschaft/aktionen/latex/

Quellen aus dem Text

[1] Fachschaft Physik: http://fachschaft.physik.uni-konstanz.de

[2] Universität Konstanz: http://www.uni-konstanz.de

[3] Homepage von Donald E. Knuth: http://www-cs-faculty.stanford.edu/~knuth/

[4] Donald E. Knuth: *The TeXbook*, Addison-Wesley, sechste Auflage von 1986, ISBN 0-201-13448-9. In der Bibliothek unter lbs 848/k69 oder kid 819/k69.

[5] Homepage von Adobe: http://www.adobe.de

[6] Den kostenlosen Acrobat Reader von Adobe gibt es hier: http://www.adobe.de/products/acrobat/readstep2.html

[7] Hàn Thế Thành: *Micro-typographic extensions to the TeX typesetting system*, Dissertation, Brünn, Oktober 2000, als PDF-Dokument im Internet unter http://www.pragma-ade.com/pdftex/thesis.pdf

[8] Homepage des gemeinnützigen Unicode Consortium, das sich um den Unicode-Standard kümmert: http://www.unicode.org

---

Weiterführende Literatur

[1] Walter Schmidt, Jörg Knappen, Hubert Partl, Irene Hyna: *LaTeX 2ε-Kurzbeschreibung*, erhältlich als PDF-Datei unter http://www.ctan.org/tex-archive/info/lshort/german/l2kurz.pdf.

[2] Tobias Oetiker, Hubert Partl, Irene Hyna, Elisabeth Schlegl: *The Not So Short Introduction to LaTeX 2ε*, erhältlich als PDF-Datei unter http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf.

[3] Helmut Kopka: *LaTeX, Band 1: Einführung*, Addison-Wesley, dritte Auflage von 2002, ISBN 3-827-37038-8. In der Bibliothek unter kid 819/k66 und praktisch in jeder Physikarbeitsgruppe ;-).

[4] Michael Goossens, Frank Mittelbach, Alexander Samarin: *The LaTeX-Companion*, Addison-Wesley, Second Edition 2004, ISBN 0-201-36299-6. In der Bibliothek unter kid 819:la92/u04 oder lbs 848/g66.

[5] Michael Goossens, Frank Mittelbach, Alexander Samarin: *Der LaTeX-Begleiter*, Addison-Wesley, Nachdruck der ersten Auflage 2002, ISBN 3-827-37044-2 (deutsche Übersetzung von [4]). In der Bibliothek unter kid 819/g66 oder lbs 848/g66.

[6] Leslie Lamport: *LaTeX*, Addison-Wesley, Second Edition 1995, ISBN 0-201-52983-1. In der Bibliothek unter kid 819/l15 oder lbs 848/l15a.

[7] Leslie Lamport: *Das LaTeX-Handbuch*, Addison-Wesley, dritte Auflage 1995, ISBN 3-893-19826-1 (deutsche Übersetzung von [6]). In der Bibliothek unter kid 819/l15b oder lbs 848/l15b.

[8] LaTeX3-Project-Team: *LaTeX 2ε for authors*, erhältlich als PDF-Datei unter http://www.ctan.org/tex-archive/macros/latex/doc/usrguide.pdf