KAPITEL 3

jetzt lerne ich

Strukturen

In diesem Kapitel geht es nun langsam los. Ich beschreibe den grundsätzlichen Aufbau einer LaTeX-Datei. Eine `.tex`-Datei als Ausgangsbasis für alle weiteren Schritte folgt einer gewissen Struktur mit Befehlen und Umgebungen, die hier beschrieben wird. Das grundsätzliche Erscheinungsbild eines Dokuments wird durch Dokumentklassen und Pakete bestimmt.

Um aus einer LaTeX-Eingabedatei eine druckfertige Ausgabe zu erhalten, muss man ein oder mehrere Programme aufrufen. Dabei entstehen neben der zu druckenden Datei auch noch mehrere weitere Dateien, die für die Verarbeitung wichtig sind. In Abschnitt 3.3 auf Seite 52 werde ich in einem kurzen Überblick zunächst zeigen, wie aus der LaTeX-Datei eine PS- oder PDF-Datei wird. Die bei der Verarbeitung genutzten und entstehenden wichtigsten Dateien sind in Abschnitt 3.4 auf Seite 57 aufgeführt.

3.1 Eine LaTeX-Datei im Überblick

Wie sieht denn nun eine LaTeX-Datei aus? In Abschnitt 1.4 auf Seite 25 habe ich schon einmal ein kurzes Beispiel gezeigt. Nun will ich anhand einer etwas umfangreicheren Version einen ersten Überblick geben, der dann in den folgenden Abschnitten und Kapiteln vertieft wird. Das folgende Beispiellisting 3.1 auf der nächsten Seite enthält viele Elemente, die regelmäßig in LaTeX verwendet werden. Ich will die einzelnen Komponenten jeweils kurz vorstellen, damit man schon einmal einen Überblick bekommt. Später wird dann natürlich genauer darauf eingegangen. Lässt

man die Beispieldatei durch LaTeX laufen, erhält man eine Ausgabe wie in Abbildung 3.1 auf der nächsten Seite.

Listing 3.1: Eine kurze LaTeX-Beispieldatei

```
\documentclass[a4paper]{article}
\usepackage[ngerman]{babel}
\begin{document}
  % Der Titelbereich
  \title{Ein \LaTeX{}-Beispiel}
  \author{Thomas Demmig\thanks{Mannheim}}
  \date{\today}
  \maketitle

  % Inhaltsverzeichnis
  \tableofcontents

  % Der eigentliche Text
  \section{Einleitung}
  F"ur einen Beispieltext ist es immer schwierig,
  sich eine Einleitung auszudenken. Aber der geneigte
  Leser mag sich selbst seinen Teil dazu denken.

  \section{Hauptteil}
  Im Hauptteil kann man sich daf"ur besser austoben.
  Hier kann man viele tolle Sachen schreiben.

  \subsection{Mathematik}
  \LaTeX{} ist ber"uhmt f"ur seinen Formelsatz. Man
  kann mitten im Text Formeln schreiben: $E = mc^2$,
  aber auch abgesetzte Formeln mit \emph{automatischer
  Nummerierung} einbauen.
  \begin{equation}
    \sum_{i=1}^n n = \frac{n(n+1)}{2}
    \qquad
    F(\omega) = \int_{-\infty}^{+\infty}\limits
    f(t)e^{-i\omega t}\,\mbox{d}t
  \end{equation}
\end{document}
```

Ein LaTeX-Beispiel

Thomas Demmig*

7. Juli 2003

Inhaltsverzeichnis

1 Einleitung

Für einen Beispieltext ist es immer schwierig, sich eine Einleitung auszudenken. Aber der geneigte Leser mag sich selbst seinen Teil dazu denken.

2 Hauptteil

Im Hauptteil kann man sich dafür besser austoben. Hier kann man viele tolle Sachen schreiben.

2.1 Mathematik

LaTeX ist berühmt für seinen Formelsatz. Man kann mitten im Text Formeln schreiben: $E = mc^2$, aber auch abgesetzte Formeln mit *automatischer Nummerierung* einbauen.

$$\sum_{i=1}^{n} n = \frac{n(n+1)}{2} \qquad F(\omega) = \int_{-\infty}^{+\infty} f(t) e^{-i\omega t}\,\mathrm{d}t \tag{1}$$

*Mannheim

Abb. 3.1: Ergebnis einer LaTeX-Datei

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

Die meisten LaTeX-Dateien beginnen mit dem Befehl `\documentclass`. Durch ihn wird der prinzipielle Aufbau und das Aussehen des Textes festgelegt. Als Parameter übergibt man ihm die gewünschte Dokumentklasse (`article`) und optionale Extra-Parameter, wie hier `a4paper`, mit dem die Papiergröße auf DIN A4 festgelegt wird. Befehle beginnen meist mit einem Backslash „\“, ihre Parameter stehen entweder in geschweiften Klammern „{…}“ oder – falls optional – in eckigen Klammern „[…]“.

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

In der zweiten Zeile wird mit dem Befehl `\usepackage` das Paket `babel` mit der Option `ngerman` geladen. Dieses Paket erweitert LaTeX um Mehrsprachigkeit, wobei mit der hier genutzten Option ein deutschsprachiger Text (in neuer Rechtschreibung – daher das „n“) beschrieben wird. Dadurch werden verschiedene Begriffe in LaTeX („Inhaltsverzeichnis“, „Kapitel“, …) eingedeutscht, die deutsche Silbentrennung aktiviert und die Eingabe von Umlauten vereinfacht.

```
\begin{document}
...
\end{document}
```

Nun geht es mit dem eigentlichen Text los. Mit `\begin` wird eine Umgebung eingeleitet, die später mit `\end` wieder endet. Solche Umgebungen dienen allen möglichen Zwecken, zum Beispiel einer Aufzählung. Hier geht es um die wichtigste Umgebung, `document`, denn in dieser findet sich der eigentliche Text.

```
% Der Titelbereich
\title{Ein \LaTeX{}-Beispiel}
\author{Thomas Demmig\thanks{Mannheim}}
\date{\today}
\maketitle
```

Mit diesen Zeilen wird der Titel des Textes definiert. Dabei geht es nicht nur um die Überschrift, sondern auch um den Autor und das Datum. Zunächst soll aber noch auf die erste Zeile dieses Blocks hingewiesen werden. Sie beginnt mit einem Prozentzeichen „%“. Alles, was nach einem Prozentzeichen kommt, ist ein Kommentar und wird von LaTeX bis zum Ende der Zeile ignoriert.

Mit `\title` wird der eigentliche Dokumenttitel festgelegt. Dieser lautet „Ein LaTeX-Beispiel“. Um die hoch- und tiefgestellten Zeichen von „LaTeX“ zu erhalten, nutzt man den Befehl `\LaTeX`. Mit `\author` wird der Autor des Dokuments angegeben, wobei man mit `\thanks` in einer Fußnote weitere

Angaben machen kann. Mittels `\date` wird das Datum angegeben, wobei man mit `\today` das Datum erhält, zu dem das Dokument mit LaTeX gesetzt wurde.

Schließlich folgt nun der eigentliche Befehl zum Generieren des Titelbereichs: `\maketitle`. Erst durch ihn werden die vorher gemachten Angaben auch umgesetzt.

```
% Inhaltsverzeichnis
\tableofcontents
```

Mit dem Befehl `\tableofcontents` wird ein Inhaltsverzeichnis ausgegeben.

```
% Der eigentliche Text
\section{Einleitung}
```

`\section` legt eine Überschrift fest. Es gibt noch weitere Befehle aus diesem Bereich, wie zum Beispiel `\subsection` und `\chapter`.

```
F"ur einen Beispieltext ist es immer schwierig,
sich eine Einleitung auszudenken. Aber der geneigte
Leser mag sich selbst seinen Teil dazu denken.

\section{Hauptteil}
Im Hauptteil kann man sich daf"ur besser austoben.
Hier kann man viele tolle Sachen schreiben.
```

Nun folgt endlich der eigentliche, normale Text. Man schreibt ihn ganz normal als ASCII-Text nieder. Zeilenumbrüche im Quelltext haben nichts mit den Zeilenumbrüchen im Endergebnis zu tun. Absätze werden durch eine Leerzeile kenntlich gemacht. Nutzt man wie in diesem Beispiel das Paket `babel` für deutsche Texte, kann man die Umlaute durch ein doppeltes Anführungszeichen kenntlich machen, also zum Beispiel `"a` für ein ä. Später werde ich auch noch eine Variante vorstellen, in der man „ganz natürlich“ normale Umlaute und Akzent-Zeichen schreiben kann.

```
\subsection{Mathematik}
\LaTeX{} ist ber"uhmt f"ur seinen Formelsatz. Man
kann mitten im Text Formeln schreiben: $E = mc^2$,
aber auch abgesetzte Formeln mit \emph{automatischer
Nummerierung} einbauen.
```

Unterhalb einer durch `\section` erzeugten Überschrift kann man mit `\subsection` einen Unterabschnitt anlegen. In diesem Absatz soll es aber eher um zwei andere, wichtige Dinge gehen:

- Einzelne Textabschnitte lassen sich hervorheben. Während man bei einer Schreibmaschine häufig Unterstreichungen genutzt hat, empfiehlt sich für einen „anständig“ gesetzten Text eine Hervorhebung durch Kursivschreibung. Diese erreicht man durch den Befehl `\emph`. Als Parameter wird der hervorzuhebende Text angegeben. Unterstreichungen sind jedoch auch möglich.
- Mathematische Formeln, die weithin bekannte Stärke von LaTeX, lassen sich entweder im Fließtext unterbringen (wie hier im Beispiel) oder als abgesetzte Formel (siehe nächstes Textstück). Eine Formel im Fließtext wird durch Dollarzeichen ($) begrenzt. Innerhalb einer solchen Formel gelten andere Regeln für Abstände. Außerdem werden Variablen kursiv geschrieben, kurz: Es wird alles dafür getan, eine Formel so aussehen zu lassen, wie sie nach typographischen Gesichtspunkten auszusehen hat.

Hier haben wir eine abgesetzte mathematische Formel in einer `equation`-Umgebung:

```
\begin{equation}
  \sum_{i=1}^n n = \frac{n(n+1)}{2}
  \qquad
  F(\omega) = \int_{-\infty}^{+\infty}\limits
              f(t)e^{-i\omega t}\,\mbox{d}t
\end{equation}
```

Auch bei diesen ist der Zeilenumbruch innerhalb des Quelltextes unerheblich. Auf den ersten Blick sieht das Ganze vielleicht verwirrend aus, aber mit wenigen Regeln erkennt man, worum es sich handelt. Dazu aber später mehr.

3.2 Aufbau einer LaTeX-Datei

Im vorigen Abschnitt haben wir uns im Schnelldurchlauf eine recht einfache LaTeX-Datei angeschaut. Im Folgenden wollen wir nun etwas ausführlicher auf die einzelnen Bestandteile eingehen.

3.2.1 Text

Dazu gibt es gar nicht so viel zu sagen. Man schreibt ihn einfach. Wie schon erwähnt sind Zeilenumbrüche im Ergebnis unabhängig von den Umbrüchen im Quelltext. Nur Absätze muss man kenntlich machen. Dies geschieht aber ganz einfach: Durch eine Leerzeile. Leerzeichen zwischen den einzelnen Worten werden natürlich berücksichtigt[1], aber man erreicht

1 Wäre auch schlecht, wenn nicht …

durch mehrere Leerzeichen hintereinander nicht mehr Abstand, was auch ganz gut so ist:

```
Dies ist ein Stück     Text.      Er sieht
etwas    seltsam aus, ist        aber nur mit
       ein paar Leerzeichen zu   viel
geschrieben.

Entscheidend sind   nur Leerzeilen, da sie die
    Absätze   in    einem Text   kenntlich
machen.
```

Das wird dann nämlich zu:

> Dies ist ein Stück Text. Er sieht etwas seltsam aus, ist aber nur mit ein paar Leerzeichen zu viel geschrieben.
>
> Entscheidend sind nur Leerzeilen, da sie die Absätze in einem Text kenntlich machen.

3.2.2 Spezielle Zeichen

In einer normalen LaTeX-Datei kann man normalerweise nur die folgenden Zeichen nutzen: `A...Z`, `a...z`, `0...9` und die Satzzeichen `. , ; : ? ! ´ ` ( ) [ ] - / * @`. Dazu kommen die Zeichen `+ = | < >`, die hauptsächlich in mathematischen Formeln genutzt werden. `+` und `=` lassen sich auch im normalen Text nutzen.

Einige Sonderzeichen werden von LaTeX als Teil eines Befehls angesehen und besonders behandelt. Sie müssen „umschrieben“ werden, wenn man sie ausgegeben haben möchte. Zu diesen Zeichen gehören `# $ % & ~ _ ^ \ { } "`. In Tabelle 3.1 ist aufgeführt, wie diese Zeichen eingegeben werden können.

Zeichen	Eingabe	Zeichen	Eingabe
#	\#	$	\$
%	\%	&	\&
~	\textasciitilde	_	_
^	\textasciicircum	\	\textbackslash
{	\{	}	\}
"	\dq	|	\textbar

Tabelle 3.1: Eingabe von Sonderzeichen

Alle weiteren Zeichen, also zum Beispiel Umlaute, akzentuierte Zeichen und ähnliches, müssen entweder umschrieben werden oder können nur dann direkt eingegeben werden, wenn mit dem Paket `inputenc` der entsprechende Zeichensatz festgelegt wurde. Auf solche Zeichen und die Möglichkeiten der Eingabe gehe ich in Kapitel 6 auf Seite 97 ein.

3.2.3 Befehle

Schon in den ersten Zeilen der Beispieldatei in Listing 1.1 auf Seite 26 lassen sich drei typische Eigenschaften eines Befehls zeigen:

- Befehle beginnen fast immer mit einem Backslash „\“.
- Benötigt ein Befehl weitere Angaben, werden diese in geschweiften Klammern „{…}“ angegeben.
- Optionale Parameter werden vor den notwendigen Parametern in eckigen Klammern „[…]“ festgelegt.

Allerdings gibt es noch eine ganze Reihe von weiteren Befehlen ohne Backslash, die meist aus einem Sonderzeichen bestehen und besondere Bedeutungen haben. Auf sie werde ich vor allem bei den Sonderzeichen und Akzenten eingehen. Einzig das Dollarzeichen ($) wird als Kennzeichnung für Formeln in Fließtexten verwendet.

Häufig wird Ihnen auch der Begriff *Makro* über den Weg laufen. Dies ist in TEX und LATEX ein Synonym für einen Befehl.

3.2.4 Umgebungen

Umgebungen dienen dazu, einen ganzen Bereich auszuzeichnen oder mit ihm besondere Dinge anzustellen. So kann man zum Beispiel durch die Umgebung `quotation` dafür sorgen, dass der darin befindliche Text beidseitig eingerückt wird.

Eine Umgebung wird immer durch den Befehl `\begin{`*`Umgebungsname`*`}` eingeleitet und mit `\end{`*`Umgebungsname`*`}` beendet. Alles zwischen diesen beiden Befehlen gehört dann zur Umgebung. So ergibt der folgende Quelltext

```
\begin{quotation}
  Dies ist eine Umgebung, in der der Text beidseitig
  eingerückt wird. Sie kann zum Beispiel dazu genutzt werden,
  Zitate in einem längeren Text kenntlich zu machen.
\end{quotation}
```

einen wie schon beschrieben beidseitig eingerückten Text:

> Dies ist eine Umgebung, in der der Text beidseitig eingerückt wird. Sie kann zum Beispiel dazu genutzt werden, Zitate in einem längeren Text kenntlich zu machen.

Umgebungen können – wie Befehle – auch Parameter besitzen, die man mitgeben muss. So wird zum Beispiel der Umgebung `tabular` gesagt, wie die einzelnen Spalten formatiert werden sollen (siehe dazu Abschnitt 7.2 auf Seite 111):

```
\begin{tabular}{lr}
  …
\end{tabular}
```

Auch mehrere Absätze können natürlich in einer Umgebung genutzt werden. In jedem Dokument kommt immer mindestens eine Umgebung vor: `\begin{document}` …`\end{document}`. Sie umfasst nach der Präambel den kompletten Text.

Es gibt eine ganze Reihe von Befehlen, die auch als Umgebung vorhanden sind, speziell bei der Auszeichnung von Textabschnitten, also zum Beispiel kursiv oder fett. Dies kann bei längeren Texten von Vorteil sein, weil dann leichter zu erkennen ist, wo eine Formatierung beginnt und endet.

Umgebungen können auch ineinander verschachtelt sein, aber sie dürfen sich nicht gegenseitig überlappen. Das heißt, eine Umgebung, die sich in einer anderen befindet, muss auch vor ihr beendet sein.

So geht es also nicht:

```
\begin{quotation}
Hier fängt ein Zitat an.
\begin{bfseries}
Das hier sollte fett sein.
\end{quotation}
Hier soll das Zitat zu Ende sein, aber fett weitergehen.
\end{bfseries}
```

Dagegen ist das folgende Vorgehen richtig:

```
\begin{quotation}
  Hier fängt ein Zitat an.
  \begin{bfseries}
    Das hier sollte fett sein.
  \end{bfseries}
  Und hier ist wieder normal formatiertes Zitat.
\end{quotation}
```

An dieser Stelle sei nochmals darauf hingewiesen, dass die Einrückungen und Zeilenumbrüche des obigen Beispiels natürlich dem Schreiber beim

Strukturieren helfen, aber auf das Verhalten von LaTeX keinen weiteren Einfluss haben. Außerdem fällt es mit Hilfe der Einrückungen leichter, zu sehen, wo begonnene Umgebungen nicht abgeschlossen sind. So werden alle drei Sätze direkt hintereinander geschrieben werden. Im Ergebnis erhält man dann das Folgende:

> Hier fängt ein Zitat an. **Das hier sollte fett sein.** Und hier ist wieder normal formatiertes Zitat.

3.2.5 Gruppen

Will man Teile eines Textes gesondert behandeln, muss man ihn in eine entsprechende Umgebung stecken oder in einer Gruppe zusammenfassen. Eine Gruppe beginnt mit einer öffnenden geschweiften Klammer { und endet an einer entsprechenden schließenden geschweiften Klammer }. Innerhalb einer solchen Gruppen kann man zum Beispiel die Schriftart oder -größe ändern und viele andere Dinge beeinflussen. Alle Änderungen enden mit der schließenden Klammer. Ebenso enden Änderungen innerhalb einer Umgebung mit dem `\end`-Befehl der Umgebung. So wird zum Beispiel aus

```
Innerhalb dieses Textes {\sffamily
ist ein Teil mit serifenloser Schrift} gesetzt.
```

der folgende Absatz:

> Innerhalb dieses Textes ist ein Teil mit serifenloser Schrift gesetzt.

3.2.6 Kommentare

Möchte man gewisse Bereiche eines Quelltextes nicht verarbeiten lassen, muss man ihn auskommentieren. Dazu dient das Prozentzeichen `%`. Alles, was innerhalb einer Zeile nach diesem Zeichen kommt, wird von LaTeX bei der Verarbeitung ignoriert.

So wird aus

```
In dieser Zeile gilt %alles, was man will
nur das, was vor dem Prozentzeichen steht.
```

das Folgende:

> In dieser Zeile gilt nur das, was vor dem Prozentzeichen steht.

Man muss bei der „normalen“ Nutzung des Prozentzeichens aufpassen, dass man nicht aus Versehen damit Teile einer Zeile auskommentiert. Es passiert leicht, dass man eigentlich schreiben will:

```
Der Anteil der erfolgreichen Messungen
betrug 96\,%. Von den
verbleibenden Messungen war die Hälfte
aufgrund äußerer Umstände gescheitert.
```

Das Ergebnis ist dann natürlich:

> Der Anteil der erfolgreichen Messungen betrug 96 verbleibenden Messungen war die Hälfte aufgrund äußerer Umstände gescheitert.

Um das gewünschte Ergebnis zu erhalten, muss man stattdessen im Quelltext das Folgende schreiben, da das Prozentzeichen maskiert werden muss:

```
Der Anteil der erfolgreichen Messungen
betrug 96\,\%. Von den
verbleibenden Messungen war die Hälfte
aufgrund äußerer Umstände gescheitert.
```

Dann erreicht man tatsächlich:

> Der Anteil der erfolgreichen Messungen betrug 96 %. Von den verbleibenden Messungen war die Hälfte aufgrund äußerer Umstände gescheitert.

3.2.7 Die Präambel – Dokumentklassen und Pakete

Allgemein bezeichnet man die Zeilen vor `\begin{document}` als Präambel. Hier werden neben der Dokumentklasse auch noch andere Festlegungen getroffen, die das gesamte Dokument betreffen. Man gibt zum Beispiel allgemeine Optionen und weitere Pakete an, die das Aussehen und Verhalten des Dokuments beschreiben.

Dokumentklassen und Pakete wirken sich auf den kompletten Text aus. Während die Klassen im Allgemeinen dazu dienen, das Aussehen des Textes festzulegen, nutzt man die Pakete meist, um weitere Optionen oder Variationen zu aktivieren.

Dokumentklassen

Mit dem Befehl `\documentclass{`*`Dokumentklasse`*`}` wird das grundlegende Aussehen und Verhalten des Dokuments festgelegt. Seine Syntax lautet:

```
\documentclass[Optionen]{Dokumentklasse}
```

Es gibt vier häufig genutzte Standarddokumentklassen und viele weitere für alle möglichen Bedürfnisse. Bei den vier Standarddokumentklassen handelt es sich um die folgenden:

`article` Die Klasse `article` ist für kürzere Texte gedacht, die keine allzu feine Untergliederung haben.

`report` Mit dieser Klasse kann man längere Texte schreiben, was aber natürlich auch mit `article` geht. Es gibt eine Überschriftenebene mehr als bei der vorigen Klasse und neue Kapitel beginnen auf einer neuen Seite.

`book` Handelt es sich bei Ihrem Text sogar um ein Buch, empfiehlt sich `book`. Hier kann man die Kapitel zusätzlich noch in Teilen zusammenfassen. Standardmäßig werden Texte in dieser Klasse zweiseitig gesetzt.

`letter` Mit Hilfe dieser Klasse lassen sich sehr rationell Briefe schreiben. Die Vorgehensweise mag zwar etwas ungewohnt sein, ist aber durchaus logisch und durchdacht.

Neben diesen vier Klassen gibt es noch weitere erwähnenswerte. Mit der Klasse `slides` lassen sich Folien für eine Präsentation erstellen. Die Standardklassen sind eher an amerikanischen Maßstäben ausgerichtet. Vor allem das Papierformat hat als Grundlage das Letter-Format. Es gibt die Möglichkeit, auch die DIN-Formate einzustellen (dazu komme ich gleich), aber vielen gefallen die Einstellungen für den Papierrand und anderes nicht, obwohl sie beliebig veränderbar sind. Als Alternative bieten sich dann die KOMA-Script-Klassen an, die in Kapitel 13.5 auf Seite 227 etwas genauer beleuchtet werden.

Dem Befehl `\documentclass` kann man weitere Optionen mitgeben. Diese werden noch vor der Angabe der Dokumentklasse in eckigen Klammern

und durch Kommata getrennt mitgeteilt. Diese Optionen wirken sich einerseits auf die Dokumentklasse aus, andererseits sind sie als „globale" Optionen auch für alle weiteren angegebenen Pakete (siehe Abschnitt „Pakete" auf der nächsten Seite) gültig, falls sie dort verwendet werden können. Die folgenden Optionen sind für die Standardklassen gültig:

`10pt`, `11pt`, `12pt` Diese drei Optionen legen die normale Schriftgröße im Dokument fest. Standardmäßig wird eine Schriftgröße von 10 Punkt[2] gewählt.

`a4paper`, `a5paper`, `b5paper`, `letterpaper`, `legalpaper`, `executivepaper` Mit diesen Optionen kann man die gewünschte Papiergröße angeben. Als Standard ist `letterpaper` eingestellt, denn schließlich kommt LaTeX aus den USA …

`oneside`, `twoside` Angabe, ob man einen ein- oder doppelseitigen Druck plant. Dies wirkt sich nicht auf die Druckeransteuerung aus, aber die Seitenränder erhalten andere Größen, und die Kopf- und Fußzeilen werden nach linken und rechten Seiten unterschieden. Während für die Dokumentklasse `book` die Option `twoside` als Standard eingestellt ist, nutzen die anderen Klassen `oneside`.

`onecolumn`, `twocolumn` Mit diesen Optionen kann man festlegen, ob das Dokument im ein- oder zweispaltigen Satz gedruckt werden soll. Beim zweispaltigen Satz gibt es einige Einschränkungen und seltsame Verhaltensweisen. Daher empfiehlt es sich eher, das Paket `multicol` zu nutzen, falls man mehrspaltig setzen möchte.

`openright`, `openany` Durch diese Optionen wird bestimmt, ob neue Kapitel immer auf einer rechten Seite beginnen müssen. Dies ist natürlich nur dann sinnvoll, wenn man auch mit der Option `twoside` oder der Dokumentklasse `book` einen doppelseitigen Druck ausgewählt hat. Während für `book` auch `openright` als Standard eingestellt ist, nutzt die Dokumentklasse `report` die Option `openany`.

`notitlepage`, `titlepage` Hiermit lässt sich einstellen, ob der Titelbereich und die Zusammenfassung auf einer eigenen Seite oder schlicht auf der ersten Seite direkt vor dem eigentlichen Text erscheinen sollen. Standard ist `titlepage`, außer für die Dokumentklasse `report`.

`final`, `draft` Der sehr komplexe Umbruch- und Silbentrennungsmechanismus findet in seltenen Fällen keine passende Stelle für ein „anständiges" Zeilenende. In solchen Fällen ragt das Ende der Zeile über

2 „Punkt" ist die übliche Größenangabe im Druck- und Schriftenbereich. Dabei entsprechen 72,27 Punkt einem Inch, was wiederum 2,54 cm sind. Also ist ein Punkt etwa 0,35 mm groß.

den rechten Rand hinaus. Nutzt man die Option `draft`, werden solche Zeilen mit einem schwarzen Balken am Rand gekennzeichet, um sie leichter auffinden zu können. Zudem wird `draft` auch vom Paket `graphicx` ausgewertet – es werden dann nur leere Bilderrahmen mit den Dateinamen der Grafiken dargestellt.

`landscape` Durch Verwendung dieser Option werden die Seiten im Querformat bedruckt.

`leqno` Formelnummern von abgesetzten Formeln werden auf der linken statt auf der rechten Seite der Formel gedruckt.

`fleqn` Abgesetzte Formeln werden linksbündig statt zentriert gesetzt.

`openbib` Ein Literaturverzeichnis (siehe Kapitel 18.1 auf Seite 313) wird im sogenannten „offenen Stil" formatiert. Standardmäßig wird der „komprimierte Stil" genutzt.

Dokumentklassen befinden sich in Dateien mit der Endung `.cls`. Die verschiedenen Optionen werden entweder direkt in diesen Klassendateien oder in speziellen Dateien mit der Endung `.clo` abgehandelt.

Pakete

Wie schon erwähnt, kann man die Funktionalität von LaTeX durch Pakete erweitern. In diesen befinden sich meist neue Befehle und Umgebungen, manchmal werden auch einfach vorhandene Befehle umdefiniert. Es gibt Pakete für alles und jeden Zweck. Mit ihnen kann man die grundlegenden Befehle zum Einbinden von Grafiken aktivieren (`graphics` oder `graphicx`) verschiedene Sprachen nutzen (`babel`), andere Fonts für den Text und die Formeln festlegen (zum Beispiel `mathpazo` oder `mathptmx` für Palatino beziehungsweise Times), chemische Strukturformeln aufbauen (`chemtex`) und vieles mehr erreichen.

Aktiviert werden Pakete durch den Befehl `\usepackage` in der Präambel des Dokuments. Auch seine Syntax ist recht übersichtlich:

```
\usepackage[Optionen]{Paketnamen}
```

Es ist möglich, durch Kommata getrennt, mehrere Pakete anzugeben. So ist zum Beispiel die folgende Angabe erlaubt:

```
\usepackage{graphics,color,alltt}
```

Als optionales Argument lassen sich auch hier Optionen angeben, die dann an alle Pakete weitergereicht werden, die aufgezählt sind. Zudem

werden alle Optionen, die bei der Dokumentklasse angegeben sind (*globale Optionen*) an alle aufgeführten Pakete übermittelt. Gibt man eine Option an, die von keiner Dokumentklasse und auch von keinem Paket verwendet wird, erhält man beim Durchlauf mit LaTeX eine Warnung.

Pakete sind in Dateien mit der Endung `.sty` definiert. Es gibt diverse Pakete, aber einige werden besonders häufig verwendet:

`babel` Dieses Paket aktiviert die Unterstützung für Fremdsprachen. Standardmäßig ist LaTeX auf Englisch ausgerichtet. Mit `babel` kann man auf eine sprachabhängige Silbentrennung, vereinfachte Eingabe von Sonderzeichen wie Umlaute oder Akzente, Übersetzungen von generierten Texten („Kapitel“, „Abbildung“, „Inhaltsverzeichnis“) und gesonderte Satzregeln zurückgreifen.

Als Optionen gibt man die Sprache(n) an, die im Dokument verwendet werden sollen. Die letzte angegebene Sprache wird als Standard ausgewählt. Mit dem Befehl `\selectlanguage{`*Sprache*`}` kann man im Dokument dann auf eine der vorher gewählten Sprachen umschalten. In Kapitel 6 auf Seite 97 gehe ich ausführlicher auf die Besonderheiten bei deutschsprachigen Dokumenten ein.

`inputenc` Mit diesem Paket kann man die Eingabe von Sonderzeichen wie Umlauten und Akzenten stark vereinfachen. Hat man keinerlei Unterstützung für deutschsprachige Dokumente aktiviert, müssen die deutschen Sonderzeichen ä, ö, ü und ß als `\"{a}`, `\"{o}`, `\"{u}` und `\ss` eingegeben werden. Mit `\usepackage[ngerman]{babel}` vereinfacht sich das zu `"a`, `"o`, `"u`, `"s`. Durch Verwendung des Pakets `inputenc` kann man nun direkt die Sonderzeichen verwenden, man gibt einfach `ä`, `ö`, `ü` und `ß` ein. Dabei muss man nur als Option angeben, mit welchem Zeichensatz man arbeitet (siehe auch Abschnitt 6.1 auf Seite 97).

`fontenc` Mit Hilfe dieses Pakets kann man Zeichensätze nutzen, die einen anderen Aufbau als die „klassischen“ TeX-Zeichensätze haben. Dies ist dann von Vorteil, wenn man Texte mit Umlauten oder akzentuierten Zeichen schreiben möchte. In einem solchen Fall würden mit den alten Zeichensätzen Wörter mit „Sonderzeichen“ nicht immer vernünftig getrennt werden.[3] Nutzt man die modernere T1-Kodierung, wie sie PostScript-Zeichensätze oder die em-Schriften bieten (siehe dazu mehr im Kapitel 13.1 auf Seite 212), tritt dieses Problem nicht auf, und die Silbentrennung funktioniert auch mit „europäischen“ Buchstaben. Diesem Zweck dient das Paket `fontenc`, das Zeichensätze mit der als Option angegebenen Kodierung aktiviert. Für den europäischen Sprachraum empfiehlt sich also die Verwendung von `\usepackage[T1]{fontenc}`.

3 LaTeX muss dabei die Sonderzeichen aus mehreren Elementen zusammensetzen, was den Trennalgorithmus durcheinander bringt.

`graphicx` Durch das Einbinden dieses Pakets kann man die verschiedenen Grafikformate verwenden. So lassen sich in Abhängigkeit vom Grafiktreiber EPS- und PDF-Dateien, sowie als Pixelgrafiken TIFF-, JPEG-, PNG- und BMP-Dateien einbinden. Mittels verschiedener Optionen beim Hauptbefehl `\includegraphics` kann man die Grafiken drehen, skalieren und beschneiden.

`color` Dieses Paket basiert auf der gleichen Schnittstelle wie `graphicx` und ermöglicht die Verwendung von Farben für Texte, Boxen und Hintergründe.

`amsmath`, `amssymb` Mit Hilfe dieser Pakete werden die Möglichkeiten zum mathematischen Formelsatz immens erweitert. Grundlage dafür war das alte $\mathcal{AMS}$-TEX der American Math Society, welches in LATEX integriert wurde.

`hyperref` Durch dieses Paket erhält man automatisch anklickbare Verweise, die sich besonders bei der Verwendung von PDF als Ausgabeformat gut nutzen lassen.

`longtable` Um Tabellen auch komfortabel über mehrere Seiten setzen zu können, sollte man auf dieses Paket zurückgreifen.

`makeidx` Zum Erstellen eines Indexes kann man durch dieses Paket eine umfassende Unterstützung aktivieren.

`multicol` Möchte man Text setzen, der in mehr als zwei Spalten pro Seite verlaufen soll, nutzt man das Paket `multicol`, das allerdings auch für eigentlich direkt von den Dokumentklassen angebotenen zweispaltigen Satz Vorteile bietet.

Nachdem wir nun die grundlegenden Elemente einer LATEX-Datei besprochen haben, soll es jetzt um den Weg von der LATEX-Quelldatei bis hin zur druckbaren Ausgabe gehen.

3.3 Vom Text zum Dokument – ein Durchlauf

Um eine LATEX-Quelldatei auf einen Drucker zu bringen oder in eine Form umzuwandeln, die lesbar ist, wie zum Beispiel eine PostScript- oder PDF-Datei, sind verschiedene Schritte notwendig.

Zunächst einmal muss man seine `.tex`-Datei mit LATEX verarbeiten. Der Text in Listing 3.1 auf Seite 38 sei in der Datei `beispiel01.tex` abgespeichert. Dann lautet der erste Schritt:

```
prompt> latex beispiel01
```

Beachtenswert ist hier, dass man die Endung `.tex` nicht mit angeben muss. Während der Verarbeitung erhält man die folgende oder eine ähnliche Ausgabe:

```
This is TeX, Version 3.14159 (Web2c 7.3.7x)
(./beispiel01.tex
LaTeX2e <2001/06/01>
Loading CZ hyphenation patterns: Pavel Sevecek, v3, 1995
Loading SK hyphenation patterns: Jana Chlebikova, 1992
Babel <v3.7h> and hyphenation patterns for english, french,
german, ngerman, catalan, croatian, czech, danish, dutch,
estonian, finnish, greek, hungarian, italian, latin,
mongolian, norwegian, polish, portuguese, romanian, russian,
slovene, serbocroat, slovak, spanish, swedish, ukenglish,
ukrainian, welsh, dumylang, nohyphenation, loaded.
(c:/TeXLive/texmf/tex/latex/base/article.cls
Document Class: article 2001/04/21 v1.4e Standard LaTeX
document class
(c:/TeXLive/texmf/tex/latex/base/size10.clo))
(c:/TeXLive/texmf/tex/generic/babel/babel.sty
(c:/TeXLive/texmf/tex/generic/babel/ngermanb.ldf
(c:/TeXLive/texmf/tex/generic/babel/babel.def)))
No file beispiel01.aux.
No file beispiel01.toc.
[1] (./beispiel01.aux) )
Output written on beispiel01.dvi (1 page, 1968 bytes).
Transcript written on beispiel01.log.
```

Man sieht hier, wie zunächst die verwendete TEX-Version ausgegeben wird. Dann folgt die zu verarbeitende Datei. Nach einer etwas länglichen Angabe der möglichen Sprach-Trennmuster folgt der Verweis auf die Dokumentklasse `article.cls`, die verwendete Option `size10.clo`[4] und das Paket `babel.sty` mit der Option `ngermanb.ldf`[5]. Dann wird darauf hingewiesen, dass die Dateien `beispiel01.aux` und `beispiel01.toc` noch nicht existieren. Das bedeutet, dass das Inhaltsverzeichnis und die Querverweise erst noch erstellt werden müssen.

Nun folgt endlich die Verarbeitung der ersten Seite. Jede erstellte Seite wird in eckigen Klammern angegeben. Dazu sieht man, dass nun die Datei `beispiel01.aux` geschrieben wird.

Schließlich folgt eine Zusammenfassung der Ausgabe: Der Name der Datei, in die geschrieben wurde (`beispiel01.dvi`) und die Anzahl der Seiten sowie die Größe der Ausgabedatei. Ganz zum Schluss folgt noch der Hinweis, dass eine Log-Datei in `beispiel01.log` geschrieben wurde. Darin

4 Die Option `10pt` ist hier zwar nicht angegeben, wird aber als Standard geladen.
5 Eigentlich hatten wir ja die Option `ngerman` gewählt, aber aus historischen Gründen wird dabei auf `ngermanb` zurückgegriffen.

finden sich die oben gezeigten Informationen und noch ein paar Details mehr.

Ein solcher Durchlauf ist natürlich nur dann möglich, wenn man keine syntaktischen Fehler in seinem Quelltext hat. Diese machen sich im Allgemeinen aber sehr schnell bemerkbar, da man eine mehr oder weniger kryptische Fehlermeldung auf dem Bildschirm erhält. Die beliebteste Ursache für Fehler sind zu wenige oder zu viele schließende Klammern. Auch eine falsche Klammer (eckig statt geschweift) wird gerne genommen.

Manchmal sieht man das Problem auf Anhieb, manchmal schlägt `latex` selber eine Korrektur vor. Aber es gibt auch genügend Fälle, in denen die Fehlermeldung erst ein ganzes Stück nach der Ursache auftaucht. Wenn nämlich zum Beispiel vergessen wurde, den schließenden Befehl für eine Umgebung anzugeben, wird `latex` erst ganz am Ende des Durchlaufs melden, dass etwas fehlt. Dann wird die Suche aufwändiger, allerdings können einem gute Editoren dabei helfen, indem zum Beispiel Klammerpaare angezeigt werden.

Man kann sich die Ausgabe nun im Prinzip schon anschauen, allerdings ist sie noch nicht vollständig. Denn für das Inhaltsverzeichnis oder Querverweise benötigt man noch einen zweiten Durchlauf. Also rufen wir LaTeX erneut auf:

```
prompt> latex beispiel01
```

Wenn man sich die diesmal ausgegebenen Informationen anschaut, sieht man, dass direkt auf die Dateien `beispiel01.aux` und `beispiel01.toc` zugegriffen wird.

Nun haben wir eine vollständige Ausgabedatei `beispiel01.dvi`.[6] Sie kann schon direkt mit entsprechenden Programmen angezeigt werden. Unter Windows nutzt man dafür zum Beispiel `windvi` (siehe Abbildung 3.2 auf der nächsten Seite), unter Linux kann man auf `xdvi` (siehe Abbildung 3.3 auf Seite 56) zurückgreifen.

Aus diesen Programmen heraus lässt sich meistens auch direkt drucken. Allerdings hat sich heute noch ein weiterer Zwischenschritt durchgesetzt. Denn um die `.dvi`-Dateien zu drucken, nutzte man früher eigens dafür gedachte Programme, die auf einen bestimmten Drucker oder eine Gruppe von Druckern zugeschnitten war. Mit der Zeit hat sich allerdings als Ausgabeformat PostScript (PS) durchgesetzt. Es kann von vielen Druckern direkt

6 Bei komplexen Dokumenten kann durchaus auch noch ein dritter Durchlauf notwendig sein, eventuell mit eingeschobenen Aufrufen von Hilfsprogrammen, zum Beispiel zum Erstellen eines Literaturverzeichnisses oder eines Indexes.

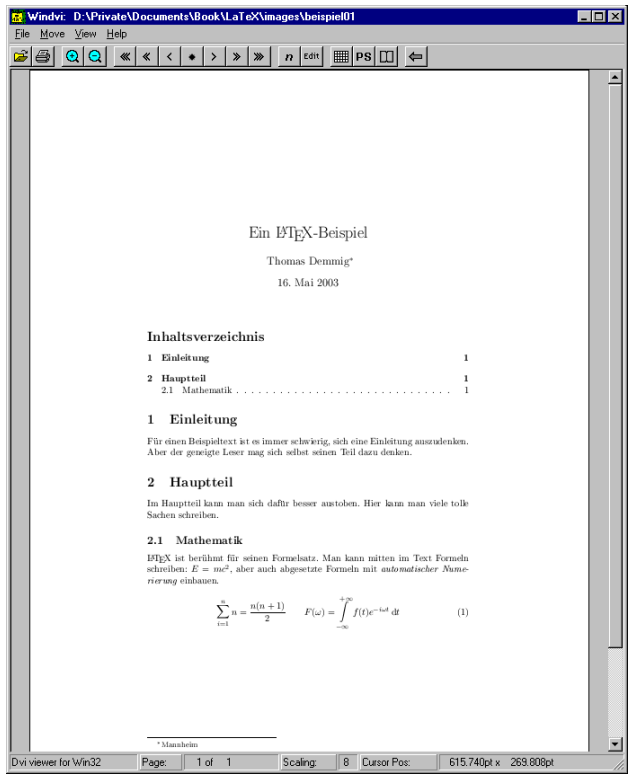

Abb. 3.2: Windvi – Ein DVI-Betrachter unter Windows

verarbeitet werden. Für alle anderen Drucker gibt es Konvertierungsprogramme. Das wohl bekannteste ist GhostScript (das Programm selber hat den Namen `gs`), das für die meisten Betriebssystemplattformen verfügbar ist.

Um aus einer DVI-Datei eine PS-Datei zu machen, nutzt man normalerweise das Programm `dvips`. Also ziehen wir diesen Schritt auch durch:

```
prompt> dvips beispiel01
```

Damit wird aus der Datei `beispiel01.dvi` eine Datei `beispiel01.ps` erzeugt. Die Ausgabe an der Befehlszeile ist nicht so interessant, es sei nur darauf hingewiesen, dass auch hier die einzelnen Seiten in eckigen Klammern angegeben werden.[7] Die PostScript-Datei kann nun direkt auf einen entsprechenden Drucker ausgegeben oder vorher mit `gs` umgewandelt werden. Man kann sich auch diese Datei am Bildschirm anschauen, dazu dient GhostView, eine grafische Oberfläche für GhostScript.

7 Dies ist bei den meisten an LaTeX beteiligten Programmen der Fall.

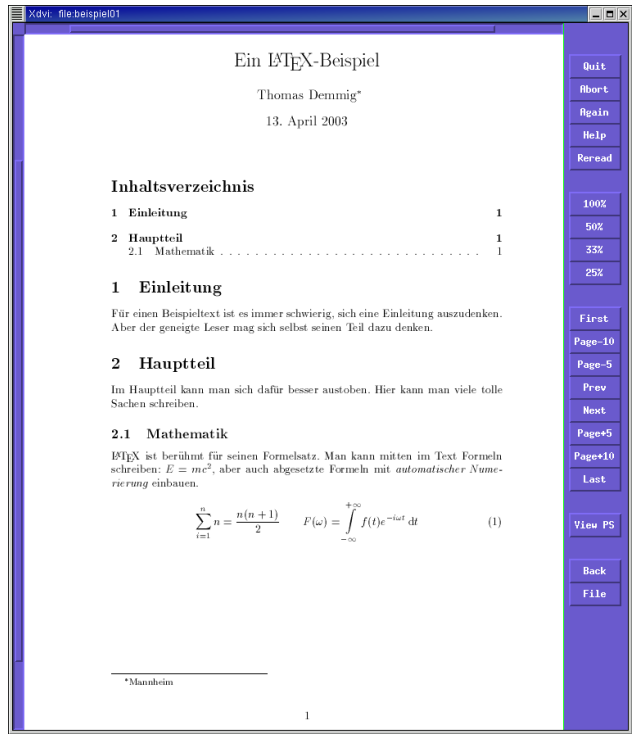

Abb. 3.3: xdvi – Ein DVI-Betrachter unter Linux

Um das Ergebnis in digitaler Form an andere weiterzugeben, bietet sich das PDF-Format von Adobe an. Solche Dateien lassen sich unter anderem mit dem kostenlos erhältlichen Acrobat Reader anzeigen und ausdrucken. Es dürfte das zur Zeit meistverwendete Format für elektronische Dokumente sein, die eine gewisse Formatierung beinhalten.[8] Es ist aus PostScript entstanden und kann zum Beispiel mit `ps2pdf` umgewandelt werden:

```
prompt> ps2pdf beispiel01
```

Es gibt allerdings auch einen direkteren Weg, um aus einer LaTeX-Datei eine PDF-Datei zu machen. Dazu nutzt man `pdflatex`, eine besondere LaTeX-Version, die aus einer `.tex`-Datei ohne Umwege eine PDF-Datei erzeugt. Darauf werde ich ausführlicher in Kapitel 19 auf Seite 341 eingehen.

Das hört sich jetzt ziemlich umständlich an, ist es aber im alltäglichen Gebrauch gar nicht. Zum einen erhält man meist durch seinen Editor Un-

8 Reiner ASCII-Text ist natürlich noch einfacher …

terstützung, zum anderen beschränkt sich der endgültige Aufruf oft auf einen doppelten Aufruf von LaTeX, gefolgt von dvips und dem Druck. Ganz modern ist zudem noch die direkte Nutzung von pdflatex, mit dem man sich manchen Schritt erspart.

Erfahrene Nutzer können sich vor allem unter Linux für umfangreiche Dokumente eine make-Datei schreiben, mit dem sich quasi alles von alleine erledigt.

3.4 Dateien

Nun sind uns schon eine ganze Reihe von verschiedenen Dateien über den Weg gelaufen. In der folgenden Aufzählung sind viele von ihnen beschrieben. Diese Liste ist natürlich nicht vollständig …

`.tex` Die eigentliche LaTeX-Quelldatei, die mit `latex` verarbeitet wird.[9]

`.cls` Dokumentklasse, die man im Befehl `\documentclass` angeben kann.

`.clo` Klassenoptionen, die im Zusammenhang mit einer `.cls`-Datei verwendet werden.

`.sty` Paket, das sich mit `\usepackage` aktivieren lässt.

`.dtx` LaTeX-Datei, in der sich der Code und die Dokumentation für eine Paketdatei befindet. Lässt man LaTeX über diese Datei laufen, erhält man die Dokumentation dazu.

`.ins` LaTeX-Datei, mit der sich die Paket-Datei mit der Endung `.sty` erzeugen lässt. Dazu wird allerdings auch die `.dtx`-Datei benötigt.

`.dvi` Ausgabedatei, die unabhängig von einem Ausgabegerät ist (daher DVI – Device Independent). Sie kann entweder direkt mit verschiedenen Programmen angezeigt oder ausgedruckt werden oder wird weiterverarbeitet und zum Beispiel in PostScript oder PDF umgewandelt.

`.aux` Datei mit den Querverweisen und Abschnitten eines Dokuments. Aus dieser Datei werden die Informationen für Verweise ausgelesen. Sie selbst wird bei einem LaTeX-Lauf erstellt.

`.toc` Datei für das Inhaltsverzeichnis. Auch sie wird erst während des LaTeX-Laufs angelegt. Für ein korrektes Inhaltsverzeichnis benötigt man also mindestens einen zweiten Durchlauf.

9 Auch „reine“ TeX-Dateien (sogenanntes Plain-TeX) haben diese Endung. Sie können normalerweise nicht mit LaTeX verarbeitet werden, sondern müssen direkt mit `tex` aufgerufen werden.

`.lof` Ähnlich wie das Inhaltsverzeichnis wird in dieser Datei das Abbildungsverzeichnis erstellt.

`.lot` Auch diese Datei wird erstellt und enthält das Tabellenverzeichnis.

`.idx` Hier sammeln sich Informationen für einen Index. Diese Datei ist in der Regel nicht direkt nutzbar. Sie dient vielmehr als Eingangsdatei für einen Aufruf von `makeindex`.

`.ind` Ausgabe eines `makeindex`-Laufs. Diese Datei wird wiederum von LaTeX eingebunden.

`.log` Protokolldatei mit recht ausführlichen Informationen über die Geschehnisse während eines LaTeX-Laufs.

`.ilg` Protokolldatei eines `makeindex`-Laufs.

`.bib` Datei mit Literaturverweisen. Diese Dateien dienen als Datenbanken für das Programm BibTeX und vereinfachen den Umgang mit großen Literaturlisten, die von vielen Leuten (wie zum Beispiel in einem Fachbereich) verwendet werden.

`.bbl` Ausgabe von BibTeX, die in der LaTeX-Datei verwendet wird.

3.5 Mehrere Dateien

Bei längeren Dokumenten wird es schnell unübersichtlich, wenn man den gesamten Text in einer einzelnen `tex`-Datei hat. In solchen Fällen empfiehlt es sich, ihn zum Beispiel kapitelweise aufzuteilen und die einzelnen Kapiteldateien dann in einer Hauptdatei zusammenzufassen. Diese Hauptdatei enthält dann üblicherweise nur die Präambel, das `\begin{document}`, die Befehle zum Einbinden der anderen Dateien und ein `\end{document}`. Eine andere Anwendung ist, umfangreiche Tabellen oder Zeichnungen in extra Dateien auszulagern, um sie dann im Text einzubinden. Für diesen Fall nutzt man am besten `\input`.

```
\input{Dateiname}
```

Der Dateiname kann ohne Erweiterung angegeben werden, wenn es sich um eine `tex`-Datei handelt. Innerhalb der eingebundenen Datei können wiederum Dateien per `\input` eingebunden werden. LaTeX verhält sich dabei so, als ob es vom „Dateiwechsel“ gar nichts weiß.

Das zu Beginn dieses Abschnitts beschriebene Szenario mit den Kapiteln in einzelnen Dateien sollte allerdings eher über einen anderen Befehl gelöst werden:

```
\include{Dateiname}
```

Mit \include wird ebenfalls der Code aus *Dateiname* eingefügt – hier ist die Erweiterung .tex wegzulassen. Der Befehl hat zwei Eigenheiten:

- Er kann nicht verschachtelt werden. Eine mit \include eingebundene Datei darf keinen weiteren \include-Befehl enthalten.
- Vor und nach dem einzufügenden Code wird automatisch ein Seitenumbruch vorgenommen. Somit sollte man den Befehl wirklich nur für ganze Kapitel oder ähnliche Bereiche nutzen.

Wenn \include doch so viel mehr Einschränkungen hat als \input, warum sollte man ihn dann nutzen? Ganz einfach: Er kann im Zusammenspiel mit \includeonly dafür sorgen, dass bei Änderungen an der aktuellen Datei nicht alles komplett neu gesetzt wird.

```
\includeonly{Dateinamen}
```

Mit diesem Befehl teilt man LaTeX mit, dass neben der Hauptdatei nur diejenigen durch Komma getrennt eingebunden werden sollen, die als Parameter angegeben sind und danach in einem \include-Befehl auch vorkommen. Dadurch ist der gesetzte Text dann zwar auf das tatsächlich Eingebundene beschränkt, aber Verweise, Inhaltsverzeichnis und Ähnliches werden so weit wie möglich beibehalten. Dies ist dann sehr hilfreich, wenn man in einem großen Projekt nacheinander einzelne Kapitel bearbeitet und beim Kompilieren nicht immer alles benötigt. Man gibt dann in \includeonly die Datei des aktuellen Kapitels an und hat eine deutlich schnellere Generierung des (Teil-)Dokuments. Somit muss man nicht mühevoll einzelne \include-Befehle aus- und wieder einkommentieren, sondern kann an einer einzelnen Stelle drehen. Ist das Dokument dann abgeschlossen, nimmt man nur den \includeonly-Befehl wieder raus und lässt den Text komplett erstellen.[10]

Eine Einschränkung gibt es noch: \include darf nicht in der Präambel stehen, \includeonly dagegen nicht nach dem \begin{document}.

3.6 Ein schneller Einstieg: empfehlenswerte Umgebung

Jetzt sind Sie vermutlich ziemlich erschlagen und wissen gar nicht so genau, was Sie alles machen sollen, oder? Das kann ich verstehen und ich verhehle auch nicht, dass es einem Einsteiger durchaus schwer fallen kann. Aber Sie können sicher sein, dass schon nach kurzer Zeit das Meiste durchaus logisch und eingängig ist.

10 So ist übrigens auch dieses Buch entstanden.

Für einen einfachen Start empfehle ich die Befehle aus Listing 3.2. Man erhält damit quasi ein Rundum-Sorglos-Paket, mit dem man zumindest die besonderen Stolperfallen europäischer Texte umgeht.

Listing 3.2: Eine empfehlenswerte Ausgangsdatei

```
\documentclass[a4paper]{article}
\usepackage[ngerman]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\begin{document}
  Hier kann jetzt Ihr Text stehen.
\end{document}
```

Durch die globale Option `a4paper` wird schon vom normalen deutschen DIN-A4-Format ausgegangen.

Die Option `ngerman` des Pakets `babel` aktiviert eine deutschsprachige Umgebung zum Beispiel für die Silbentrennung und generierte Bezeichnungen. Man kann auch die Option `german` verwendet, um dadurch die Regeln der alten deutschen Rechtschreibung zu aktivieren. Ansonsten gibt es keinen Unterschied zur „normalen“ Option.

Mit dem Paket `inputenc` wird die Eingabe „exotischerer“ Zeichen unterstützt. Umlaute, Akzente und ähnliches sind nun kein Problem mehr, da sie direkt über die Tastatur eingegeben werden können. Man muss nur als Option den Zeichensatz angeben, den das Betriebssystem verwendet. Für Windows gilt hier im Allgemeinen `ansinew`, während man für Linux die Option `latin1` wählen sollte.

Das letzte geladene Paket `fontenc` schließlich aktiviert Zeichensätze mit der T1-Kodierung, wodurch Silbentrennungen auch in Wörtern mit Umlauten oder akzentuierten Zeichen ordentlich funktionieren.

3.7 Kontrollfragen

1. Welcher Befehl und welche Umgebung sind für jedes LaTeX-Dokument notwendig?
2. Wie kann man Kommentare im Quelltext einfügen?