LaTeX in 14 Minuten

Hans Peter Wolf

12. November 2008

Inhaltsverzeichnis

1 Ziel und Gegenstand

Diese Kurzbeschreibung aus 2×7 Kapiteln richtet sich an diejenigen, die noch nichts mit Hilfe von TeX oder LaTeX geschrieben haben, nun aber doch einen Hauch davon einatmen möchten. Es werden die allerwichtigsten Konzepte und Kommandos vorgestellt, mit denen schon viele einfache Texte gelingen werden.

TeX ist ein von D.E. Knuth entwickelter Textformatierer, der den Text einer Input-Datei nach bestimmten Regeln setzt und das Ergebnis als geräteunabhängige Datei ausgibt. Mit Hilfe von sogenannten Makros oder Befehlen lässt sich das Ergebnis beeinflussen – der Text gestalten. Da die von TeX bereitgestellten Befehle für viele Textersteller zu elementar sind, wird gern auf eine Sammlung zusätzlicher Befehle zurückgegriffen, die mit dem Kürzel LaTeX oder mit LaTeX2ϵ bezeichnet wird.

2 Umgang mit LaTeX

Wie erhält man einen mit LaTeX gesetzten Text? Für einen Ausdruck sind fünf Schritte umzusetzen:

1. Schreibe mit einem beliebigen Editor den gewünschten Text in eine Datei, deren Name auf `.tex` enden muss, z. B. `test.tex`.

2. Platziere in den Text LaTeX-Befehle, um das gewünschte Erscheinungsbild zu erhalten.

3. Formatiere den Text. Heißt die erstellte Datei `test.tex`, formatiert der Befehl `pdflatex` den Text und erstellt eine `pdf`-Datei, die wie gewohnt angeschaut oder gedruckt werden kann.

3.* Alternativ funktioniert auch die klassische Vorgehensweise: a) Formatiere die `tex`-Datei mit dem Befehl `latex test`. b) Das Ergebnis – die Datei `test.dvi` – kann dann mit einem *Viewer* wie `yap` der TEX-Version `miktex` oder `xdvi` von `teTeX` betrachtet werden. c) Zum Ausdruck muss `test.dvi` in eine druckbare Datei, z. B. mittels `dvipdf test` oder `dvips test` überführt werden; diese Befehle rufen die neuen Dateien `test.pdf` bzw. `test.ps` hervor.

Die verwendeten Befehle können auf den verschiedenen Rechnern etwas unterschiedlich aussehen. Am Prinzip ändert sich dadurch aber nichts. In der Regel verwenden TEXniker kleine Umgebungen, die zentral einen Editor enthalten und die angesprochenen Verarbeitungsprozesse per Knopfdruck auslösen, wie: TeXnicCenter, TeXShop, Texmaker.[1]

3 LATEX-Befehle

3.1 Wie kann man Gestaltungswünsche ausdrücken?

Gestaltungswünsche können sich auf das gesamte Dokument, auf eine Seite, auf die Strukturierung, auf einen Absatz, auf einzelne Worte und anderes beziehen. LATEX-Befehle stellen eine Sprache dar, mit der man auf jeder dieser Ebenen seine Wünsche artikulieren kann. Demgemäß muss man sich immer fragen, auf welche Ebene oder auf welchen Bereich sich ein Wunsch bezieht, und kann dann unter den Befehlen dieser Ebene den passenden auswählen.

Es gibt also Befehle, die eine Wirkung auf das gesamte Dokument entwickeln, auf einzelne Seiten und so fort. Dieser Einteilung entsprechen auch interne Abarbeitungsprozesse. Jedes Text-Zeichen, das nicht zu einem Befehl gehört, wird in eine kleine Box gesteckt. Aus diesen kleinen Boxen werden Wort-Boxen konstruiert, die wiederum zu Paragraphen zusammengefügt werden und so fort. Zwischen Boxen wird ein elastisches, aber unsichtbares Material eingefügt. Der Formatierer verschiebt dann solange die Boxen, bis *er* mit dem Ergebnis zufrieden ist.

3.2 Wie sieht ein LATEX-Befehl aus?

Befehle und Nichtbefehle müssen erkannt werden. Alle Buchstabenfolgen, die mit einem \ beginnen, sind Befehle. Der Schrägstrich ist also ein Erkennungszeichen. Viele Befehle besitzen Argumente, die jeweils in geschweiften Klammern dem Befehlsnamen folgen müssen. Diese Argumente spezifizieren die Wirkung des Befehls genauer. Einigen Befehlen können außerdem noch zusätzlich optionale Argumente mitgegeben werden. Diese befinden sich dann in eckigen Klammern.

Es gibt zwar noch andere Strukturen wie Zähler, Maße und Register, doch sollte man diese im Fortgeschrittenenkurs diskutieren. Wohl aber erscheint ein Hinweis auf zwei Design-Prinzipien angebracht. Es gibt Befehle, deren Wirkung solange gilt, bis ein entsprechend anderer sie aufhebt. Auf der anderen

[1] Vgl. auch http://en.wikipedia.org/wiki/Category:TeX_editors.

Seite findet man Befehle, die nur paarweise auftreten. Mit dem ersten beginnt eine Zustandsänderung, die durch den zweiten wieder zurückgesetzt wird. Die erwähnten Klammern { und } sowie [und] entsprechen der zuletzt genannten Philosophie.

4 Das Dokument

4.1 Wie ist ein Dokument aufgebaut?

Ein LaTeX-Dokument besteht aus einem Vorspann (Präambel) und dem Rest. In der Präambel werden alle Befehle eingetragen, die sich auf das gesamte Dokument beziehen sollen. Sie wird eingeleitet mit:

```
\documentclass{article}
```

für eine 10-pt-Größe oder für etwas größere Schrift:

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

Die Präambel endet mit dem Beginn des wirklichen Dokumentes durch den Befehl:

```
\begin{document}
```

Der letzte Befehl eines Dokumentes schließt die geöffnete Klammer:

```
\end{document}
```

In der Präambel lässt sich zum Beispiel das Paket `ngerman` laden, mit dem einige Spezialitäten der deutschen Sprache (Trennungen) unterstützt werden.

```
\usepackage{ngerman}
```

Andere Pakete (vgl. die Einbindung von `helvet`, `mathpazo`, `courier`) stellen weitere Besonderheiten (wie z. B. Schriften) zur Verfügung. Wer gleich ganze Bücher schreiben will, sollte als Dokumentenklasse nicht `article`, sondern `book` verwenden. Zur Veränderung von Texthöhe, -breite und Rändern lese man nach unter:

```
\textwidth, \textheight, \topmargin, \oddsidemargin.
```

Doch statt diese Größen umzudefinieren, sollte man lieber ein Paket einbinden, das ausgewogene Maße setzt.

4.2 Wie lassen sich Titel und Inhaltsverzeichnis erstellen?

Viele Dokumente beginnen mit Titel, Autor, Datum und Inhaltsverzeichnis. Solche Typen von Textelementen sollte ein Autor orientiert an den jeweiligen Bedeutungen festlegen können. LaTeX bietet folgerichtig entsprechende sachlogische Befehle an. Im vorliegenden Dokument kommen am Anfang folgende fünf zum Einsatz:

```
\title{\LaTeX\ in 14 Minuten}
\author{Hans Peter Wolf}
\date{12. November 2008}
\maketitle
\tableofcontents
```

4.3 Wie lässt sich ein Dokument strukturieren?

Texte gliedern sich normalerweise in Abschnitte, Unterabschnitte usw. Dieser Idee entsprechen die Befehle

```
\section{...}, \subsection{...}, ...
```

Diesen Befehlen sind die gewünschten Überschriften mitzugeben. Abstände und Schriftarten werden automatisch gewählt.

5 Feinheiten

5.1 Wie lässt sich Text einrücken?

LaTeX kennt dazu *Umgebungen*. In diesen gelten modifizierte Gesetze. Zum Beispiel rückt die `quote`-Umgebung den eingeklammerten Text ein. Verwendung:

```
\begin{quote}
    der hier stehende Text wird einger\"uckt
\end{quote}
```

5.2 Wie kann man Text zentrieren?

Ganz einfach!

```
\begin{center}
     zu zentrierender Text
\end{center}
```

zentriert den Text mit einem Schuss Luft über und unter der Zeile:

zu zentrierender Text

Die beiden verwendeten Befehle entsprechen der Klammerphilosophie, die oben erwähnt wurde.

5.3 Wie lassen sich Aufzählungen realisieren?

Die `enumerate`-Umgebung gestaltet Aufzählungen. Es sind folgende Punkte zu berücksichtigen:

1. Sie beginnt mit einem Beginn-Befehl.
2. Sie endet mit einem Ende-Befehl.

3. Items werden entsprechende Befehle vorangestellt.

4. Aufzählungsumgebungen können geschachtelt auftreten.

Wie wurde dieses spezielle Beispiel erstellt?

```
\begin{enumerate}
\item Sie beginnt mit einem Beginn-Befehl.
\item Sie endet mit einem Ende-Befehl.
\item Items werden entsprechende Befehle vorangestellt.
\item Aufz\"ahlungsumgebungen k\"onnen geschachtelt auftreten.
\end{enumerate}
```

Die Umgebung `itemize` setzt statt der Zahlen einen fetten Punkt vor die Aufzählungselemente. Übrigens lassen sich Umgebungen schachteln, so dass ein Aufzählungspunkt wieder eine Aufzählung enthalten kann.

5.4 Wie kann man Größe und Art der Schrift ändern?

Dafür gibt es natürlich Befehle. Für die Schriftgrößenwahl wollen

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

eingesetzt werden, Schriftarten werden durch

```
\rm, \it, \sc, \bf, \sl, \sf, \tt
```

herbeizitiert. Die Kürzel seien kurz übersetzt. Der erste Befehl ist die Bezeichnung für die Normalschrift (Roman), der zweite für *kursive Schrift*, der dritte für SMALL CAPS, der vierte für **Fettdruck**, der fünfte für *Slanted*, der sechste für Sans Serif und der letzte für `Typewriter`. Jeder dieser parameterfreien Befehle wirkt solange, bis ein entsprechend anderer einen neuen Zustand herbeiführt. Soll eine Wirkung nur für einen bestimmten Bereich gelten, lässt dieses sich durch Einsatz einer expliziten Klammerung { und } erzielen:

```
Normalschrift {\bf Fett} Normalschrift
```

erzeugt: Normalschrift **Fett** Normalschrift. Für Einschübe sind die folgenden Befehle zu verwenden: `\textit`, `\textsc`, `\textbf`, `\textsl`, `\textsf`, `\texttt`. Diesen müssen die anders zu schreibenden Inhalte als Argument mitgegeben werden. Wie erhalten wir wohl: *textit*, TEXTSC, **textbf** *textsl*, textsf, `texttt`? Richtig, bspw. durch: `\texttt{texttt}`!

5.5 Wie bekommt man besondere Zeichen?

Mathematische und einige besondere Zeichen müssen mittels eines Befehls hervorgerufen werden. So erzeugt `$\rightarrow$` einen Pfeil nach rechts: $\rightarrow$. Die Dollar-Zeichen schalten vom Normalzustand in den mathematischen Modus um bzw. wieder zurück. Im mathematischen Modus lassen sich leicht ganze Formeln setzen, was aber hier nicht vorgeführt wird. Einige weitere Zeichen:

Befehl:	`\#`	`\$`	`\&`	`\_`	`\%`	`\{`	`\}`	`\S`
Ergebnis:	#	$	&	_	%	{	}	§

Was macht man bei Umlautproblemen? Umlaute werden klassisch durch Tüddelchen und Buchstaben ohne Punkte zusammengebaut: `\"a` → ä, `\"o` → ö, `\"u` → ü, `\"A` → Ä, `\"O` → Ö, `\"U` → Ü, `{\ss}` → ß. Das geht – vom ß abgesehen – immer. Unter Verwendung von `ngerman` kann man diese Befehle abkürzen durch `"a` usw.

Viel eleganter ist es, Zeichensätze mit echten Umlauten zu benutzen. Mit der Anweisung `\usepackage[T1]{fontenc}` werden die sogenannten EC-Fonts angefordert, die bspw. Umlaute enthalten. Damit die richtigen Umlaute den eingegebenen Zeichen zugeordnet werden, muss weiterhin angegeben werden, nach welcher Kodierung die Umlaute in der `tex`-Datei abgelegt werden. Dieses wird durch Einbindung von `\package[Kodierung]{inputenc}` in der Präambel festgelegt. Für den Platzhalter `Kodierung` muss anlagenspezifisch der Name der Kodiertabelle angegeben werden; typisch sind hierbei: `latin1, cp850, isolatin, macce, applemac, utf8`.

Der Autor hat zur erfolgreichen Formatierung auf verschiedenen Anlagen ohne Änderung des `inputenc`-Parameters die Datei `uml-ok.tex` entworfen. Erhältlich ist sie über die Homepage des Autors. Nach Einbindung per `\input{uml-ok.tex}` in der Präambel werden dann verschiedene Umlaute-Kodierungen in klassische TEX-Sequenzen übersetzt. Damit sollte der Ausdruck von Umlauten auch kein Problem mehr sein.

5.6 Wie lassen sich Zwischenräume verändern?

Einzelne Paragraphen werden durch Leerzeilen voneinander getrennt, mehr nicht. Der Abstand zwischen zwei Paragraphen lässt sich mit den Befehlen `\smallskip`, `\medskip` und `\bigskip` vergrößern.

Horizontalen Leerraum in der Größe des Wortes `Platz` verschafft man sich durch `\phantom{Platz}`. Das Zeichen ~ erzeugt ein geschütztes Leerzeichen. Um die Einrückung eines Paragraphen, wie am Anfang dieses Satzes, zu verhindern, ist der Befehl `\noindent` dem Paragraphen voranzustellen.

Dirty Tricks: Nur ganz ausnahmsweise sollte man sich vertikalen oder horizontalen Platz durch `\vspace*` bzw. `\hspace*` verschaffen; beispielsweise fügt `\vspace*{10mm}` vertikale 10mm Luft ein und `\hspace*{-2cm}` führt zu einem negativen Sprung, der für Überschreibungen genutzt werden kann. Probiere: `~~:--)~~und~~,\hspace*{-1mm}':\hspace*{-0.8mm}--)~` – diese führt zu dem Ergebnis :–) und ‚:–)

5.7 Weitere Hinweise

Die bisher vorgestellten Befehle lösen viele Gestaltungsfragen. Wie sehen die weiteren Schritte aus? Dazu soll mit einer Tabelle Neugier geweckt werden:

Befehl:	Ergebnis:
`\\`	bricht Zeile um
`\newpage`	bricht Seite um
`\-`	zeigt in einem Wort eine mögliche Trennstelle an
`\emph{...}`	bewirkt Texthervorhebung durch Schriftartwechsel
`$y_t=a+cx^2_t$`	zeigt, wie sich $y_t = a + cx_t^2$ erstellen lässt
`\tableofcontents`	erzeugt Inhaltsverzeichnis

Weitere Abschnitte müssten sich nun Tabellen und mathematischen Formeln widmen, und es fehlen auch noch Hinweise darauf, wie Fehlermeldungen des

Formatierungsprozesses zu verstehen sind. Zunächst sei dazu auf die Literatur verwiesen.

6 Wieso ist LaTeX ein *Wytiwyg*-Instrument?

Spätestens jetzt – oder vielleicht nach einer kleinen Pause – sollten Sie die gelesenen Dinge ausprobieren. Denn ohne eigene Experimente werden Sie den nächsten Abschnitt kaum verstehen. Dann werden Sie sehen, dass man mit LaTeX das bekommt, was man verdient: What you type is what you get!

7 Fehler, Fehler, Fehler

Wo gehobelt wird, da fallen Späne. So ist es auch bei der Texterstellung und auch bei der Textverarbeitung. Inhaltliche Fehler können unerkannt bleiben. Syntax-Fehler führen jedoch zu Fehlermeldungen. Hierbei bildet LaTeX keine Ausnahme, so dass nun noch etwas über Fehler gesagt wird.

Vermeide Fehler! Das sagt sich sehr leicht, jedoch kann man sich einen Fehler vermeidenden Arbeitsstil angewöhnen. Dieser beinhaltet Klammerpaare immer so hinzuschreiben, dass zugehörige Klammern sofort gesehen werden. Tabellen sollten auch im Editor übersichtlich sein. Wähle immer erst eine einfache Struktur, die dann typographisch verbessert werden kann. Unterscheide Phasen, in denen der Text vorangetrieben wird, von Formatierungsphasen. Benutze logische Bausteine und vermeide ad hoc überlegte Lösungen.

Studiere Fehlermeldungen! TeX-Fehler erkennt man bei den meisten LaTeX-Produkten daran, dass der Formatierungsprozess anhält und eine Fehlermeldung angezeigt wird. Beispielsweise verursacht ein Text mit der Zeichenkette „`verb+\unbekannt+`“ aufgrund eines fehlenden Rückstriches vor `verb` folgende Meldung:

```
(/usr/share/texmf/tex/latex/psnfss/t1phv.fd) [3] [4] [5] [6] [7]
! Undefined control sequence.
l.440 Zeichenkette "‘verb+\unbekannt
                                    +"’ aufgrund
?
```

Die Zahlen in eckigen Klammern repräsentieren fertig gesetzte Seiten. Die Zeile beginnend mit `l.` zeigt die Nummer der Zeile, die LaTeX Probleme bereitet hat. Von dieser Zeile ist ein Teil abgedruckt, es folgt ein Sprung und dann der Rest der Zeile. Die Schwierigkeit ist bei der Interpretation der Zeichenkette aufgetaucht, die unmittelbar vor dem Sprung steht. Den wichtigen Hinweis auf den Fehlertyp enthält die Zeile mit dem Ausrufungszeichen. In diesem Fall versteht man ihn schnell: Der Befehl (*control sequence*) `\unbekannt` ist unbekannt und die Fehlerursache. Das Fragezeichen ist ein Eingabeaufforderungszeichen. Typische Eingaben sind `h` für *help*, `x` für *exit*, `q` für *quiet*. Die Fehler-Meldung wird übrigens zusammen mit weiteren Meldungen in der

Datei mit einer Endung `log` abgelegt und kann dort noch einmal nachgelesen werden.

Finde Fehlerort und -ursache! Die Eingabe des Buchstabens `h` zeigt einen kleinen Hilfetext an, der manchmal mehr Klarheit bringt. Oft ist die Eingabe von `q` zu empfehlen; dann läuft der Prozess ohne Stocken durch und man kann weitere Indizien anhand des Formatierergebnisses sammeln: Dieses kann darauf hinweisen, dass die Ursache weiter vorn liegen muss oder gibt uns (z. B. durch falsche Schriften) Hinweise zur Fehlerqualität. Als Strategien kommen immer wieder zum Einsatz: *Fehlerstelle einkreisen* durch ein temporär eingefügtes `\end{document}` sowie *Fehler entfernen* durch Vereinfachungen. Bei völlig unerklärbarem Verhalten sei geraten, die Datei mit der Endung `aux` zu löschen.

Es sei zugegeben, dass dieses Kapitel ein Erinnerungsposten ist. Denn es ist nichts schwieriger als kurz, aber erschöpfend, über Fehler zu philosophieren. Vielleicht hilft die Anregung, ein Logbuch über Fehler zu führen, auf dessen Basis dann dieses Kapitel verbessert werden kann. Mit dem Hinweis, dass erstaunlich oft eine kurze Arbeitspause bei der Fehlersuche weiterhilft, endet der erste Teil der Abhandlung.

8 Gegenstand des zweiten Teiles

Dieser Teil widmet sich in weiteren *sieben* Kapiteln zwei von drei wichtigen Punkten, die in dem ersten Teil noch nicht zur Sprache gekommen sind:

- Wie erstellt man einfache Tabellen?
- Wie setzt man mathematische Formeln?
- Wie wird man mit Fehlermeldungen fertig?

Auch für diesen Teil gilt das Prinzip:

> Wenn du etwas bekommen willst, musst du deinen Wunsch formulieren. Hierfür bedarf es einer Sprache, um über die relevanten Dinge sprechen zu können, — einer Verabredung zwischen dem, der sich etwas wünscht, und dem Erfüller der Wünsche.

So wünscht sich der Ersteller einer Tabelle, die Tabelle genau so zu bekommen, wie er sie sich vorstellt. Der Textformatierer hat den Wunsch umzusetzen. Für die Formulierung des Wunsches muss der Ersteller auf die Sprachkonstrukte zurückgreifen, die der Formatierer versteht. Ohne die Sprache zu kennen, kann man davon ausgehen, dass sie für bestimmte Tabellen oder Formeln geeignet und für andere weniger geeignet sein wird. (Nebenbei bemerkt gehören die Mausklicks vieler Software-Anwendungen mit zu deren Sprache, und auch diese Sprachelemente müssen erlernt werden.) Wenden wir uns nun der Sprache zu, die LaTeX versteht.

9 Tabellenerstellung

9.1 Wie erstellt man eine einfache Tabelle?

Zur Beschreibung einer Tabelle bietet LaTeX die *tabular*-Umgebung an. Sie beginnt mit

`\begin{tabular}{...}`

und endet mit

`\end{tabular}`.

Tabellen bestehen aus Spalten und Zeilen. An der Stelle der drei Punkte `...` muss man gemäß seinem Wunsch beschreiben, wie die Spalten — die großen Teile einer Tabelle — auszusehen haben. In dieser Spaltenbeschreibung ist festzulegen, wie viele Spalten man bekommen möchte und ob deren Elemente linksbündig, zentriert oder rechtsbündig gesetzt werden sollen. Die entsprechenden Symbole, die statt der drei Punkte eingesetzt werden können, sind `c`, `r` und `l`, wobei für jede Spalte genau eines dieser Symbole gewählt werden muss. Wir wollen ein Beispiel betrachten:

lfd. Nr.	Sprachelement	Bedeutung
1	l	am linken Rand ausgerichtet
2	c	zentriert
3	r	am rechten Rand ausgerichtet
4	\\	Zeilenumbruch
5	&	Trennzeichen

Diese Tabelle besteht aus einer zentrierten, einer rechts- und einer linksbündigen Spalte. Die TeX-Datei dazu hat folgenden Inhalt:

```
\begin{center}
\begin{tabular}{crl}
lfd. Nr. & Sprachelement & Bedeutung\\
1        &       l       & am linken Rand ausgerichtet\\
2        &       c       & zentriert\\
3        &       r       & am rechten Rand ausgerichtet\\
4        &  \verb-\\-    & Zeilenumbruch\\
5        &  \verb-&-     & Trennzeichen
\end{tabular}
\end{center}
```

Wir erkennen, dass die Tabelle in einer *center*-Umgebung steht, die Spaltenbeschreibung `{crl}` lautet, Zeilen (bis auf die letzte!) durch `\\` beendet und die Felder in einer Zeile durch das Trennzeichen `&` voneinander abgetrennt werden. Damit lassen sich einfache Tabellen leicht bauen.[2] Die Fußnote ist übrigens erstellt worden durch:

```
... bauen.\footnote{Will man ... ausgegeben.} Die ...
```

[2] Will man den Code von LaTeX-Kommandos in einen Text integrieren, dürfen diese nicht ausgewertet werden. Hierfür gibt es den Befehl `\verb`, der leider in Fußnoten Probleme bereitet. Das Zeichen, das dem `\verb`-Befehl folgt, erhält die Bedeutung eines Trennzeichens. Zwischen diesem Zeichen und seinem Wiederauftreten werden keine Befehle ausgewertet, sondern direkt ausgegeben.

9.2 Wie beschreibt man horizontale Linien und vertikale Zwischenräume?

Nach dem Kartoffeltheorem gilt: Jetzt kann das Werkzeug Linien erstellen, dann werden auch Linien in die Tabelle eingetragen. Deshalb wünschen sich einige Leute jeglichem data ink ratio zum Trotz viele Linien in ihren Tabellen. Eine horizontale Linie bekommt man durch den Befehl `\hline`.

Dirty Tricks: Häufig gefällt dann der Zwischenraum zwischen Linie und Text nicht mehr. Ein zusätzlicher, vertikaler Zwischenraumwunsch kann dem `\\`-Befehl als optionaler Parameter mitgegeben werden. Folgendes Ergebnis erhält man durch Einfügen eines Zeilenumbruchs mit `\\` mit dem Argument `.5ex` bzw. `-1ex` (`ex` ist eine Maßeinheit; `1ex` entspricht der Höhe von einem x):

Nr.	Sprachelement	Bedeutung
1	`\\[.5ex]`	neue Zeile plus `.5ex` vertikalem Zwischenraum
2	`\hline`	horizontale Linie
3	`\\[-1ex]`	neue Zeile minus `1ex` vertikalem Zwischenraum

Und hier kommt der TEX-Code:

```
\begin{tabular}{cll}
Nr. & Sprachelement   & Bedeutung   \\[0.5ex]\hline \\[-1ex]
1   & \verb?\\[.5ex]? & neue Zeile plus \verb?.5ex?
                        vertikalem Zwischenraum     \\
2   & \verb?\hline?   & horizontale Linie           \\
3   & \verb?\\[-1ex]? & neue Zeile minus \verb?1ex?
                        vertikalem Zwischenraum
\end{tabular}
```

9.3 Wie beschreibt man vertikale Linien und Spaltenzwischenräume?

Diese Wünsche werden bei der Definition der Spalten festgelegt. Zwischen den Buchstaben zum Spaltentyp (`{crl}`) sind dazu spezielle Beschreibungselemente einzufügen. Ein „`|`“ in der Spaltenbeschreibung erzeugt eine vertikale Linie. Soll zwischen Spalten oder zwischen Spalte und einer vertikalen Linie `etwas` (z. B. leerer Raum) eingefügt oder @ddiert werden, ist an der entsprechenden Stelle `@{etwas}` einzufügen. Es folgt wieder ein Beispiel.

1.	`|`	:	platziert vertikale Linie
2.	`@{:}`	:	platziert zwischen Spalten :
3.	`@{\quad:\quad}`	:	platziert zwischen Spalten : und etwas Luft
4.	`@{\hspace{1cm}}`	:	platziert 1cm Luft zwischen Spalten
5.	`@{}`	:	entfernt den kleinen Zwischenraum, der sonst zwischen die Spalten gesetzt wird.

Versuchen Sie, die Sprachelemente im Original wiederzufinden:

```
\begin{tabular}{||l|l@{\quad:\quad}p{8cm}||}
\hline
1. & \verb-|-              & platziert vertikale Linie          \\
2. & \verb?@{:}?           & platziert zwischen Spalten :       \\
3. & \verb?@{\quad:\quad}? & platziert zwischen Spalten :
                             und etwas Luft                     \\
4. & \verb?@{\hspace{1cm}}? & platziert 1cm Luft zwischen Spalten \\
5. & \verb?@{}?            & entfernt den kleinen Zwischenraum, der
          \newline sonst zwischen die Spalten gesetzt wird.     \\
\hline
\end{tabular}
```

9.4 Wie trägt man Textstücke in Tabellen ein?

Manchmal möchte man für Texte eine Spalte von vorgegebener Breite, z. B. 8 cm, bekommen. Hierzu trägt man das Sprachelement `p{8cm}` in die Spaltenbeschreibung ein (s.o.). (In verkleinerten Kopien sind solche Breiten natürlich auch reduziert.) Passt der Inhalt nicht in die 8 cm, wird er umgebrochen. Ein Umbruch innerhalb eines Feldes, kann mit `\newline` erzwungen werden.

9.5 Wie vereinigt man in einer Zeile mehrere Felder?

Bisweilen möchte man innerhalb einer Tabellenzeile mehrere Spalten vereinigen. Dieser Wunsch kommt z. B. für übergreifende Überschriften auf. Hierfür gibt es den Befehl

`\multicolumn{`*Spaltenanzahl*`}{`*Spaltenbeschreibung*`}{`*Text*`}`.

Der letzte Parameter beinhaltet den gewünschten Text. Der erste Parameter bestimmt die Anzahl der Spalten, die zusammengefasst werden sollen. Der mittlere charakterisiert den Stil. Für den Stil gelten die Regeln der Spaltenbeschreibungssprache, die ja nun schon bekannt ist.

10 Mathematische Formeln

Was wäre TEX ohne mathematische Formeln, und was wären mathematische Formeln ohne TEX? TEX ist gerade auch für den Satz von Formeln entwickelt worden und damit ohne Formeln undenkbar. Formeln selbst würden ohne TEX viel von ihrer Schönheit (?) einbüßen.

10.1 Wie kündigt man Formeln an?

Ein Papier mit Formeln besteht in der Regel aus Text, in den Formeln eingestreut sind, sowie aus abgesetzten Formeln, in denen auch wieder etwas Text vorkommen kann. Solche qualitativ unterschiedlichen Einheiten müssen vom Formatierer erkannt werden. Je nach Qualität nimmt der Formatierer während der Formatierung einen bestimmten Zustand an, in dem er dann nach festen Regeln seine Satzaufgabe löst.

Zur Umschaltung in den mathematischen Modus innerhalb eines Textabschnittes dient das $-Zeichen. Das nächste $-Zeichen schaltet wieder in den Textmodus zurück: Die Summe von a und b ist $a+b$. wird erreicht durch:

```
{\sf Die Summe von $a$ und $b$ ist $a+b$.}
```

Soll eine Formel etwas abgesetzt und zentriert erscheinen, muss eine *Umgebung* her. An dem Sprachelement `\[` erkennt der Formatierer den Beginn der abgesetzten Formel, `\]` zeigt ihm das Ende der Formel an. Die Umgebung

```
\begin{equation} ... \end{equation}
```

arbeitet wie die `\[-\]`-Umgebung, nummeriert aber zusätzlich die Formeln. Innerhalb des mathematischen Modus lässt sich etwas von der Qualität normaler Text mit `\mbox{Normaltext im Mathemodus}` einbringen. Damit sind die wichtigsten äußerlichen Dinge über den Formelsatz erklärt.

10.2 Wie sieht das Innere einer Formel aus?

Formeln bestehen in der Regel aus seltsamen Zeichen, die in einer noch seltsameren Weise angeordnet oder positioniert werden müssen. Für die vielen Zeichen müssen entsprechende Befehle her, die zum Teil sehr einprägsame Namen besitzen. Auf die Zeichenfrage kann also eine große Tabelle antworten. Schwieriger ist es schon, eine geeignete Sprache zur Positionierung zu entwerfen. Dieses Problem entspricht genau dem, eine Formel über Telefon (natürlich nicht per Fax!) zu übermitteln. Probieren Sie es mit folgender Formel:

$$\int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{1}{2}(\frac{v-\mu_i}{\sigma_i})^2} dv$$

Die Lösung von LaTeX kommt hoffentlich Ihrem Vorschlag bis auf syntaktische Kleinigkeiten sehr nahe.

```
\[
  \int_{-\infty}^x
      \frac{1}
           {
            \sqrt{2\pi\sigma_1^2}
           }
      e^{
         -\frac{1}{2}
         (\frac{v-\mu_i}{\sigma_i})^2
        }
      dv
\]
```

Einige Bemerkungen sind zu der Formel angebracht. Das Sprachelement zur Anforderung eines Integralzeichens ist `\int`, für ein Summenzeichen entsprechend `\sum`. Die Untergrenze ist in der dem Unterstrich folgenden Gruppe definiert. Eine Gruppe wird, wie auch in anderem Zusammenhang schon angedeutet, mit dem Klammerpaar `{` und `}` definiert. Mit dem Unterstrich

lassen sich auch Indizes umsetzen. `\infty` erzeugt das ∞-Zeichen. Zur Erzeugung von Obergrenzen und Exponenten wird das `^`-Zeichen eingesetzt. Da für die Obergrenze im Beispiel nur ein Buchststabe, nämlich `x` verwendet wird, konnte auf die Gruppierungsklammern verzichtet werden. Brüche werden mit dem Befehl `\frac` erzeugt. Diesem sind zwei Parameter mitzugeben: zuerst der Zähler, dann der Nenner. Die Wurzel im Nenner erzeugt `\sqrt`. Da griechischen Buchstaben entsprechend lautende Befehle zugeordnet sind, liefert `\sigma_i^2` gerade σ_i^2. Damit müsste das Prinzip klar sein. Was noch empfohlen werden kann, ist, die Beschreibung der Formel übersichtlich anzuordnen. Dies verhindert zum Beispiel Klammerfehler.

11 LaTeX-Sünden und Hinweise

- Verändere erst nach sorgfältiger Überlegung die vorgeschlagenen Maße und lade nur unbedingt nötige Pakete!
- Vergiss über Layout-Fragen nicht die Inhalte und wähle die Formatierung möglichst einfach: keep it simple!
- Beende Zeilen nicht und Abschnitt nie mit doppelten Rückstrichen!
- Verwende möglichst wenige Schriften und möglichst wenige Striche in Tabellen, setze mathematische Größen und Ausdrücke immer im Mathe-Stil und vermeide möglichst Fußnoten!
- Befrage einen Fachmann erst nach gründlichem Studium der Fehlermeldung und der FAQ-Liste [6]!

Für das weitere Studium ist ein kleines Literaturverzeichnis angefügt. Ein solches Verzeichnis wird mit einer `thebibliography`-Umgebung erzeugt. Sie verwendet `\begin{thebibliography}{mylit}` und `\end{thebibliography}` als Klammern. Die einzelnen Einträge beginnen mit `\bibitem{xyz}`, wobei „`xyz`“ eine frei zu wählende Kennung ist. Mit `\cite{Kop97}` referenzieren wir den Eintrag mit der Kennung `Kop97`, also die Angabe mit dem Eintrag `\bibitem{Kop97}...`, und im Text wird dann später bspw. [1] stehen.

Literatur

[1] KOPKA, H., 1997:
LaTeX—Einführung, Bd 1, 3. Aufl., Pearson Studium (ca. 40 EUR).

[2] GOOSSENS, M., MITTELBACH, F., SAMARIN, A., 2005:
Der LaTeX Begleiter. 2. Aufl., Pearson Studium (ca. 60 EUR).

[3] KNUTH, D.E., 1986: The TeX-book. Addison-Wesley.

[4] LaTeX—A document preparation system. 2. Aufl.,Addison-Wesley.

[5] Schmidt, W. et al. 2003: LaTeX2$_\varepsilon$-Kurzbeschreibung.
ftp://ftp.dante.de/tex-archive/info/lshort/german/l2kurz2.pdf

[6] FAQ-Liste: http://www.dante.de/faq/de-tex-faq/html/de-tex-faq.html

A Anhang: Einige Übersichten für Formeln

Ein Anhang wird mit dem Befehl `\appendix` eingeleitet und führt zu einer Verwendung von großen Buchstaben zur *Nummerierung*.

A.1 Griechische Buchstaben

Die Regel lautet: Schreibe, wie man es spricht! Leider gibt es nicht für alle griechischen Buchstaben auch die große Form.

Befehl	`\alpha`	···	`\omega`	`\Gamma`	···	`\Omega`
Ergebnis	α	···	ω	Γ	···	Ω

A.2 Binäre Operatoren und Relationen

Befehl	`+`	`-`	`\pm`	`\cap`	`\vee`	`\mp`	`\cup`	`\in`
Ergebnis	$+$	$-$	$\pm$	$\cap$	$\vee$	$\mp$	$\cup$	$\in$
Befehl	`\wedge`	`\cdot`	`\times`	`\ast`	`\star`	`\bullet`	`\div`	`\sim`
Ergebnis	$\wedge$	$\cdot$	$\times$	$\ast$	$\star$	$\bullet$	$\div$	$\sim$

Befehl	`<`	`>`	`=`	`\leq`	`\geq`	`\equiv`
Ergebnis	$<$	$>$	$=$	$\leq$	$\geq$	$\equiv$
Befehl	`\mid`	`\subset`	`\supset`	`\approx`	`\subseteq`	`\supseteq`
Ergebnis	$\mid$	$\subset$	$\supset$	$\approx$	$\subseteq$	$\supseteq$
Befehl	`\ni`	`\simeq`	`\doteq`	`\perp`	`\parallel`	`\propto`
Ergebnis	$\ni$	$\simeq$	$\doteq$	$\perp$	$\parallel$	$\propto$

Verneinungen werden mit dem Befehl `\not` umgesetzt: `\not<` erzeugt z. B. $\not<$.

A.3 Akzente und Punkte

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

Akzente sind auch in Texten sehr wesentlich:

```
ò=\‘{o}  ó=\’{o}  ô=\^{o}  õ=\~{o}  ō=\={o}  ȯ=\.{o}
ŏ=\u{o}  ǒ=\v{o}  ő=\H{o}  o̧=\c{o}  ọ=\d{o}  o̲=\b{o}
```

Befehl	`\ldots`	`\cdot`	`\cdots`	`\vdots`	`\ddots`	`$\bullet \bullet$`
Ergebnis	$\ldots$	$\cdot$	$\cdots$	$\vdots$	$\ddots$	$\bullet\bullet$

A.4 Funktionsnamen

Funktionsnamen, wie die der trigonometrischen Funktionen, werden gern in *roman* gesetzt. Um nicht immer mit einer `\mbox`-Konstruktion arbeiten zu müssen, werden auch hierfür Befehle bereitgestellt. Zum Beispiel schreibt man `$\sin x$` und bekommt: $\sin x$. Besonders erwähnt seien: `\det`, `\exp`, `\lg`, `\lim`, `\ln`, `\log`, `\min` und `\max`.

A.5 Klammern und Pfeile

Befehl	`(`	`[`	`\{`	`\lbrack`	`\lfloor`	`\lceil`	`\langle`
Ergebnis	(	[	{	[	⌊	⌈	⟨
Befehl	`)`	`]`	`\}`	`\rbrack`	`\rfloor`	`\rceil`	`\rangle`
Ergebnis	)	]	}	]	⌋	⌉	⟩

Vergrößerte Klammern können durch Vorschaltung eines der Vergrößerungsbefehle umgesetzt werden. Diese lauten: `\big`, `\Big`, `\bigg` und `\Bigg`. Nun einige Pfeile:

`\leftarrow` ←	`\longleftarrow` ⟵	`\uparrow` ↑	`\Leftarrow` ⇐
`\Longleftarrow` ⟸	`\Uparrow` ⇑	`\rightarrow` →	`\longrightarrow` ⟶
`\downarrow` ↓	`\Rightarrow` ⇒	`\Longrightarrow` ⟹	`\Downarrow` ⇓
`\leftrightarrow` ↔	`\longleft...` ⟷	`\updownarrow` ↕	`\Leftrightarrow` ⇔
`\Longleft...` ⟺	`\Updownarrow` ⇕	`\mapsto` ↦	`\longmapsto` ⟼
`\nearrow` ↗	`\searrow` ↘	`\swarrow` ↙	`\nwarrow` ↖

A.6 Hoch und Tief sowie Zwischenräume

Für die Unter- und Überstreichungen verwendet man `\underline` und `\overline`. `$\underline{\overline{a+b}}$` liefert $\underline{\overline{a+b}}$. Mit den Makros `\underbrace` und `\overbrace` lassen sich unter und über Ausdrücken geschweifte Klammern anbringen. Der Befehl `\stackrel` dient dazu, beliebige Objekte über einem Formelteil anzubringen: `$\stackrel{hoch}{tief}$` erzeugt $\stackrel{hoch}{tief}$. Für in der Kombinatorik so wichtigen Binomialkoeffizienten verwende man `\choose`. `${n \choose k}$` führt zum Setzergebnis: $\binom{n}{k}$. `\atop` setzt einen Binomialkoeffizienten ohne Klammern: `${n \atop k}$` → ${n \atop k}$.

Zur Gestaltung von Zwischenräumen in Formeln gibt es die Befehle `\!`, `\,`, `\:`, `\;`, `\quad` und `\qquad`. Der erste Befehl erzeugt einen negativen Zwischenraum (=−3/18 quad), die übrigen positive Zwischenräume: 3/18 quad, 4/18 quad und 5/18 quad.

A.7 Arrays

Die Konstruktion von Matrizen und mathematischen Arrays verläuft entsprechend zu den Text-Tabellen. Für die ersten Versuche muss man nur wissen, dass die Umgebung mit `\begin{array}{`*Spaltenbeschreibung*`}` beginnt und mit `\end{array}` endet. Weitergehende Versuche erfordern einen Blick in eine umfangreichere LaTeX-Beschreibung.

B Weisheit

Man kann ein Pferd zum Wasser bringen, aber nicht zum Trinken zwingen.