Skript zum LaTeX–Kurs

Adrian Bunk (bunk@fs.tum.de)
Jutta Kiener (kiener@fs.tum.de)
Martin Landers (landers@fs.tum.de)
Hannes Keil (keil@fs.tum.de)
Max Berger (max.berger@fs.tum.de)
Michael Rauch (mrauch@fs.tum.de)

20./27. Mai 2000

Version 1.2
(7. April 2004)

Inhaltsverzeichnis

Kapitel 1

LaTeX – Was ist das?

Dies ist eigentlich eine gute Frage. Das erste Kapitel soll einen kurzen Überblick darüber geben, was alles „TeX“ heißt, wer es geschrieben hat und wo die Unterschiede zu anderen Programmen liegen.

1.1 The name of the game

Es gibt viele Produkte, die alle ein TeX im Namen Tragen: TeX, LaTeX, emTeX, teTeX, AMS-TeX, BibTeX. . . . Was wir hier besprechen, heißt im einzelnen:

TeX ist das ursprünglich von Donald E. Knuth geschriebene Textsatzprogramm.

LaTeX ist ein Makro-Paket für TeX vom Leslie Lamport (**La**mports TeX). Die aktuelle Version ist LaTeX 2_ε [1].

teTeX, emTeX, mikTeX sind sog. LaTeX-Distributionen. D.h. hier hat jemand alles, was man zum normalen Arbeiten mit LaTeX benötigt, zusammengestellt. teTeX ist für Unix, emTeX für Dos und mikTeX für Win32 (95/98/NT/2000).

1.2 Und warum jetzt TeX?

In den späten siebziger Jahren schrieb ein Informatikprofessor namens Donald E. Knuth von der Stanford Universität an einer Neuauflage seines Buches „The art of computer programming“. Als Informatiker natürlich mit einem Computer. Doch der dabei entstandene Textsatz war höchst miserabel. Deswegen schrieb er ein Programm namens TeX, das auf den selben Regeln basiert, nach denen Setzer schon seit Jahrhunderten arbeiten.

[1] ε deshalb, weil es schon etwas mehr als Version 2, aber noch weit von Version 3 entfernt ist, sozusagen in einer ε-Umgebung von Version 2

1.3 Was bringt mir dann LaTeX?

Da TeX selbst sehr umständlich zu bedienen ist, gibt es einen Aufsatz, der LaTeX heißt. LaTeX bietet vor allem das sog. „logische Markup“, das in Kapitel 3 beschrieben wird.

Vergleicht man es mit dem traditionellem Buchdruck, so übernimmt TeX die Rolle des Setzers und LaTeX die des Layouters.

1.4 Vorteile von LaTeX gegenüber einem Wortprozessor

Ok. Wir haben also einen Setzter und einen Layouter. Aber wo liegt jetzt der Unterschied zu einem Wortprozessor? Nun, ein Wortprozessor geht folgendermaßen vor: „Was kann ich mit dem Computer machen? Ok, dann mache ich genau das“. Die Philosophie hinter LaTeX hingegen ist: „Wie sieht ein Dokument aus? Ok, das muß ich meinem Computer beibringen“. Das führt dazu, daß LaTeX Dokumente im allgemeinen professioneller aussehen als entsprechende Dokumente aus Wortprozessoren.

TeX ist bei Formeln bisher ungeschlagen. Bereits ein Jahr nach den ersten Anfängen von TeX setzte sich Donald Knuth mit der AMS (American Mathematical Society) zusammen. Selbst große Formeln gehen leicht mit TeX. Das System ist so gut, das sich viele andere Formeleditoren daran orientieren.

TeX hat keine Probleme mit großen Dokumenten. Dadurch, das TeX vom verwendeten Editor unabhängig ist (siehe Kapitel 2), ist es das einfachste Programm zum Schreiben großer Dokumente. Kein langes Warten auf irgendeinen Seitenaufbau, wie man es von WYSIWYG[2] Programmen gewöhnt ist, sondern schnelles und zügiges Arbeiten.

[2]WYSIWYG = What You See is What You Get (was du siehst bekommst du auch). Manche LaTeX-Fans sagen, es geschehe den Wortprozessor Benutzern ganz recht, wenn sie das bekommen, was sie sehen. LaTeX wird in diesem Zusammenhang manchmal auch als WYGIWYW (What you Get Is What You Want, was du bekommst ist das, was du willst) bezeichnet. Aber dazu mehr in Kapitel 2

Kapitel 2

LaTeX-Grundlagen

2.1 Aufruf von LaTeX und auftretende Dateitypen

Wie funktioniert LaTeX? Offenbar ja nicht so wie Word oder andere WYSIWYG (what you see is what you get) Textverarbeitungen. Der in LaTeX verwendete Ansatz wird oft als WYGIWYW (what you get is what you want) beschrieben. Dabei schreibt der Autor den „rohen“ Text – ergänzt um spezielle Befehle – in eine Datei. Den so „angereicherten“ Text nennt man auch *LaTeX-Dokument*, und die Dateien dazu heißen traditionell `*.tex` bzw. `*.ltx`.

Aus dieser „Beschreibung“ des späteren Dokuments erzeugt LaTeX dann – ähnlich einem Compiler für eine Programmiersprache – eine nach den Anweisungen gesetzte, druckfähige Version in Form einer sogenannten *Device Independent Datei* (`*.dvi`). Dies geschieht auf praktisch allen Systemen in der Regel einfach, indem LaTeX mit dem Namen der Sourcecode-Datei aufgerufen wird, also z.B.

```
> latex dokument.tex
```

In der entstehenden DVI Datei (in diesem Fall `dokument.dvi`) ist dann eine (Ausgabe-) geräteunhabhängige, grafische Beschreibung der Seiten des Dokuments gespeichert. Um eine solche Datei am Computer zu betrachten, gibt es unter UNIX das Programm `xdvi`, das einfach mit dem Namen der DVI-Datei aufgerufen wird, z.B.

```
> xdvi dokument.dvi
```

Unter Windows dient hierzu das Programm `yap`. Allerdings ist das DVI-Format auf den meisten Computern nicht ohne weiteres druckbar. Um eine besser für den Ausdruck geeignete *PostScript-Datei* (`*.ps`) zu erzeugen, kann unter UNIX

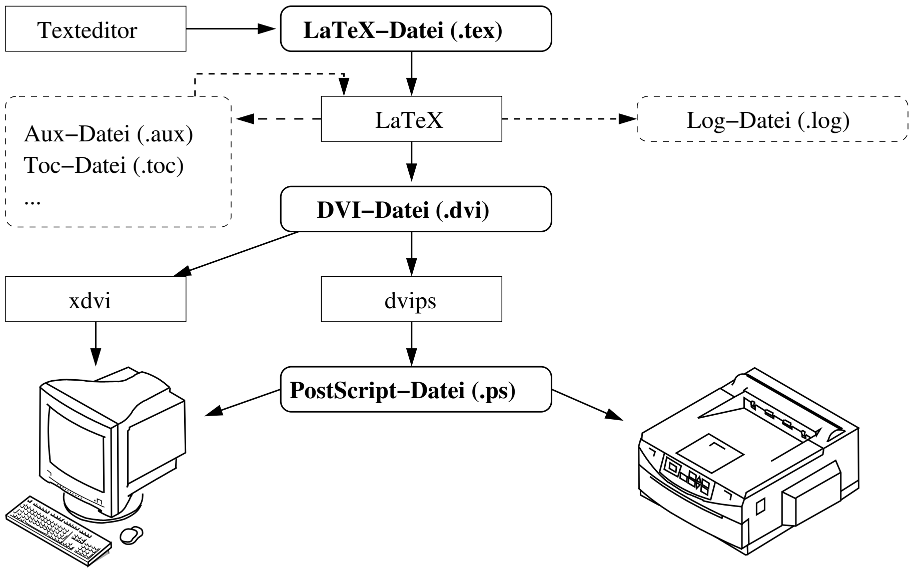


Abbildung 2.1: Datenfluß bei der Arbeit mit LaTeX

das Tool `dvips` verwendet werden. Der Aufruf zur Erzeugung einer Datei ist hier ein wenig komplizierter, da die erzeugte *PostScript Datei* standardmäßig zum Drucker geschickt wird. Der korrekte Aufruf lautet:

```
> dvips -o dokument.ps dokument.dvi
```

In Abbildung 2.1 sind die Zusammenhänge zwischen den gängigen Tools und Dateitypen nochmals illustriert. Anzumerken ist, daß die Namen der Tools sowie ihr korrekter Aufruf nicht eindeutig festgelegt sind und sich von System zu System, bzw. von LaTeX-Distribution zu LaTeX-Distribution unterscheiden können. Allerdings steht auf praktisch jedem System ein sogenannter *local guide* zur Verfügung, der die Besonderheiten der lokalen LaTeX-Installation erläutert.

Häufig beim Arbeiten mit LaTeX auftretende Dateien sind:

`*.tex` *(LaTeX-Dokument)* Ein LaTeX-Dokument.

`*.sty` *(LaTeX-Makropaket)* Ein Makropaket, das mit Hilfe des Befehls `\usepackage` geladen werden kann.

`*.cls` *(LaTeX-Klassendatei)* Diese Dateien enthalten die Definition einer Dokumentklasse, die mit dem `\documentclass` Befehl verwendet werden kann.

`*.dtx` *(Documented TeX)* In dieser Form werden Makropakete für LaTeX aus dem Internet angeboten. Übersetzt man eine solche Datei mit LaTeX, so erhält man den dokumentierten Sourcecode des Pakets.

`*.ins` *(Paketinstallationsdatei)* Diese Datei dient bei Makropaketen für LaTeX dazu, die zugehörige `*.dtx` Datei korrekt „auszupacken“, damit das Paket lokal verwendet werden kann.

`*.dvi` *(Device Independent File)* Beschreibung eines Dokuments in einer (Ausgabe-) geräteunabhängigen Art und Weise.

`*.ps` *(PostScript Datei)* Geräteabhängige Beschreibung eines Dokuments, die von vielen Druckern „verstanden“ wird. Dieses Format wird z.B. durch das Tool `dvips` aus `*.dvi` Dateien erzeugt.

Die folgenden Dateien entstehen oft bei einem LaTeX-Lauf:

`*.log` *(Logdatei)* Eine Logdatei, in der LaTeX Meldungen über die „Kompilation“ des Dokuments ausgibt.

`*.aux` *(Hilfsdatei)* Eine allgemeine Hilfsdatei, in der LaTeX Informationen für einen 2. Lauf speichert, z.B. Informationen über Querverweise.

`*.toc` *(Inhaltsverzeichnis Datei)* Eine Hilfsdatei zur Erzeugung des Inhaltsverzeichnisses, die beim ersten LaTeX-Lauf erzeugt wird und beim zweiten LaTeX-Lauf wieder eingelesen wird (**t**able **o**f **c**ontents).

`*.lof` *(Bildverzeichnis Datei)* Eine Hilfsdatei zur Erzeugung des Bildverzeichnisses, ähnlich einer `*.toc`-Datei (**l**ist **o**f **f**igures).

`*.lot` *(Tabellenverzeichnis Datei)* Eine Hilfsdatei zur Erzeugung des Tabellenverzeichnisses, ähnlich `*.toc` (**l**ist **o**f **t**ables).

`*.idx` *(Rohindex Datei)* Eine Hilfsdatei zur Erzeugung des Indexes, die beim ersten LaTeX-Lauf erzeugt und mit `makeindex` bearbeitet werden muß.

`*.ind` *(Index Datei)* Der fertige Index aus der `*.idx` Datei zur Einbindung in das LaTeX-Dokument.

`*.ilg` *(Index Log)* Eine Logdatei, in der `makeindex` Meldungen ausgibt.

2.2 Befehle und Umgebungen

2.2.1 Besondere Zeichen

Damit LaTeX weiß wie der Text gesetzt werden soll, muß er mit Befehlen versehen werden. Diese Befehle beginnen alle mit einem `\`, um sie vom normalen Text abzusetzen. Weitere für LaTeX besondere Zeichen sind: # % & ˜ _ ˆ $ { }.

Die Bedeutung dieser Zeichen ist:

{,}	Klammern zur Kennzeichnung von Gruppen.
$	Dient der Markierung math. Formeln.
&	Trennt Tabellenspalten.
#	Kennzeichnet Parameter.
_	Tiefstellung in Formeln.
^	Hochstellung in Formeln.
%	Kommentarkennzeichen. Der Rest der Zeile wird ignoriert.
~	Untrennbares Leerzeichen.

Will man ein solches Zeichen im Text verwenden (und nicht als Befehl an LaTeX), muß man ihm einen \ voranstellen. Also:

```
\# \$ \& \~{} \_ \^{} \% \{ \}
```

$ & ~ _ ^ % { }

Die Klammern {} bei der Tilde (˜) und dem Dach (ˆ) sind nötig, damit nicht versucht wird, diese Zeichen als Akzent für das nachfolgende Zeichen zu verwenden (s. auch Seite 13). Macht man das nicht, passiert nämlich folgendes:

```
\~ a \^ b
```

ã b̂

Damit liegt auch schon nahe, wie man LaTeX die deutschen Umlaute eingibt:

```
Da\ss{} in diesem Satz aber
ja \"a \"o und \"u
vorkommen!
```

Daß in diesem Satz aber ja ä ö und ü vorkommen!

Allerdings kann man sich ob der nicht allzu geringen Menge dieser Zeichen in deutschen Texten das Leben noch einfacher machen. Das Paket **german** erlaubt unter anderem die folgende, vereinfachte Eingabe:

```
Da"s in diesem Satz aber
ja "a "o und "u
vorkommen!
```

Daß in diesem Satz aber ja ä ö und ü vorkommen!

Nimmt man noch das Paket **inputenc** hinzu, so können die Umlaute direkt eingegeben werden, allerdings um den Preis einer gewissen Abhängigkeit von der Umgebung, da diese Zeichen je nach Rechner oder Betriebssystem unterschiedlich dargestellt werden. Nähere Informationen zu Paketen im allgemeinen und diesen beiden Paketen gibt es in Kapitel 11.

Will man schließlich einen normalen \ im Text verwenden, so benutzt man nicht etwa \\ (das ist ein Kommando für einen Zeilenumbruch), sondern das Kommando `\textbackslash`.

2.2.2 Format der Befehle

Wie genau sieht ein solcher Befehl in LaTeX aus? Wie eben schon bemerkt, beginnt ein Befehl in LaTeX immer mit einem \, gefolgt von einem einzigen

Sonderzeichen oder einem Namen der nur aus Buchstaben besteht. Außerdem kann ein Befehl noch Argumente erfordern.

Ein einfacher Befehl an LaTeX ist etwa:

```
\hline
```

Dieser Befehl veranlaßt LaTeX, eine horizontale Linie in einer Tabelle einzufügen, ähnlich dem Befehl <hr> in HTML, nur halt auf Tabellen beschränkt. Ein etwas komplexerer Befehl ist z.B.

`\chapter`[*Titel f. Inhaltsverzeichnis*]{*Titel*}

Dieser Befehl teilt LaTeX den Beginn eines neuen Kapitels im Dokument mit, wobei ein unterschiedlicher *Titel für das Inhaltsverzeichnis* verwendet werden soll als der *Titel* für den Kapitelanfang.

Allgemein haben Befehle in LaTeX die folgende Form:

`\befehl`[*opt. Argument*]{*Argument*} {*Argument*}...

Dabei gilt: Argumente in [eckigen] Klammern sind sogenannte *optionale Argumente*,

die (inkl. den zugehörigen eckigen Klammern) weggelassen werden können. Argumente in {geschweiften} Klammern hingegen sind *verpflichtende Argumente*, die immer angegeben werden müssen. Die Anzahl der verpflichtenden und optionalen Argumente ist von Befehl zu Befehl unterschiedlich. Insbesondere gibt es auch Befehle, die ganz ohne Argumente auskommen, wie das bereits erwähnte `\hline`. Außerdem ist es wichtig zu wissen, daß LaTeX Groß- und Kleinschreibung bei Befehlen unterscheidet. Daher gilt:

`\befehl` ist nicht gleich `\Befehl` oder `\bEfeHl`

Im Englischen spricht man auch von *case-sensitiveness*. Ein weiterer wichtiger Punkt ist, daß LaTeX sämtliche Leerzeichen nach Befehlen ohne Argumente ignoriert. Um LaTeX daran zu hindern, nach einem solchen Befehl sämtliche Leerzeichen „aufzufressen", muß man entweder `{}` an den Befehl anhängen und ihn damit als „beendet" erklären, oder mit `\ ` (einem \ gefolgt von einem Leerzeichen) explizit ein Leerzeichen verlangen. Ein Beispiel:

```
I read that Knuth divides the
people that work with \TeX{} into
\TeX{}nicians and \TeX perts.\newline
Is your \TeX \ good enough to be
a \TeX    pert?
```

I read that Knuth divides the people that work with TeX into TeXnicians and TeXperts.
Is your TeX good enough to be a TeXpert?

Bei Befehlen mit Argumenten ist dies wie gesagt *nicht* nötig.

2.2.3 Format von Umgebungen

Neben einzelnen Befehlen gibt es in LaTeX noch eine zweite große Klasse von Anweisungen, die *Umgebungen* genannt werden. Eine Umgebung unterscheidet sich von einem einfachen Befehl dadurch, daß sie einen Anfang und ein Ende hat, die durch spezielle Befehle gekennzeichnet werden. Allgemein hat eine Umgebung in LaTeX die folgende Form

`\begin`{*Umgebungsname*}[opt. Argument]{*Argument*}{*Argument*}...
LaTeX-Befehle und Text
`\end`{*Umgebungsname*}

und ist eigentlich nur eine Sonderform der Gruppe, die mittels {...} markiert wird. Sowohl für Gruppen als auch für Umgebungen gilt: Alle Einstellungen die darin vorgenommen werden (also z.B. die Auswahl eines bestimmten Fonts o.ä.) gelten nur bis zum jeweiligen Ende. Anfang und Ende einer Umgebung werden durch die speziellen Befehle `\begin{`*`Umgebungsname`*`}` und `\end{`*`Umgebungsname`*`}` definiert. Außerdem kann eine Umgebung genauso wie ein normaler Befehl optionale und verbindliche Argumente erfordern. Ein Beispiel für eine einfache Umgebung ist die `center` Umgebung, die den darin enthaltenen Text zentriert setzt.

```
\begin{center}
Dieser Text wird\\
zentriert gesetzt!!
\end{center}
```

Dieser Text wird
zentriert gesetzt!!

2.2.4 Längenangaben

Abschließend soll hier noch auf einen oft in LaTeX benötigten Parametertypus eingegangen werden: Die Angabe von Längen. Viele Befehle brauchen als Argument eine Längenangabe. Diese hat in LaTeX allgemein die Form einer vorzeichenbehafteten Dezimalzahl (mit Dezimalpunkt oder -komma) mit einer Einheit. Also z.B.

...{-3.0cm} oder ...{5,3em}

Als Einheiten können verwendet werden: bp, cc, cm, dd, em, ex, mm, in, pc, pt und sp. Dabei lassen sich diese Einheiten wie folgt umrechenen: 1sp = 65536pt; 12pt = 1pc; 1238pt = 1157dd; 12dd = 1cc; 72.27pt = 1in = 72bp = 25,4mm = 2,54cm. Die Einheiten em und ex sind von der aktuellen Schriftgröße abhängig und geben die Breite eines „x“ (ex) beziehungsweise die Höhe des Buchstaben „M“ (em) an.

2.3 Aussehen einer LaTeX-Datei

Wir wissen bereits, daß ein LaTeX-Dokument aus dem „rohen“ Text, angereichert um Befehle an LaTeX, besteht. Doch wie genau sieht so eine Datei aus? Und was gibt es zu beachten? Darauf soll hier eingegangen werden.

Ein erster wichtiger Punkt bei der Erstellung einer LaTeX-Datei ist zu wissen, wie LaTeX mit sogenanntem *„whitespace“* (etwa Leerzeichen, Tab oder Zeilenumbruch) umgeht. LaTeX behandelt nämlich alle diese sog. whitespace-Zeichen als „Leerzeichen“. Mehrere *aufeinanderfolgende* whitespace-Zeichen werden als *ein einziges* „Leerzeichen“ aufgefaßt, whitespace-Zeichen am Anfang einer Zeile werden normalerweise ignoriert und ein einfacher Zeilenumbruch gilt als ein einzelnes „Leerzeichen“. Eine Leerzeile zwischen zwei Textzeilen beginnt einen neuen Absatz. Mehrere *aufeinanderfolgende* Leerzeilen werden als *eine* Leerzeile aufgefaßt. Ein Beispiel soll dies verdeutlichen:

```
Wie man    sehen kann  gelten
mehrere   Leerzeichen und sogar
Zeilenumbr"uche   als einzelne
"‘Leerzeichen"’.

Und eine  Leerzeile beginnt
einen neuen Absatz.
```

Wie man sehen kann gelten mehrere Leerzeichen und sogar Zeilenumbrüche als einzelne „Leerzeichen“.

Und eine Leerzeile beginnt einen neuen Absatz.

Allgemein gilt: Ein LaTeX-Dokument gliedert sich in zwei Teile, die sog. *Präambel*, manchmal auch *Header* genannt und das eigentliche Dokument, selten auch *Body* genannt. In der Präambel, die mit dem `\documentclass` Befehl beginnt, wird der Typ des Dokuments festgelegt, eventuell benötigte Zusatzpakete mit der `\usepackage` Anweisung geladen und sämtliche nötigen, globalen Einstellungen vorgenommen. Danach beginnt die `document`-Umgebung, in der das eigentlich Dokument enthalten ist. Der Befehl `\begin{document}` markiert gleichzeitig das Ende der Präambel und den Anfang des eigentlichen Dokuments. Ein Beispiel:

```
\documentclass[a4paper]{article}
\usepackage{german}
\author{L. A. Tex}
\title{Der Erste (von Vielen?)}

\begin{document}


\maketitle


Dies ist mein erster Text mit \LaTeX{} und
ich glaube, es werden noch viele folgen.


\end{document}
```

Dieses komplette und syntaktisch korrekte erste Beispiel gliedert sich wie oben erklärt in die folgende Präambel:

```
\documentclass[a4paper]{article}
\usepackage{german}
\author{L. A. Tex}
\title{Der Erste (von Vielen?)}
```

und den folgenden Dokumentkörper:

```
\begin{document}

\maketitle

Dies ist mein erster Text mit \LaTeX{} und
ich glaube es werden noch viele folgen.

\end{document}
```

Die Präambel beginnt mit dem Befehl `\documentclass[a4paper]{article}`. Dieser teilt LaTeX mit, daß es sich bei diesem Dokument um einen kurzen Artikel handelt, der besonders auf eine Ausgabe auf A4 Papier hin optimiert werden soll. Die Angabe des `\documentclass` Befehls ist übrigens verpflichtend – ohne diese Angabe geht in LaTeX nichts. Die gängigsten Dokumentklassen und wichtige Optionen dafür sind im Kapitel 11 aufgeführt.

Als nächstes wird mit `\usepackage{german}` ein Paket zur besseren Unterstützung der deutschen Sprache geladen und mit `\author` und `\title` der Autor und der Titel des Dokuments für die Nachwelt festgehalten. Außerdem könnte hier noch das Datum mittels `\date` festgelegt werden. Im eigentlichen Dokument wird dann zuerst mit `\maketitle` aus den Angaben in der Präambel eine Titelseite für das Dokument erzeugt (die im Falle eines `article`s allerdings nicht eine ganze eigene Seite bekommt) gefolgt von normalem Fließtext. Was das Kommando `\LaTeX` tut, ist vermutlich einfach zu erraten. Nein? Na dann – weil's so schön ist – gleich ein paar mehr Kommandos dieser Sorte:

`\TeX{} \LaTeX{} \LaTeXe{}`	TeX LaTeX LaTeX 2_ε

Damit sind wir am Ende dieses Kapitels angelangt, und nachdem hoffentlich ein erster Eindruck von der „Wirkungsweise“ von LaTeX und einige Grundlagen vermittelt worden sind, können wir uns nun dem Konzept des logischen Markups zuwenden.

Kapitel 3

Logisches Markup

Auf den ersten Blick mag es so scheinen, als ob der Hauptunterschied zwischen WYSIWYG (what you see is what you get) Programmen und LaTeX darin besteht, daß man in LaTeX eine Textdatei editiert und erst nach einer Kompilation das Dokument betrachten kann. Außerdem fällt einem vielleicht auch noch auf, daß man in LaTeX normalerweise nicht mit der Maus arbeitet. Der größte Unterschied besteht aber in Wirklichkeit zwischen logischem Markup in LaTeX und visuellem Markup in vielen anderen Programmen.

3.1 Was ist Markup?

Als *Markup* wurden früher die Anweisungen an den Setzer bezeichnet, wie ein Manuskript zu setzen ist. Ein Beispiel dafür wäre (aus [GMS95]):

> Diese Überschrift in 12 Punkt Helvetica kursiv über einen 10 Punkt Text setzen, ausgerichtet an einem 22 Pica Typenstein mit 1 em Einzug links und ohne rechts.

Heutzutage werden mit *Markup* jene Anweisungen bezeichnet, die besagen, wie ein Dokument formatiert werden soll.

3.2 Visuelles Markup

Bei visuellem Markup wird die *Formatierung* des Dokuments beschrieben.

Ein Beispiel für eine Überschrift:

Meine Katze

Bei visuellem Markup würde diese Überschrift folgendermaßen beschrieben werden:

> Setze den Text „Meine Katze“ zentriert in einer fetten Schrift der Größe 17 Punkt.

Beispiele für Programme, die mit visuellem Markup arbeiten, sind TEX (das Programm auf dem LATEX aufbaut) und die meisten gängigen Textverarbeitungsprogramme.

3.3 Logisches Markup

Bei logischem Markup („logisch“ weil die logische Struktur des Dokuments beschrieben wird) wird die *logische Struktur* des Dokuments beschrieben und der Inhalt ist von der Formatierung getrennt.

Unser Beispiel von gerade eben

Meine Katze

würde bei logischem Markup folgendermaßen beschrieben werden:

> Hierhin kommt eine Überschrift mit dem Text „Meine Katze“.

und an anderer Stelle (im Dokument oder in einer eigenen Beschreibungsdatei) würde stehen:

> Setze alle Überschriften zentriert in einer fetten Schrift der Größe 17 Punkt.

Beispiele für die Verwendung von logischem Markup sind neben LATEX z.B. SGML und XML.

3.4 Vorteile von logischem Markup

Logisches Markup hat die folgenden Vorteile gegenüber visuellem Markup:

- Visuelles Markup verlangt viel Wissen darüber, was gut aussieht bzw. gut lesbar ist. Bei logischem Markup kann man einfacher auf das Wissen von anderen zurückgreifen.
- Bei logischem Markup ist der Inhalt von der Formatierung getrennt. Das heißt, es ist egal, ob man zuerst den Text schreibt und sich dann eine Formatierung überlegt oder umgekehrt. Es kann sich auch jemand anders um die Formatierung des Dokuments kümmern.

3.5 Vor- und Nachteile beim Arbeiten mit LaTeX

LaTeX ist nicht das optimale für jemanden, der sagt: *„Ich weiß selber am Besten, wie mein Dokument aussehen soll.“*. Man kann zwar mit LaTeX im Prinzip alles machen, aber man muß entweder Glück haben und jemand anders hat schon ein entsprechendes Paket (siehe Abschnitt 11.2) geschrieben, das man verwenden kann, oder man muß oft sehr viel Wissen über TeX haben.

LaTeX ist dagegen sehr gut geeignet, wenn man sagt *„Ich möchte mich nur um den Inhalt, aber nicht um die Formatierung kümmern. Es soll aber trotzdem gut aussehen.“* oder wenn man sich einmal um die Formatierung kümmern und dann viele gleichaussehende Dokumente erstellen möchte.

3.6 Logisches Markup in LaTeX

In LaTeX arbeitet man mit logischem Markup, indem man vordefinierte und selbstdefinierte Befehle und Umgebungen verwendet. Wie man das macht, wird in Kapitel 5 beschrieben. Die Formatierung wird dann bei der Definition des Befehls oder der Umgebung festgelegt.

Kapitel 4

Gliederungs- und Gestaltungskommandos

4.1 Dokumentklassen

Jedes Dokument hat ein Grundlayout, welches durch eine Dokumentklasse festgelegt wird.

Die normalerweise verwendeten Dokumentklassen sind:

- `article` (kleines Dokument)
- `report` (größeres Dokument)
- `book` (Buch)

Auf die genauen Unterschiede zwischen diesen Dokumentklassen wird in Kapitel 11 eingegangen.

4.2 Dokumentuntergliederung

Die logische Unterteilung des Dokuments in Kapitel, welche aus Abschnitten, welche wiederum aus Unterabschnitten usw. bestehen, wird durch die in Tabelle 4.1 aufgeführten Befehle erreicht.

`\chapter` (nur in den Dokumentklassen `report` und `book`)
`\section`
`\subsection`
`\subsubsection`
`\paragraph`
`\subparagraph`

Tabelle 4.1: Dokumentuntergliederung (in absteigender Reihenfolge)

`\chapter` ist in der Dokumentklasse `article` nicht vorhanden, damit man mehrere Artikel leichter zu einem größeren Tagungsband, Buch o.ä. zusammenfassen kann.

4.3 Beispiel für eine Dokumentuntergliederung

```
\documentclass[a4paper]{report}

\usepackage{german}

\begin{document}

\chapter{Meine Hobbies}
Text über meine Hobbies

\chapter{Meine Haustiere}
\section{Meine Katze}
Text über meine Katze

\section{Mein Hund}
Text über meinen Hund

\end{document}
```

In diesem Beispiel gibt es zwei Kapitel von denen das Zweite noch einmal in zwei Abschnitte unterteilt ist.

4.4 Inhaltsverzeichnis

Dank logischem Markup bekomme ich in LaTeX das Inhaltsverzeichnis mit einem Befehl (`\tableofcontents`):

```
\documentclass[a4paper]{report}

\usepackage{german}

\begin{document}

\tableofcontents

\chapter{Meine Hobbies}
Text über meine Hobbies

\chapter{Meine Haustiere}
```

```
\section{Meine Katze}
Text über meine Katze

\section{Mein Hund}
Text über meinen Hund

\end{document}
```

Hierdurch wird das Inhaltsverzeichnis automatisch erzeugt.

Achtung: Damit das Inhaltsverzeichnis erzeugt wird, muß man LaTeX *zweimal* aufrufen! Das liegt daran, daß bei einem Aufruf von LaTeX die Informationen über das Inhaltsverzeichnis in eine spezielle Datei geschrieben werden, welche erst beim nächsten Aufruf neu eingelesen wird.

Genauso funktionieren auch `\listoffigures` für ein Abbildungsverzeichnis und `\listoftables` für ein Verzeichnis aller Tabellen.

4.5 Seitenstile

Das allgemeine Aussehen der Seiten (Kopf- und Fußzeilen) kann man ab der aktuellen Seite mit `\pagestyle` und nur für die gerade aktuelle Seite mittels `\thispagestyle` festlegen. Die Standardseitenstile sind in Tabelle 4.2 aufgeführt.

`empty`	Kopf- und Fußzeile sind leer
`plain`	Kopfzeile leer, Fußzeile enthält die Seitennummer
`headings`	Kopfzeile: Informationen über das Kapitel, Fußzeile: leer

Tabelle 4.2: Standardseitenstile in LaTeX

Mit `\pagestyle`{*plain*} kann man also z.B. dafür sorgen, daß die Kopfzeilen leer sind und in der Fußzeile nur die Seitennummer steht.

Zur Definition von eigenen Seitenstilen sei auf ([GMS95], Seiten 91-99) verwiesen.

4.6 Verschiedene Gestaltungskommandos

4.6.1 `\emph`

In einem Text zum Durchlesen sollten Hervorhebungen (engl. *emphasize*) nicht aufdringlich durch fette Schrift sondern durch `\emph{kursiv}` *kursiv* erfolgen.

Man kann `\emph` auch umdefinieren. Wie das geht, wird in Kapitel 5 beschrieben.

4.6.2 Aufzählungen

Es gibt in LaTeX numerierte und nicht numerierte Aufzählungen, die man auch ineinander verschachteln kann. Bei numerierten Aufzählungen erfolgt die Durchnumerierung automatisch.

Beispiel für eine nicht numerierte Aufzählung:

```
\begin{itemize}
  \item irgendetwas
  \item noch etwas anderes
\end{itemize}
```

- irgendetwas
- noch etwas anderes

Anmerkung: Die Einrückungen ignoriert LaTeX, sie verbessern aber die Lesbarkeit.

Beispiel für eine numerierte Aufzählung:

```
\begin{enumerate}
  \item irgendetwas
  \item noch etwas anderes
\end{enumerate}
```

1. irgendetwas
2. noch etwas anderes

Bei numerierten Aufzählungen kümmert sich LaTeX automatisch um die Numerierung.

Man kann `\item` auch einen Parameter mitgeben, der aussagt, was statt der normalen Numerierung bzw. dem Symbol dargestellt werden soll. Bei numerierten Aufzählungen wird dabei die Numerierung nicht weitergezählt:

```
\begin{enumerate}
  \item[-+=] irgendetwas
  \item noch etwas anderes
  \item und noch etwas
\end{enumerate}
```

-+= irgendetwas

1. noch etwas anderes
2. und noch etwas

4.6.3 Ein etwas komplizierteres Beispiel für Aufzählungen

```
\begin{enumerate}
  \item[abc] irgendetwas
  \item noch etwas anderes
    \begin{itemize}
      \item nichtnumerierte
            Aufz"ahlung
        \begin{itemize}
          \item nichtnumerierte
        Aufz"ahlung
          \item[***] etwas Text
        \end{itemize}
      \item es geht noch weiter
        \begin{enumerate}
          \item noch mehr Text
        \end{enumerate}
      \item der letzte Text
    \end{itemize}
\end{enumerate}
```

abc irgendetwas

1. noch etwas anderes
   - nichtnumerierte Aufzählung
     – nichtnumerierte Aufzählung
     *** etwas Text
   - es geht noch weiter
     (a) noch mehr Text
   - der letzte Text

Kapitel 5

Eigene Befehle und Umgebungen

Folgt man dem in LaTeX vorgesehenen Weg des logischen Markups (s. Kapitel 3), so dauert es nicht lange, bis man an den Punkt kommt, an dem es einfach keinen vordefinierten Befehl gibt, der ein gewisses Textstück ideal beschreiben würde. In anderen Umgebungen wäre man jetzt vielleicht gezwungen zurückzustecken und sich mit einem eventuell unbefriedigenden Kompromiß zufriedenzugeben. Nicht so in LaTeX! Hier existiert eine sehr elegante Methode zur Lösung dieses Problems: Die Definition eigener Befehle und Umgebungen. Dazu dienen die Befehle `\newcommand` und `\newenvironment`, die eine sehr bequeme Definition, sogar inklusive optionaler und verbindlicher Parameter, erlauben.

5.1 Erstellung eigener Befehle

Es gilt hier zwei unterschiedliche Situationen zu unterscheiden. Existiert das Kommando, das ich definieren will schon, oder noch nicht? Falls ein Kommando mit dem gewählten Namen schon existiert, muß der Befehl umdefiniert werden. Falls so ein Kommando nicht existiert, muß es neu definiert werden.

5.1.1 Neudefinition von Befehlen

Die Definition eines eigenen Befehls mittels `\newcommand` funktioniert folgendermaßen:

`\newcommand{\`*`Name`*`}[`*Parameter*`] [`*Default*`]{`*Definition*`}`

Name ist naheliegenderweise der Name des zu definierenden neuen Befehls und muß den Regeln für alle Befehle in LaTeX (dem sog. *Befehlsformat*) entsprechen, d.h. entweder aus einem Sonderzeichen oder mehreren Buchstaben bestehen. Mit *Parameter* kann, falls nötig, angegeben werden, wieviele Parameter der Befehl bekommt, wobei diese Anzahl auf maximal 9 beschränkt ist. Wird der

Parameter *Default* angegeben, so bewirkt er, daß das erste Argument des neuen Befehls optional wird und gibt, für den Fall daß es nicht angeben wird, den Standardwert vor. In der *Definition* des Befehls steht letztlich der Text und die Befehle, die anstelle des Befehls eingesetzt werden. Dabei kann auf die Parameter mit #1 bis #9 zugegriffen werden. Einige Beispiele sollen dabei helfen, diese etwas komplizierte Materie zu verdeutlichen.

So etwa definiert das folgende Beispiel einen neuen Befehl `\werbung` und wendet Ihn gleich an.

```
\newcommand{\werbung}
{\LaTeX{} ist toll!}
Und jetzt: \werbung
```

Und jetzt: LaTeX ist toll!

Ein etwas komplizierteres Beispiel:

```
\newcommand{\hello}[1]
{#1
world!}

\hello{Hello}\\
\hello{Good morning}
```

Hello world!
Good morning world!

Hier sieht man, wie mit Hilfe von #1 auf den ersten (und einzigen) Parameter des Kommandos zugegriffen werden kann. Außerdem ist festzuhalten, daß sich LaTeX auch bei der Definition eines Befehls nicht an eventuellen Zeilenumbrüchen stört. Aber mal ehrlich, das haben wir doch schon vermutet, oder? Wichtig ist hierbei noch, daß beim Versuch das Kommando mit der falschen Anzahl an Parametern (insbesondere *ohne* Parameter) aufzurufen, ein Fehler auftritt. In dieser Hinsicht können wir unseren neuen Befehl also noch verbessern:

```
\newcommand{\hello}[1]
[Hello]{#1 world!}

\hello\\
\hello[Hello]\\
\hello[Good night]
```

Hello world!
Hello world!
Good night world!

Durch die Angabe des *Default*-Wertes für das erste Argument in den zweiten eckigen Klammern wird das erste Argument jetzt optional, und unser neuer Befehl kann auch ganz ohne Argumente aufgerufen werden. Wichtig ist hierbei, daß das erste Argument jetzt, wenn es angegeben wird, in *eckigen* Klammern angegeben werden muß – wie es eben Konvention in LaTeX ist.

Um hier noch ein etwas sinnvolleres Beispiel aufzuführen, soll hier ein kleines Kommando für die Anrede in einem Brief deklariert werden, allerdings nicht ohne darauf hinzuweisen, daß Briefeschreiben mit Paketen wie **dinbrief** oder **g-brief** deutlich leichter von der Hand geht als von selbiger. Genug der Vorrede, definiert den Befehl!

```
\newcommand{\anrede}[2]
[Sehr geehrte(r)]
{#1 #2,\\}

\anrede{Herr Mayer}
wie in unserem
letzten Schreiben\ldots

\anrede[Liebe]{Frau K.}
wie ist es Ihnen
ergangen\ldots
```

Sehr geehrte(r) Herr Mayer,
wie in unserem letzten Schreiben...

Liebe Frau K.,
wie ist es Ihnen ergangen...

5.1.2 Redefinition von Befehlen

Was aber passiert, wenn es schon einen gleichnamigen Befehl gibt? Falls bereits ein Befehl des selben Namens existiert und versucht wird ihn mit `\newcommand` neu zu definieren, so bricht LaTeX mit einer Fehlermeldung ab, um eine versehentliche Neudefinition zu vermeiden. Will man aber absichtlich einen schon existierenden Befehl neu definieren, so gibt es dafür den Befehl `\renewcommand`. Dieser Befehl funktioniert genauso wie `\newcommand` mit dem Unterschied, daß er *ausschließlich* bereits existierende Befehle verändert. D.h. man erhält einen Fehler, wenn der Befehl noch *nicht* definiert war bei der Anwendung von `\renewcommand`. Ein in dieser Hinsicht fehlerfreies (allerdings auch sinnfreies) Beispiel:

```
\newcommand{\test}{test}
\test 'mer mal.

\renewcommand{\test}[1]
{#1}
\test{probiern} 'mer mal.
```

test'mer mal.

probiern 'mer mal.

An dieser Stelle ist auch noch einmal schön zu sehen, daß LaTeX sämtliche Leerzeichen nach einem Befehl *ohne Argumente* ignoriert, während sie nach einem Befehl mit Argumenten beachtet werden.

Gewissermassen als ein Zwitter aus `\newcommand` und `\renewcommand` existiert noch ein dritter Befehl zur Definition eigener Befehle namens `\providecommand`. Damit wird ein neuer Befehl *nur dann* neu definiert, wenn es ihn noch *nicht* gibt.

Zusätzlich existiert von jedem der drei Befehle noch eine *-Variante, die als Argumente jeweils nur max. einen Absatz akzeptiert und einem damit in der Regel eine ganze Menge Fehlersuche abnimmt, denn wenn ein Befehl mehr als einen einzelnen Absatz als Argument bekommt, ist zumeist ohnehin irgendetwas gründlich schiefgegangen. Daher sollte `\newcommand*` der Normalfall sein.

5.2 Erstellung eigener Umgebungen

Neben der Definition eigener Befehle ist es in LaTeX auch möglich, eigene Umgebungen zu definieren. Dies geschieht in Analogie zur Definition eigener Befehle (durch `\newcommand`) mittels `\newenvironment`. Der Aufruf von `\newenvironment` ähnelt dabei stark dem von `\newcommand` und hat die folgende Form:

`\newenvironment`{*Name*}[*Parameter*][*Default*]{*Start-Definition*} {*End-Definition*}

Name ist dabei wieder der Name der neu zu definierenden Umgebung, darf aber aus Buchstaben, Ziffern und dem * bestehen (das ist das sog. *Umgebungsformat*). Die Angabe des optionalen und der verpflichtenden Parameter mittels *Parameter* und *Default* funktioniert auf die gleiche Weise wie bei der Definition eigener Befehle. Die Änderung besteht darin, daß es statt einer einzelnen Definition eine *Start-Definition* und eine *End-Definition* gibt. Dabei werden die Befehle und der Text der *Start-Definition* am Anfang der neuen Umgebung eingefügt und Befehle und Text der *End-Definition* an deren Ende. Innerhalb der *Start-Definition* – und nur dort – kann wie bei der Definition neuer Befehle mittels #1 bis #9 auf die Parameter zugegriffen werden. Ein einfaches Beispiel:

```
\newenvironment{stack}
{Top of Stack\\}{Stack Bottom}

\begin{stack}
135\\
'a'\\
\end{stack}
```

Top of Stack 135 'a' Stack Bottom

In diesem Beispiel wird die einfache Umgebung `stack` definiert, die nur einen entsprechenden Text an ihrem Anfang und Ende ausgibt. Etwas komplizierter wird es, wenn man noch die Möglichkeit zur Parameterübergabe dazunimmt.

```
\newenvironment{stack}
[1][Top of Stack]
{#1\\}
{Stack Bottom}

\begin{stack}
135\\
'a'\\
\end{stack}

\begin{stack}[Oben]
'b'\\
'a'\\
\end{stack}
```

Top of Stack 135 'a' Stack Bottom Oben 'b' 'a' Stack Bottom

Wie bei der Definition einfacher Befehle gibt es auch hier noch einen Befehl `\renewenvironment` und die Formen mit *, die nur je max. einen Absatz als Parameter akzeptieren. Allerdings gibt es kein Kommando `\provideenvironment`!

Wirklich sinnvoll sind neue Umgebungen normalerweise aber ohnehin erst in Kombination mit der Definition dazu passender, neuer Befehle. Richtig angewendet sind sie dann aber ein sehr mächtiges Werkzeug.

5.3 Befehle zum visuellen Markup

Mit der Möglichkeit, eigene Befehle in LaTeX zu definieren, ergibt sich schnell die Notwendigkeit, neben dem logischen nun doch auch visuelles Markup einzusetzen, um die neuen Befehle „mit Leben zu füllen“. In diesem Falle spricht auch nichts dagegen visuelles Markup einzusetzen, da ja die logische Struktur durch die Einführung neuer Befehle aufrechterhalten wird. LaTeX stellt daher auch Befehle zum visuellen Markup zur Verfügung. Allerdings sollte aus den in Kapitel 3 genannten Gründen von einem häufigen Einsatz außerhalb eigener Befehle im normalen Fließtext abgesehen werden.

5.3.1 Schriftart und -größe

Eine der häufigsten Anpassungen visueller Art in der Textverarbeitung sind wohl Veränderungen von Schriftgröße und Schriftart, insbesondere da diese in den meisten WYSIWYG-Textverarbeitungen wie etwa Word sehr einfach vorzunehmen sind. Auch in LaTeX ist es natürlich möglich, die Schriftgröße und Schriftart des Textes zu beeinflussen, allerdings auf eine andere Art als in Word o.ä. Im Gegensatz zu den festen Angaben, die man in einer WYSIWYG-Textverarbeitung macht, wie etwa „Diesen Text in 16pt mit der Schriftart Arial“, gibt es in LaTeX nur „relative“ Befehle in der Art von „Setze diesen Text etwas größer als normal, mit einem Font ohne Serifen“. Dieser Ansatz mag zwar etwas umständlich klingen, hat aber durchaus Vorteile. Zum einen läßt sich später die Schriftgröße im ganzen Dokument sehr einfach verändern, da sämtliche Angaben relativ zu einer Grundschriftgröße sind. Und zum anderen stört es LaTeX nicht im Geringsten, wenn es keine Schrift mit dem Namen „Arial“ gibt, da es nur weiß, daß es eine Schrift ohne Serifen verwenden soll. (Die Problematik „fehlender Schriften“ dürfte jedem www-Benutzer bekannt sein, der schon einmal mit einem UNIX-Netscape eine für Windows „optimierte“ Seite betrachtet hat : −))

Die zur Verfügung stehenden Befehle zur Festlegung der Textgröße in LaTeX werden in der nächsten Tabelle aufgelistet.

`\tiny`	ganz wuzlig	`\Large`	noch größer
`\scriptsize`	Indexgröße	`\LARGE`	g'scheit groß
`\footnotesize`	wie eine Fußnote		
`\small`	klein	`\huge`	riesig
`\normalsize`	normal		
`\large`	groß	`\Huge`	monströs

Wobei `\normalsize` die in der `\documentclass` (s. Kapitel 11) ausgewählte Grundschriftgröße des Dokuments ist. Die Befehle zur Änderung der Schriftgröße wirken dabei wie „Schalter", die solange den nachfolgenden Text beeinflussen, bis sie entweder ausgeschaltet werden (etwa durch `\normalsize`), oder bis die aktuelle Gruppe oder die aktuelle Umgebung zu Ende ist. Hierzu ein Beispiel:

```
Mach mal \Large gro"s
aber dann wieder
\normalsize normal.

Mit einer
{\Large Gruppe}
hat man das fast
noch besser im Griff.
```

Mach mal groß aber dann wieder normal.

Mit einer Gruppe hat man das fast noch besser im Griff.

Kann man sich die Unart, fixe Schriftgrößen anzugeben, gar nicht abgewöhnen, oder benötigt man ein besonders großes Font, kann man auf die folgende Weise auch „vom Pfad der Tugend" abweichen: `\fontsize`{*Größe*}{*Zeilenabstand*} `\selectfont` wobei der Zeilenabstand nicht direkt angegeben wird, sondern als Zeilenabstand für ein Font der Größe *Zeilenabstand*. Allerdings geht hierbei auf jeden Fall der Vorteil der einfachen, dokumentweiten Änderung der Schriftgröße verloren, und daher ist diese Möglichkeit nur als absolute Notlösung anzusehen.

```
\fontsize{30pt}{30pt}\selectfont
GROSSER TEXT!
\normalfont
```

GROSSER TEXT!

Zur Auswahl einer Schriftart existieren die folgenden Befehle:

`\textrm{...}`	`\rmfamily`	Font mit Serifen
`\textsf{...}`	`\sffamily`	Serifenloses Font
`\texttt{...}`	`\ttfamily`	`Maschinenschrift (Typewriter)`
`\textmd{...}`	`\mdseries`	Medium Font
`\textbf{...}`	`\bfseries`	**Fettes Font**
`\textup{...}`	`\upshape`	Aufrechtes Font (nicht kursiv)
`\textit{...}`	`\itshape`	*Kursives Font*
`\textsl{...}`	`\slshape`	*Geneigtes Font*
`\textsc{...}`	`\scshape`	FONT MIT KAPITÄLCHEN
`\textnormal{...}`	`\normalfont`	Defaultfont des Dokuments

Die Befehle existieren, wie in der Auflistung zu sehen, sowohl als Schalter, die wirken bis sie aufgehoben werden bzw. das Ende der aktuellen Gruppe oder Umgebung erreicht ist, als auch als einfacher Befehl mit einem Argument. Auch hierzu ein Beispiel:

```
Kommt ein \textsc{Kapit"alchen} zum
\bfseries \itshape fetten, kursiven
Font \normalfont und bestellt:
{\sffamily Einmal
\texttt{Schreib\-maschine} bitte.}
```

Kommt ein KAPITÄLCHEN zum ***fetten, kursiven Font*** und bestellt: Einmal `Schreibmaschine` bitte.

Für längere Textpassagen gibt es sogar noch die Möglichkeit, eine Umgebung zu verwenden:

```
\begin{Large}
Des is gro"s!
Und des a no!
\end{Large}
```

Des is groß! Und des a no!

Eine Möglichkeit, Fonts direkt per Name wie in WYSIWYG-Textverarbeitungen auszuwählen, ist dem Verfasser nicht bekannt, und er ist froh darüber...

5.3.2 Textausrichtung

Eine fast noch häufiger gebrauchte Möglichkeit des visuellen Markups ist die Ausrichtung von Text bzw. Absätzen. Dafür gibt es in LaTeX natürlich entsprechende Umgebungen und Schalter.

`flushleft`	`\raggedright`	Linksbündiger Text
`flushright`	`\raggedleft`	Rechtsbündiger Text
`center`	`\centering`	Zentrierter Text

Anders als bei der Schriftgröße und -art gibt es hierbei keinen Ausschalter, der die Wirkung vorheriger Schalter aufhebt, daher ist man hier, will man nicht die Umgebungen verwenden, stärker auf die Verwendung von Gruppen angewiesen. Ein Beispiel zur Verdeutlichung:

```
\begin{center}
Dieser Text wird zentriert
gesetzt.
\end{center}

{\raggedleft
Und dieser hier\\
rechtsb"undig.\\
}

Der Rest ist dank
der Verwendung einer
Gruppe wieder "‘normal"’.
```

Dieser Text wird zentriert gesetzt.

Und dieser hier
rechtsbündig.

Der Rest ist dank der Verwendung einer Gruppe wieder „normal“.

5.3.3 Umbrüche & Abstände

Im Normalfall bestimmt LaTeX sämtliche Umbrüche und Abstände selbst und macht das recht gut. Möchte man von Hand etwas an Abständen und Umbrüchen ändern, so stellt einem LaTeX dazu Befehle zur Verfügung.

Zum manuellen Einfügen von Zeilenumbrüchen stehen zur Verfügung:

`\\` oder `\\*`	Erzwingt einen Zeilenumbruch. Der * verhindert einen Seitenumbruch nach dem Zeilenumbruch
`\newline`	wie `\textbackslash`
`\\[abstand]`	Erzwingt einen Zeilenumbruch mit zusätzlichem `abstand`. Auch hier gibt es eine *-Variante

Hierzu ein Beispiel:

```
Eine Zeile\\
Noch eine\\[2cm]
Und die hier
kommt erst 2cm sp"ater.
```

Eine Zeile
Noch eine

Und die hier kommt erst 2cm später.

Für manuelle Seitenumbrüche steht das Kommando `\newpage` zur Verfügung.

Weiterhin gibt es noch die Befehle `\linebreak[`*num*`]` und `\nolinebreak[`*num*`]`, die mit einem Parameter *num* von 0 bis 4 eine *Empfehlung* für oder gegen einen Zeilenumbruch an der jeweiligen Stelle aussprechen (wobei der Wert 4 gleichbedeutend mit zwingend ist). Wird kein Parameter angegeben, so wird zwingend umgebrochen bzw. nicht umgebrochen. Ein weiterer Unterschied zu `\\` und `\newline` ist, daß die entstehende „kurze“ Zeile statt linksbündig im Blocksatz gesetzt wird.

Möchte man beliebige Abstände, etwa Absatzabstände einfügen, verwende man für horizontale Abstände:

`\hspace{abstand}`	Erzeugt einen Abstand der Länge `abstand`
`\hspace*{abstand}`	Fügt den Abstand auch am Zeilenanfang oder -ende ein.
`\hfill`	Äquivalent zu `\hspace{\fill}`. Fügt soviel Leerraum ein bis die Zeile „voll“ ist.
`\hrulefill`	Wie `\hfill`, nur daß mit einem Unterstrich „gefüllt“ wird.
`\dotfill`	Wie `\hfill`, nur daß mit ... „gefüllt“ wird.
`\,`	Ein sehr kleiner Abstand.
`\enspace`	Ein Abstand so breit wie eine Ziffer im momentanen Font.
`\quad`	Doppelt so breit wie `\enspace`.
`\qquad`	Doppelt so breit wie `\quad`.

Und für vertikale Abstände:

`\vspace{abstand}`	Erzeugt einen vertikalen Abstand (Absatzabstand) der Höhe `abstand`
`\vspace*{abstand}`	Fügt den Abstand auch am Seitenanfang oder -ende ein.
`\vfill`	Äquivalent zu `\vspace{\fill}`+. Fügt soviel Leerraum ein bis die Seite „voll“ ist.
`\bigskip`	Fügt einen großen `\vspace` ein
`\medskip`	Fügt einen mittleren `\vspace` ein
`\smallskip`	Fügt einen kleinen `\vspace` ein

Dazu ein etwas umfangreicheres Beispiel:

```
Hier bitte etwas \qquad Platz.
\smallskip\\
Anfang \hfill Ende.
\vspace{0.3cm}\\
Und noch was lustiges:
\medskip\\
Links\\
\hspace*{\fill}Mitte\hspace*{\fill}\\
\hspace*{\fill}Rechts
\bigskip\\
Name: \hrulefill\enspace
Unterschrift: \dotfill
```

Hier bitte etwas   Platz.
Anfang   Ende.

Und noch was lustiges:

Links
                    Mitte
                                        Rechts

Name: ____________ Unterschrift: ....

An diesem Beispiel ist auch zu sehen was passiert, wenn man mehrere „Füllkommandos“ wie `\hfill` oder `\dotfill` in einer Zeile verwendet: Der verlangte Platz verteilt sich entsprechend, um die Zeile mit proportionalen Abständen zu füllen. Werden mehrere solche Kommandos hintereinander angegeben, verteilt sich der Platz entsprechend:

```
A \dotfill\dotfill B \hfill C
```

A ....B   C

Proportionalen Leerraum kann man auch noch mit dem „magischen“ Argument `\stretch{nr}` für `\hspace` erzeugen, wobei `nr` die Proportion angibt.

```
x\hspace{\stretch{1}}
x\hspace{\stretch{3}}
x\hspace{\stretch{2}}
x
```

x	x	x	x

5.3.4 Absatzeinzüge

Abschließend soll hier noch auf ein paar Gestaltungskommandos eingegangen werden, die ein wenig aus dem Rahmen fallen, aber für deutsche Texte so wichtig sind, daß sie hier mit aufgenommen werden sollen. Um in LATEX von den im amerikanischen Raum üblichen Absätzen (geringer Abstand, dafür die erste Zeile eingezogen) auf die in Deutschland übliche Form umzustellen muss man die Längen *parskip* und *parindent* setzen. Diese Regeln wieviel Abstand Absätze voneinander haben, und wie weit die erste Zeile eines neuen Abstandes eingezogen wird. Um also deutsche Absätze herzustellen bedarf es:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{1ex}
```

Abgesehen von dieser Methode kann mit den Befehlen `\indent` bzw. `\noindent` am Anfang eines Absatzes der Einzug selektiv ein- bzw. ausgeschaltet werden. Das hat aber natürlich keinen Effekt mehr, wenn *parindent*, z.B. durch die obige Befehlssequenz, schon auf 0 Punkte gesetzt wurde.

Damit wären einige der wichtigsten Befehle zum visuellen Markup erklärt, und wir wollen es an dieser Stelle bewenden lassen. Allerdings gibt es noch deutlich mehr Möglichkeiten das Aussehen des eigenen Dokuments in LATEX zu beeinflußen. Hierzu sei aber an [Lam94] oder [GMS94] verwiesen.

Kapitel 6

Mathematische Formeln Teil 1

Mathematische Ausdrücke kommen in Texten sehr häufig vor. Bei einer Quadratzahl oder einem Index ist es gerade noch möglich, mit einem herkömmlichen Textverarbeitungsprogramm das gewünschte Ergebnis zu bekommen. Wenn man aber früher oder später doch mal eine Ausarbeitung oder ein (Pro-) Seminar verfassen möchte, benötigt man ein besseres Werkzeug.

Und da kommt auch schon wie gerufen LaTeX. Damit kann man eine so häßliche, lange Formel wie z.B.

$$\ln\left[\lim_{\delta\to\infty}\left[\left[(X')^{-1}-(X^{-1})'\right]!+\frac{1}{\delta}\right]^{\delta}\right]+(\sin^2 q+\cos^2 q)=\sum_{n=0}^{\infty}\frac{\cosh p\sqrt{1-\tanh^2 p}}{2^n}$$

relativ einfach verfassen. Und man muß sich nicht damit herumärgern, daß das Textverarbeitungsprogramm möglicherweise zusammenbricht.

6.1 Die Umgebung

Mathematische Formeln werden in einer *mathematischen Umgebung* geschrieben, d. h. man muß LaTeX mitteilen, wo der mathematische Ausdruck beginnt und endet. Das ist auf mehrere Arten möglich, die alle gleichberechtigt sind:

- `\begin{math}` *Formel* `\end{math}`
  oder kürzer
- `$` *Formel* `$`
- `\(` *Formel* `\)`

Der mathematische Ausdruck erscheint jetzt im Fließtext. Das kann praktisch sein, wenn er nur kurz erwähnt wird.

```
z.B. kann man definieren,
da"s $ x = a + b$ gelten soll.
Es ist auch m"oglich \( a^2 \) und
\begin{math} r^2 \end{math}
zu schreiben.
```

z.B. kann man definieren, daß $x = a + b$ gelten soll. Es ist auch möglich a^2 und r^2 zu schreiben.

Dabei werden die Buchstaben in einer solchen mathematischen Umgebung *kursiv* geschrieben, die Zahlen selbst normal.

Spezielle mathematische Funktionen, wie beispielsweise `\cos` oder `\sin` werden gerade geschrieben. Alle anderen Buchstaben erscheinen in einer mathematischen Umgebung *kursiv* und vor allem auch ohne Abstand voneinander (s. auch Kap. 7.1).

Es ist auch möglich, die Formeln abgesetzt und zentriert zu schreiben. Dafür schreibt man die Formel nicht wie oben in \$...\$, sondern verwendet einer der folgenden drei Umgebungen:

- `\begin{displaymath}` *Formel* `\end{displaymath}`
  oder, wieder kürzer
- `$$` *Formel* `$$`
- `\[` *Formel* `\]`

```
Diese Art f"ur Formeln
eignen sich besonders dann,
wenn die Formel
\[ x = a + b \]
sehr wichtig ist.
genauso wie
$$ r^2 =  1 $$
oder
\begin{displaymath}
f(t) = t^2
\end{displaymath}
```

Diese Art für Formeln eignen sich besonders dann, wenn die Formel

$$x = a + b$$

sehr wichtig ist. genauso wie

$$r^2 = 1$$

oder

$$f(t) = t^2$$

Jetzt kann es natürlich sein, daß man sich später wieder auf eine Formel beziehen möchte. Dazu ist es sinnvoll, diese zu numerieren. Auch diesen Wunsch kann LaTeX erfüllen. Dafür beginnt man einfach die Formel anstatt mit den obigen Umgebungen mit `\beginequation` und beendet sie mit `\endequation`.

Man kann mit dem Befehl `\label{`*Name*`}` eine Gleichung markieren und sich später mit `\ref{`*Name*`}` wieder darauf beziehen.

Hier gleich nochmal ein Beispiel für alle drei Arten, also ein mathematischer Ausdruck im Fließtext, eine abgesetzte Formel und eine Formel mit Nummerierung.

```
So sieht also eine Formel mit
laufender Nummerierung aus.
\begin{equation}
1 = 3 - 2
\label{gleichung}
\end{equation}
Leider f"allt mir nichts ein, bei dem
ich mich auf die obige Gleichung
(\ref{gleichung}) beziehen k"onnte.

Es k"onnen auch Formeln ohne laufende
Nummern wichtig sein, wie z.B.
\[ 2 = 4 - 2 \]
Die laufende Nummer l"auft nat"urlich
durch, wie man bei
\begin{equation}
3 = 3
\end{equation}
sieht.
```

So sieht also eine Formel mit laufender Nummerierung aus.

$$1 = 3 - 2 \tag{6.1}$$

Leider fällt mir nichts ein, bei dem ich mich auf die obige Gleichung (6.1) beziehen könnte.

Es können auch Formeln ohne laufende Nummern wichtig sein, wie z.B.

$$2 = 4 - 2$$

Die laufende Nummer läuft natürlich durch, wie man bei

$$3 = 3 \tag{6.2}$$

sieht.

6.2 Mathematische Zeichen

Wie auch bei Computersprachen oder Programmen (*Maple* oder *Matlab*) werden einige Zeichen ganz normal geschrieben. Dazu gehören

```
+ - = < > : / ! ( ) [ ] '
```

Ein Beispiel dazu ist

```
\[
y(x)'' = [(3! - 4) + 5] y(x)' + x
\]
```

$$y(x)'' = [(3! - 4) + 5]y(x)' + x$$

Exponeten und Indizes werden mit `^` (Dach) und `_` (Unterstrich) geschrieben. Soll ein Term Exponenten oder Indizes mit mehreren Zeichen haben, sind diese mit geschweiften Klammern `{...}` zu schreiben. Das wird relativ schnell an den Beispielen klar. Bei einem einzigen Zeichen als Exponent oder Index müssen die geschweiften Klammern `{...}` nicht vorhanden sein.

```
$ x^2, x^{2^2} $\\
$ x^1+3 $ wohingegen $ x^{1+3} $\\
$ y_i+1 $ wohingegen $ y_{i+1} $
```

x^2, x^{2^2}
x^1+3 wohingegen x^{1+3}
y_i+1 wohingegen y_{i+1}

Da waren aber noch keine Brüche, Wurzeln oder Integrale dabei. Und jetzt zeigt sich das wirklich angenehme an LaTeX: Man muß seine Formel einfach nur in ganz normalen Buchstaben tippen. Keine Verrenkungen zu `Strg-Alt-Esc`. Stattdessen kann man ganz entspannt vor der Tastatur sitzen und einfach nur tippen

Bruch	`\frac{Zähler}{Nenner}`	$\frac{Zähler}{Nenner}$
Wurzel	`\sqrt[n]{Ausdruck}`	$\sqrt[n]{Ausdruck}$
Summe	`\sum_{unten}^{oben}`	$\sum_{unten}^{oben}$
Produkt	`\prod_{unten}^{oben}`	$\prod_{unten}^{oben}$
Integral	`\int_{unten}^{oben}`	$\int_{unten}^{oben}$
Limes	`\lim_{unten}`	$\lim_{unten}$

Hier waren die Grenzen ein wenig rechts vom jeweiligen Symbol angeordnet. Sollen die Grenzen direkt unter dem Symbol stehen, so muß man zwischen dem Symbolnamen und den Grenzen noch den Befehl `\limits` einfügen. Komplett sieht das dann so aus:

```
$ \int_{0}^{1} (\ln x)^n dx
= (-1)^n n! $ \\[1ex] % ohne \limits
im Gegensatz zu \\[1ex]
$ \int\limits_{0}^{1} (\ln x)^n dx
= (-1)^n n! $ % mit \limits
```

$\int_{0}^{1} (\ln x)^n dx = (-1)^n n!$
im Gegensatz zu
$\int\limits_{0}^{1} (\ln x)^n dx = (-1)^n n!$

Umgekehrt kann man, falls bei einem Symbol die Zeichen ungewollt direkt darüber und darunter stehen, sie mit dem Befehl `\unlimits` auf die rechte Seite verschieben.

6.3 Buchstaben und Zeichen

Wer schreibt schon eine mathematische Formel mit Zahlen?

Buchstaben sind das einzig wahre. :)

6.3.1 Griechische Buchstaben

Das A und Ω ist dabei, daß man weiß, wie die Buchstaben heißen, die man schreiben möchte. Dann gibt man ganz einfach zuerst einen Backslash \ und direkt anschließend ohne Leerschritt den Namen des Buchstabens ein. Dabei werden kleine griechische Buchstaben komplett klein geschrieben (z.B. `\alpha`), griechische Großbuchstaben, sofern sie von den deutschen Buchstaben abweiche, werden mit dem ersten Buchstaben groß geschrieben (z.B. `\Gamma` $\Rightarrow \Gamma$).

Hier eine Übersicht über sämtliche verfügbaren griechischen Buchstaben:

α	\alpha	β	\beta	γ	\gamma	δ	\delta
ϵ	\epsilon	ζ	\zeta	η	\eta	θ	\theta
ι	\iota	κ	\kappa	λ	\lambda	μ	\mu
ν	\nu	o	o	π	\pi	ρ	\rho
σ	\sigma	τ	\tau	υ	\upsilon	ϕ	\phi
χ	\chi	ψ	\psi	ω	\omega		

Γ	\Gamma	Δ	\Delta	Θ	\Theta	Λ	\Lambda
Π	\Pi	Σ	\Sigma	Υ	\Upsilon	Φ	\Phi
Ψ	\Psi	Ω	\Omega	Ξ	\Xi		

Manche Buchstaben kann man auf zwei Arten schreiben. Es handelt sich dabei um die folgenden.

ϵ	\epsilon	ε	\varepsilon	ϕ	\phi	φ	\varphi
π	\pi	ϖ	\varpi	ρ	\rho	ϱ	\varrho
σ	\sigma	ς	\varsigma	θ	\theta	ϑ	\vartheta

6.3.2 Mengensymbole

Wichtig für jeden Zahlenjongleur sind die Zeichen, mit denen so wichtige Mengen wie die der natürlichen ($\mathbb{N}$) oder der reellen Zahlen ($\mathbb{R}$) bezeichnet werden. LaTeX selbst bietet hier keine Zeichen, aber mit Hilfe `amsmath`-Pakets (siehe speziell Abschnitt 8.15) ist das kein großes Problem.

6.3.3 Kalligraphische Buchstaben

Vor allem in (Linearer) Algebra werden gerne noch mehr Zeichen verwendet. Um den eigenen Buchstabenvorrat zu erweitern, kann man hier die *kalligraphischen Buchstaben* zu Hilfe nehmen, die sich recht gut für Mengenbeschreibungen eignen. Es handelt sich dabei um

$$\mathcal{A},\mathcal{B},\mathcal{C},\mathcal{D},\mathcal{E},\mathcal{F},\mathcal{G},\mathcal{H},\mathcal{I},\mathcal{J},\mathcal{K},\mathcal{L},\mathcal{M},\mathcal{N},\mathcal{O},\mathcal{P},\mathcal{Q},\mathcal{R},\mathcal{S},\mathcal{T},\mathcal{U},\mathcal{V},\mathcal{W},\mathcal{X},\mathcal{Y},\mathcal{Z}$$

die erzeugt werden mit

```
$ \mathcal{ A, B, C,\dots } $
bzw.  $ {\cal A, B, C,\dots } $
```

$\mathcal{A},\mathcal{B},\mathcal{C},\dots$ bzw. $\mathcal{A},\mathcal{B},\mathcal{C},\dots$

6.3.4 Andere Zeichen

Ansonsten sind so gut wie alle mathematischen Zeichen darstellbar. Man muß sie nur noch finden. Zur Erleichterung sollen die folgenden Auflistungen dienen

Binäre Zeichen

$\pm$	`\pm`	$\cap$	`\cap`	$\circ$	`\circ`	$\bigcirc$	`\bigcirc`
$\mp$	`\mp`	$\cup$	`\cup`	$\bullet$	`\bullet`	$\Box$	`\Box`
$\times$	`\times`	$\uplus$	`\uplus`	$\diamond$	`\diamond`	$\Diamond$	`\Diamond`
$\div$	`\div`	$\sqcap$	`\sqcap`	$\lhd$	`\lhd`	$\bigtriangleup$	`\bigtriangleup`
$\cdot$	`\cdot`	$\sqcup$	`\sqcup`	$\rhd$	`\rhd`	$\bigtriangledown$	`\bigtriangledown`
$\ast$	`\ast`	$\vee$	`\vee`	$\unlhd$	`\unlhd`	$\triangleleft$	`\triangleleft`
$\star$	`\star`	$\wedge$	`\wedge`	$\unrhd$	`\unrhd`	$\triangleright$	`\triangleright`
$\dagger$	`\dagger`	$\setminus$	`\setminus`	$\oslash$	`\oslash`	$\oplus$	`\oplus`
$\ddagger$	`\ddagger`	$\wr$	`\wr`	$\odot$	`\odot`	$\ominus$	`\ominus`
$\amalg$	`\amalg`					$\otimes$	`\otimes`

Vergleichssymbole und deren Negationen

<table>
<tr><td>$\le$</td><td><code>\le</code></td><td>$\ge$</td><td><code>\ge</code></td><td>$\neq$</td><td><code>\neq</code></td><td>$\sim$</td><td><code>\sim</code></td></tr>
<tr><td>$\leq$</td><td><code>\leq</code></td><td>$\geq$</td><td><code>\geq</code></td><td></td><td></td><td></td><td></td></tr>
<tr><td>$\ll$</td><td><code>\ll</code></td><td>$\gg$</td><td><code>\gg</code></td><td>$\doteq$</td><td><code>\doteq</code></td><td>$\simeq$</td><td><code>\simeq</code></td></tr>
<tr><td>$\subset$</td><td><code>\subset</code></td><td>$\supset$</td><td><code>\supset</code></td><td>$\approx$</td><td><code>\approx</code></td><td>$\asymp$</td><td><code>\asymp</code></td></tr>
<tr><td>$\subseteq$</td><td><code>\subseteq</code></td><td>$\supseteq$</td><td><code>\supseteq</code></td><td>$\cong$</td><td><code>\cong</code></td><td>$\smile$</td><td><code>\smile</code></td></tr>
<tr><td>$\sqsubset$</td><td><code>\sqsubset</code></td><td>$\sqsupset$</td><td><code>\sqsupset</code></td><td>$\equiv$</td><td><code>\equiv</code></td><td>$\frown$</td><td><code>\frown</code></td></tr>
<tr><td>$\sqsubseteq$</td><td><code>\sqsubseteq</code></td><td>$\sqsupseteq$</td><td><code>\sqsupseteq</code></td><td>$\propto$</td><td><code>\propto</code></td><td>$\bowtie$</td><td><code>\bowtie</code></td></tr>
<tr><td>$\in$</td><td><code>\in</code></td><td>$\ni$</td><td><code>\ni</code></td><td>$\prec$</td><td><code>\prec</code></td><td>$\succ$</td><td><code>\succ</code></td></tr>
<tr><td>$\vdash$</td><td><code>\vdash</code></td><td>$\dashv$</td><td><code>\dashv</code></td><td>$\preceq$</td><td><code>\preceq</code></td><td>$\succeq$</td><td><code>\succeq</code></td></tr>
<tr><td>$\models$</td><td><code>\models</code></td><td>$\perp$</td><td><code>\perp</code></td><td>$\parallel$</td><td><code>\parallel</code></td><td>$\mid$</td><td><code>\mid</code></td></tr>
<tr><td></td><td></td><td></td><td></td><td>$\Vert$</td><td><code>\|</code></td><td>$\vert$</td><td><code>|</code></td></tr>
</table>

$\not<$	`\not<`	$\not>$	`\not>`	$\not=$	`\not=`
$\not\le$	`\not\le`	$\not\ge$	`\not\ge`	$\not\equiv$	`\not\equiv`
$\not\prec$	`\not\prec`	$\not\succ$	`\not\succ`	$\not\sim$	`\not\sim`
$\not\preceq$	`\not\preceq`	$\not\succeq$	`\not\succeq`	$\not\simeq$	`\not\simeq`
$\not\subset$	`\not\subset`	$\not\supset$	`\not\supset`	$\not\approx$	`\not\approx`
$\not\subseteq$	`\not\subseteq`	$\not\supseteq$	`\not\supseteq`	$\not\cong$	`\not\cong`
$\not\sqsubseteq$	`\not\sqsubseteq`	$\not\sqsupseteq$	`\not\sqsupseteq`	$\not\asymp$	`\not\asymp`
$\not\in$	`\not\in`	$\notin$	`\notin`		

Pfeil- und Zeichensymbole

$\leftarrow$	\leftarrow	$\longleftarrow$	\longleftarrow	$\uparrow$	\uparrow
$\Leftarrow$	\Leftarrow	$\Longleftarrow$	\Longleftarrow	$\Uparrow$	\Uparrow
$\rightarrow$	\rightarrow	$\longrightarrow$	\longrightarrow	$\downarrow$	\downarrow
$\Rightarrow$	\Rightarrow	$\Longrightarrow$	\Longrightarrow	$\Downarrow$	\Downarrow
$\leftrightarrow$	\leftrightarrow	$\longleftrightarrow$	\longleftrightarrow	$\updownarrow$	\updownarrow
$\Leftrightarrow$	\Leftrightarrow	$\Longleftrightarrow$	\Longleftrightarrow	$\Updownarrow$	\Updownarrow
$\mapsto$	\mapsto	$\longmapsto$	\longmapsto	$\nearrow$	\nearrow
$\hookleftarrow$	\hookleftarrow	$\hookrightarrow$	\hookrightarrow	$\searrow$	\searrow
$\leftharpoonup$	\leftharpoonup	$\rightharpoonup$	\rightharpoonup	$\swarrow$	\swarrow
$\leftharpoondown$	\leftharpoondown	$\rightharpoondown$	\rightharpoondown	$\nwarrow$	\nwarrow
$\rightleftharpoons$	\rightleftharpoons	$\leadsto$	\leadsto		

Verschiedene sonstige Symbole

$\aleph$	\aleph	$\prime$	\prime	$\forall$	\forall		
$\hbar$	\hbar	$\emptyset$	\emptyset	$\exists$	\exists		
$\imath$	\imath	∇	\nabla	$\neg$	\neg	$\triangle$	\triangle
$\jmath$	\jmath	$\surd$	\surd	$\flat$	\flat	$\clubsuit$	\clubsuit
ℓ	\ell	∂	\partial	$\natural$	\natural	$\diamondsuit$	\diamondsuit
$\wp$	\wp	$\top$	\top	$\sharp$	\sharp	$\heartsuit$	\heartsuit
$\Re$	\Re	$\bot$	\bot	$\Vert$	\\|	$\spadesuit$	\spadesuit
$\Im$	\Im	$\vdash$	\vdash	$\angle$	\angle	$\Join$	\Join
$\mho$	\mho	$\dashv$	\dashv	$\backslash$	\backslash	∞	\infty

Mathematische Akzente

$\hat{a}$	\hata	$\breve{a}$	\brevea	$\grave{a}$	\gravea	$\bar{a}$	\bara	$\dot{a}$	\dota
$\check{a}$	\checka	$\acute{a}$	\acutea	$\tilde{a}$	\tildea	$\vec{a}$	\veca	$\ddot{a}$	\ddota

6.4 Es wird anspruchsvoller . . .

6.4.1 Matrizen

Jeder kommt mit ihnen in Berührung. Und sie sind auch ganz wichtig. Die Rede ist natürlich von Matrizen.

$$I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Und obwohl sie so wichtig sind, werden sie nicht hier besprochen, sondern in Kap. 9, ebenso wie auch z.B. stückweise definierte Funktionen wie

$$|x| = \begin{cases} -x & \text{für } x < 0 \\ x & \text{für } x \geq 0 \end{cases}$$

6.4.2 Gleichungssysteme in drei Teilen

Eine Gleichung hat ja immer drei Teile:

linker Ausdruck = *rechter Ausdruck*

Um nun ein Gleichungssystem hübsch darstellen zu können, gibt es den Befehl

```
\begin{eqnarray}
Gleichung(en)
\end{eqnarray}
```

Hier werden auch Gleichungsnummer mit ausgegeben, auf die man sich später in einem Text beziehen kann. Die Nummerierung wird generell unterdrückt bei dem Befehl

```
\begin{eqnarray*}
Gleichung(en)
\end{eqnarray*}
```

sowie in einer einzelnen Zeile der Gleichung mit dem Befehl `\nonumber`.

```
Wenn ich also wei"s, da"s
\begin{eqnarray}
a + b+ c &=& y \nonumber
\label{gleichung1}\\
y &=& m + n + k
\label{gleichung2}
\end{eqnarray}
gilt, dann wei"s ich auch, da"s mit
(\ref{gleichung1}) und
(\ref{gleichung2})
\begin{eqnarray*}
a + b + c & = &  m + n + k \\
y & = & y
\end{eqnarray*}
gilt.
```

Wenn ich also weiß, daß

$$\begin{aligned} a+b+c &= y \\ y &= m+n+k \qquad (6.3) \end{aligned}$$

gilt, dann weiß ich auch, daß mit (6.3) und (6.3)

$$\begin{aligned} a+b+c &= m+n+k \\ y &= y \end{aligned}$$

gilt.

Hier tauchen auf einmal unversehens die Zeichen & und \\ auf. Diese Schreibweise ist angeglichen zu der Schreibweise bei Tabellen. Deshalb sei auf weiteres dazu verwiesen auf Kap. 9.

6.4.3 Größenanpassung von Klammern

In einer langen Formel kann eine normale Klammer zu klein sein. Um sie auf die richtige Form anzupassen, kann man sich folgender Schreibweise bedienen:

```
\leftKlammerzeichen ...\right Klammerzeichen
```

Aus den zu kleinen Klammer

```
\[
  [
   \int\limits_0^1
       (\ln x)^n dx
  ] - (-1)^n
  \cdot n! = 0
\]
```

$$[\int\limits_0^1 (\ln x)^n dx] - (-1)^n \cdot n! = 0$$

werden dann welche, die ihr Argument umfaßen.

```
\[
  \left[
   \int\limits_0^1
       ( \ln x )^n dx
  \right] - (-1)^n
  \cdot n! = 0
\]
```

$$\left[\int\limits_0^1 (\ln x)^n dx\right] - (-1)^n \cdot n! = 0$$

Wichtig ist, daß die Anzahl der `\left`-Befehle gleich der Anzahl der `\right`-Befehle ist. Soll trotzdem mal nur eine der beiden Klammern, z.B. die linke, erscheinen, so kann die andere, in diesem Fall die rechte, mit dem Befehl `\right.` unterdrückt werden.

```
\[
   \left[
     \int \int
   \right.
\]
```

$$\left[\int\int\right.$$

6.4.4 Abstände in Formeln

LaTeX bestimmt selbst, wie groß oder klein die Abstände in der Formel werden. Gefällt einem der Abstand nicht, kann man ihn auch selbst verändern. Man sollte von dieser Möglichkeit aber nur Gebrauch machen, wenn man wirklich sicher ist, daß es notwenig ist.

Die verschiedenen Abstände sind

`\,`	kleiner Zwischenraum
`\:`	mittlerer Zwischenraum
`\;`	großer Zwischenraum
`\!`	kleiner negativer Zwischenraum

Schauen wir uns wieder obiges Beispiel an. Es ist recht wenig Abstand zwischen dem Exponenten n und dem dx-Zeichen. Und von den eckigen Klammern `[` und `]` könnte auch ein bißchen mehr Abstand sein. Hier kann man von

```
\[
  \left[
    \int\limits_0^1
      \left(
        \ln x
      \right)^n dx
  \right] - (-1)^n
  \cdot n! = 0
\]
```

$$\left[\int\limits_0^1 \left(\ln x \right)^n dx \right] - (-1)^n \cdot n! = 0$$

übergehen zu

```
\[
    \left[ \;
      \int\limits_0^1
        \left(
          \ln x
        \right)^n \: dx \;
    \right] - (-1)^n
    \cdot n! = 0
\]
```

$$\left[\; \int\limits_0^1 \left(\ln x \right)^n \: dx \; \right] - (-1)^n \cdot n! = 0$$

6.4.5 Normale Worte in Formeln

Buchstaben und Buchstabenfolgen, die LaTeX nicht erkennt, erscheinen in Formeln *kursiv* und ohne Abstand. Möchte man aber doch mal einen ganz normalen, kurzen Text in einer Formel stehen haben, so kann man ihn in eine Textumgebung innerhalb der Formel schreiben. Der Befehl dafür lautet `\mbox{normaler Text}`.

Das sieht also dann so aus

```
$ f(x) = 1 \mbox{ f"ur } x > 7 $
```

$f(x) = 1 \mbox{ für } x > 7$

Nicht kursiv, sondern gerade geschrieben werden Wörter, die LaTeX als Funktionen erkennt, also z.B. `\sin`, `\cos`, `\ln`, `\dots`

Dabei handelt es sich um

`\arccos`	`\cos`	`\csc`	`\esp`	`\ker`	`\limsup`	`\min`	`\sinh`
`\arcsin`	`\cosh`	`\deg`	`\gcd`	`\lg`	`\ln`	`\Pr`	`\sup`
`\arctan`	`\cot`	`\det`	`\hom`	`\lim`	`\log`	`\sec`	`\tan`
`\arg`	`\coth`	`\dim`	`\inf`	`\liminf`	`\max`	`\sin`	`\tanh`

Ganz wichtig: Damit LaTeX auch wirklich erkennt, daß es sich um diese Funktionen handelt, darf der Backslash \ nicht fehlen!

```
Es gibt einen Unterschied
zwischen $ ln x $ und
$ \ln x $.
```

Es gibt einen Unterschied zwischen $ln x$ und $\ln x$.

6.4.6 Specials

Man braucht sie nicht oft, aber dann sucht man mit unter lange danach. Gemeint sind Ausdrücke wie

$$\underbrace{1 + 1 + \cdots + 1}_{n} = n$$

$$\vec{x} \stackrel{\text{def}}{=} (1, 1, 1)^T$$

$$\overline{z_1 \cdot z_2} = \overline{z_1} \cdot \overline{z_2}$$

Damit Du nicht so lange danach suchen mußt, hier eine kurze Erklärung:

Der Befehl `\stackrel{oberes Symbol}{unteres Symbol}` bewirkt, daß über dem unteren Symbol das obere Symbol steht (wer hätt's gedacht).

Also eben gerade

```
\[
    \vec{x}
    \stackrel{\textrm{\tiny def}}%
    {=} (1, 1, 1)
\]
```

$$\vec{x} \stackrel{\text{def}}{=} (1, 1, 1)$$

Der Befehl `\tiny` bewirkt dabei, daß `def` in einer sehr kleinen Schrift geschrieben wird.

Der Befehl `\overline{formelteil}` bzw. `\underline{formelteil}` bewirkt, daß der Formelteil über- bzw. unterstrichen wird.

```
\[
    \overline{z_1 \cdot z_2} =
    \overline{z_1} \cdot
    \overline{z_2}
\]
```

$$\overline{z_1 \cdot z_2} = \overline{z_1} \cdot \overline{z_2}$$

Ähnlich funktionieren die beiden Befehle `\overbrace{formelteil}` und `\underbrace{formelteil}`. Der Formelteil erhält statt eines Striches eine geschweifte Klammer. Soll diese noch einen Text darüber oder darunter bekommen, kann das folgendermaßen dargestellt werden:

```
\[
    \underbrace{1 + 1 + %
        \ldots + 1}_{n} = n
\]
```

$$\underbrace{1 + 1 + \ldots + 1}_{n} = n$$

6.4.7 Weiteres

LaTeX wird vor allem dafür verwendet, um mathematische Texte zu schreiben. Es gibt deshalb eine Vielzahl von Kleinigkeiten, die auch sehr nützlich sind, hier aber mangels Platz nicht erwähnt wurden. Der geneigte Leser sei deshalb aufgefordert, sich in weiteren Büchern kundig zu machen.

Als kleine Anregung: [Kop96], [Lam95].
☺

Kapitel 7

Aus Fehlern lernen . . .

Fehler macht jeder. Und ganz besonders leicht passieren sie einem in LaTeX . Aber hier ist der große Vorteil, daß sie auch ganz einfach wieder korrigiert werden können.

7.1 Typische Fehler

Wenn man den Text in einen Editor eingetippt hat und dann den LaTeX-Lauf startet, gibt es immer wieder typische Fehler:

- es fehlt `\end{`*`umgebung`*`}`
- es fehlt eine einzelne Klammer }
- Befehle sind falsch getippt

Es gibt bei LaTeX auch oft Fehler, die ganz typisch für bestimmte Umgebungen sind.

Im *Mathematik-Modus* muß man sich ja eine Umgebung schaffen, in die man seine mathematischen Zeichen schreiben darf. Wenn man nun diese Umgebung mit $ geöffnet hat, dann vergißt man oft, sie auch wieder mit $ zu schließen.

Bei der Eingabe in Zeile 43 wurde bei der Eingabe von ‚`$ a = b + c`' leider etwas vergessen:

```
! Missing $ inserted.
<inserted text>
                      $
l.43

?
```

Oder man verwendet ein Zeichen, bei dem man gar nicht daran denkt, daß LaTeX es als ein mathematisches Zeichen ansieht. Der Unterstrich _ zählt dazu. Auch hier beschwert sich das Programm beim Compilerlauf.

Falls im Fließtext der Satz ‚`Fehlermeldung für Unterstrich _`' auftaucht, dann sieht die Fehlermeldung im Compilerlauf dazu folgendermaßen aus:

```
! Missing $ inserted.
<inserted text>
                $
l.54 Fehlermeldung für Unterstrich _

?
```

Arbeitet man mit Tabellen (Kap. 9), so sind alle Zeilen mit `\\` abzuschließen. Läßt man den in einer Zeile den Doppel-Backslash weg

```
\begin{tabular}{ll}
Zeile 1, Spalte 1 & Zeile 1, Spalte 2   % Kommentar: hier fehlt \\
Zeile 2, Spalte 2 & Zeile 2, Spalte 2
\end{tabular}
```

kommt es zu der Fehlermeldung

```
! Extra alignment tab has been changed to \cr.
<recently read> \endtemplate

l.78 Zeile 2, Spalte 2 &
                         Zeile 2, Spalte 2
?
```

Eine recht ausführliche Beschreibung verschiedener Fehler findet sich in [Kop96].

7.2 Fehlerbehandlung

Die Fehlermeldungen sind oft etwas kryptisch, wenn man sie noch nicht gewohnt ist. Tritt ein Fehler im Compiler-Lauf (also nach Eingabe des Kommandos `latex <Dateiname.tex>`) auf, dann wartet das Programm auf eine Eingabe. Die wichtigsten Eingaben sind dabei

Optionen	Name	Was bewirkt das
`<Enter>`		überspringt diesen Fehler und geht zum nächsten.
`s`	scroll mode	Programm läuft komplett durch, als würde bei jedem Fehler `<Enter>` gedrückt werden. Die Fehlermeldungen erscheinen auf dem Bildschirm. Es stoppt nur, wenn eine Datei (z.B. ein Style `*.sty`) nicht existiert.
`r`	run mode	wie `s`, es findet aber nicht einmal bei fehlenden Dateien eine Unterbrechung statt.
`q`	quiet mode	wie `r`, die Fehlermeldungen werden aber nicht auf dem Bildschirm, sondern in der Datei `<Dateiname>.log` abgespeichert.
`x`	exit	Das Programm bricht sofort komplett ab, es wird kein `.dvi`-File erzeugt.
`i`	insert	man kann danach etwas eingeben.
`h`	help	liefert eine Fehlerbeschreibung.

Auch hier sei nochmals auf eine ausführlichere Beschreibung in [Kop96] hingewiesen.

Hilfe bietet auch das Programm `lacheck` (LaTeX-Check). Dieses findet man unter `ftp://sunsite.auc.dk/pub/text/lacheck/`

In jedem Fall gilt aber, daß es am Anfang einfach schwer ist, mit der Flut von Fehlern, die wohl über einen hereinbrechen werden, zurechtzukommen. Trotzdem sollte man daran nicht verzweifeln. LaTeX ist trotz der Fehlermeldungen nicht so schwer zu bedienen.

Außerdem: Irgendeinen Nachteil muß es doch geben!

Kapitel 8

Mathematische Formeln Teil 2

Da Donald Knuth sein Textsatzsystem unter anderem auf mathematischen Formelsatz optimiert hat, kann TEX schon exzellent mit Wurzeln, Brüchen und dergleichen um sich werfen. Hier mußte Lamport also nicht mehr viel hinzufügen als er LATEX schrieb und so hat er einfach die Schnittstellen zum Formelsatz ein wenig vereinfacht und sich ansonsten vorrangig auf den Textsatz konzentriert.

Parallel zur Entwicklung von LATEX arbeitete die American Mathematical Society ebenfalls an einer Weiterentwicklung von TEX genannt $\mathcal{AMS}$-TEX. Diese Version zeichnet sich dadurch aus, daß sie besonderen Wert auf einen korrekten Formelsatz legt und dabei den Text ein wenig vernachlässigt. So fehlt etwa die Möglichkeit, Stellen im Text zu referenzieren sowie eine automatische Nummerierung.

Dies führte zu einigen Ärgernissen, da die jeweils benötigte Fähigkeit natürlich immer nur im jeweils anderen Programm implementiert war. So machte sich die American Mathematical Society daran, beide Programme miteinander zu verbinden. Herausgekommen ist $\mathcal{AMS}$-LATEX, das bei Bedarf als Paket in LATEX 2_ε eingebunden werden kann.

8.1 Möglichkeiten mit $\mathcal{AMS}$-LATEX

Dieses Paket ist also immer dann von Nutzen, wenn ein Text ansteht, der zu einem großen Teil aus Formeln besteht. Aber was genau hat man dann, was es vorher noch nicht gab? Ein paar der Möglichkeiten sind

- In mathematischen Formeln werden Funktionen wie Sinus oder Logarithmus aufrecht gesetzt, während Variablen kursiv erscheinen (so etwa: $2x+\sin(\phi+\pi)$). Jetzt können sehr einfach neue Funktionsnamen definiert werden, die dann auch richtig gesetzt werden.
- Die `eqnarray`-Umgebung, mit der man mehrzeilige Formeln setzen kann setzt dem Schreiber recht enge Grenzen. Dieses Problem wird behoben, indem mehrere neue Umgebungen angeboten werden.

- Mit LaTeX ist es zwar möglich, Formeln zu nummerieren, es ist aber nicht möglich, das Erscheinungsbild dieser Nummern seinen Bedürfnissen anzupassen. Auch hierfür gibt es jetzt ein Makro.
- Außerdem können mehrere zusammengehörige Formeln auch mit Unternummern versehen werden. Etwa (1.3a), (1.3b) und (1.3c).
- AMS-LaTeX bringt viele neue Symbole mit.
- Der Text unter und über Summen- oder Produktzeichen kann nun auch mehrzeilig gesetzt werden.
- Viele, viele weitere Gimmicks, die einem das Leben beim Setzen von Formeln doch sehr vereinfachen. Eigene Matrix-Umgebung, partiell definierte Funktionen, Indizes an allen vier Ecken eines Symbols, vierte Ableitung für Physiker, Symbole in dicker Schrift und natürlich das wichtige `\text`-Kommando

8.2 Was zu tun ist bevor's losgeht

Um ein Paket zu nutzen, muß man es erst einmal mit `\usepackage` in sein Dokument einbinden. AMS-LaTeX besteht aus mehreren Paketen, die je nach Bedarf geladen werden. Die wichtigsten hierbei sind

amsmath Das ist so etwa das Hauptpaket, das immer eingebunden wird, wenn mit AMS-LaTeX gearbeitet wird.

amstext Wie schon erwähnt, gibt es das `\text`-Makro. Da es sich hierbei um ein sehr nützliches Makro handelt, das auch so immer mal wieder gern verwendet wird, hat es ein eigenes Paket.

amsfonts Hierin ist unter anderem die Schriftart enthalten, mit der man „echte" Mengensymbole erhält.

amssymb Viele viele Symbole, die für spezielle Gebiete der Mathematik benötigt werden.

8.3 Neue Umgebungen für Formeln

Wenn wir mit LaTeX arbeiten, stehen uns zwei Sorten von Umgebungen für abgesetzte Umgebungen zur Verfügung. Eine Umgebung für eine Formel und eine für mehrere Zeilen voller Formeln. Dieses Konzept stößt aber oft genug an seine Grenzen. Zum Beispiel, wenn man eine sehr lange Formel hat, die sich über mehrere Zeilen erstreckt, oder wenn man ein paar kleine Formeln nebeneinander haben will. Außerdem erlaubt die `eqnarray`-Umgebung ein Ausrichten der Formeln nur an zwei Stellen. Was aber, wenn man ein lineares Gleichungssystem hat, bei dem doch bitte alle x_i auf jeweils gleicher Höhe stehen sollen?

8.3.1 multline

```
\begin{multline}
Sehr sehr lange Formel
\end{multline}
```

`multline` ist für sehr lange Formeln gemacht, die nicht in eine Zeile passen. Die erste Zeile der Formel wird linksbündig gesetzt, die weiteren Zeilen mittig bis zur letzten Zeile, die rechtsbündig gesetzt wird.

```
\begin{multline}
    \framebox[.65\columnwidth]
        {erster Teil} \\
    \framebox[.5\columnwidth]
        {Mittelteil} \\
    \framebox[.65\columnwidth]
        {Schlu"steil}
\end{multline}
```

$$\begin{multline}\boxed{\text{erster Teil}} \\ \boxed{\text{Mittelteil}} \\ \boxed{\text{Schlußteil}} \tag{8.1}\end{multline}$$

8.3.2 equation

Das ist die ganz normale Umgebung für eine einzelne abgesetzte Formel. Die Leerräume beim „=“ wurden ein wenig angepaßt, aber sonst ist kein großer Unterschied.

```
\begin{equation}
    a^2 + b^2 = c^2
\end{equation}
```

$$a^2 + b^2 = c^2 \tag{8.2}$$

8.3.3 split

Diese Umgebung ist so eine Art Mischung aus `multline` und `eqnarray`. Zwar handelt es sich um eine lange umgebrochene Formel, aber man kann mit Hilfe des &-Zeichens eine Anordnung ins Spiel bringen. Split nummeriert nicht und es gibt auch keine Form, die Nummern vergibt, da die Umgebung dafür gedacht ist, innerhalb einer `equation` eingebettet zu werden.

```
\begin{equation}
\label{grosseFormel}
\begin{split}
    H_c &= \frac{1}{2n} \sum^n_{l=0}
        (-1)^l (n-l)^{p-2} \sum_{l_1
+ \dots + l_p=l} \prod^p_{i=1}
\binom{n_i}{l_i} \\ &\quad\cdot
[(n-l)-(n_i-l_i)]^{n_i-l_i}
\end{split}
\end{equation}
```

$$\begin{split} H_c &= \frac{1}{2n} \sum^n_{l=0} (-1)^l (n-l)^{p-2} \sum_{l_1 + \dots + l_p=l} \prod^p_{i=1} \binom{n_i}{l_i} \\ &\quad\cdot [(n-l)-(n_i-l_i)]^{n_i-l_i} \end{split} \tag{8.3}$$

8.3.4 gather

Sollen mehrere Formeln mittig untereinander gesetzt werden, so nimmt man `gather`. Die Formeln werden dann wie auch bei den anderen Umgebungen für mehrzeiligen Formelsatz durch einen doppelten Backslash gegeneinander abgetrennt.

```
\begin{gather}
    a_1 = b_1 + c_1 \\
    x_n = 3\pi \cdot \sin\phi_n
\end{gather}
```

$$a_1 = b_1 + c_1 \tag{8.4}$$
$$x_n = 3\pi \cdot \sin \phi_n \tag{8.5}$$

8.3.5 align und flalign

Mit dem `eqnarray` kann man ja schon die enthaltenen Formeln mit dem &-Symbol aufeinander abstimmen, indem man beispielsweise die Gleichheitssymbole untereinander setzt (Alignment). Allerdings kann man bei jeder Formel immer nur einen linken Teil, eine Mitte und einen rechten Teil definieren, die dann aufeinander abgestimmt werden. Dieses Manko beseitigt die `align`-Umgebung, in der beliebig viele &-Symbole vorkommen können. Diese Symbole markieren dann Positionen in der Formel, die im fertigen Text untereinander gessetzt werden sollen.

Sehr ähnlich ist die `flalign`-Umgebung. Der einzige Unterschied zu `align` besteht darin, daß `flalign` seine Inhalte an den Rand schiebt.

```
\begin{align*}
    c_{11}& = a_{11} + b_{11}
        & c_{12}& = a_{12}
    + b_{12} \\
    c_{21}& = a_{21} + b_{21}
        & c_{22}& = a_{22}
    + b_{22}
\end{align*}
```

$$\begin{aligned} c_{11} &= a_{11} + b_{11} & c_{12} &= a_{12} + b_{12} \\ c_{21} &= a_{21} + b_{21} & c_{22} &= a_{22} + b_{22} \end{aligned}$$

```
\begin{flalign}
        c_{11}& = a_{11} + b_{11}
                & c_{12}& = a_{12}
    + b_{12} \\
        c_{21}& = a_{21} + b_{21}
                & c_{22}& = a_{22}
    + b_{22}
\end{flalign}
```

$$c_{11} = a_{11} + b_{11} \qquad c_{12} = a_{12} + b_{12} \tag{8.6}$$
$$c_{21} = a_{21} + b_{21} \qquad c_{22} = a_{22} + b_{22} \tag{8.7}$$

8.3.6 aligned / gathered

Die letzte Umgebung in diesem Paket läßt schon darauf schließen, daß sie etwas ähnliches macht, wie `align`. Der Hauptunterschied ist der, daß `aligned` nur innerhalb einer der anderen mathematischen Umgebungen angewendet werden darf. Mit ihr können dann kleinere Blöcke à la `align` gesetzt werden.

Ebenso kann man mit `gathered` kleine Blöcke innerhalb einer abgesetzten Formelumgebung à la `gather` setzen.

```
\begin{equation*}
    \begin{gathered}
        1 = \alpha + \beta \\
        \gamma + \delta = 2\\
        3 = \eta + \tau
    \end{gathered}
    \qquad\text{wohingegen}\qquad
    \begin{aligned}[t]
        12 &= 3\pi + \xi \\
        21 &= 3\xi + \pi \\
        -1 &= e^{i\pi}
    \end{aligned}
\end{equation*}
```

$$\begin{gathered} 1 = \alpha + \beta \\ \gamma + \delta = 2\\ 3 = \eta + \tau \end{gathered} \qquad\text{wohingegen}\qquad \begin{aligned}[t] 12 &= 3\pi + \xi \\ 21 &= 3\xi + \pi \\ -1 &= e^{i\pi} \end{aligned}$$

Das kleine „t“ hinter aligned sagt aus, welcher Teil des Kastens, der die Formeln enthält auf Höhe der Mittellinie gesetzt wird. Läßt man es weg, so wird der Kasten mittig gesetzt. Das t steht für Top und b wäre für Bottom.

8.4 Das Makro `\text`

$\mathcal{A}\mathcal{M}\mathcal{S}$-LaTeX gibt einem die Möglichkeit, ohne große Probleme Text in Formeln einzufügen. Das Makro hierfür heißt `\text`. Unter LaTeX gibt es ja für sowas die `\mbox`. Die hat aber den Nachteil, daß man die Schriftgröße in Indizes selbst anpassen muß. Außerdem kann man sich `\text` ganz klar besser merken, als `\mbox`.

8.5 Partiell definierte Funktionen mit `cases`

In der Mathematik gibt es öfters mal Funktionen, die auf verschiedenen Bereichen verschieden definiert sind. Normalerweise schreibt man solche Funktionen mit einer großen geschweiften Klammer, die dann alle möglichen Definitionen umfaßt. In LaTeX wird so eine Schreibweise dadurch simuliert, daß man eine große geschweifte Klammer öffnet, dann ein `array` mit allen Definitionen angibt und die große Klammer mit einem Punkt wieder schließt (jedes `\left\{` benötigt ein `\right\}`. Aber es geht schon auch ein `\right.`, dann gibts einfach gar keine Klammer).

Nachdem dieses Vorgehen recht umständlich ist, gibt es auch hierfür eine neue Umgebung mit Namen `cases`.

```
\[
P_{r-j} = \begin{cases}
    0& \text{falls
        $r-j$ ungerade ist} \\
    r!(-1)^{(r-j)/2}& \text{sonst}
\end{cases}
\]
```

$$P_{r-j} = \begin{cases} 0 & \text{falls } r-j \text{ ungerade ist} \\ r!(-1)^{(r-j)/2} & \text{sonst} \end{cases}$$

8.6 Nummern an den Formeln verschieben

Wenn man mal längere Formeln nummeriert, kann es schon mal vorkommen, daß die Formel ihre Nummerierung überschreibt. AMS-LaTeX versucht dem so weit wie möglich zu entgehen und die Formel im Notfall ein Stückchen höher oder tiefer zu setzen. Fall das mal nicht gelingt, gibt es das Makro `\raisetag`{*Höhe*}, das man in einer Formel angeben kann um die Nummer per Hand noch in der Höhe zu verschieben. Das ist aber eine echte Feintuningmaßnahme, die bis zum Schluß warten kann.

8.7 Kurze Texteinschübe in Formeln

Muß man mal in einem Feld aus mehreren Formeln eine oder zwei Zeilen Text einschieben, will aber, daß die Ausrichtung der Formeln erhalten bleibt, so kann man das Makro `\intertext` verwenden.

```
Wenn man sagt, da"s gilt
\begin{align}
    a& = b& b& = c
\intertext{dann gilt auch}
    a& = c
\end{align}
```

Wenn man sagt, daß gilt

$$a = b \qquad b = c \tag{8.8}$$

dann gilt auch

$$a = c \tag{8.9}$$

8.8 diverse Matrizen

Anstatt sich eigene Makros für Matrizen machen zu müssen, existieren im AMS-LaTeX-Paket einige Umgebungen für Matrizen. Es handelt sich um **pmatrix**, **bmatrix**, **Bmatrix**, **vmatrix** und **Vmatrix**. Diese Umgebungen funktionieren wie ein Array, setzen aber die Klammern und die einzelnen Werte etwas dichter als ein Array. Der Unterschied liegt in den Klammern die gesetzt werden, nämlich

pmatrix	bmatrix	Bmatrix	vmatrix	Vmatrix
( )	[ ]	{ }	| |	‖ ‖

8.9 Matrizen für Fließtext

Man kann eine Matrix auch innerhalb von Fließtext setzen. Die Matrix sollte allerdings nicht zu groß sein, da der Text sonst zu sehr auseinandergezogen wird. Mit der Umgebung `smallmatrix`, kann man kleine Matrizen speziell für Fließtext setzen. Man muß sich allerdings selbst um seine Klammern kümmern.

```
Wenn man einen Text schreibt, in
dem es um Matrizen geht, ist es
nat"urlich w"unschenswert, da"s
man auch eine zeigen kann, zum
Beispiel diese hier:
$\bigl(\begin{smallmatrix}
a & b \\ c & d
\end{smallmatrix} \bigr)$.
Das ist dann sehr sch"on
anzusehen, da der Text nicht all
zu sehr zerrissen wird.
```

Wenn man einen Text schreibt, in dem es um Matrizen geht, ist es natürlich wünschenswert, daß man auch eine zeigen kann, zum Beispiel diese hier: $\bigl(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\bigr)$. Das ist dann sehr schön anzusehen, da der Text nicht all zu sehr zerrissen wird.

8.10 Pfeile mit Text

Kommt einmal ein wenig mehr Text auf einen Pfeil, so wär's natürlich toll, wenn er sich entsprechend verlängert. Alles kein Problem, für sowas gibt's ja `\xleftarrow` und `\xrightarrow`

```
\begin{equation*}
    A \xleftarrow{n+\mu-1}
    B \xrightarrow[T]{n\pm i-1}
    C
\end{equation*}
```

$$A \xleftarrow{n+\mu-1} B \xrightarrow[T]{n\pm i-1} C$$

8.11 Das Doppelleben des vertikalen Balkens

In der Mathematik ist der senkrechte Strich mit vielerlei Bedeutungen beladen. Er kann heißen, daß es sich um einen Absolutbetrag handelt, ist er doppelt, so handelt es sich um eine Norm. In der Definition einer Menge heißt er „für die gilt", bei der Integration findet er Freunde, und die Stochastiker haben ihn sowieso ganz besonders gern.

Das widerspricht natürlich ein wenig dem Prinzip vom logischen Markup und so gibt es wenigstens für den Betrag und die Norm eigene Makros. Ein weiterer

Vorteil dieser Makros ist der, daß die Balken etwas näher an die Variablen gesetzt werden. Das erhöht die Lesbarkeit, wenn mehrere Beträge hintereinander kommen.

```
Ein Betrag: $\lvert x \rvert$
\qquad
Eine Norm: $\lVert x \rVert$
```

Ein Betrag: $\lvert x \rvert$ $\qquad$ Eine Norm: $\lVert x \rVert$

8.12 Breite Symbole

Um ein mathematisches Symbol breit zu schreiben, gibt es zwei Möglichkeiten. Die eine ist das Makro `\boldsymbol`. Dieses Makro wählt einfach im aktuellen Schriftsatz das breite Pendant aus und druckt es. Falls aber im aktuellen Schriftsatz keine breite Version des Zeichens definiert ist, so gibt es noch `\pmb`, das „poor mans bold". Mit ihm wird das Zeichen einfach mehrfach übereinander gedruckt und jedesmal minimal verrutscht.

8.13 Mehrzeilige Summationsgrenzen für Summen

Was in LaTeX nur mit dem häßlichen TeX-Kommando `\atop` geht, nämlich die Summationsgrenzen unter Summenzeichen mehrzeilig zu setzen, ist nun auch auf saubere Weise mit dem Makro `\substack` möglich. Die einzelnen Zeilen werden wie so oft durch einen doppelten Backslash abgegrenzt.

```
\begin{equation*}
    \sum_{
    \substack{0 \le i \le m \\
    0 < j < n}}
    P(i,j)
\end{equation*}
```

$$\sum_{\substack{0 \le i \le m \\ 0 < j < n}} P(i,j)$$

8.14 Neue Funktionen definieren

In Formeln werden öfter mal so Funktionen wie `\sin` oder `\ln` verwendet. Diese Kommandos haben die schöne Eigenschaft, daß der Funktionsname nicht kursiv gesetzt wird und die Namen teilweise auch subscripts annehmen, wie zum Beispiel `\lim`, bei dem öfter mal $x \to 0$ geht.

Falls man nun einen neuen mathematischen Operator (er-)findet, für den es keinen LaTeX-Befehl gibt, so kann man `\DeclareMathOperator` verwenden.

```
\DeclareMathOperator{\limsup}{limsup}
\DeclareMathOperator*{\liminf}{liminf}
```

Erst kommt der Name des Makros (wie bei `\newcommand`) und dann der Text, der diese Funktion bezeichnet. Die Sternform sagt aus, daß direkt unter dem Funktionsnamen noch Grenzen gesetzt werden können.

8.15 Mengensymbole

$\mathcal{AMS}$-LaTeX hat *wirklich* viele neue Schriften. Allein schon um die ganzen Symbole unterzubringen. Die meisten Symbole benötigt man erst, wenn man schon sehr tief in die Materie eingedrungen ist. Es gibt allerdings eine Schriftart, die immer wieder gerne benutzt wird – die blackboard font.

```
Auf vielen Aufgabenbl"attern wird
das Symbol f"ur die Menge der
ganzen Zahlen noch immer dadurch
dargestellt, da"s zwei "`Z"'s eng
aneinander ger"uckt werden, dabei
gibt es doch \AmSLaTeX{}, mit dem
man sehr sch"on $\mathbb{Z}$
schreiben kann. Des weiteren
$\mathbb{Q}$, $\mathbb{C}$ und
viele viele mehr. Wenn einem das
Ganze zu viel Tipperei wird, hilft
\cmd{newcommand}
```

Auf vielen Aufgabenblättern wird das Symbol für die Menge der ganzen Zahlen noch immer dadurch dargestellt, daß zwei „Z"s eng aneinander gerückt werden, dabei gibt es doch $\mathcal{AMS}$-LaTeX, mit dem man sehr schön $\mathbb{Z}$ schreiben kann. Des weiteren $\mathbb{Q}$, $\mathbb{C}$ und viele viele mehr. Wenn einem das Ganze zu viel Tipperei wird, hilft `\newcommand`

8.16 Ableitungen und Integrale

Physiker leiten gerne nach der Zeit ab. Das sagen sie, indem sie Punkte über ein Zeichen machen. Das geht bei LaTeX aber nur bis zur zweiten Ableitung, dann gibt es kein passendes Makro mehr. Verwendet man das **amsmath**-Paket, so gibt es auch die Kommandos `\dddot` für die dritte und `\ddddot` für die vierte Ableitung.

Auch bei den Integralen sind ein paar neue dabei. Die Makros `\iint`, `\iiint` und `\iiiint` für Mehrfachintegrale und falls man wiklich noch mehr Integrale hintereinander stellen will, so gibt es noch `\idotsint`, das zwei Integralzeichen setzt mit drei Punkten dazwischen.

Kapitel 9

Tabellen

Ein weiterer wichtiger Teil in Dokumenten sind alle Arten von Tabellen. Unter Tabelle hat man dabei einfach eine Anordnung von Text in Zeilen und Spalten zu verstehen. Dazu gehören normale *Text* - Tabellen genau so wie *mathematische* „Tabellen“ ; z.B. Matrizen oder Vektoren (in Koordinatendarstellung). Außerdem stellt LaTeX noch eine Umgebung zur Verfügung, die mit Tabulatoren arbeitet.

9.1 Text - Tabellen

Für Text - Tabellen gibt es in LaTeX die Umgebung `tabular` . Die Syntax lautet wie folgt:

```
\begin{tabular}{Position}
Zelle & Zelle & ...\\
Zelle & Zelle & ...\\
Zelle & Zelle & ...
\end{tabular}
```

Dabei sind die einzelnen Zellen durch das &-Zeichen voneinander getrennt, Zeilen werden durch Doppel-Backslash (\\ ) voneinander abgetrennt. Nur hinter der allerletzten Zeile steht kein Doppel-Backslash.

Das Argument *Position* gibt an, wie der Text in den Spalten gesetzt wird. Folgende Einträge sind möglich:

l Ausrichtung des Textes am linken Rand

r Ausrichtung des Textes am rechten Rand

c zentrierte Ausrichtung des Textes

p{*Länge*} Blocksatz
  Der (verpflichtende) Parameter *Länge* gibt an, wie breit die Spalte ist.

| vertikale Linie

|| vertikale Doppellinie

@{*Trennzeichen*} *Trennzeichen*
Anstelle von Leerraum werden die beiden Spalten durch das/die Zeichen *Trennzeichen* getrennt

Für jede Spalte muß genau eines der vier Positionierungsargumente `l,r,c,p{}` stehen. Beim Positionierungsargument `p{}` muß die Breite mit angegeben werden. Bei den anderen drei berechnet LaTeX die Breite der Spalte automatisch so, daß der längste Eintrag genau in die Spalte paßt. Ist die Gesamtbreite der Tabelle insgesamt größer als die Textbreite, wird beim LaTeX-Lauf eine Warnmeldung (Overfull `\hbox`) ausgegeben.

Zum besseren Verständnis folgt ein Beispiel:

```
Adrians Katze: \\[1ex]
\begin{tabular}{c|r@{.}l||p{3cm}|}
Gesamtl"ange: & 37 & 5 & Adrians
  Katze ist ganz sch"on lang. \\
Breite: & 4 & 27 &
  Daf"ur ist sie nicht so breit.
\end{tabular}
```

Adrians Katze:

Gesamtlänge:	37.5	Adrians Katze ist ganz schön lang.
Breite:	4.27	Dafür ist sie nicht so breit.

Außerdem wird eine Tabelle von LaTeX nicht vom normalen Text abgesetzt. Dies muß der Autor, falls gewünscht, selbst vornehmen.

Zur Tabellengestaltung gibt es noch drei weitere Kommandos. Diese sind:

`\hline` fügt eine horizontale Linie ein

`\cline{von-bis}` fügt eine horizontale Linie von Spalte *von* bis Spalte *bis* ein

`\multicolumn{Spaltenanzahl}{Position}{Text}` spannt einen Eintrag über mehrere Spalten

Der Befehl `\hline` darf nur am Anfang einer neuen Tabellenzeile (nach \\ ) stehen. Für eine Doppellinie schreibt man `\hline\hline`.

Bei `\multicolumn` gibt dabei *Spaltenanzahl* die Anzahl der überspannten Spalten an. Bei *Position* dürfen dieselben Argumente wie in der `tabular`-Umgebung selbst stehen; es muß genau eines der Argumente `r,c,l,p{}` vorkommen. Die ursprüngliche Positionierung in der Tabellen-Definition wird dabei aufgehoben; falls rechts von der letzten überspannten Spalte eine vertikale Linie steht, wird diese ebenfalls aufgehoben und muß eventuell im Positionierungsargument von `\multicolumn` wiederholt werden. *Text* ist der Inhalt der Tabellenzelle.

Mit Hilfe von `\multicolumn` ist es auch möglich, die Positionierung für eine einzelne Tabellenzelle zu ändern. Dazu schreibt man einfach `\multicolumn{1}{`*`Position`*`}{`*`Zelleninhalt`*`}`.

Ein abschließendes Beispiel verdeutlicht die Befehle nochmals:

```
\begin{tabular}{|l||r|r@{:}l|r|p{4cm}|}
\multicolumn{6}{c}{Tabelle vom 30. April 2000} \\
\hline
Verein & Spiele & \multicolumn{2}{|c|}{Tore} & Punkte
   & Kommentar \\
\hline \hline
Bayer Leverkusen & 32 & 45 & 38 & 64 & \\
Bayern München  & 32 & 48 & 37 & 61 & Rekordmeister und
   Pokalsieger 2000 \\
HSV & 32 & 59 & 38 & 55 & \\
\multicolumn{6}{l}{\dots} \\
MSV Duisburg & 32 & 36 & 66 & 21 & Schlußlicht \\
\hline
\end{tabular}
```

Tabelle vom 30. April 2000

Verein	Spiele	Tore	Punkte	Kommentar
Bayer Leverkusen	32	45:38	64	
Bayern München	32	48:37	61	Rekordmeister und Pokalsieger 2000
HSV	32	59:38	55	
...				
MSV Duisburg	32	36:66	21	Schlußlicht

9.2 Mathematische Tabellen

Braucht man im Mathematik-Modus eine Anordnung in Zeilen und Spalten, so kann man natürlich auch die `tabular`-Umgebung nutzen. Dabei wird innerhalb der Tabelle der Mathematik-Modus aufgehoben und alles im Textstil außerhalb des Mathematik-Modus gesetzt. Normalerweise ist das jedoch nicht das, was man wirklich will. Für Matrizen z.B. sollte auch der Inhalt der Tabelle im Mathematik-Modus gesetzt werden. Man könnte jetzt zwar für jede einzelne Zelle mit $ *Inhalt* $ explizit in den Mathematik-Modus umschalten, aber LaTeX hat dafür eine elegantere Lösung:

Die `array`-Umgebung. Ihre Syntax lautet wie folgt:

```
\begin{array}{Position}
Zelle & Zelle & ...\\
Zelle & Zelle & ...\\
Zelle & Zelle & ...
\end{array}
```

Für *Position* dürfen dabei alle Zeichen, die auch in der `tabular`-Umgebung erlaubt waren, eingesetzt werden, es müssen jedoch wieder genau so viele `l,c,r,p{}` vorkommen wie Spalten in der Tabelle. Auch innerhalb der `array`-Umgebung sind die Befehle `\multicolumn` und `\hline` erlaubt, sie haben dieselbe Syntax und dieselben Auswirkungen wie in einer `tabular`-Umgebung. Der Inhalt der Zellen wird jedoch im mathematischen Modus gesetzt, und eine `array`-Umgebung darf nur im mathematischen Modus vorkommen.

Eine der häufigsten Anwendungen dieser Umgebung sind Vektoren:

```
\[
\vec{r} = \left(
  \begin{array}{c}
   x \\
   y \\
   z
  \end{array}
 \right)
\]
```

$$\vec{r} = \left(\begin{array}{c} x \\ y \\ z \end{array} \right)$$

und Matrizen:

```
\[
1_2 = \left(
  \begin{array}{cc}
    1 & 0 \\
    0 & 1 \\
  \end{array}
 \right)
 \]
```

$$1_2 = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \\ \end{array} \right)$$

Eine weitere wichtige Anwendung in der Mathematik sind Fallunterscheidungen. Diese kann man ebenfalls mit der `array`-Umgebung darstellen.

```
\[ \delta_{ij} = \left\{
 \begin{array}{cl}
  1 & \mbox{ f"ur $i=j$} \\
  0 & \mbox{ f"ur $i \neq j$}
\end{array} \right. \]
```

$$\delta_{ij} = \left\{ \begin{array}{cl} 1 & \mbox{ für $i=j$} \\ 0 & \mbox{ für $i \neq j$} \end{array} \right.$$

Insbesondere die zweite Spalte kann jedoch relativ aufwendig werden, da zunächst in den Textmodus mit `\mbox` geschalten wird, um dann teilweise wieder mit `$ ...$` in den Mathematikmodus zurückzuschalten. Mit $\mathcal{AMS}$-LaTeX gibt es dafür eine eigene `cases`-Umgebung, die in Kapitel 8 erklärt wird (s. S. 54).

9.3 Tabulatoren

Neben diesen beiden Umgebungen stellt LaTeX noch eine dritte zur Verfügung, die sich zwar auch mit der Anordnung von Text in Zeilen und Spalten beschäftigt, aber mehr den Tabstopps einer Schreibmaschine ähnlich ist: die `tabbing`-Umgebung.

Ihre Syntax ist recht einfach:

```
\begin{tabbing}
Text
\end{tabbing}
```

Der interessantere Teil sind die zusätzlichen Kommandos, mit deren Hilfe man innerhalb des Textes Tabulatoren setzen und sie anspringen kann. Zur Übersicht folgt zunächst eine Aufstellung aller Befehle, dann eine Erläuterung im Detail.

`\=` setzt einen Tabstopp an der aktuellen Position

`\>` springt zum nächsten Tabstopp

`\+` rückt alle folgenden Zeilen um einen (weiteren) Tabstopp ein

`\-` rückt alle folgenden Zeilen um einen Tabstopp weniger ein

`\<` rückt die aktuelle Zeile um einen Tabstopp weniger ein

`\'` setzt den vorangegangenen Text der Zelle an den rechten Rand und springt zum nächsten Tabstopp

`` \` `` springt ans Ende der Zeile und setzt den Text dort rechtsbündig

`\\` beendet die aktuelle Zeile

`\kill` löscht die aktuelle Zeile

In einer `tabbing`-Umgebung müssen zunächst einmal die Tabstopps gesetzt werden. Dies geschieht mit dem Befehl `\=`, der an der aktuellen Position einen Tabstopp einfügt. Der nullte Tabstopp ist dabei immer der linke Rand.

Der Befehl `\>` springt zum nächsten Tabstopp. Dabei wird derjenige benutzt, der in der numerischen Reihung als nächster folgt, nicht, wie man vielleicht auch erwarten könnte, der nächste rechts vom Textende. Dies kann insbesondere bedeuten, daß LaTeX eventuell auch zurückspringt und anderen Text überdruckt.

```
\begin{tabbing}
 Dies ist ein \= Text. \\
 Diese Zeile ist zu lang.
   \> / / / / /
\end{tabbing}
```

Dies ist ein Text.
Diese Zeile i/st/z/u /la/ng.

Oft kommt es vor, daß man in der ersten Zeile die Tabstopps nicht passend setzen kann, z.B. weil man weiß, daß später ein etwas längerer Text kommt, der ebenso zwischen zwei Tabstopps passen soll. Dafür gibt es die Möglichkeit, Text zuerst von LaTeX setzen zu lassen und dabei die Tabstopps zu verteilen, und dann mit `\kill` den Text wieder verwerfen, so als ob er nie gesetzt worden wäre. Wenn wir dies auf unser obiges Beispiel anwenden, sieht es so aus:

```
\begin{tabbing}
 Diese Zeile ist zu lang. \= \kill
 Dies ist ein \> Text. \\
 Diese Zeile ist zu lang.
   \> / / / / /
\end{tabbing}
```

Dies ist ein	Text.
Diese Zeile ist zu lang.	/ / / / /

Zeilenumbrüche werden innerhalb der `tabbing`-Umgebung übrigens von LaTeX nicht selbst durchgeführt, sondern müssen vom Benutzer jeweils selbst mit `\\` angegeben werden. Außerdem wirken die Tabstopps so, als wären die Zellen in geschweifte Klammern eingeschlossen. D.h. global wirkende Definitionen wie z.B. `\bfseries` werden durch den Sprung zum nächsten Tabstopp automatisch beendet.

```
\begin{tabbing}
\itshape Kursiver Text
  \= nicht mehr kursiv
\end{tabbing}
```

Kursiver Text nicht mehr kursiv

LaTeX bietet auch die Möglichkeit, nach einem `\\`-Kommando den Text an einem anderen als dem nullten Tabstopp beginnen zu lassen. Dafür gibt es `\+`, was den Text ab der nächsten Zeile jeweils einen Tabstopp weiter links beginnen läßt. Das Gegenteil davon ist `\-`, was den Text ab der nächsten Zeile wieder einen Tabstopp weiter links beginnen läßt (jedoch nie mehr links als der linke Rand). Der Befehl `\<` darf nur am Anfang einer Zeile stehen, und setzt nur für diese Zeile den Text einen Tabstopp weiter nach links. Diese drei Befehle dürfen auch geschachtelt werden, d.h. der Befehl `\<\<` läßt z.B. die aktuelle Zeile zwei Tabstopps weiter links beginnen.

```
\begin{tabbing}
 Katze: \= Bernhardiner \= \kill
 Hund: \> Bernhardiner
    \> bellt \+\+ \\
       gro"s \- \\
    Dackel \> bellt auch \+ \\
              klein \\
 \< \< Katze: \> \> miaut
\end{tabbing}
```

Hund:	Bernhardiner	bellt
		groß
	Dackel	bellt auch
		klein
Katze:		miaut

Manchmal kommt es auch vor, daß man kurzfristig die Tabstopps verändern möchte, um danach wieder auf die ursprünglichen zurückzuschalten. Auch dafür gibt es Befehle. `\pushtabs` speichert die aktuellen Tabulatorpositionen, `\poptabs`i stellt die zuletzt gespeicherte Einstellung wieder her. Es können auch mehrere Ebenen gespeichert werden, es müssen aber genau so viele `\pushtabs`-wie `\poptabs`-Befehle auftreten.

Die beiden Befehle `\'` und `` \` `` dienen dazu, Text rechtsbündig zu setzen. Der Befehl `` \` `` setzt den gesamten folgenden Text rechtsbündig an den rechten Rand. Nach diesem Befehl darf bis zum Zeilenendebefehl keiner der Befehle `\>`, `\=` oder `\'` stehen. Der Befehl `\'` setzt den vorangegangenen Text der aktuellen Zelle an den rechten Rand der vorangegangenen Zelle und den folgenden Text in die aktuelle Zelle. Dieser Befehl kann übrigens auch dazu benutzt werden, Text in der `tabbing`-Umgebung rechtsbündig an den linken Rand zu setzen.

```
\begin{tabbing}
Gewinne und Einnahmen:
  \= 100.000 \= DM \\
Verluste: \> --- \' 100.000 \> DM \\
Gesamt: \` nichts "ubrig
\end{tabbing}
```

Gewinne und Einnahmen:	100.000	DM
Verluste: —	100.000	DM
Gesamt:		nichts übrig

Zum Schluß gebe ich noch mal ein ausführliches Beispiel, das die meisten Funktionen verdeutlicht:

```
\begin{tabbing}
 Einkaufsliste: \= Schokolade
   \= 2 St"uck \= \kill
 Einkaufsliste: \>
   Zucker \> 500g \>
     braun \+ \\
   Mehl \> 1000g \>
     fein \+ \\
            750g \>
     grob \- \\
   Eier \> 2 St"uck \>
     wei"s \+\+ \\
     Gr"o"se M \-\- \\
   \pushtabs
   Schokolade \=
     Vollmilch \= 100g \+ \\
     Nougat \> 100g \\
     wei"s \> 200g \- \\
   \poptabs
   Spielzeug \> Rassel \+ \\
                Kuscheltier \\
         und \' Schnuller \\
   \` Das war's.
\end{tabbing}
```

Einkaufsliste:	Zucker	500g	braun	
	Mehl	1000g	fein	
		750g	grob	
	Eier	2 Stück	weiß	
			Größe M	
	Schokolade	Vollmilch	100g	
		Nougat	100g	
		weiß	200g	
	Spielzeug	Rassel		
		Kuscheltier		
	und	Schnuller		
				Das war's.

Kapitel 10

Einbinden von Grafik

10.1 Der Grundgedanke oder wie sag ichs meinem TEX

TEX war von Knuth ursprünglich dafür ausgelegt, seine mathematischen Abhandlungen zu setzen. Also primär für Fließtext mit Formeleinschüben sowohl im Text, als auch in eigenen Absätzen. Um gute Schriftzeichen zu erhalten, entwickelte er hierfür ein eigenes Programm – METAFONT. Die Arbeitsteilung zwischen METAFONT und TEX (das ja LATEX zugrunde liegt) sieht nun folgendermaßen aus: METAFONT kümmert sich um das Aussehen der einzelnen Schriftzeichen und sagt TEX nur, wie hoch und breit jedes einzelne Zeichen ist.

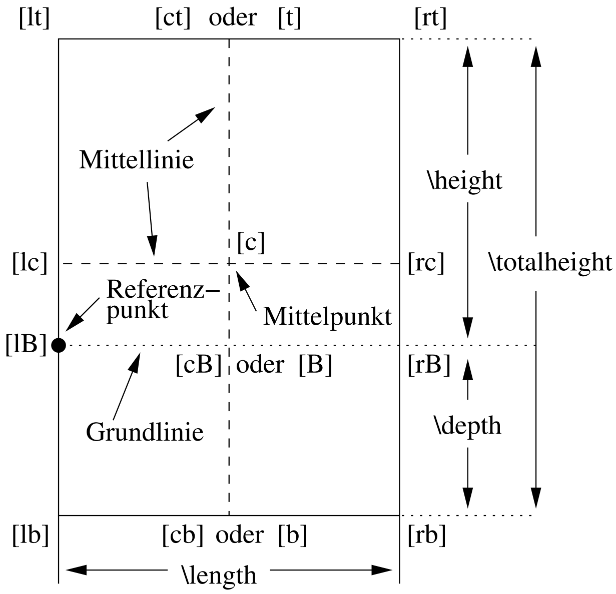


Abbildung 10.1: Eine TEX-BoundingBox

Das genügt TEX völlig, um seine Arbeit zu tun. Die besteht nämlich nur darin, die Seiten mit verschieden großen Rechtecken (*BoundingBoxes*; siehe auch die Abbildung nebenan) zu bestücken. Dabei sollten sich die Rechtecke nicht überlappen und der Freiraum zwischen ihnen sollte nicht übermäßig groß sein. Dieses Verhalten kann man sich zu nütze machen, um nicht nur Text, sondern beliebigen Inhalt auf eine Seite zu bekommen. Man muß TEX nur sagen, wie hoch und breit der zu reservierende Kasten ist.

Genau das macht das **graphicx**-Paket. Mit ihm ist es möglich, Bilder in Texte einzufügen. Nebenbei stellt dieses Paket noch einige Routinen bereit, die auch mit Text arbeiten. So ist es möglich, beliebige Inhalte zu stauchen, zerren, spiegeln oder zu drehen.

10.2 Das graphicx-Paket

Unter PostScript gilt alles auf dem Papier als Graphik. Sogar die einzelnen Schriftzeichen. Diese Vorgehensweise ist im digitalen Textsatz nicht all zu ungewöhnlich und so werden auch vom **graphicx**-Paket Makros bereitgestellt, die das Erscheinungsbild des Textes modifizieren.

10.2.1 Rotation

Das Kommando `\rotatebox` packt einen Text in eine `\mbox` und rotiert diese um einen frei wählbaren Ursprung.

`\rotatebox`[*optionen*]{*Winkel*}{*Text*}

Folgende Optionen können verwendet werden

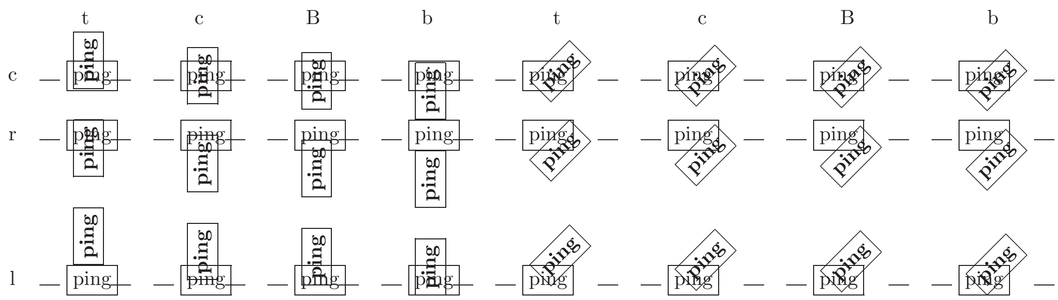
origin Diese Einstellung gibt an, welcher Punkt als Ursprung gewählt wird. So rotiert etwa `[origin = rB]` um den Punkt, der am rechten Rand der BoundingBox auf der Grundlinie liegt. Eine „Auflistung“ aller Ursprungspunkte findet sich in der Abbildung auf Seite 66.

x Feintuning für den Ursprung. Der Ursprung wird noch einmal verschoben. Diesmal relativ zu dem Punkt, der unter `origin` bezeichnet ist.

y Genauso wie `x` möglich sind zum Beispiel `[x = 0.5cm, y = -1cm]` oder andere von LaTeX verwendete Längenangaben.

units Hiermit wird festgelegt, wie groß ein voller Kreis ist. So kann man etwa mit `[units = 6.283185]` auf Bogenmaß schalten, oder mit `[units = −360]` die Drehrichtung ändern.

Ein kleines Beispiel, in der eine Kiste mit Text um jeweils 90° bzw. 45° um alle namentlich ansteuerbaren Ursprungspunkte gedreht wird.



10.2.2 Skalierung

Skalierung um einen Faktor

Um ein paar Wörter mal schnell zu vergrößern, zu verkleinern oder sonst wie zu verzerren gibt es das Kommando `\scalebox`. Die Syntax dazu sieht folgendermaßen aus

`\scalebox`{*h-Faktor*}[*v-Faktor*]{*Text*}

Die h- und v-Faktoren geben an, um welchen Wert der Text skaliert werden soll. Bleibt der vertikale Faktor leer, so wird die vertikale Skalierung übernommen. Unterscheiden sich beide Werte, so wird der Text eben gezerrt, so wie dieser hier. Einen Spezialfall der `\scalebox` stellt die `\reflectbox` dar.

`\reflectbox`{*Text*}

Der Effekt ist der gleiche, wie bei `\scalebox`{-1}[1]{*Text*}. Statt „viel blah blah“, erhält man “halb halb leiv„.

Skalierung auf gewünschte Maße

Anstatt einen Skalierungsfaktor anzugeben kann man seinen Text auch auf eine feste Größe skalieren. Hierzu gibt es die `\resizebox`, von der es auch gleich mal zwei Varianten gibt:

`\resizebox`{*Breite*}{*Höhe*}{*Text*} `\resizebox*`{*Breite*}{*Höhe*}{*Text*}

Die beiden Versionen unterscheiden sich in der Interpretation der Höhenangabe. Während die „normale“ Form damit die Höhe des Buchstabens ab der Grundlinie meint, geht die Sternform davon aus, daß mit *Höhe* die Größe nach oben plus die Tiefe gemeint ist. Für die Größenangaben können auch die Werte `\height`, `\width`, `\totalheight` und `\depth` verwendet werden.

Außerdem kann in einem von beiden Argumenten ein Ausrufezeichen verwendet werden. Dann sucht sich LaTeX den Wert so, daß der Text nicht verzerrt wird.

`\resizebox`{1cm}{!}{Ein Text}: Ein Text

`\resizebox`{0.8`\width`}{1cm}{Noch ein Text}: Noch ein Text

10.3 Bilder einbinden

Nachdem wir uns nun an Schriftzeichen ausgetobt haben, können wir langsam zu „echten“ Grafiken hinübergleiten. Das **graphicx**-Paket versucht es einem hier so einfach wie möglich zu machen. Der Befehl um ein Bild einzubinden heißt sinnigerweise `\includegraphics` und benötigt so gut wie keiner Erklärung. Die Bildformate, die hiermit verarbeitet werden können hängen vom dvi-Treiber ab, **eps** (encapsulated PostScript) funktioniert so gut wie immer und meistens noch **tiff** und **pcx**. Der Dateityp wird dann an der Endung erkannt. Hier erst mal die Syntax

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

Auch hier gibt es wieder eine Form mit Stern und eine ohne. Die Form mit Stern schneidet alles vom Bild ab, was aus einem angegebenen Rechteck herauslappt. Die Form ohne Stern überschreibt dann einfach mit dem Bild den Text drum rum.

Zu diesem Befehl gibt es noch ein paar Optionen. Werden mehrere Optionen angegeben, so werden diese durch Kommata getrennt. Die einzelnen Optionen werden von links nach rechts ausgewertet, was bei manchen Operationen durchaus wichtig sein kann.

bb Hiermit wird die *BoundingBox* angegeben. Die (vier) Werte müssen durch Leerräume getrennt sein. Die Werte bezeichnen der Reihe nach „x-Wert links unten“, „y-Wert links unten“, „x-Wert rechts oben“ und „y-Wert rechts oben“.

bbllx, bblly, bburx, bbury Alternative zu **bb**. Es geht entweder „**bbllx = a, bblly = b, bburx = c, bbury = d**“ oder „**bb = a b c d**“.

natwidth, natheight Noch eine alternative zu **bb**. Die Option „natheight = h, natwidth = w“ ist gleichbedeutend mit „bb = 0 0 h w“.

hiresbb PostScript-Dateien enthalten normalerweise einen Kommentar, in dem die BoundingBox der Seite angegeben ist (`%%BoundingBox`). Außerdem gibt es einen Kommentar, der einen hochauflösenderen Rahmen anbietet (`%%HiResBoundingBox`). Wenn der Schaltern **hiresbb** auf true gesetzt wird (standardmäßig an), schaut LaTeX nach dem HiRes-Kommentar

viewport Mit dieser Option kann man sich Teile des Bildes anzeigen lassen. Die Syntax ist genau wie bei **bb**. Es werden vier Werte benötigt, die durch Leerzeichen getrennt werden müssen. Diese Werte sind relativ zum Ursprung des Bildes (der implizit durch die Boundingbox gesetzt wurde). Maßeinheit ist hier der typographische Punkt *1 pt* von denen 72 Stück eine Länge von einem inch haben.

trim Hierdurch wird das Bild am Rand beschnitten. Erwartet werden wieder vier Werte. So bewirkt zum Beispiel **trim = 1 2 3 4**, daß das Bild am linken Rand einen Punkt gekürzt wird, unten zwei, rechts drei und oben vier Punkte (72pt/inch)

angle Winkel um den das Bild gedreht wird

origin Der Ursprung für die Rotation, erlaubt sind die gleichen Werte wie bei `\rotatebox`. Siehe hierzu auch die Abbildung auf Seite 66

width Wird dieser Parameter angegeben, so wird das Bild auf die gegebene Breite skaliert.

height Wie bei **width** nur für die Höhe

totalheight Gibt die Gesamthöhe an, auf die das Bild skaliert werden soll. Dieser Wert bezieht alles über und unter der Grundlinie mit ein. Achtung: Wenn das Bild vorher rotiert wurde, verändert sich dadurch die Höhe

keepaspectratio Ein Boolescher Wert. Wird er auf true gesetzt, so verzerren die Optionen **height** und **width** das Bild nicht. Es wird vielmehr so skaliert, daß keiner der gegebenen Parameter überschritten wird.

scale Skalierungsfaktor. Wird auf Höhe und Breite angewendet.

clip Ebenfalls ein Boolescher Wert (true oder false, standardmäßig auf true gesetzt). Die Angabe von **clip** ist gleichbedeutend mit der Verwendung der Sternform. Überstehende Teile des Bildes werden weggeschnitten.

draft Boolescher Wert. LaTeX schaltet temporär in den *draft*-Modus. Das Bild wird also zugunsten eines schnelleren Übersetzungslaufes nicht mit eingebunden, aber schon mal Platz reserviert.

type Der Typ der Grafikdatei

ext Die Endung der Datei. Nur mit **type** verwenden.

read Die Endung der zu lesenden Datei. Nur mit **type** verwenden.

command Hier wird das Kommando angegeben, mit dem man von der zu lesenden Datei zur einzufügenden Datei kommt. Ebenfalls nur mit **type** verwenden.

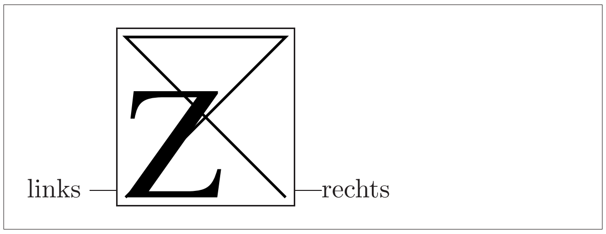
Es gilt noch eine kleine Falle bei der Verwendung der Optionen zu umschiffen. Die Optionen werden von links nach rechts ausgewertet. Also wird bei `[angle=90, height=1cm]` das Bild erst rotiert und dann auf einen Zentimeter Höhe gebracht, während `[height=1cm, angle=90]` zuerst das Bild skaliert und dann rotiert. Die Ergebnisse der beiden Fälle unterscheiden sich dadurch.

Noch eine Bemerkung zu den letzten paar Optionen. Es ist möglich, eine Datei on the fly zu erzeugen, indem man zum Beispiel seine PostScript-Dateien gezippt lagert und von LaTeX vor dem einfügen entpacken läßt. Hierfür sind diese Optionen gedacht.

Das `\includegraphics`-Makro erlernt man am besten, indem man ein wenig mit den verschiedenen Optionen herumspielt.

Erst einmal ganz ohne Optionen

```
links ---%
\fbox{\includegraphics{z}}%
---rechts
```

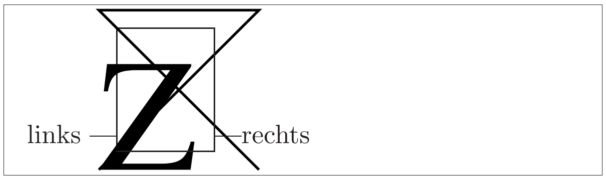


Dann ein bißchen kleiner

```
links ---%
\fbox{\includegraphics%
[scale = 0.4]{z}}%
---rechts
```

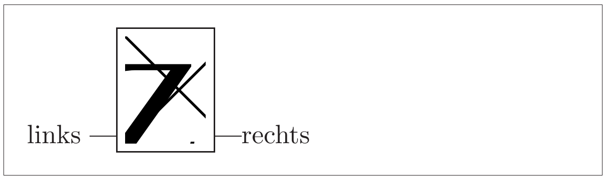
Oder nur ein kleiner Ausschnitt ...

```
links ---%
\fbox{\includegraphics%
[bb = 10 10 40 50]{z}}%
---rechts
```



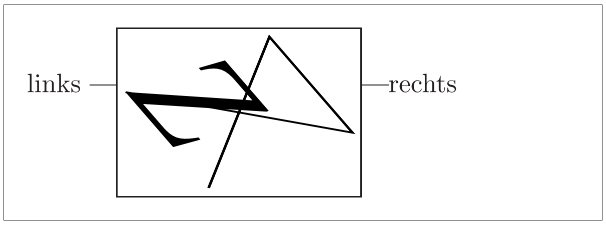
Hoppla, da stand wohl grad was über, besser wir schneiden das mit `clip` ab

```
links ---%
\fbox{\includegraphics%
[bb = 10 10 40 50, clip]{z}}%
---rechts
```



Zum Schluß noch ein etwas surrealistischeres. Zu beachten ist hierbei, daß das Bild erst gedreht wird und dann zurechtgestutzt wird.

```
links ---%
\fbox{\includegraphics%
[angle = -60,%
totalheight=20mm,width=30mm]%
{z}}%
---rechts
```



10.4 Einstellmöglichkeiten am graphicx-Paket

Es gibt noch einige Makros, mit denen man quasi Umgebungsvariablen des Paketes setzen kann. Vorrangig sind das der Suchpfad nach Bildern und die Standard-Dateiendung.

```
\graphicspath{Verz-Liste}
```

Mit diesem Komando gibt man den Suchpfad an, in dem das System nach Bilddateien suchen soll. In der Liste ist jeder Eintrag in ein paar geschweifter Klammern eingebettet (selbst wenn es nur einen Eintrag gibt). Also etwa so `\graphicspath{{eps/}{tiff/}}`. Bei diesem Beispiel handelt es sich um die UNIX-Schreibweise. Auf anderen Systemen muß der Pfad evtl. anders angegeben werden.

```
\DeclareGraphicsExtension{Endungsliste}
```

Wenn in einer `\includegraphics`-Anweisung keine Dateiendung angegeben wird, so sucht das System, ob es eine Datei mit dem gegebenen Anfang und einer der Standardendungen findet. Die Liste besteht aus verschiedenen Dateiendungen, die durch Kommata abgetrennt sind.

```
\DeclareGraphicsRule{Endung}{Dateityp} {zu lesen}{Kommando}
```

Hiermit kann für festgelegt werden, wie sich das System verhalten soll, wenn es auf eine Datei mit der angegebenen Endung stößt. Auf diese Weise kann man einen Satz von Standardoptionen für das `\includegraphics`-Kommando schaffen. Statt die Endung anzugeben ist es auch erlaubt, ein Sternchen einzufügen. Dadurch wird die Standardvogehensweise für alle unbekannten Dateien festgelegt. Also etwa so, um alle *.ps.gz Dateien vorher durch `zcat` zu pipen und die Informationen über die BoundingBox aus der zugehörigen .ps.bb-Datei zu holen:

```
\DeclareGraphicsRule{.ps.gz}{eps}{.ps.bb}{'zcat #1}
```

10.5 Text der um Bilder fließt

10.5.1 Das floatflt-Paket

Jetzt können wir also Bilder einbinden, aber bisher wird die Grafik noch in eine Zeile Text gesetzt, was das Textbild ein wenig zerreißt. Alternativ kann das Bild von zwei Leerzeilen eingefaßt werden, wodurch es einen eigenen Absatz erhält. Schön wär es aber, wenn das Bild am Rand der Seite abgebildet wird und der Text drum herum fließt.

Für so etwas gibt es das **floatflt**-Paket. Es ist eine Erweiterung der normalen figure- bzw. table-Umgebung. Die Umgebung selbst wird wie die beiden anderen gehandhabt, allerdings wird der Text außen herumgeleitet.

Das Paket wird im Kopf des Dokuments mit eingebunden und erhält bei bedarf noch eine Option

`\usepackage`[*Ausrichtung*]{floatflt}

Die Ausrichtung gibt an, an welche Kante der Seite die Umgebung standardmäßig gesetzt wird.

rflt Die Umgebung wird immer, wenn nicht anders angegeben auf die rechte Seite gesetzt.

lflt Die Umgebung wird als Standard auf die linke Seite gesetzt.

vflt Auf ungeradzahligen Seiten wird die Umgebung rechts gesetzt, auf geradzahligen Seiten an den linken Rand. Diese Option ist standardmäßig gesetzt.

10.5.2 Verwendung der Umgebung

Wenn der Teil erst mal geschafft ist, hat man zwei neue Makros zur Verfügung.

`\begin`{floatingfigure}[*Ausrichtung*]{*Breite*}
`\end`{floatingfigure}

`\begin`{floatingtable}[*Ausrichtung*]{*Breite*}
`\end`{floatingtable}

Für die Ausrichtung gibt es vier Möglichkeiten

r Die Umgebung wird im aktuellen Absatz an den rechten Rand der Seite gesetzt.

l Die Umgebung wird im aktuellen Absatz an den linken Rand der Seite gesetzt.

p Die Umgebung landet rechts, wenn die Seitenzahl ungerade ist, sonst links.

v Die gleiche Option wird verwendet, wie sie beim einbinden des Paketes angegeben wurde.

Wird keine Option verwendet, so wird **p** verwendet. Innerhalb der beiden Umgebungen kann der Befehl `\caption` verwendet werden. Er nimmt einen Text als Argument, der dann als Bildunterschrift abgedruckt wird. Hierbei zeigt sich auch der Unterschied zwischen **floatingtable** und **floatingfigure**. Während beim einen in der Bildunterschrift ein „Abbildung x.x“ vorangestellt wird, steht beim anderen „Tabelle x.x“.

Kapitel 11

`\documentclass`-Optionen und Einbindung von Paketen

Nicht für alles sind im Standard-LaTeX Befehle vorhanden, und manchmal soll eine bestimmte Aufgabe auf die eine Art gelöst werden, und ein anderes Mal auf eine andere. Vor solchen Problemen steht im allgemeinen nicht nur ein Benutzer, sondern sehr viele, und es wäre recht mühsam, müsste jeder für sich das Rad neu erfinden. Deshalb bietet LaTeX in der Präambel die Möglichkeit, verschiedene Layouts auszuwählen und *Pakete*, „Problemlösungen für ein spezielles Gebiet“, zusätzlich zu laden.

11.1 `\documentclass`-Optionen

Als erster Befehl in einem LaTeXDokument steht immer `\documentclass`.

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

Damit wählt man ein bestimmtes Grundlayout des Dokuments. *Klasse* gibt dabei eine bestimmte Richtung an, mit den *Optionen* lassen sich einige Parameter verändern.

Für Klasse sind folgende Werte üblich und auf allen LaTeX-Installationen vorhanden:

`article` für kurze Texte, Artikel in Zeitschriften, Praktikums-, Seminararbeiten, Berichte u.v.a.

`report` für längere Texte aus mehreren Kapiteln, Diplom-, Doktorarbeiten, Skripten, kleine Bücher

`book` für normale Bücher

`letter` Ein Brief, allerdings ein amerikanischer. Diese Dokumentklasse ist zur Erzeugung von Briefen wie sie im amerikanischen Raum verwendet werden gedacht. Hierzulande verwendet man besser die, leider nicht unbedingt in jeder LaTeX-Distribution enthaltenen, Dokumentklassen **dinbrief** oder **g-brief**. Falls nötig sind sie vom *CTAN* (s. 12.2) zu beziehen.

`slides` Diese Dokumentklasse ist zur Erstellung von Overhead-Folien für Vorträge gedacht. Eine evtl. bessere bzw. einfacher zu handhabende Alternative dazu ist die Klasse **foils** aus dem Paket foiltex das ebenfalls bei *CTAN* erhältlich ist.

Die Klassen unterscheiden sich z.B. in ihrem Verhalten, was als Standard für bestimmte *Optionen* voreingestellt ist. Mögliche *Optionen* sind:

`10pt, 11pt, 12pt` stellt die Standard-Schriftgröße ein und passt die absoluten Werte der Schriftgrößen-Befehle an. Voreingestellter Wert ist `10pt`.

`*paper` gibt die Papiergröße an. LaTeX berechnet daraus auch die entsprechenden Werte für den Textbereich. Gültige Werte sind `a4paper, a5paper, b5paper, letterpapaer, executivepaper und legalpaper`, Standard ist `letterpaper`.

`landscape` setzt den Text im Querformat, d.h. vertauscht aktuelle Papierbreite und -höhe

`fleqn` setzt Gleichungen linksbündig mit etwas Einrückung vom linken Rand. Standard ist zentriert.

`leqno` setzt die Formelnummer für abgesetzte Formeln links von der Gleichung. Standard ist rechts von der Gleichung.

`titlepage, notitlepage` gibt an, ob nach der Titel„seite“ eine neue Seite gestartet werden soll. Standard ist nein für `article` und ja für `report` und `book`.

`twocolumn` weist LaTeX an, den Text zweispaltig zu formatieren. Standard ist einspaltig.

`twoside, oneside` gibt an, ob LaTeX den Text für zweiseitigen Druck formatieren soll. Dies betrifft insbesondere die Einstellungen für eine eventuelle Kopfzeile und den automatischen Seitenlängenausgleich. Standard ist einseitige Formatierung für `article` und `report`, und zweiseitig für `book`.

`openright, openany` gibt bei zweiseitiger Formatierung an, ob ein neues Kapitel(`\chapter`) auf jeder oder nur auf einer rechten Seite beginnen darf. Diese Option ist für `article` nicht sinnvoll, da es dort den Befehl `\chapter` nicht gibt. Standard für `report` ist auf jeder Seite, für `book` nur auf rechten Seiten.

Sollen mehrere Optionen angegeben werden, schreibt man diese in beliebiger Reihenfolge durch Komma getrennt in das optionale Argument.

Für alle Einstellungen, die den endgültigen Ausdruck betreffen, gilt ein wichtiger Hinweis. LaTeX kann *nur* das *Layout* des Textes beeinflussen, es hat *keinen* Zugriff darauf, wie das Dokument letztendlich ausgedruckt wird. So kann die Option `twoside` z.B. nicht den Drucker anweisen, das Dokument tatsächlich zweiseitig auszudrucken. Es kann nur die Voraussetzungen zum besten Aussehen dafür machen. Dies gilt insbesondere auch für die Optionen zur Papiergröße. Das dvi-Format speichert intern keine Angaben zur Papiergröße, daher kann LaTeX diese Informationen nicht an das Anzeigeprogramm oder den Druckertreiber weitergeben. Dies muss der Benutzer selbst vornehmen. Im Normalfall muss man sich als Benutzer auf der anderen Seite aber darüber keine Gedanken machen. Meist ist auf europäischen Computersystemen als Standard das Papierformat A4 eingestellt, was in der Regel auch das Format ist, in dem die Dokumente erstellt werden. Ansonsten bieten die Hilfe-Dateien der jeweiligen Programme Unterstützung, um die Ausgabe wie gewünscht durchzuführen.

Für Seiten im Querformat lauten für die beiden gängigsten Programme `xdvi` und `dvips` die Befehlsaufrufe wie folgt:

```
> xdvi -paper a4r <Datei>
> dvips -t a4 -t landscape <Datei>
```

Für Querformat gibt es in LaTeXauch einen Befehl, der diese Information an dvips sozusagen „durchreicht“. Dieser lautet `\special{landscape}` .

11.2 Pakete

Der Standard-Befehlssatz von LaTeX bietet für die meisten Dokumente eine ausreichende Funktionalität. Es kann aber auch vorkommen, dass für bestimmte Aufgaben zusätzliche Funktionen benötigt werden. Dann kann man sich diese über `\newcommand`-Befehle entweder selbst definieren, oder im `CTAN` (s. Kapitel 12.2) nachschauen, ob jemand bereits eine Lösung für diese Aufgabe geschrieben hat, ein sogenanntes Paket.

Pakete werden mit folgendem Befehl geladen, der nur in der Präambel des Dokuments stehen darf:

```
\usepackage[Option]{Paket}
```

Dabei gibt *Paket* den Namen des Paketes an, das geladen werden soll, und *Option* wählt analog zum Befehl `\documentclass` für etwas eine andere Variante. Was für *Option* erlaubt ist, findet sich in der Dokumentation zum jeweiligen Paket.

Die Anzahl bereits geschriebener Pakete ist sehr groß, es gibt jedoch drei Pakete, die in deutschsprachigen Texten oft benötigt werden und die teilweise implizit bereits benutzt wurden, ohne genau zu wissen, was sie eigentlich für Auswirkungen haben.

Da LaTeX mit Ausrichtung für den amerikanischen (Sprach-)Raum geschrieben wurde, wird für deutschsprachige Texte normalerweise immer das Paket `german` geladen. Es bietet eine vereinfachte Eingabe für Umlaute (die in diesem Text bereits immer vorausgesetzt wurde), es ändert vordefinierte Namen wie z.B. *Chapter* für Kapitelüberschriften in ihr deutsches Pendant *Kapitel* und es verändert einige interne Einstellungen auf Werte, die im deutschen Buchdruck üblich sind.

Ein weiteres Paket, das eine verbesserte Unterstützung für das in Europa übliche A4-Format bringt, ist das Paket `a4`. Es setzt im wesentlichen die Textbreite und -höhe auf Werte, die ein A4-Blatt optisch besser füllen als Standard-LaTeX. Dieses Paket besitzt eine Option `widemargins`, das die Textbreite etwas verkleinert und den Textrand dafür etwas vergrößert.

Das dritte Paket betrifft die Verarbeitung von Zeichen, die nicht zum normalen Zeichensatz gehören. Normalerweise erkennt LaTeX nur Zeichen, die zum sogenannten ASCII-Zeichensatz gehören. Dieser besteht aus 128 Zeichen und enthält alle Buchstaben, Ziffern sowie einige Sonderzeichen (\$,%, ˜, . . .). Insbesondere gehören dazu aber nicht die deutschen Umlaute und ß. Heutige Computersysteme arbeiten inzwischen mit Zeichensätzen aus 256 Zeichen. In dem in Westeuropa genutzten Zeichensatz *ISO-Latin-1* finden sich die Umlaute und ß in den zusätzlichen 128 Zeichen. Das Paket `inputenc` erlaubt die Eingabe dieser Zeichen und setzt sie intern in die entsprechenden LaTeX-Befehle um. Als optionales Argument erwartet das Paket die verwendete Zeichensatztabelle, für ISO-Latin-1 ist dies `latin1`. Damit ist es z.B. möglich, Umlaute und ß direkt als solche in ein LaTeX-Dokument einzugeben. Dies kann besonders nützlich sein, falls man einen bereits bestehenden Text mit Umlauten in ein LaTeX-Dokument verwandeln will, da dies einiges an Konvertierungsarbeit spart.

In der Präambel dürfen beliebig viele `\usepackage`-Befehle stehen. Mehrere Pakete, die dieselben Optionen besitzen, können aber auch mit einem `\usepackage`-Befehl geladen werden. Die einzelnen Paketnamen werden durch Kommata voneinander getrennt. Es darf im verpflichtenden Argument *Paket* des `\usepackage`-Befehls aber kein Paket stehen, dem nicht sämtliche Optionen des optionalen Arguments *Option* bekannt sind. So ist der Befehl

```
\usepackage{a4, german}
```

erlaubt, der genau so als

```
\usepackage{a4}
\usepackage{german}
```

geschrieben werden kann. Nicht erlaubt hingegen ist

```
\usepackage[widemargins]{a4, german}
```

da das Paket `german` keine Option `widemargins` besitzt. Dies *muss* als

```
\usepackage[widemargins]{a4}
\usepackage{german}
```

geschrieben werden.

Außerdem werden alle Optionen des `\documentclass`-Befehls ebenfalls an alle geladenen Pakete weitergegeben. Hier werden unbekannte Optionen von den Paketen aber einfach ignoriert.

Kapitel 12

Informationsquellen

12.1 Gedrucktes

Über LaTeX gibt es viel zu sagen. Und viel zu schreiben. Das haben sich einige Menschen gedacht und deshalb gleich damit begonnen, ihr Wissen in Bücher zu packen. Um beim ersten Kontakt ein bißchen zu helfen, sind hier drei Bücher erwähnt, die man kennen sollte und die einem weiterhelfen können.

- Einführende Literatur
  - Leslie Lamport: *LaTeX: A document preparation system.* Addison-Wesley
    Hierbei handelt es sich um *das* Buch zu LaTeX, geschrieben von Lamport himself und deshalb natürlich in englisch. Wer sich auch mit einer Übersetzung zufrieden gibt, findet die allerdings auch.
  - Helmut Kopka: *LaTeX, Bd.1, Einführung* Addison-Wesley, München
    Das deutsche Gegenstück zu dem Lamport-Buch. Es gibt noch weitere Teile.
  - ... in jeder Bibliothek gibt es weitere Bücher. Am besten, man blättert sie selbst mal durch, leiht sie aus und schaut, wie man damit zurecht kommt.
- Weiterführende Literatur
  - Michel Goossens: *Der LaTeX-Begleiter* Addison-Wesley, München
  - ... wieder in jeder Bibliothek

12.2 Infos am Rechner und übers Internet

Da wir an einem Rechner arbeiten, ist es sinnvoll, sich gleich hier nach Hilfestellungen umzusehen.

Je nach LaTeX-Distribution gibt es mehr oder weniger viel Dokumentation zu LaTeX selbst, sowie zu den installierten Paketen.

Die beste Unterstützung findet man aber sicherlich im Internet. Die *Deutschsprachige Anwendervereinigung TeX e.V*, kurz *Dante* bietet auf ihrer Homepage `http://www.dante.de` so gut wie alles an, was das LaTeX-Herz begehrt.

Da sind zum einen die FAQ (*frequently asked questions*), die einem bei Problemen weiterhelfen können. Man sollte sie sich immer zuerst durchlesen, bevor man sich an „lebende " Menschen im Internet oder in Newsgroups wendet, damit man von dort keine Antworten im Stil von „Read the FAQ " oder ein unfreundliches „RTFM " bekommt.

Wenn man aber dort keine Hilfe gefunden hat, kann man sich natürlich an eine Newsgroup wenden und dort sein Problem formulieren (z.B. `de.comp.text.tex`).

Und dann gibt es unter dem Begriff CTAN (*Comprehensive TeX Archive Network*) ein großes Archiv von Packages, die einem das Leben leichter machen. Wie man diese verwendet, steht in Kapitel 11.

Unter `www.deja.com/usenet` findet man ein Archiv des `usenet`, wo man in früheren eMails nach Beiträgen zum eigenen Programm suchen kann und meist auch fündig wird.

12.3 Woher bekomme ich mein LaTeX denn?

Einfach ist es für alle Benutzer einer Linux-Distribution, denn LaTeX befindet ist dort normalerweise mit dabei.

Ansonsten findet man Bezugsquellen für TeX-Distributionen im WWW unter `http://www.dante.de/faq/de-tex-faq/html/woher.html` Für Windows gibt es z.B. *MiKTeX*. Es ist nicht sehr schwer, sich damit LaTeX auf den heimischen PC zu holen. Nur sollte man hier wirklich die `Readme.txt` lesen. Dort steht nämlich drin, wie man LaTeX in den Systempfad einträgt. Dannach funktioniert LaTeX wie gewoht. Das DVI-Betrachtungsprogramm hier heißt `yap` und kann auch direkt drucken.

12.4 Die Zeit nach dem LaTeX-Kurs

An den zwei Tagen, an dem der Kurs stattfindet, kann man die Leute fragen, die da sind und so aussehen, als wüßten sie etwas. In so gut wie allen Fällen werden sie weiterhelfen können.

Für die Zeit nach dem Kurs wurde eine Mailing-Liste eingerichtet, wo Du Dich mit anderen austauschen kannst. Die Addresse ist `latex-fragen@fs.tum.de`. Draufschreiben lassen kannst du dich mit einer eMail an `majordomo@fs.tum.de`, Subject ist egal, im Inhalt steht „`subscribe` *`deine@email`* `latex-fragen`"

Ansonsten gilt, wie immer bei Rechner-Problemen:

Ask your local guru!

Kapitel 13

Ergänzungen

13.1 Querverweise

Querverweise sind in LaTeX recht einfach möglich. Man muß nur die entsprechende Stelle mit `\label`{*Markierung*} markieren. Man kann dann mit `\pageref`{*Markierung*} die Seitenzahl der Seite auf welcher sich die Markierung befindet erhalten. `\ref`{*Markierung*} liefert einem je nach dem ob sich `\label`{*Markierung*} in einer Umgebung mit eigener Nummerierung wie z.B. `enumerate` oder `equation` (abgesetzte Formel, Beispiel auf Seite 36) befindet die aktuelle Nummerierung (z.B. die Formelnummer) an der Stelle wo `\label`{*Markierung*} steht bzw. sonst die entsprechende Stelle in der Dokumentuntergliederung zurück (wir sind hier z.B. in Abschnitt 13.1).

```
Wir sind hier\label{hier}
auf Seite~\pageref{hier}.
```

Wir sind hier auf Seite 81.

Achtung: Wenn man eine der folgenden beiden Fehlermeldungen bekommt:

```
LaTeX Warning: There were undefined references.
```

oder

```
LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
```

muß man noch einen LaTeX-Lauf machen damit alle Querverweise richtig sind. Wenn die erste Fehlermeldung danach immer noch erscheint gibt es bei mindestens einer Referenz entweder das entsprechende `\label` nicht oder man hat sich dabei vertippt.

13.2 Titelseite

Man kann entweder `\maketitle` verwenden und dafür die in Tabelle 13.1 angegebenen Deklarationen machen, was dann z.B. so aussieht:

`\title{`*text*`}`	Titel des Dokuments
`\author{`*namen*`}`	durch `\and` getrennte Namen der Autoren
`\thanks{`*text*`}`	fügt dem Titel eine Fußnote hinzu
`\date{`*datum*`}` (optional)	setzt das Datum; ansonsten wird das aktuelle Datum verwendet

Tabelle 13.1: Deklarationen für `\maketitle`

	itemize	enumerate	Zähler für `\enumerate`
1.Stufe	`\labelitemi`	`\labelenumi`	`enumi`
2.Stufe	`\labelitemii`	`\labelenumii`	`enumii`
3.Stufe	`\labelitemiii`	`\labelenumiii`	`enumiii`
4.Stufe	`\labelitemiv`	`\labelenumiv`	`enumiv`

Tabelle 13.2: Befehle für die Markierung von `\item` und Zähler für `\enumerate`

```
\title{Unser Dokument}
\author{Adrian Bunk\\
        Informatik\thanks{bunk@fs.tum.de} \and
        Michael Rauch\\
        Physik\thanks{mrauch@fs.tum.de}}
\date{27. Mai 2000}
\maketitle
```

oder die `titlepage`-Umgebung verwenden, innerhalb der man die Titelseite frei gestalten kann. Z.B.:

```
\begin{titlepage}
\LARGE
\textbf{Dies ist}\\[5cm]
\textsc{meine Titelseite}
\end{titlepage}
```

(OK, ich gebe zu, daß dieses Beispiel nicht besonders gut aussieht. . .)

13.3 Änderung der Markierung von `\item`

Für die Markierung von `\item` wird jeweils der entsprechende in Tabelle 13.2 aufgeführte Befehl verwendet. Durch Umdefinieren des entsprechenden Befehls kann man die Markierung ändern:

```
\renewcommand{\labelitemi}{+-=}
\begin{itemize}
        \item ein Punkt
        \item noch ein Punkt
\end{itemize}
```

+-= ein Punkt

+-= noch ein Punkt

`\arabic`	arabische Ziffern
`\roman`	kleine römische Ziffern
`\Roman`	große römische Ziffern
`\alph`	Kleinbuchstaben
`\Alph`	Großbuchstaben

Tabelle 13.3: Numerierungsstile für `\enumerate`

Bei einer numerierten Aufzählung kann man auf die in Tabelle 13.2 aufgeführten Zähler für `\enumerate` zurückgreifen, und diese mittels der in Tabelle 13.3 aufgeführten Numerierungsstile anzeigen. Ein Beispiel dafür wäre

```
\renewcommand{\labelenumi}{\arabic{enumi}}
\renewcommand{\labelenumii}{\arabic{enumi}.\arabic{enumii}}
\renewcommand{\labelenumiii}{\arabic{enumi}.\arabic{enumii}.\arabic{enumiii}}
```

```
\begin{enumerate}
  \item dies ist
  \item eine Aufz"ahlung;
  \begin{enumerate}
    \item und das
    \begin{enumerate}
      \item sind auch
    \end{enumerate}
    \item Aufz"ahlungen
  \end{enumerate}
\end{enumerate}
```

1 dies ist

2 eine Aufzählung;

  2.1 und das

    2.1.1 sind auch

  2.2 Aufzählungen

13.4 Schriftgröße in Formeln

Analog zu den auf Seite 29 vorgestellten Befehlen kann man auch in Formeln die Schriftgröße ändern:

`\scriptstyle`	$\scriptstyle klein$
`\scriptscriptstyle`	$\scriptscriptstyle sehr\ klein$

13.5 Fußnoten

Mit `\footnote`{*Fußnotentext*} kann man eine Fußnote erzeugen. Für genauere Einzelheiten und insbesondere auch wie man Fußnoten in Umgebungen in denen es mit diesem Befehl nicht geht (z.B. in Tabellen) erzeugt findet man in [Lam95] ab Seite 199.

```
Dieser Text hat eine
Fu"snote.\footnote{Dies
ist eine Fu"snote.}
```

Dieser Text hat eine Fußnote.[a]

[a]Dies ist eine Fußnote.

13.6 Randbemerkungen

Randbemerkungen kann man mit `\marginpar`{*Randbemerkung*} erzeugen.
Text am Rand
Die neben diesem Absatz stehende Randbemerkung wurde z.B. mit `\marginpar{Text am Rand}` erzeugt. Randbemerkungen werden bei einseitigem einspaltigem Druck an den rechten Rand und bei zweiseitigem einspaltigem Druck an den äußeren Rand gesetzt. Im letzteren Fall kann man durch `\marginpar`[*linke Randbemerkung*]{*rechte Randbemerkung*} je nachdem ob die Randbemerkung auf einer linken oder einer rechten Seite auftritt verschiedene Texte vorgeben.
$\Rightarrow$
Einen auf den Text zeigenden Pfeil wie man ihn hier sieht erzeugt man z.B. mit `\marginpar[$\Rightarrow$]{$\Leftarrow$}`.

13.7 Noch ein bißchen was zu Fehlermeldungen

Zwei häufigere Fehlermeldungen sind noch:

```
! LaTeX Error: There's no line here to end.
```

In diesem Fall hat man ein `\newline` oder `\\` am Anfang einer Zeile geschrieben was LaTeX nicht mag. Als Workaround kann man davor ein `\mbox{}` setzen.

```
*
```

Wenn man anstatt eine genauere Fehlermeldung zu bekommen nur ein Sternchen bekommt und LaTeX auf eine Eingabe wartet hat man das `\end{document}` vergessen.

Eine Übersicht über viele häufige Fehlermeldungen findet man in [Lam95].

Anhang A

Danksagung

Unser Dank gilt ...

- den Fakultäten für Mathematik, Physik und Informatik und ihren Studiendekanen, ohne deren finanzielle Unterstützung dieser Kurs nie hätte stattfinden können.
- der Rechnerbetriebsgruppe der Informatik, die uns die Rechnerhalle zur Verfügung stellt.
- Donald Knuth und Leslie Lamport für ihre tolle Software
- allen Tutoren, die uns beim Betreuen geholfen haben.
- Und natürlich im Besonderen: Heike, Harald, Carmen, Marcel, Simon, Dagmar, Wilma, Simone ... und allen, die wir vergessen haben.

Anhang B

Verwendete Tasten

In diesem Begleitheft wurden fast alle Tasten auf unseren Tastaturen verwendet. Zwar sind alle Tasten gleich, aber manche sind eben doch gleicher und so gilt unser ganz spezieller Dank den folgenden Tasten.

(sp)[1]	22849	p	2116	M	342	2	164	ˆ	61
e	21225	f	2089	X	328	0	161	˜	54
n	12186	w	1512	F	326	’	158	/	53
i	10602	x	1471	ß	302	G	144	Y	48
t	10282	k	1412	E	301	H	141	>	44
a	7775	.	1371	”	296	N	137	!	42
r	7637	(ht)[2]	1182	ä	288	=	136	?	37
s	6603	,	1156	P	274	V	130	9	35
d	6111	z	1130	[	241	‘	126	6	30
l	5308	v	900	]	240	%	113	7	27
m	5026	&	748	1	239	R	109	8	26
\	4986	T	720	U	228	j	97	#	25
o	4727	D	706	Z	224	C	89	<	23
(nl)[3]	4710	$	647	)	216	_	83	J	20
c	4410	y	557	I	213	*	81	@	20
h	4342	B	532	ö	211	O	79	;	20
u	4163	A	498	(	210	4	73	Ü	15
g	3391	S	468	W	203	3	69	Q	14
}	3280	ü	456	K	200	|	66	Ä	8
{	3280	L	456	q	198	5	64		
b	2929	-	445	:	191	+	64		

[1]space
[2]horizontal tabulator
[3]newline

Anhang C

Kurzreferenz vi

Der vi ist auf UNIX-Systemen der am weitesten verbreitete Editor. Auf der Beliebtheitsskala teilt er sich den ersten Platz mit dem Emacs, der allerdings deutlich größer und schwerfälliger ist `;)` Die Bedienung erscheint erst recht kryptisch, allerdings läßt sich nach ein wenig Einarbeitung sehr schnell und effizient arbeiten.

Inzwischen existieren mehrere vi-Klone. Der bekannteste und wohl auch beste darunter ist der `vim` (VI iMproved).

Die wichtigste Eigenart, die man über den vi wissen muß, sind seine Modi. Text eingeben geschieht im Eingabemodus, die meisten Befehle hingegen werden im Kommandomodus ausgeführt.

Vielen der Befehle kann man eine Zahl vorstellen, um den Befehl damit zu vervielfachen (3x löscht zum Beispiel drei Buchstaben)

Dateikommandos

Datei öffnen	`:e` *`datei`*
Datei speichern	`:w`
vi beenden	`:q`
Hilfe	`:help`

Modi

Eingabemodus	i
Kommandomodus	ESC
Ex-Modus	:

Suchen

vorwärts suchen	/
rückwärts suchen	?

Bewegung

links	←, h
runter	↓, j
hoch	↑, k
rechts	→, l
ein Wort vor	w
ein wort zurück	b

Cut, Copy, Paste

Zeile kopieren	yy
Zeile löschen	dd
Buchstabe löschen	x
Wort löschen	dw
einfügen	p

Diese Auswahl ist relativ willkürlich und erfaßt nur den Bruchteil des vi, der zum Arbeiten absolut nötig ist. Sehr zu empfehlen ist hier [LR99]. Da die Einstellung des `vim` in der Rechnerhalle der Informatik unbrauchbar sind, könnt ihr euch die Datei `~keil/.vimrc` in euer home kopieren.

Anhang D

Kurzreferenz Emacs

D.1 Warum EMACS

Emacs ist auf vielen Unix-Systemen vorhanden, und vor allem für den weniger geübten Benutzer zu empfehlen, da die wichtigsten Kommandos intuitiver zu erlernen sind als beim vi[1].

D.2 Die wichtigsten Tasten

In der folgenden Liste heißt `C-` `Ctrl` (`Strg` auf deutschen Tastaturen) und `M-` `Meta` (♦ auf Sun-Tastaturen, `Alt` auf manchen Linux–Rechnern). Statt `Meta` kann man auch `ESC` drücken, muß das dann allerdings loslassen.

Dateikommandos	
Datei laden	`C-x C-f`
Datei speichern	`C-x C-s`
Emacs beenden	`C-x C-c`
Suchen	
Vorwärts suchen	`C-s`
Rückwärts suchen	`C-r`
Suchen und Ersetzen	`M-%`
Sonstiges	
Abbruch	`C-g`

Bewegen	
Zeichen links	`C-b, links`
Zeichen rechts	`C-f, rechts`
Wort links	`M-b, M-links`
Wort rechts	`M-f, M-rechts`
Zeile hoch	`C-p, hoch`
Zeile runter	`C-n, runter`
Zeilenanfang	`C-a`
Zeilenende	`C-e`
Dataianfang	`Pos1`
Dateiende	`End`

Der Emacs bietet standardmäßig leider eine Unterstützung für amerikanische Anführungzeichen beim Drücken der ¨-Taste(“ und ”). Abhilfe hierbei schafft es, in der Sunhalle meine `.emacs` Datei (`cp ~berger/.emacs ~/`)und den `emacs20` zu benutzen.

[1] Viele Menschen führen Kleinkrieg zwischen vi und Emacs. Emacs ist einfach besser `;)`!

Anhang E

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document „free“ in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of „copyleft“, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. Applicability and definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **„Document“**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **„you“**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **„Modified Version“** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **„Secondary Section“** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **„Invariant Sections“** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **„Cover Texts“** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **„Transparent“** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not „Transparent“ is called **„Opaque“**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **„Title Page“** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, „Title Page“ means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **„Entitled XYZ“** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **„Acknowledgements“**, **„Dedications“**, **„Endorsements“**, or **„History“**.) To **„Preserve the Title“** of such a section when you modify the Document means that it remains a section „Entitled XYZ'äccording to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. Verbatim copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. Copying in quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled „History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled „History"ïn the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the „History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled „Acknowledgements" or „Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled „Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled „Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled „Endorsements“, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. Combining documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled „History“ in the various original documents, forming one section Entitled „History“; likewise combine any sections Entitled „Acknowledgements“, and any sections Entitled „Dedications“. You must delete all sections Entitled „Endorsements“.

6. Collections of documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. Aggregation with independent works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an „aggregate“ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled „Acknowledgements“, „Dedications“, or „History“, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. Future revisions of this license

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License „or any later version“ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

Addendum: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled „GNU Free Documentation License“.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the „with...Texts.“ line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Literaturverzeichnis

[GMS94] Michael Goosens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, 1994.

[GMS95] Michael Goosens, Frank Mittelbach, and Alexander Samarin. *Der LaTeX Begleiter*. Addison-Wesley, 1995. Deutsche Übersetzung von [GMS94].

[GR+99] Michael Goosens, Sebastian Rahtz, et al. *The LaTeX Web Companion*. Addison-Wesley, 1999.

[GRM97] Michael Goosens, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Addison-Wesley, 1997.

[Kop96] Helmut Kopka. *LaTeX: Band 1 – Einführung*. Addison-Wesley, second edition, 1996.

[Kop97a] Helmut Kopka. *LaTeX: Band 2 – Ergänzungen*. Addison-Wesley, second edition, 1997.

[Kop97b] Helmut Kopka. *LaTeX: Band 3 – Erweiterungen*. Addison-Wesley, 1997.

[Lam94] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, second edition, 1994.

[Lam95] Leslie Lamport. *Das LaTeX Handbuch*. Addison-Wesley, second edition, 1995. Deutsche Übersetzung von [Lam94].

[LR99] Linda Lamb and Arnold Robbins. *Textverarbeitung mit dem vi-Editor*. O'Reilly, 1999. Originaltitel: Learning the vi-Editor.

[O+99] Tobias Oetiker et al. *The Not So Short Introduction to LaTeX 2_ε*, 1999. CTAN: `/tex-archive/info/lshort/english/`.

[SKPH99] Walter Schmidt, Jörg Knappen, Hubert Partl, and Irene Hyna. *LaTeX 2_ε–Kurzbeschreibung*, 1999. CTAN: `/tex-archive/info/lshort/german/`.

[SP97] Stefan Schwarz and Rudolf Potuček. *Das TeXikon: Referenzhandbuch für TeX und LaTeX*. Addison-Wesley, 1997.

Index