# Wichtige LaTeX Befehle

Christian Feuersänger

19. Oktober 2010

**Zusammenfassung**

Dieses Dokument entstand über mehrere Jahre, in denen ich das Resultat von zum Teil stundenlangem Suchen nach LaTeX-Einstellung gesammelt habe. Es enthält – neben mancherlei wohlbekannten Formatierungselementen – auch immer wieder speziellere Hinweise oder Literaturangaben, wie man über das 0815 Layout hinaus arbeiten kann.

## Inhaltsverzeichnis

# 1 Paketdokumentationen

Für (fast) alle vorinstallierten LaTeX-Pakete gibt es auch vorinstallierte Manuals. Mit

`texdoc` ⟨*paketname*⟩

wird nach verfügbarer Dokumentation gesucht und ggf. ein Anzeigeprogramm gestartet.

Beispiel: `texdoc pgfplots` started bei mir `kpdf` mit `pgfplots.pdf`.

Ansonsten findet man unter

`http://tug.ctan.org/search.html`

auch sehr gut Dokumentation. Dabei sollte man „Search the package descriptions“ verwenden.

# 2 Grundlagen

- `"‘` (shift+neben Backspace) und `"’` (shift+neben return)
  „ und “
- ‘ und ’
  Englische einfache Anführungszeichen: ‘ und ’
- ‘‘ und ’’
  Englische doppelte Anführungszeichen: “ und ”
- `\"a`, `\"o` (Korrekte Syntax, auch ohne Paket german)
  ä, ö, (für ‘ß’ weiss ich es nicht)...
- `\usepackage{german}`
  erlaubt `"a, "s, "o` für ä, ß, ö und macht einige weitere Anpassungen.
- `\usepackage[latin1]{inputenc}` oder meistens viel besser
  `\usepackage[utf8]{inputenc}`
  erlaubt direkte Eingabe der Zeichen `ä, ß, ö` = ä, ß, ö. Damit sagt man LaTeX, in welchem encoding die Datei gespeichert worden ist, was natürlich vom Rechner und Editor abhängt. `latin1` ist Ascii und wird (noch) standardmäßig bei uns verwendet. Moderner und besser ist `utf8` (Unicode). Bei dem Editor vim kann man eine Datei mit unicode schreiben durch eingabe von `set fileencoding=utf8` (die wird dann auch immer korrekt gelesen).
- `\usepackage{eurosym}`
  stellt € ( `\euro{}` ) zur Verfügung

# 3 Gliederung

- `\section{...}  \subsection{...}`
  Alles einfach ausprobieren ...
- `\subsubsection{...}`
- `\paragraph{...}  \subparagraph{...}`
- `\appendix`
  Schaltet auf Seitennumerierung mit Buchstaben um
- `\part{...}  \chapter{...}`
  Bei report oder book. part beeinflusst dabei nicht die Numerierung der Kapitel.

Siehe auch Abschnitt 13.4.1 für Feinheiten bei der Seitennumerierung dabei.

# 4 einfache Strukturelemente

## 4.1 itemize

z.B.:

```
\begin{itemize}
  \item erstes
  \item zweites
    \begin{itemize}
      \item zweites erstes
          \begin{itemize}
             \item und so weiter
          \end{itemize}
      \item zweites zweites
    \end{itemize}
\end{itemize}
```

Ergibt eine einfache Aufzählung der Art:

- erstes
- zweites
  - zweites erstes
    - und so weiter
  - zweites zweites

Siehe [7] für eine Anwendung von `\begin{list}` für benutzerdefinierte Listen.

## 4.2 enumerate

Funktioniert exaxt genauso wie itemize, nur mit Nummern.

- Gut zu wissen: mit `\setcounter{enumi}{3}` kann man die Aufzählung bei 3 beginnen lassen.
- Sicher auch nicht uninteressant: das unterbrechen einer Aufzählung kann man – ein wenig technisch – auch machen:

```
\begin{enumerate}
\item Eins.
\item Zwei.
\xdef\letzterwert{\the\value{enumi}}
\end{enumerate}
Unterbrechung.
\begin{enumerate}
\setcounter{enumi}{\letzterwert}
\item Drei
\item Vier
\end{enumerate}
```

Das führt zu:

1. Eins.
2. Zwei.

Unterbrechung.

3. Drei
4. Vier

Also: mit `\xdef\letzterwert{\the\value{enumi}}` wird der aktuelle Wert des counters `enumi` in ein Makro namens `\letzterwert` expandiert geschrieben. Mit `\setcounter` wird der gespeicherte wieder gesetzt[1].

## 4.3 description

Beispiel:

```
\begin{description}
  \item[erstes] Eine Beschreibung zu erstes
  \item[zweites] Eine Beschreibung zu zweites
\end{description}
```

Ergibt

**erstes** Eine Beschreibung zu erstes

**zweites** Eine Beschreibung zu zweites

[1] Das `\xdef` ist notwendig; es heisst „definiere global und expandiere das Resultat“. Siehe [2].

# 5 Tabellen

```
\begin{tabular}{SPALTENDEFINITION}
ZEILEN \\
ZEILEN \\
\end{tabular}
```

| Anweisung | Beschreibung | weiteres |
|---|---|---|
| r | Spalte wird nach rechts ausgerichtet | |
| l | Spalte wird nach links ausgerichtet | |
| c | Spalte wird nach zentriert | |
| `p{6cm}` | Spalte mit fixer Breite; Zeilenumbruch wird gemacht | |
| `\|` | vertikaler Balken trennt Spalten | Bsp: `{l\|l\|l}` |
| & | trennt einzelne Spalten | |
| `@{}` | ersetzt Abstand zwischen Spalten durch Nutzerbefehl | Bsp.: `{ll@{a}ll@{}}` |
| `*{7}{c}` | macht 7 zentrierte (c) Spalten | Bsp.: `{l*{2}{l}r}` |
| \\ | beendet eine Tabellenspalte | |
| `\hline` | macht einen horizontalen Strich | |
| `\hline\hline` | macht einen doppelten horizontalen Strich | |
| `m{Breite}` | wie p, nur Felder sind vertikal zentriert | (braucht `\usepackage{array}`) |
| `b{Breite}` | wie m, nur Ausrichtung an Fußzeile | (braucht `\usepackage{array}`) |
| `\|` | vertikaler Strich wie vorher plus Breite des Striches | (braucht `\usepackage{array}`) |
| `!{Sep}` | wie `\|`, aber mit Sep als Separator | (braucht `\usepackage{array}`) |
| `>{Dekl}` | kann vor l, r, c, b, m, p stehen, fügt Dekl vor jedes Feld der Spalte ein | (braucht `\usepackage{array}`) |
| `<{Dekl}` | kann nach l, r, c, b, m, p stehen, fügt Dekl nach jedem Feld der Spalte ein | (braucht `\usepackage{array}`) |

Beispiel:

```
\begin{tabular}{lp{6cm}l}
Anweisung & Beschreibung & weiteres\\
\hline
r   & Spalte wird nach rechts ausgerichtet & \\
l   & Spalte wird nach links ausgerichtet & \\
c   & Spalte wird nach zentriert & \\
\end{tabular}
```

Weitere Anmerkungen:

- `\multicolumn{3}{c}{Text}`
  macht eine zentrierte Zeile, die über 3 Spalten geht
- Mit '`p`', '`m`' und '`b`' kann man *vertikales* Alignment machen. Dann ist standardmäßig aber die horizontale Ausrichtung auf *linksbündig*. Will man vertikal *und* horizontal ausrichten, braucht man die '`>`' und '`<`' optionen. Beispiel:

  ```
  \newcolumntype{C}{>{\centering\arraybackslash}p{5cm}}
  ```

```
\begin{tabular}{C}
    Daten
\end{tabular}
```

Wozu dieses '`\arraybackslash`' ist, verstehe ich leider nicht. Jedenfalls geht es nicht ohne.

- `\renewcommand{\arraystretch}{1.5}` Erlaubt eine Vergrößerung der Höhe.
- Man kann mit dem Paket colortbl Spalten, Zeilen oder auch einzelne Zellen einfärben (siehe [12]). Achtung: Es sieht oft so aus, als würden durch die Einfärbung vertikale oder horizontale Linien überschrieben. Scheinbar ist das ein Viewer-Problem; beim Druck sind die vorhanden.

Noch ein paar Sachen dazu:

- `http://www.uni-bielefeld.de/lili/personen/luecking/ltx-spez.pdf`
- `ftp://ftp.dante.de/pub/tex/info/german/tabsatz/tabsatz.pdf`

# 6 Querverweise

Für alle Querverweise muss man mehrfach den Übersetzungsbefehl ausführen (mehrfach „latex“ aufrufen).

## 6.1 label, ref, pageref

- `\label{marker}, \ref{marker}, \pageref{marker}`
  Mit label setzt man einen Marker an eine Stelle, wenn man dann woanders `\ref{den_richtigen_Namen}` macht kommt da ein ordentlicher Verweis hin. Bei Formeln z.B. die Formelnummer, ansonsten die Seitennummer oder Kapitel oder so.
- `\cite[text]{key_list}`
  Zum Zitieren von Quellen. In den Text wird ein Verweis eingefügt, dessen Format etwas vom verwendeten Paket oder Stil abhängt. Normalerweise sieht der so aus: „`[15]`“, vllt. noch mit dem optionalen text aus den eckigen Klammern. Ganz unten am Dokument stehen alle Quellen in einer Liste, das wird auch direkt das Literaturverzeichnis. Darauf verweist der `{key_list}`.
- `\usepackage{showkeys}` Ein draft / debug Modus, der im gesamten Dokument alle internen labels anzeigt.

## 6.2 cite, bibtex

- `\nocite{key_list}`
  Genauso wie `\cite`, allerdings wird in den Text nichts eingefügt – das ist nur für bibtex interessant, sodass die keys in der Literaturliste auftauchen.

- bibtex
  Oft verwendet man bibtex – das ist sogar anzuraten, denn der kümmert sich um ein schönes Aussehen der Literaturliste. bibtex ist ein eigenständiges Programm, was nach latex aufgerufen wird. Dazu:
  - `\usepackage{bibgerm}`
    In der Präambel, setzt den Stil
  - `\bibliographystyle{gerplain}`
    Vor `begin{document}`; setzt den Stil fest.
  - `\bibliography{literatur.bib}`
    Anstelle dieses Befehls werden ein- oder mehrere Seiten mit Literaturangaben generiert. Die Datei „literatur.bib“ enthält dabei alle bekannten. Es werden nur die Einträge in dem Dokument angegeben, die auch zitiert wurden (entweder mit `\cite` oder mit `\nocite`).
- `\begin{thebibliography}`
  Eine alternative zu bibtex, die bibliography anzugeben. Bsp:

  ```
  \begin{thebibliography}{XXXX}
  \bibitem{amsmanual} AMS Anleitung.
  \end{thebibliography}
  ```

  Das fügt den angegebene Eintrag in die Liste ein – und zwar so, wie er hier hingeschrieben wird – und macht ihn unter dem key „amsmanual“ im Dokument verfügbar.

## 6.3 Anklickbare Querverweise

Mit `\usepackage{hyperref}` kann man dafür sorgen, dass sämtliche Links im PDF-Dokument anklickbar werden. Mit

```
\usepackage{hyperref}
\hypersetup{pdfborder={0 0 0}}
```

werden die Links nicht visuell markiert.

## 6.4 Stichwortindex

Mit

```
\usepackage{makeidx}
\makeindex

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

kann man folgende Indexeinträge erzeugen:

| | | |
|---|---|---|
| `\index{hello}` | hello, 1 | Plain entry |
| `\index{hello!Peter}` | Peter, 3 | Subentry under „hello“ |
| `\index{Sam@\textsl{Sam}}` | *Sam*, 2 | Formatted entry |
| `\index{Lin@\textbf{Lin}}` | **Lin**, 7 | Same as above |
| `\index{Jenny\|textbf}` | Jenny, **3** | Formatted page number |
| `\index{Joe\|textit}` | Joe, *5* | Same as above |
| `\index{ecole@\'ecole}` | école, 4 | Handling of accents |
| `\index{Eintrag\|(}` | | Eintrag von Seite |
| `\index{Eintrag\|)}` | | Eintrag bis Seite |

die Einträge werden gesondert in externe Dateien gesammelt. Schliesslich muss man noch das externe Programm `makeindex` ⟨*filename*⟩ aufrufen, das aktualisiert den Index.

### 6.4.1 Feinkonfiguration des Index

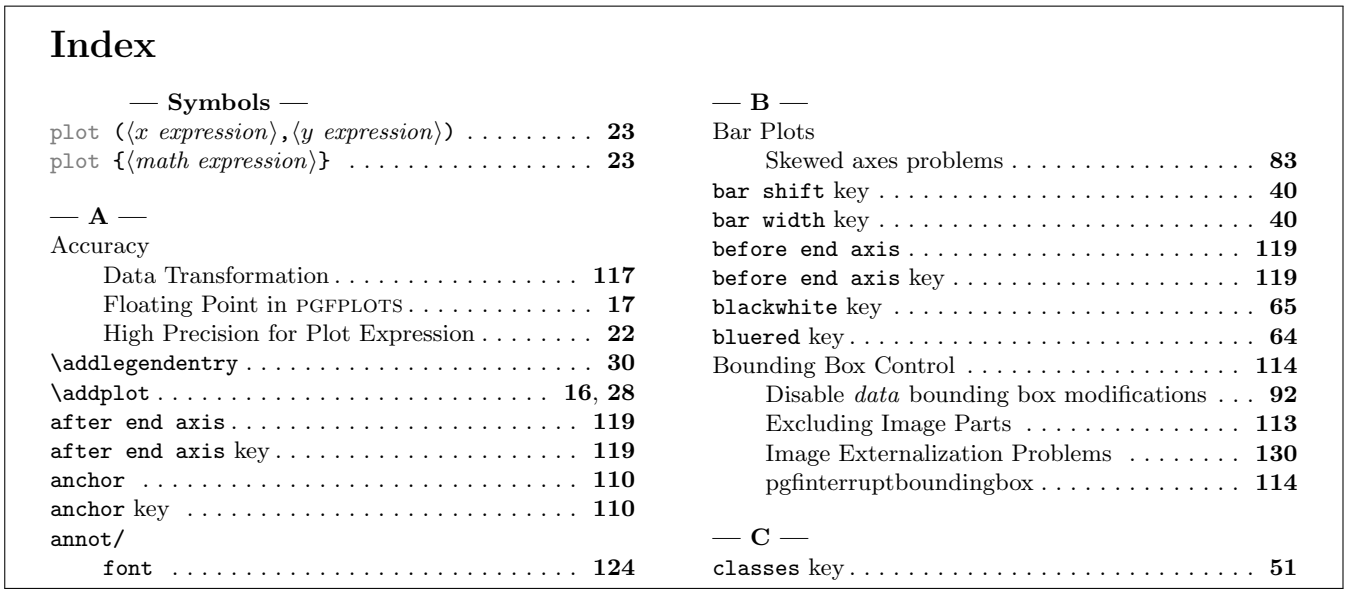
Wenn man eine Datei namens ⟨*filename*⟩`.mst` anlegt, wird die automatisch von `makeindex` als Konfigurationsdatei eingelesen - aber nur, wenn man auch ⟨*filename*⟩ bearbeitet. Ich fand z.B.

```
headings_flag 1
heading_prefix "{\\bf ---~"
heading_suffix "~---}"
delim_0       "\\dotfill\\indexpageno{"
delim_1       "\\dotfill\\indexpageno{"
delim_2       "\\dotfill\\indexpageno{"
delim_t       "}"
delim_n       "}, \\indexpageno{"
delim_r       "} -- \\indexpageno{"
suffix_2p     "}f.{"
suffix_3p     "}ff.{"
symhead_negative "Symbols"
symhead_positive "Symbols"
numhead_negative "Numbers"
numhead_positive "Numbers"
quote         '+'
line_max      255
```

hübscher als die Standardeinstellung. Das produziert sowas wie in Abbildung 1.

# 7 Fussnoten

- `\footnote{text}`
  Fuegt mitten im Text die Zahl der Fussnote ein, die eigentliche Fussnote kommt dann nach unten. Beachte: Mehrzeilige Fussnoten sehen manchmal was komisch aus. Man kann aber einiges Interessantes im Paket `footmisc` einstellen.



# Index



Abbildung 1: Beispiel für Indexkonfiguration (entnommen aus meinem `pgfplots` Paket). Der anfängliche Zeileneinschub ist ein Fehler irgendwo in der Implementierung (glaube ich).

# 8 Mathematik

Auf jeden Fall AMS Manual lesen! Viele der folgenden Sachen gehen nur mit den Paketen
`\usepackage{amsmath, amsthm, amssymb}`

- Im fliessenden Text `$1+1=1$` gibt: $1+1=1$
  Das macht man auch, wenn man nur Variablen erwaehnt - dann haben alle Mathesachen dieselbe Schrift...
- Abgesetzte Formel: `\[ 1+1=1 \]`
  ohne Numerierung
- oder `\begin{equation} 1+1=1 \end{equation}`
  mit Numerierung. Beachte: die Numerierung ist standartmäßig mit ganzen Zahlen `(1), (2), ...`. Das ist nicht unbedingt zweckmäßig. Man kann das umstellen mit `\numberwithin{equation}{section}` in der Preambel. Das geht auch prinzipiell mit anderen Sachen die numeriert werden. Man kann natürlich auch z.B. subsection angeben.
- Mit `\begin{equation*} \tag{} \end{equation*}`
  wird die Autonumerierung ausgeschaltet und ersetzt durch eine manuelle, die mit `\tag{..}` auch wieder abgerufen werden kann.
- `\text{...}`
  Damit kann man im Mathemodus ganz normalen Text einfuegen - das geht sonst nicht. Es gibt oft `underful vboxes` oder so. Wenn die durch viele

mehrzeilige Formeln verursacht wurden kann man den Befehl `\allowdisplaybreaks` lokal oder global anwenden (amsmath).

- `\R, \N, \Z, \Q` als $\mathbb{R}, \mathbb{N}, \mathbb{Z}, \mathbb{Q}$
  Man verwende

  ```
  \newcommand{\R}{\mathbb{R}}
  \newcommand{\N}{\mathbb{N}}
  \newcommand{\Z}{\mathbb{Z}}
  \newcommand{\Q}{\mathbb{Q}}
  ```

  Das geht auch mit anderen Buchstaben.

- $\|x\|, |x|, \langle x, y\rangle$ mittels `\norm{x}, \abs{x}, \innerProd{x}{y}`
  Man verwende

  ```
  \providecommand{\norm}[1]{\lVert #1 \rVert}
  \providecommand{\abs}[1]{\lvert #1 \rvert}
  \providecommand{\innerProd}[2]{\langle #1, #2\rangle}
  ```

- Neue Mathe-Operatoren
  Man kann mit `\DeclareMathOperator{\grad}{grad}` neue Mathematik-Operatoren definieren. Das ergibt dann `$\grad u$` $= \operatorname{grad} u$ im Vergleich zu `$grad u$` $= gradu$.

## 8.1 Spezielle Mathesachen

- `\eqref{...}`
  Dereferenzierung von Gleichung

- Option `intlimits`
  setzt Global limits für Integral *unter* das Integralzeichen, nicht an die Seite

- `\substack{...}`
  mehrzeilige Grenzen, z.B. `\[\sum_{\substack{1\le i\le n\\ 1 \le j \le m}} a_{ij}\]` ergibt

$$\sum_{\substack{1\le i\le n\\ 1\le j\le m}} a_{ij}$$

- `\boldsymbol{...}`
  setzt das Argument im Mathemodus in fetter Schrift, z.B. `$\boldsymbol{\pi}$` liefert $\boldsymbol{\pi}$

- `\boldmath \[ ... \] \unboldmath` setzt soviel Elemente der folgenden Formel(n) in fetter Schrift wie geht. Vorsicht: muss ausserhalb der Formel stehen!

- Abstände im Mathemodus
  - Zusammenrücken: `\!`, `\negmedspace`, `\negthickspace`
  - Platz einfügen: `\,` `\:` `\;` oder `\quad` und `\qquad`

- `\overset{}{}` und `\underset{}{}` setzte Formelbestandteile über- oder untereinander, bspw. liefert `$\overset{O}{U} \quad \underset{O}{U}$`

$$\overset{O}{U} \quad \underset{O}{U}$$

- `\underbrace{a+b}_\text{Schritt 1}` und `\overbrace{e^2 - \pi}^\text{Schritt 2}`

$$\underbrace{a+b}_{\text{Schritt 1}} = \overbrace{e^2 - \pi}^{\text{Schritt 2}}$$

- `\newcommand{\LL}{\ensuremath{L_2}}` erlaubt sowas wie
  `Dies ist normaler Text mit \LL.`
  Dies ist normaler Text mit $L_2$.

# 9 Aussehen

## 9.1 Layout

- `\setlength{\parindent}{0cm}`
  Ändert den horizontalen Einschub, der am Anfang eines jeden neuen Absatzes eingefügt wird.
- `\setlength{\parskip}{0.2cm}`
  Ändert den vertikalen Abstand zwischen zwei Absätzen.

## 9.2 Farbe

- `\usepackage{color}`
  Das Paket für Farben
- `\textcolor{...}`
  Farben wie red blue black ...
- `\usepackage{colortbl}`
  Erlaubt das Einfärben von Tabellenspalten, -Zeilen und -Zellen. Siehe [12].

## 9.3 Textarten

- Stil:
  Es gibt zwei Arten, fett kursiv etc. einzustellen:

```
Dieser \textrm{Text}
ist ganz \textit{schön}
komisch ... \textbf{Und}
alle Vögel { \rmfamily
\itshape \bfseries
fliegen tief $:-)$ }
```

Dieser Text
ist ganz *schön*
komisch ... **Und**
alle Vögel ***fliegen tief*** : −)

Das eine geht mit den `\kram{...}` - Sachen, das andere mit Schaltern, die im aktuellen Bereich ( z.B. mit `{ ... }` abgegrenzt ) gelten. Es gibt noch (aus dem Manual kopiert):

- `\emph`
  Emphasis (toggles between `\textit` and `\textrm`)
- `\textmd (\mdseries)`
  Medium weight (default). The opposite of boldface.
- `\textup (\upshape)`
  Upright (default). The opposite of slanted.
- `\textsl (\slshape)`
  Slanted.
- `\textsf (\sffamily)`
  Sans serif.
- `\textsc (\scshape)`
  Small caps.
- `\texttt (\ttfamily)`
  Typewriter.
- `\textnormal (\normalfont)`
  Main document font.

### 9.3.1 Weitere Textarten

Mit

```
\usepackage[normalem]{ulem}
```

- `\uline{important}` Underlined text, breaks over multiple lines
- `\uuline{urgend}` double-underlined text urgend
- `\uwave{boat}` boat
- `\sout{wrong}` ~~wrong~~
- `\xout{removed}` ~~removed~~

### 9.4 minipage, scalebox

- `\minipage`
  Erlaubt die Erstellung von einer box, die genauso gehandhabt wird wie eine kleine Seite von vorgegebener Größe. Nutzung

  ```
  \begin{minipage}[position]{width}
  text
  \end{minipage}
  ```

  Nutzen: man kann z.B. mit `\scaleBox` ganze Seiten skalieren oder z.B. in tabular-umgebungen, die sonst etwas wählerisch sind, andere Umgebungen reinpacken.

- `\scalebox`
  Erlaubt Skalierung von beliebigen boxen. Bspw:

  ```
  \scalebox{0.4}{Text zu skalieren}
  ```

  Zuweilen muss man vllt. mit minipage arbeiten, um Zeilenumbrüche u.ä. hinzubekommen. Achtung: das Argument von `\scalebox` ist ein Skalierungs*faktor*, keine Längenangabe.

### 9.5 Schriftgroessen

lassen sich „verbal“ einstellen mit:

- `\tiny \scriptsize \footnotesize \small \normalsize (default)`
- `\large \Large \LARGE \huge \Huge`

## 10 Graphik

Dieser Abschnitt behandelt nur das Einbinden, nicht die `figure` Umgebungen. Siehe Abschnitt 12.

### 10.1 Externe Bilddateien

Man kann diverse Graphiken einbinden. Dazu lädt man mit

```
\usepackage{graphicx}
```

(man beachte das ‘x’) das zugehörige Paket und kann dann mit

```
\includegraphics[width=5cm]{my_image}
```

das gewünschte Bild einbinden. Die Endung kann angegeben werden, allerdings macht es mehr Sinn, dies offenzulassen. Man kann auch mit

```
\usepackage[draft]{graphicx}
```

dafür sorgen, dass zwar der finale Platz reserviert wird, aber LaTeX keine Bilder einbindet. Das geht einiges schneller.

Die einbindbaren Bildformate unterscheiden sich je nach verwendetem LaTeX-Übersetzer. Bei Verwendung von „pdflatex“ kann man JPG, PNG, PDF (und noch ein paar) einbinden. Bei Verwendung des standard „latex“ wird üblicherweise nur EPS (encapsulated postscript) verwendet. In jedem Fall sollte man Zeichnungen, Illustrationen und Skizzen mit einem Vektorformat machen, dass auch nach einer Skalierung immer noch gut aussieht. Dafür ist EPS oder PDF das geeignetste Format (es gibt auch einen `epstopdf`).

Man kann den Suchpfag für Bilder mittels

```
\graphicspath{{path_to_images/}}
\graphicspath{{the/first/path/}{the/second/path/}}
```

setzen, wobei die doppelten Klammern und der abschliessende Strägstrich obligatorisch sind.

### 10.1.1 Beschneiden

Man kann nur ein Teil eines Bildes einbinden, indem man sowas wie

```
\includegraphics[trim=0 0 2cm 0,clip,width=\textwidth]%
   {Kapitel08-img1}
```

verwendet. Das reduziert den Bildausschnitt um 2cm horizontal (Das Keyword `trim` reduziert die Größe um `links unten rechts oben`. Mit `clip` schliesslich wird alles ausserhalb des verbleibenden Rechtecks entfernt). Siehe das Manual von graphicxs oder auch [4].

## 10.2 Graphiken mit `PGF/tikz`

Man kann sehr viele graphische Sachen mit `\usepackage{tikz}` erreichen. Der Name heißt „TikZ ist *kein* Zeichenprogramm“. Es erlaubt die einfache Erstellung von professionellen Graphiken innerhalb von LaTeX, insbesondere auch Graphen und Bäume.

Generell ist das manual mit sehr vielen Beispielen versehen, es lohnt sich, da mal reinzuschauen.

### 10.2.1 Einbinden und Laden

Man lädt TikZ mit

```
\def\pgfsysdriver{pgfsys-dvipdfm.def}
%\def\pgfsysdriver{pgfsys-pdftex.def}
%\def\pgfsysdriver{pgfsys-dvipdfm.def}
%\def\pgfsysdriver{pgfsys-dvips.def}
\usepackage{tikz}
%\usetikzlibrary{trees,arrows}
```

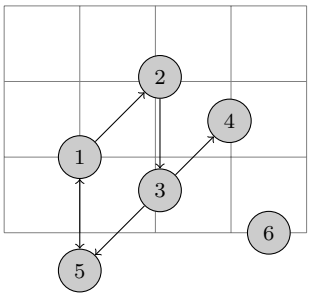



Abbildung 2: Ein Beispielgraph mit dem Paket `tikz`.

wobei man durch eine der Zeilen vor `\usetikzlibrary{tikz}` den Treiber auswählen muß, mit dem nachher die DVI-Datei weiterverarbeitet wird. Ergänzungen lädt man mit `\usetikzlibrary{...}`, eine Liste gibts im Manual [20].

**Achtung**: Irgendwie scheint sich `TikZ` nicht so gut mit `listings` und `\lstset{language=tex}` zu vertragen. Es scheint zu funktionieren, wenn man `listings` *vor* `TikZ` lädt:

```
\usepackage{listings}

\lstset{language=[LaTeX]tex} % BEFORE loading tikz!?

\def\pgfsysdriver{pgfsys-dvipdfm.def}
\usepackage{tikz}
```

### 10.2.2 Ein Graph

Die folgenden LaTeX Befehle setzen den in Abbildung 2 gezeigten Graphen. Man beachte, daß die Platzierung der Knoten relativ zueinander oder absolut angegeben werden kann und daß die Kanten quasi in Form von Adjazenzmatrizen einfach angegeben werden können.

```
\begin{tikzpicture}[node distance=1.5cm]
% Im folgenden wird JEDER `\node' mit dem angegebenen style
% gemalt:
\tikzstyle{every node}=[circle,draw=black,fill=black!20,font=\
footnotesize]

% gitterlinien, die (0,-1) starten und bis (3,2) gehen.
% schrittweite ist 1cm, wird durch den style `help lines'
% festgelegt.
\draw[style=help lines] (0,-1) grid (3,2);

% \node packt einen knoten an die AKTUELLE STELLE des path.
% das ist normalerweise (0,0).
% Das (1) setzt ein internes label, das {1} definiert den Text
% innerhalb des node.
\node (1)       {1};
```

```
% relativ zu (1) platziert:
\node (2) [above right of=1]  {2};
\node (3) [below of=2]   {3};
\node (4) [above right of=3,node distance=1.3cm] {4};
\node (5) [below of=1]   {5};

% Absolut auf (2.5cm,-1cm) platziert:
\node (6) at (2.5,-1)    {6};

% jetzt werden alle Kanten definiert.
% Das \path[->] heisst: alle Kanten sind standardmäßig -> pfeile.
% \path leitet einen "pfad" ein, der erst beim ';' endet.
%
% (COORD) edge (COORD) malt ein edge zwischen den zwei koords.
\path[->] (1) edge  (2)
              edge[<->] (5)
          (2) edge  (3)
          (3) edge  (4)
              edge  (5)
;
\end{tikzpicture}
```

Die Erläuterungen stehen im Text. Interessant wird es, wenn man Beziér Kurven haben will, die von den einzelnen Nodes zu anderen gehen sollen. Dazu braucht man verschiedene Sachen:

- Beziér Kurven (`\draw .. controls<c> and<d> ..`), [20, Abschnitt 11.1]

  (0,1)

  (0,0) (1,0)

  ```
  \begin{tikzpicture}[scale=1.5]
  \draw (0,0) .. controls (0,1) and (1,1) .. (1,0);
  \end{tikzpicture}
  ```

  Das malt eine Kurve von (0,0) zu (1,0), wobei die Kontrollen jeweils angeben, aus welcher Richtung die Kurve in den Start/Zielknoten eintreffen.

- Relative Koordinaten, um `<c>` und `<d>` relativ zu Start/Zielknoten anzugeben, [20, Kap. 10]

  ```
  \begin{tikzpicture}
  \draw (0,0) .. controls +(0,1) and +(0,1) .. (1,0);
  \end{tikzpicture}
  ```

  Erläuterung: der erste Kontrollpunkt wird relativ zum Startpunkt interpretiert $((0,0)+(0,1))$ und der zweite relative zum Endpunkt. Man kann auch `(90,1)` schreiben, d.h. Winkel 90° und 1cm. Alternativ kann man auch `(top,1)` schreiben, was äquivalent zu 90° ist. Weiteres siehe [20, Kap. 10].

- Die option, um Kurven in `edge` einzutragen, [20, Abschnitt 13.11]

Zusammengenommen kann man folgendes erreichen:

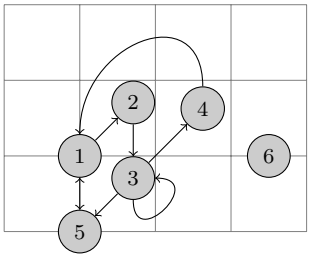



Abbildung 3: Ein Graph mit Beziér Kanten

### 10.2.3 Ein Graph mit gebogenen Kanten

Das folgende Listing generiert den Graph aus dem letzten Paragraphen, nur mit gebogenen Kanten. Das Resultat ist in Abbildung 3 gezeigt.

```
% mit ‘x=’ und ‘y=’ gibt man standart-einheiten für z.B. (1,2) an
\begin{tikzpicture}[x=1cm,y=1cm]
[ ... dasselbe wie oben ]
\path[->] (1) edge  (2)
              edge[<->] (5)
          (2) edge  (3)
          (3) edge  (4)
              edge  (5)
;
\path[->] (3) edge[to path={
                  .. controls +(down:1) and +(right:1) .. (\
                  tikztotarget)
              }] (3)
          (4) edge[to path={
                  .. controls +(up:1.5) and +(up:1.5) .. (\
                  tikztotarget)
              }]  (1)
;
\end{tikzpicture}
```

### 10.2.4 Bäume

Man kann mit `\usepackage{tikz}` recht leicht Bäume machen. Dazu folgen Beispiele aus [20, Kapitel 15]. Das folgendes Listing generiert den Baum aus Abbildung 4:

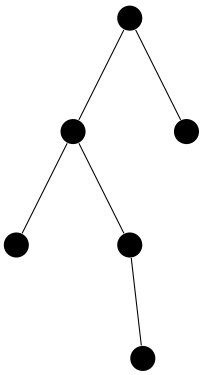


Abbildung 4: Ein Beispielbaum mit `tikz`.

```
\begin{tikzpicture}
\tikzstyle{every node}=[circle,fill=black]
\node {}
    child {node {}
        child {node {}}
        child {node {}
            child[right] {node {}}
        }
    }
    child {node {}};
\end{tikzpicture}
```

Man kann das auch tunen, sodaß die Abstände von Level zu Level sich ändern. Dazu gibt es die Styles `\tikzstyle{level X}`, die man neu definieren kann. Das Resultat zu Folgendem LaTeX-Code ist in Abbildung 5 zu sehen.

```
\begin{tikzpicture}[level distance=8mm]
\tikzstyle{level 1}=[sibling distance=4cm]
\tikzstyle{level 2}=[sibling distance=2cm]
\tikzstyle{level 3}=[sibling distance=1cm]
\tikzstyle{every node}=[circle,fill=black]
\node {}
    child {node {}
        child {node {}
            child {node {}}
            child {node {}}
        }
        child {node {}
            child {node {}}
            child {node {}}
        }
    }
    child {node {}
        child {node {}
            child {node {}}
```

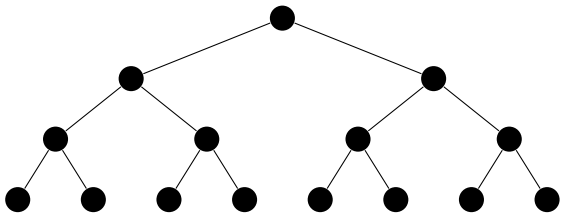


Abbildung 5: Ein zweiter Baum, der die „`level X`“ Styles zur Formatierung verwendet.

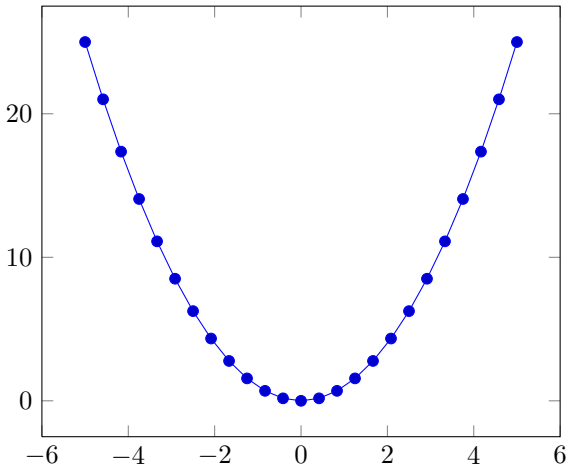
```
            child {node {}}
        }
        child {node {}
            child {node {}}
            child {node {}}
        }
    }
;
\end{tikzpicture}
```

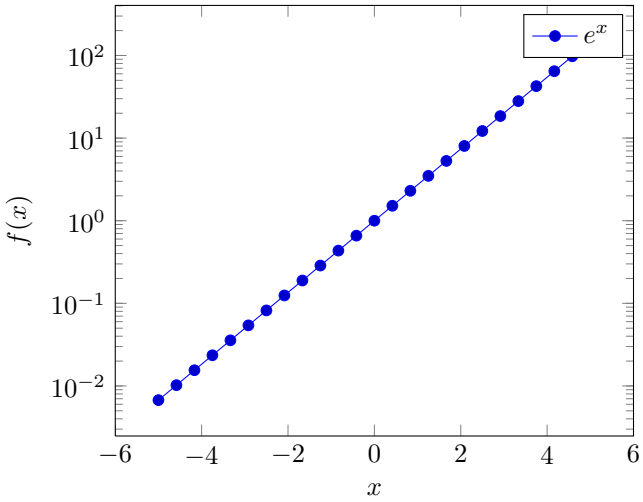
Interessant: man kann *denselben* Baum wie in Abbildung 5 mit folgendem Listing erreichen – durch die `foreach`-Konstruktion wird derselbe Code erzeugt:

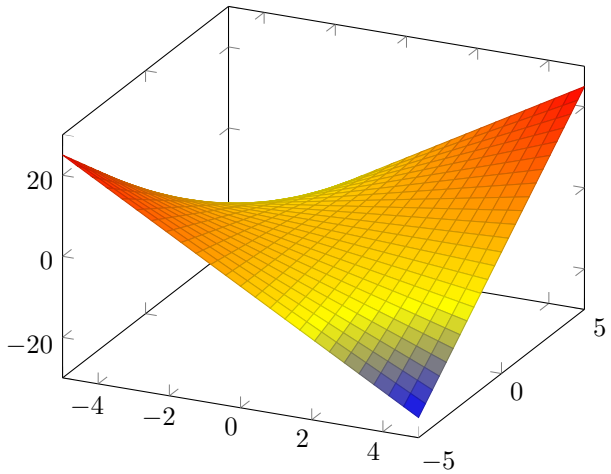
```
\begin{tikzpicture}[level distance=8mm]
\tikzstyle{level 1}=[sibling distance=4cm]
\tikzstyle{level 2}=[sibling distance=2cm]
\tikzstyle{level 3}=[sibling distance=1cm]
\tikzstyle{every node}=[circle,fill=black]
\node {}
    child foreach \i in {0,1} {node {}
        child foreach \j in {0,1} {node {}
            child foreach \k in {0,1} {node {}}
        }
    }
;
\end{tikzpicture}
```

## 10.3 Plots

Man kann plots in LaTeX mit `pgfplots` machen.




```
\begin{tikzpicture}
    \begin{axis}
    \addplot (\x,\x^2);
```

```
    \end{axis}
\end{tikzpicture}

\begin{tikzpicture}
    \begin{semilogyaxis}[xlabel=$x$,ylabel=$f(x)$]
    \addplot function {exp(x)};
    \legend{$e^x$}
    \end{semilogyaxis}
\end{tikzpicture}

\begin{tikzpicture}
    \begin{axis}
        \addplot3 {x*y};
    \end{axis}
\end{tikzpicture}
```

Auch hier hat das Manual [15] sehr viele Beispiele.

### 10.3.1 Zeichnungen

**Hinweis:** Ich bin mitlerweile der Meinung, dass TikZ [20] *das* Paket der Wahl für jegliche Art von Zeichnungen ist. Es kann so ziemlich alles was man will und hat ein leicht erlernbares Benutzerinterface – es ist dann stärker und schneller und besser als alle mir bislang bekannten Alternativen. Folgender Hinweis ist demzufolge **veraltet**.

Zum Erstellen von kleinen Illustrationen und Zeichnungen eignet sich `xfig` recht gut. Es kann zudem LaTeX code so in das Bild einbauen, dass die Graphischen Sachen in postscript (oder pdf) dargestellt werden und die LaTeX Elemente von LaTeX bearbeitet werden.

Seitdem das Paket `pgf` / `tikz` rausgekommen ist (s.o.) würde ich allerdings nicht mehr empfehlen, mit `xfig` zu arbeiten. Da ist es doch sowohl einfacher als auch hochwertiger, mit `tikz` zu arbeiten.

Dennoch eine Erläuterung hier zu Archivzwecken:

- man füge innerhalb von xfig einen text ein, bei dem man das „Special Flag“ auf „Special“ setzt.

  Das geht entweder, indem man *vor* einfügen des Textes unten die Text Flags einstellt oder aber für existierenden Text mit dem „EDIT“ Dialog draufklickt und das erwähnte Feld setzt.

- Innerhalb des xfig-Textfeldes kann man nun ganz normal LaTeX Befehle (z.B. `$1+1 = \frac{1}{2}$`) schreiben.

- Anschliessend exportiert man die Datei mithilfe von ‘FILE → EXPORT → LANGUAGE „Combined PDF/LaTeX (both parts)“’

- die Datei-Endung sollte „.pdf“ sein

- im LaTeX Dokument bindet man die Graphik dann mit `\input{datei.pdf_t}` ein.
- Achtung: beim exportieren wird u.U. ein Pfad von xfig in die Datei „datei.pdf_t“ mit eingegeben. Wenn der nicht stimmt, werden nur die LaTeX Textteile angezeigt, nicht aber die Graphik

Besonderheiten:

- man kann den export-vorgang automatisieren (z.B. im Makefile), indem man das Kommando `fig2dev` aufruft. Dieses kann auch noch Skalierungen durchführen (s.u.). Ein sinnvolles Beispiel fuer fig2dev könnte in einem Makefile auftauchen:

  ```
  %.pdf_t: %.fig %.fig.scale
  ifeq "$(HAS_TRANSFIG_WITH_PDF_T__SUPPORT)" "1"
  @SCALE=1; \
  if [ -f ${<}.scale ]; then \
      SCALE=‘cat ${<}.scale‘; \
  fi && \
  echo "converting␣pdf/latex␣figure␣$<␣with␣scale␣$$SCALE␣..." 
  && \
  fig2dev -L pdftex -m $$SCALE $< ${@:.pdf_t=.pdf} && \
  fig2dev -L pdftex_t -m $$SCALE -p ${@:.pdf_t=} $< $@
  else # HAS_TRANSFIG_WITH_PDF_T__SUPPORT = 0
  @SCALE=1; \
  if [ -f ${<}.scale ]; then \
      SCALE=‘cat ${<}.scale‘; \
  fi && \
  echo "converting␣fig␣->␣ps␣->␣pdf/latex␣figure␣$<␣with␣scale␣
  $$SCALE␣..." && \
  fig2dev -L pstex -m $$SCALE $< ${@:.pdf_t=.ps} && \
  fig2dev -L pstex_t -m $$SCALE -p ${@:.pdf_t=} $< $@ && \
  ps2pdf ${@:.pdf_t=.ps} ${@:.pdf_t=.pdf}
  endif #HAS_TRANSFIG_WITH_PDF_T__SUPPORT
  ```

  Die Idee davon ist, dass man im Makefile z.B. sagt

  ```
  FORCE:

  meindokument.pdf: meineabbildung.pdf_t FORCE
  ```

  Damit wird dann die Datei „meineabbildung.fig“ nach „pdf_t“ konvertiert. Falls auch noch die Datei „meineabbildung.fig.scale“ existiert und darin eine Zahl enthalten ist, wird die Abbildung um diese Groesse noch skaliert.

- Skalierungen: normalerweise kann man beim einbinden von Graphik eine skalierung durchführen. Das geht jetzt nicht mehr so einfach. Es geht
  - mit xfig beim exportieren
  - mit fig2dev mit „-m SCALE“
  - innerhalb von LaTeX mit einer `\scalebox` (s.o.).

# 11 Silbentrennung

Deutsche Silbentrennung wird mit dem Paket „german“ gemacht – allerdings funktioniert das standardmäßig nicht mit Umlauten. Da gibt es (zumindest) folgende Möglichkeiten:

1. Man gibt an den Stellen, wo LaTeX die Trennung nicht hinbekommt, die Trennmöglichkeiten von Hand ein:
   `Zie\-gen\-kä\-se`
2. Man verwendet T1-Fonts (`\usepackage[T1]{fontenc}`). Leider haben diese den Nachteil, daß zur Konvertierung nach pdf häßliche Bitmapfonts verwendet werden. Beim Druck sieht das gut aus, aber am Bildschirm ist es häßlich.
3. Man verwendet T1-Fonts zusammen mit dem Paket `\usepackage{lmodern}`, daß moderne Postskript Type 1 Fonts inclusive Umlaute frei zur Verfügung stellt. Das sieht so aus wie die standard fonts und LaTeX kümmert sich gewissenhaft um die gesamte Trennung. Ausserdem funktioniert dann auch copy–paste von Umlauten korrekt. Also:

   ```
   \usepackage{german}
   % erlaubt direkte Nutzung von Umlauten im TeX dokument:
   \usepackage[utf8]{inputenc}
   % hiermit kann man auch umlaute copy-pasten
   % ausserdem ist silbentrennung mit umlauten besser:
   \usepackage[T1]{fontenc}
   \usepackage{lmodern}
   ```

## 11.1 Waagerechte Striche

Eine Übersicht über die verschiedenen LaTeX-Striche wird in der folgenden Tabelle aus [14] zitiert:

| | Art des Striches | Mögliche Trennungen |
|---|---|---|
| `-` | Bindestrich, der andere Trennungen unterdrückt: `Mess-Ergebnis` | `Mess-Ergebnis` |
| `"=` | Bindestrich, der andere Trennungen erlaubt: `Mess"=Ergebnis` | `Mess-Er-geb-nis` |
| `"~` | Bindestrich, an dem nicht getrennt werden darf: `Mess"~, Schätz"~ und andere Ergebnisse` | `Mess-, ... Ergebnisse` |
| `\-` | Trennmöglichkeit, die andere Trennungen ausschließt: `Ur\-instinkt` | ... `Ur-instinkt` |
| `"-` | Trennmöglichkeit, die andere Trennungen nicht ausschließt, wenn Tex mal was nicht findet: `Am"-nestie` | ... |
| `""` | Trennmöglichkeit, bei der kein Trennstrich benötigt wird: `(Mess"~)""Ergebnisse Messer/""Gabel` | ... (Mess-) Ergebnisse |
| `"\|` | Auflösen einer Ligatur und Schaffung einer Trennmöglichkeit: `Auf"\|forderung` und für Frakturfreunde die legendäre `Wachs"\|tube` bzw. `Wach\-stube` | Wachs-tube Wach-stube |

`--` Der längere, meist auch dünnere horizontale Strich heißt – je nach Verwendung – Gedanken-, Erstreckungs-, Auslassungs- oder Spiegelstrich.

- In der ersten Zeile zwei typische Gedankenstriche.
- Auf den Seiten `28--32` finden Sie Erstreckungsstriche, diese werden ohne Leerraum gesetzt!
- Bei glatten Preisen ersetzt man gern die Cent durch einen Auslassungsstrich: 17,–€
- Die Zeilen dieser Aufzählung beginnen mit Spiegelstrichen.

Als Spiegelstrich kann man auch den längeren englischen Gedankenstrich `---` einsetzen.

`$-$` Das Minuszeichen unterscheidet sich in manchen Schriften nicht vom Gedankenstrich, erhält aber andere Leerräume um sich herum (Spationierung). Bei TeX auch für simple Ausdrücke stets in den Mathemodus schalten! `$11-5=6$`

# 12 figures

LaTeX arbeitet normalerweise mit Fließumgebungen

```
\begin{figure}
   Figure inhalt ...
   \caption{Beschreibung}
   \label{...}
\end{figure}
```

und man beschreibt die Abbildung unter Verwendung des labels. Man kann mit

```
\begin{figure}[t]
\begin{figure}[b]
\begin{figure}[h]
```

sagen, wohin die figure soll (top, bottom, here). Das here funktioniert aber irgendwie nie.

Manchmal ist das aber weder gewollt noch schön und man möchte, dass die figures dort im Dokument auftauchen, wo man sie auch hingeschrieben hat. Das geht bspw. mit

```
\usepackage{float}
\begin{figure}[H]
\end{figure}
```

Siehe die Paketdokumentation von float für mehr Informationen. Auf diese Weise taucht die Figure auf jeden Fall dort auf, wo sie definiert wird (es sei denn, die Seite ist zu Ende).

## 12.1 Umflossene Abbildungen

Das ganze mit umflossenen Abbildungen ist nicht sehr verläßlich, scheint mir. Aber es geht immerhin, wenn auch grundsätzlich nicht im Zusammenhang mit Seitenumbruch. Ein paar Pakete dazu sind auf `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=textflow` aufgelistet. Hier sind meine Erfahrungen mit ein paar davon:

### 12.1.1 picinpar

Ist etwas unkonventionell in der Bedienung, hat aber wenigstens das getan, was ich wollte: es hat Text über, seitlich von oder unter der Figure eingefügt (man kann die Zahl der Zeilen einstellen). Davon wurde abgeraten weil es wohl veraltet ist. Eine Anleitung findet sich unter [1]. Ich zitiere hier das erste Beispiel aus [1]:

```
\begin{figwindow}[2,r,{%
\unitlength1cm
\begin{picture}(3,1.4)
\put(0.7,0.7){\circle*{0.2}} \put(0.7,0.7){\circle{1.2}}
\put(0.7,0.7){\vector(0,1){0.6}} \put(2.5,0.7){\circle*{0.5}}
\end{picture}
},{Kreise und Pfeile}]
Was leisten nun diese Macros ...
... sieht hierbei wie oben aus.
\end{figwindow}
```

Was leisten nun diese Macros? Einen kleinen Eindruck hat man schon zu Anfang dieses Artikels bekommen: der erste Buchstabe des Absatzes ist in einer anderen Schriftgröße gedruckt und in den Absatz eingepaßt. Es muß aber nicht unbedingt Text sein, der so eingerückt wird. Es kann ebenso eine pictureUmgebung sein, die so in den Absatz eingefugt wird. Das rechts stehende Beispiel, das aus „LaTeX, Eine Einführung“ bekannt sein dürfte, ist in einer minipageUmgebung gesetzt worden. Die Eingabe im Text sieht hierbei wie oben aus.

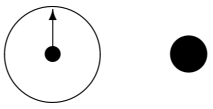


**Abbildung 6:** Kreise und Pfeile

Ende des Zitats. Die allgemeine Syntax für `\begin{figwindow}` ist

```
\begin{figwindow}[ZEILEN ÜBER FIGURE,lrc,{FIGURE},{CAPTION}]
...
\end{figwindow}
```

### 12.1.2 picins

Zu finden unter `ftp://cam.ctan.org/tex-archive/systems/msdos/picins/picins.zip` mit einer deutschen Dokumentation im dvi format.

Mit

```
\parpic{\fbox{Ein Bild}}
Ein frisch startender Beispielabsatz wird jetzt neben das Bild
gesetzt und umflossen.
```

Ein Bild Ein frisch startender Beispielabsatz wird jetzt neben das Bild gesetzt und umflossen.

Die wichtigsten Optionen sind wohl `\parpic[r]{...}` für ein Bild rechts und ein paar Alignmentsachen (siehe das Manual dazu). Interessant ist noch `\picskip{0}`, womit man den Absatz (nach 0 Zeilen) neben dem Bild beenden kann. Es gibt auch ne Option für doppelseitige Styles.

Es kann auch noch shadowboxen machen mit

```
\begin{shadowenv}
...
\end{shadowenv}
```

Falls man das zusammen mit picinpar verwenden will, muss man picinpar danach definieren.

### 12.1.3 floatflt

Hat mich nicht überzeugt: Text kann nicht oberhalb von einer Abbildung sein.

### 12.1.4 wrapfig

Genauso gut oder schlecht. Irgendwie habe ich das nicht richtig verwenden können.

```
\begin{wrapfigure}[height of figure in lines]{l|r,...}[overhang]{
width}
  figure, caption, etc.
\end{wrapfigure}
```

## 12.2 Preventing figures from appearing on a page by themselves. ([5])

LaTeX's figure placement algorithm is quite biased in favor of putting figures on a page by themselves, instead of on the top of a page with some text below it. Often, I find the result esthetically unappealing (to be polite). Fortunately, the parameters of the algorithm can be changed. The main problem is that LaTeX per default only allows a part of the top of a text-page (70%) to contain figures,

and requires at least 20% of a page to be text when text and figures share a page. These parameters should be set to more reasonable values, for example 85% and 10%.

```
\renewcommand{\topfraction}{0.85}
\renewcommand{\textfraction}{0.1}
```

This helps, but sometimes LaTeX puts a figure on a page by itself, although it would fit perfectly well on the top of a page. This happens when the figure will not fit on the page where it was defined. LaTeX then attempts to put it on a figures-only page before it attempts to put it at the top of the next page. A page may contain figures alone if the figure(s) use at least half the page. To prevent half-empty pages this limit should probably be increased to around 75%.

```
\renewcommand{\floatpagefraction}{0.75}
```

Be careful not to make `\floatpagefraction` larger than `\topfraction`, then you risk to produce a figure that can neither go on the top of a text page, nor on a page by itself. If that happens, the figure and all later figures will be postponed until next time a `\clearpage` is executed (typically at the end of a chapter or the end of the document). This will also happen if a figure is too large to fit on a page.

# 13 Einige Spezialitäten, gut zu kennen

## 13.1 No room for a new dimen - Fehler

Aus alten TEX Zeiten gibt es die Einschränkung, das nur jeweils 256 Register für Zahlen und Größenangaben zur Verfügung sind. Da kommt man mitlerweile problemlos drüber, wenn man mehrere Pakete nutzt. Abhilfe:

```
\usepackage{etex}
```

vor der erste `\usepackage` schreiben. Das vergrößert diese Zahl auf 32768.

Es scheint in seltenen Fällen auch notwendig zu sein,

```
\usepackage{etex}
\reserveinserts{30}
```

zu nutzen. Das vergrößert irgendeinen anderen Parameter, den ich nicht verstanden habe.

## 13.2 Zeilenabstand

`\linespread{ZAHL}`
Einstellen des Zeilenabstands (1 ist normal, 1.3 ist anderthalbfach, 1.5 doppelt). Beachte: Der Befehl wird erst nach dem nächsten Fontwechsel aktiv! Man kann allerdings faken, indem man z.B. sowas wie `\tiny \normalsize` macht.

## 13.3 Andere Header/Footer

Im Paket `fancyhdr` kann man schön die Header/Footer einstellen. Es bietet sich an, die pagestyles zu überschreiben/neue zu machen:

```
% 'plain': nur seitennummer in der Mitte
\fancypagestyle{plain}{
    \fancyhead{\centering \thepage}
    \fancyfoot{}
    \renewcommand{\headrulewidth}{0pt} % keine Linie
    \renewcommand{\footrulewidth}{0pt}
}
```

## 13.4 Seitennumerierung

- `\thepage`
  liefert die Seitenzahl (kann man auch aendern) Dazu ein Anwendungsbeispiel. Nehmen wir an, wir wollen alle Figures auf der jeweils nächsten Seite darstellen. Die Seitennumerierung soll diese nächste einfach überspringen; stattdessen soll z.B. 3a da stehen. Dazu:

  ```
  \newcommand{\figurepage}[1]{
      \afterpage{
          \addtocounter{page}{-1}
          \thispagestyle{figurepage}
          #1
          \clearpage
      }
  }
  ```

  allerdings braucht man dazu das Paket `afterpage`.

- Ändern des Anzeigeformats für Seitenzahlen:

  ```
  \pagenumbering{arabic} % Arabic numerals
  \pagenumbering{roman} % Lowercase Roman numerals
  \pagenumbering{Roman} % Uppercase Roman numerals
  \pagenumbering{alph} % Lowercase letters
  \pagenumbering{Alph} % Uppercase letters
  ```

### 13.4.1 Seitennumerierung nach Kapiteln

Man kann—zumindist in der `book` Klasse— zwischen verschiedenen Bereichen umschalten:

```
\documentclass{book}
...
\begin{document}
    \frontmatter
```

```
    \include{titlepg}  % include title page
    \include{preface}  % include preface
    \include{other}   % include other frontmatter

    \mainmatter

    \include{chapter1}  % include first chapter
    \include{chapter2}  % include second chapter
    \include{chapter3}  % include third chapter, etc.

    \appendix
    \include{history}  % include first appendix
    \include{tables}  % include second appendix, etc.

    \backmatter

    \include{bibliography} % include bibliography
    \include{index}   % include index
\end{document}
```

## 13.5 Datumsangaben

Man kann mit `\today` die Ausgabe des aktuellen Datums in einem eingebauten Format erzeugen. Sollte das nicht genug sein kann man auch mit

- `\number\month` den aktuellen Monat,
- `\number\year` das aktuelle Jahr,
- `\number\day` den aktuellen Tag

ausgeben. Beispiele:

- `today`: 19. Oktober 2010
- `\number\day.\number\month.\number\year`: 19.10.2010

## 13.6 Suchpfade

Umgebungsvariable TEXINPUTS:
bspw.
`:/home/zulficar/usr/local/share/texmf//` '`:`' ist trennzeichen; wenns am Anfang steht: behalte default sachen '`//`' = durchsuche rekursiv

## 13.7 Überschreiben von LaTeX Sachen

Ich habe noch keine ordentliche Doku gefunden, wie man LaTeX Sachen überschreibt. Nur soviel: in den Implementierungen kommen immer @ Zeichen vor, sodaß man

```
\makeatletter
...
\makeatother
```

schreiben muß, um das @ Zeichen zu behandeln.

Was man darüber hinaus wohl tuen kann, ist die Datei `latex.ltx` zu finden, darin den gesuchten Befehl suchen und den entsprechenden Teil in seiner Präambel neu zu definieren. Beispiel:

```
\let\oldItemizeBegin=\itemize
\let\oldItemizeEnd=\enditemize

\renewenvironment{itemize}{%
    \begingroup%
    \oldItemizeBegin%
    \setlength{\parskip}{1cm}%
}{%
    \oldItemizeEnd%
    \endgroup%
}
```

Ab jetzt werden itemize Umgebungen mit `\parskip=1cm` gesetzt.

## 13.8 Farbige Boxen

Paket `\usepackage{fancybox}`.

## 13.9 Zeigen von Codezeilen

Das Paket `listings` erscheint mir ausgezeichnet. Damit habe ich alle LaTeX-Zeilen hier gesetzt. Das kann

- indentation,
- Syntax highlighting,
- ca. 60 Programmiersprachen,
- neue Sprachen definieren.

### 13.9.1 `listings` und Copy/Paste

Mit `listings` werden auch normale LaTeX-Fonts so gesetzt, daß sie gleichen Abstand haben. Dazu müssen Leerzeichen zwischen einzelnen Buchstaben eingefügt werden, was insbesondere bei späteren Copy-Paste-Versuchen aus dem fertigen PDF zu Frust führt.

Abhilfe: man verwende

```
\usepackage{listings}
% \usepackage{courier}
\lstset{
    language=[LaTeX]tex,
    tabsize=4,
    breaklines=true,
    breakindent=0pt,
    columns=fullflexible,
    basicstyle=\ttfamily} % otherwise copy-and-paste won't work
```

in der Präambel. Das `\usepackage{courier}` sorgt dafür, daß `\ttfamily` auch mit Boldface gedruckt werden kann (was mit den Standard-TEX-fonts nicht geht). Dafür sieht `courier` nicht ganz so schön aus. Mit `basicstyle` werden optionen für alle Listings und für alle Teile eines Listings gesetzt.

## 13.10 Text im Randbereich

`\marginpar[left]{right}`
fügt Anmerkungen am Rand ein

## 13.11 Beliebig Positionieren

### 13.11.1 Mit TikZ

Das `tikz` paket erlaubt sowohl absolute Positionierungen (d.h. in Koordinaten auf der aktuellen Seite des Ausgabedokuments) als auch Positionierung relativ zur aktuellen "Cursor" Position (d.h. wo man gerade schreibt).

Für absolute Positionierung siehe [20].

Relative Positionierung geht, indem man einen `\node` mit dem `overlay` style erzeugt. Ich habe z.B. in Vortragsfolien seit kurzem folgendes Hilfsmakro verwendet:

```
\tikzset{
    every overlay node/.style={
        draw=black,fill=white,rounded corners,anchor=north west,
    },
}

% Usage:
% \tikzoverlay at (-1cm,-5cm) {content};
% or
% \tikzoverlay[text width=5cm] at (-1cm,-5cm) {content};
\def\tikzoverlay{%
    \tikz[baseline,overlay]\node[every overlay node]
}%
```

Beispiel:

```
\tikzoverlay[text width=6cm] at (9.3cm,5cm) {
```

```
    \begin{itemize}
        \item \emph{Derive subclass} from \texttt{G
        \item one \emph{variable definition} per op
        \item \emph{Default Values}
    \end{itemize}
};
```

- *Derive subclass* from `GetOptWrapper`
- one *variable definition* per option
- *Default Values*

[hier steht `\tikzoverlay` ...]

Das ganze tut keine Zauberei, es ist "nur" ein `\node` von Tikz, der durch `\tikzoverlay` gestartet wird und vom Nutzer zu Ende geführt wird (inklusive des obligatorischen TikZ semikolons am Ende).

Man muss overlays immer als letztes Element bringen, damit die zuoberst gemalt werden.

Innerhalb von `beamer` habe ich das oft zusammen mit `\only<2>{\tikzoverlay ...}` verwendet. Dann gibt es Probleme, wenn ein neuer Paragraph innerhalb des overlays erzeugt wird (ich glaube, das kommt von `\only`). Falls es jemandem hilft: man kann `\newcommand{\parx}{\par}` definieren und dann `\parx` zum erstellen eines neuen Paragraphen verwenden. Das klappt.

### 13.11.2 Mit `textpos`

Man kann mit dem Paket

```
\usepackage{textpos}
\setlength{\TPHorizModule}{1cm} % Horizontale Einheit
\setlength{\TPVertModule}{1cm} % Vertikale Einheit
...
\begin{textblock}{WIDTH}(OFFSET_X,OFFSET_Y)
    INHALT
\end{textblock}
...
```

beliebige Positionen angeben. Beispiel für eine Positionsangabe relativ zur aktuellen "Cursor" Position:

```
\definecolor{boxcol}{gray}{0.89}
\begin{textblock}{3}(8,-3)
\fcolorbox{black}{boxcol}{%
    \begin{minipage}{\textwidth}
    \setlength{\parindent}{0pt}%
    \setlength{\parskip}{0.1cm}%
    Ein Hinweis, der mittels Koordinatenanga
    \end{minipage}
}%
\end{textblock}%
```

Ein Hinweis, der mittels Koordinatenangabe plaziert wurde

[ Hier steht `\begin{textblock} ...` ]

Die Koordinaten sind immer relativ zu dem Punkt, wo `\begin{textblock}` steht. Was welchen Teil überschreibt wird bestimmt nach dem Motto „Wer zuletzt

kommt, malt zuoberst“. Siehe die Paketdokumentation zu `textpos` im CTAN, [11]. Es gibt aber auch absolute Positionierungsmöglichkeiten.

## 13.12 Spaces in der Eingabe ingorieren

Es gibt den TEX Befehl `\ignorespaces`, mit dem man Spaces ignorieren kann. Einige Beispiele aus

`http://en.wikibooks.org/wiki/Programming:TeX_%5Cignorespaces`:

```
\def\test#1{(#1)}
\test{a} b % Generate "(a) b" with space
\def\test#1{(#1)\ignorespaces}
\test{a} b % Generate "(a)b" without space
% Macro invocations
\def\test{ a}
(\test)b % Generate "( a)b" with space
(\ignorespaces \test)b % Generate "(a)b" without space
% Placeholder invocations
\def\test#1{(#1)}
\test{ a}b % Generate "( a)b" with space
\def\test#1{(\ignorespaces #1)}
\test{ a}b % Generate "(a)b" without space
```

## 13.13 Klick im Viewer öffnet Editor und umgekehrt

Man kann in LATEX spezielle Befehle in den Quellcode einbauen, die

- im *Viewer* einen Klick auf ne beliebige Zeile erlauben und dann den Editor öffnen,
- im *Editor* mit einem Befehl den Viewer an der richtigen Stelle öffnen.

Das geht mit „Dvi Specials“. In LATEX braucht man folgendes:

```
\usepackage{srcltx}
```

Das kommentiert man dann aus, wenn das Dokument fertig ist.

### 13.13.1 Verwendung von `Vim` und `xdvi/kdvi`

`Vim` → `kdvi`: Mit der Zeile

```
:noremap <buffer> <C-Right> :exec "silent
    !kdvi --unique file:".expand("%:r").'.dvi\#src:'.line(".").
    expand("%")." &"<CR>
```

in `VIMDIR/ftplugin/tex.vim` kann man durch Drücken von Control-PfeilRechts `kdvi` an der Stelle unterm Cursor zu öffnen.

`Vim` → `xdvi:` Dasselbe geht mit der Zeile

```
:noremap <buffer> <C-Right> :exec "silent
    !xdvi -sourceposition ".line(".").':'.col(".")."%".' '."%:r
    .dvi"<CR>
```

für `xdvi`.

`kdvi` → `vim:` Das kann man direkt im Menu vom `kdvi` unter *Einstellungen* einstellen; keinerlei weitere Schritte sind notwendig. Man klickt einfach mit der mittleren Maustaste irgendwo im Viewer hin und der `vim` öffnet sich an der richtigen Stelle.

### 13.13.2 Besonderheiten

Damit das mit dem Hin- und Herspringen auch dann funktioniert, wenn man `\input ...` benutzt, muß man Folgendes beachten:

- `\input foo` → `\input{foo}`
- `\input{foo.tex}` → `\input{foo}`
- Es kann sein, daß `\usepackage{srcltx}` das Spacing ändert.

Dann öffnet der `kdvi` beim Mittelmaus–Klick an die entsprechende Stelle die Datei `foo.tex` an der richtigen Stelle.

## 13.14 Automatische Vermeidung von overful/underful Fehlern

Man kann in LaTeX die Qualität der Ausgabe, d.h. der Paragraphbildung, konfigurieren. Standardmäßig ist das auf höchste Qualität eingestellt.

```
\tolerance=200
```

Je höher der Wert, desto mehr Zwischenraum wird zwischen Worten eingefügt, um den Blocksatz hinzubekommen. Je kleiner der Wert, desto weniger Zwischenraum wird eingefügt und desto eher wird Silbentrennung benutzt.

Wenn LaTeX es nicht schafft, diese Toleranz einzuhalten, wird eine „Overfull hbox“ Warnung ausgegeben und der Text guckt rechts aus dem Blocksatz raus. Hier muss man also VON HAND nachbessern, indem

- der Satz umgestellt wird oder
- die breite des Absatzes größer gemacht wird oder
- bei der Silbentrennung nachgeholfen wird.

Der Standardwert ist „`\tolerance=200`“. Defensiver ist `\tolerance=2000`.

```
\setlength{\overfullrule}{5pt}}
```

Erlaubt die Angabe einer schwarzen Rand–Markierung, wenn eine „overful hbox“ produziert wurde.

```
\emergencystretch=10pt
```

`\emergencystretch` läßt den Leerraum zwischen den Wörtern beim Umbruch weiter werden. Damit gibt man den Leerraum an, der innerhalb einer Zeile zusätzlich verteilt werden darf, wenn der normale Umbruch zu einer überlangen Zeile führt.

Das ist normalerweise besser als riesige `\tolerance` werte Standard ist 0pt.

```
\hbadness=1000
```

Mit `\hbadness` kann man steuern, wann im *logfile* Warnungen wegen „zu geringer Qualität“ ausgespuckt werden. Je höher der Wert, desto weniger Warnungen gibt es.

Der Standardwert ist `\hbadness=1000`.

**Vorschlag also:**

```
% Fix overful hboxes automatically:
\tolerance=2000
\emergencystretch=10pt
```

## 13.15 Silbentrennung verbieten

Es mag schonmal vorkommen, dass man die Silbentrennung ausschalten will, aber trotzdem ohne `\raggedright` arbeiten möchte (z.B. in einem `beamer` Vortrag).

Dann hilft

```
\pretolerance=10000
```

weiter – das kann man auch gut lokal in einem einzigen Slide hinschreiben. Der Effekt: es werden noch nichtmal Kandidaten für Silbentrennung gesucht.

Ich vermute, auch `\hyphenpenalty=10000` würde den Effekt erreichen.

Quelle: [17, P. 96].

## 13.16 Copy-Paste von Umlauten im PDF

Wenn man Umlaute verwendet und diese aus dem finalen PDF mit copy-paste kopiert, erhält man zuweilen

```
k¨mmert
 u
```

also die Zusammensetzung von Punkten und normalen Buchstaben. Abhilfe schafft hier das richtige Font-Encoding. Empfehlung:

```
\usepackage{german}
% erlaubt direkte Nutzung von Umlauten im TeX dokument:
```

```
\usepackage[utf8]{inputenc}
% hiermit kann man auch umlaute copy-pasten
% ausserdem ist silbentrennung mit umlauten besser:
\usepackage[T1]{fontenc}
\usepackage{lmodern}
```

### 13.17 \protect

Manchmal bekommt man kryptische Fehlermeldungen, wenn man irgendein Makro an die falsche Stelle gesetzt hat (bspw. in `\section` oder `\footnote` oder `\caption`). Dann hilft es oft, wenn man `\protect` einfach vor das Makro setzt.

### 13.18 Anpassungen des Inhaltsverzeichnis

Siehe `http://www.image.ufl.edu/help/latex/toc.shtml` für eine interessante Übersicht.

### 13.19 Technisch: Verbieten von Seitenumbrüchen

Etwas, das mich regelmäßig auf die Palme bringt, sind Seitenumbrüche. Wenn es darauf ankommt, bekomme ich es selten hin.

Aber zur Sache: Es passiert, dass man einen Seitenumbruch verhindern will. TEX/LATEX bieten dafür zwei Methoden an:

1. Das Verbieten von Seitenumbrüchen mithilfe von `\nobreak`,
2. Das Ermutigen zu Seitenumbrüchen an anderen Stellen mit `\pagebreak[zahl]`. Ohne Zahl wird auf jeden Fall umgebrochen, ansonsten bedeuted eine höhere Zahl eine stärkere Empfehlung (1-4).

Zweiteres klappt meistens – ist aber ganz sicher nur zweite Wahl! Denn in der Regel ergänzt man irgendwas am Dokument, und schon muss man alle "Empfehlungen" überarbeiten. Aber wenigstens klappt es.

Ersteres klappt nur in Sonderfällen, obwohl es genau das ist, was ich oft suche. Warum klappt es nicht? Nun, das vergesse ich jedesmal. Darum hier die Hintergründe:

TEX bricht Zeilen oder Seiten üblicherweise nach Whitespaces (bspw. an `\hskip` oder `\vskip`) um. Wenn man `\nobreak` vor den Leerraum setzt, wird der Umbruch verboten. Das dumme: wenn ich es richtig verstehe, muss es *direkt* davor stehen. Aber die LATEX Umgebungen erlauben das nicht; man kommt schlicht nicht an Stelle direkt vor Leerräumen dran.

Es klappt aber, wenn man `\nobreak` direkt *vor* den Leerraum setzen kann. Beispielsweise ist '`~`' definiert als `\nobreak\ `, was `~` zu einem non-breaking-space macht. Man kann auch vor einem neuen Paragraph `\nobreak` schreiben (muss

aber sicherstellen, dass `\nobreak` *nach Beendigung* des alten Paragraphen, d.h. im vertikalen Modus, steht):

```
Ein Paragraph

\nobreak% dies ist ein VERTIKALER \nobreak
Ein neuer, vor dem kein Seitenumbruch passieren darf.
```

## 13.20 Vertikales Alignment

Eine oftmals schwer anmutende Sache ist vertikales Alignment. Beispiel: man möchte gerne ein Textfeld neben einem Bild haben.

Das Textfeld kann man mit `minipage` hinbekommen (die Breite muss man allerdings wissen). Aus dem Handbuch von LaTeX entnimmt man, dass die Option `\begin{minipage}[b]{`⟨*breite*⟩`}` unten („bottom“) ausrichtet und `[c]` vertikal mittig zentriert.

Aber `[t]` richtet *nicht* am oberen Rand aus! Die Option `[t]` richtet *an der Basislinie der ersten Zeile* in der `minipage` aus. Wenn man am oberen Rand ausrichten will, muss die erste Zeile in der `minipage` die Höhe 0 haben. Das geht z.B. mit `\vspace{0cm}`:

```
\begin{minipage}[t]{5cm}
\vspace{0pt}
Diese ist eine Minipage
\end{minipage}
Hier gehts ausserhalb weiter.
```

liefert:

Diese ist eine Minipage Hier gehts ausserhalb weiter.

Ich weiss, das Beispiel ist nicht gut. Ich bin für Verbesserungen offen.

## 13.21 Acrobat Reader und Reload Document

Neue Versionen von Acrobat Reader unterstützen das von Haus aus.

Ansonsten: Unter

`http://tug.ctan.org/tex-archive/support/acroreloadpdf/`

gibt es ein kleines Skript, das für die Linux Version von Acrobat Reader eine Reload Funktion bereit stellt.

Beachte, dass es unter Linux zahlreiche Viewer mit der Funktionalität gibt – und unter Windows gibt es Sumatra PDF, der sehr gut mit LaTeX zusammenarbeitet.

## 13.22 Suche nach einem bestimmten LaTeX Symbol

Es gibt die “Comprehensive LaTeX Symbol List”, [18].

Ferner kann man Symbole *online* suchen, indem man sie grob einzeichnet, siehe

`http://detexify.kirelabs.org/classify.html`.

## 13.23 Kommandozeilenargumente ans Dokument

Als Skript-begeisterter hatte ich zuweilen den Wunsch, Kommandozeilenargumente an ein LaTeX-Dokument zu übergeben. Eine Anwendung dafür ist eine `Makefile`, das ein teures `make dist` target enthält (in meinem Fall habe ich damit vollautomatische Bildexternalisierung oder vollautomatische Erstellung von Querverweisen gesteuert).

Im Prinzip ist das einfach; es besteht aus zwei Schritten:

1. man muss die Argumente beim Aufruf von `latex` bzw. `pdflatex` übergeben und
2. man muss die Argumente verarbeiten im Dokument können.

Angenommen, das Hauptdokument heisst `dokument.tex`. Der erste Schritt geht dann wie folgt: man tippe

```
pdflatex dokument
```

wenn man Argument weglassen will und alternativ

```
pdflatex '\def\meinargument{1}\input{dokument}'
```

wenn man es setzen will (das geht analog mit `latex` statt `pdflatex`). Dazu muss man wissen, dass `pdflatex dokument` im Wesentlichen nichts anderes tut als `pdflatex '\input{dokument}'` aufzurufen – unser Statement oben macht also dasselbe, nur dass zuvor noch ein Zusatzbefehl ausgeführt wird. Das `\def` definiert ein Makro und ist quasi dasselbe wie `\newcommand` (nur auf niedrigerer Ebene). In meinem Fall setze ich den Wert auf "1", weil ich einfach nur prüfen will, ob das Makro existiert oder nicht.

Der zweite Schritt, die Abfrage im Hauptdokument, geht wie folgt: wir können einfach prüfen, ob das Argument definiert ist oder nicht. Das geht beispielsweise mit der etwas umständlichen Konstruktion

```
\expandafter\ifx\csname meinargument\endcsname\relax
   % Ah -- es ist NICHT definiert
   %
   % tue irgendwas...
\else
   % Ah -- das makro IST definiert.
   %
   % tue irgendwas anderes...
\fi
```

Zur Erläuterung für Interessierte: Das `\csname meinargument\endcsname` erlaubt die Erstellung von Makronamen ("**c**ontrol **s**equence name"). Wenn das resultierende Makro nicht existiert, wird es auf `\relax` gesetzt. Das `\ifx` prüft, ob die

zwei folgewerte gleich sind. Das `\expandafter` führt `\csname ...\endcsname` aus und fügt stattdessen das Resultat an die Stelle ein. Mit anderen Worten: Falls `\meinargument` definiert ist, wird obiger Check zu `\ifx\meinargument\relax` (was falsch ist und daher in den `\else` Teil springt). Falls es nicht definiert ist, wird es zu `\ifx\relax\relax` (was wahr ist und daher in den ersten Teil springt).

Man kann dann also irgendwas an Initialisierung tun, z.B. Pakete laden wenn man das in der Präambel abfragt.

Vorteil: das Dokument kompiliert auch, wenn man das Argument *nicht* explizit angibt.

In obigem Beispiel wird der Wert ignoriert. Man könnte natürlich auch Werte abfragen, z.B. mit dem LaTeX Befehl `\ifthenelse`.

### 13.24 Weitere offene Fragen?

Siehe [3].

## Literatur


[1] *Beispiele zu picinpar.sty.* `http://www.ifi.uio.no/it/latex-links/picinpar.pdf`.

[2] *Einführung in TEX.* `http://www.ruhr-uni-bochum.de/www-rz/schwanbs/TeX/einfuehrung-in-tex.pdf`. Empfehlenswert für Experten und solche, die es werden wollen.

[3] *Frequently asked questions.* `http://www.faqs.org/faqs/de-tex-faq/part1/`. Lesenswert.

[4] *Graphics and Colour with LaTeX.* `http://tex.loria.fr/graph-pack/grf/grf.htm`.

[5] *Jacobs LaTeX Tips.* `http://www.fysik.dtu.dk/~schiotz/comp/latextips`.

[6] *LaTeX for Complete Novices.* `http://theoval.sys.uea.ac.uk/~nlct/latex/novices/novices_a4.pdf`.

[7] *LaTeX-Fortgeschrittene Anwendungen.* `ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/broschueren/a0279510.pdf`.

[8] *LaTeX-Kurzreferenz auf 5 Seiten.* `http://www.gasperl.at/michael/downloads/latexkurzreferenz.pdf`.

[9] *LaTeX2e for class and package writers.* Teil des LaTeX2e Pakets, kommt als `clsguide.pdf`, findet sich aber auch zuweilen im Netz, bspw. unter `http://www.latex-project.org/guides/clsguide.pdf`. Es enthält Befehle für Dependencies in eigenen Paketen, Optionsbearbeitung u.ä.

[10] *The LaTeX2e Sources.* Das Dokument findet sich als `sources2e.tex` in der LaTeX Distribution. Man kann es zuweilen als fertiges PDF im Internet finden, bspw. unter `http://www.tug.org/texlive/Contents/live/texmf-dist/doc/latex/base/source2e.pdf`. Es enthält sehr viele low-level LaTeX Befehle, die für Paketschreiber interessant sind.

[11] *Online Paketreferenzen.* `http://tug.ctan.org/search.html`.

[12] *Package colortable.* `http://tug.ctan.org/tex-archive/macros/latex/contrib/colortbl/colortbl.pdf`.

[13] *Using Imported Graphics in LaTeX and pdfLaTeX.* Zu finden bspw. unter `ftp://ftp.tex.ac.uk/tex-archive/info/epslatex.pdf`.

[14] *Warum Typografie?* `http://homepage.ruhr-uni-bochum.de/Georg.Verweyen/latexfuerword.html`.

[15] Feuersänger, Christian: *Package PGFPlots manual.* `http://pgfplots.sourceforge.net/`.

[16] Grahn, Alexander: *AcroReloadPDF.js.* `http://tug.ctan.org/tex-archive/support/acroreloadpdf/`.

[17] Knuth, D.: *Computers & Typesetting – The TeXbook.* Addison Wesley, 1986.

[18] Pakin, Scott: *The Comprehensive LaTeX Symbol List*, 2002.

[19] Society, American Mathematical: *User's Guide for the amsmath Package*, 1992.

[20] Tantau, Till: *TikZ and pgf Manual v.1.18.* Zu finden bspw. unter `http://tug.ctan.org/tex-archive/graphics/pgf/doc/generic/pgf/version-for-pdftex/en/pgfmanual.pdf` oder mit `texdoc pgfmanual`.