

WEITERFÜHRENDES L^AT_EX

EIGENE ENVIRONMENTS BIS HIN ZU EIGENEN PACKAGES

Alexander **Phi** Goetz | info@phictional.de

Fachschaft Informatik Uni Tübingen
fsi@fsi.uni-tuebingen.de

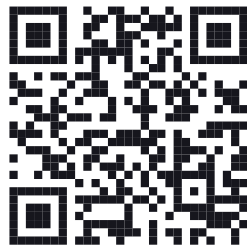
14. April 2022

Disclaimer:

Dieser Workshop baut direkt auf dem Workshop von heute Vormittag auf!


- 1 Commands
- 2 Environments
- 3 Fonts
 - Font Styles
 - Andere Fonts
- 4 Packages
- 5 **M↓** Markdown & pandoc

 Material



<https://phictional.de/tutor/latex/>

Commands

 live/01_commands_and_environments/

Commands

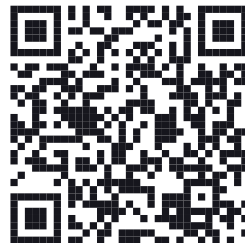
Commands ohne Argument:

`\leq` $\Rightarrow \leq$

Commands **mit** [optionalen] Argumenten:

`\sqrt{4}` $\Rightarrow \sqrt{4}$

`\sqrt[3]{4}` $\Rightarrow \sqrt[3]{4}$



<https://www.caam.rice.edu/~heinken/latex/symbols.pdf>

Definition

1 `\newcommand {name}[n][defaultFirst]{body}`

- **name:** Name des Befehls
- **n:** Anzahl von Parametern
- **defaultFirst:** Standardwert für ersten Parameter
- **body:** Hier werden Parameter mittels **#x**, $x \in \{1, \dots, n\}$ verwendet

Beispiel (`\hello`)

```
1 \newcommand{\hello}[1][Phi]{Hallo #1, wie geht's so?}  
2  
3 \hello \\  
4 \hello[Jules]
```

Beispiel (`\hello`)

```
1 \newcommand{\hello}[1][Phi]{Hallo #1, wie geht's so?}  
2  
3 \hello \\  
4 \hello[Jules]
```

Ausgabe

Hallo Phi, wie geht's so?
Hallo Jules, wie geht's so?

Definition

99% identisch zu `\newcommand`⁶

```
1 \renewcommand {name}[n][defaultFirst]{body}
```

- **name**: Name des Befehls
- **n**: Anzahl von Parametern
- **defaultFirst**: Standartwert für ersten Parameter
- **body**: Hier werden Parameter mittels **#x**, $x \in \{1, \dots, n\}$ verwendet

Beispiel (`\square`)

```
1  $\Box\square$  
2  
3  \renewcommand{\square}{^2}  
4  $\Box\square$
```

Beispiel (`\square`)


```
1  $\Box\square$  
2  
3  \renewcommand{\square}{^2}  
4  $\Box\square$
```

Ausgabe

$\square\square$

\square^2

Environments

 `live/01_commands_and_environments/`

Environments

Die bekannte Umgebung **table**:

```
1 \begin{table}
2   \begin{tabular}{l|c}
3     Workshop & Teilnehmer \\
4     \hline
5     Bash      & 20 \\
6     Git       & 20 \\
7     Python   & 20 \\
8     LaTeX    & 20 \\
9     \dots
10  \end{tabular}
11  \caption{Teilnehmer}
12 \end{table}
```

Workshop	Teilnehmer
Bash	20
Git	20
Python	20
LaTeX	20
...	

Tabelle: Teilnehmer

Definition

1 `\newenvironment {name}[n][defaultFirst]{before}{after}`

- **name**: Name des Befehls
- **n**: Anzahl von Parametern
- **defaultFirst**: Standartwert für ersten Parameter
- **before**: Der Code der *vor* dem Inhalt kommt
- **after**: Der Code der *nach* dem Inhalt kommt

Zwischen **before** und **after** landet der Code, der zwischen `\begin {name}` und `\end {name}` geschrieben wird.

Definition

```
1  \newcounter{name}           % Zähler anlegen
2  \newenvironment{name}[n][defaultFirst]{
3      \refstepcounter{name} % Referenzieren zum Inkrementieren
4      \thename               % Zahl abrufen
5      before
6  }{after}
```

Funktioniert so auch in normalen **Commands**. Siehe `\section`.

Beispiel (`\newenvironment`)

```
1 \newenvironment{para}[1]{
2     \begin{minipage}{1.5em}
3         \rotatebox{90}{\textsc{#1}}
4     \end{minipage}\begin{minipage}{\linewidth}
5 }{
6     \end{minipage}\smallskip
7 }
```

- minipage ermöglicht es horizontal "Boxen" anzulegen.
- `\rotatebox[Winkel]{Inhalt}` dreht den Inhalt um den angegebenen Winkel.

Beispiel (`\newenvironment`)

```
1 \newenvironment{para}[1]{  
2     \begin{minipage}{1.5em}  
3         \rotatebox{90}{\textsc{#1}}  
4     \end{minipage}\begin{minipage}{\linewidth}  
5 }{  
6     \end{minipage}\smallskip  
7 }
```

Ausgabe

TEST

Hier steht ein bisschen Fülltext.

Beispiel (`\newenvironment`)

```
1 \newcounter{joke}
2 \newenvironment{joke}{
3     \refstepcounter{joke}
4     \noindent
5     \colorbox{gray!50!white}{
6         \textbf{Witz~\thejoke}
7     } \!\! [.5em]
8 }{\medskip}
```

Ausgabe

Witz 1

Wie nennt man eine Zauberin in
der Wüste?

Sand Witch!

Definition

```
\renewenvironment {name}[n][defaultFirst]{before}{after}
```

Auch hier gibt es wieder ein `\renewenvironment`. Es verhält sich genauso wie `\newenvironment` und überschreibt die vorher definierte/importierte Umgebung.

Aufgaben

Aufgaben

Es gibt den Befehl `\mathbb` {} für den Mathe-Modus. `\mathbb {R} \Rightarrow \mathbb{R}`

Aufgabe 1: Zahlenräume

Schreibe

$$\mathbb{N} \subset \mathbb{Z} \subset \mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$$

mittels Custom Commands nach dem Muster

$$\texttt{\backslash bbR} \Rightarrow \mathbb{R} \quad \text{oder} \quad \texttt{\backslash bb} \{R\} \Rightarrow \mathbb{R}$$

Aufgabe 2: Aufgaben-Section mit Punkten

Ziemlich was der Titel sagt. Entwerfe einen Befehl `\aufgabe`, der **zwei** Argumente erwartet:

- Nummer der Aufgabe
- Punkte für die Aufgabe

Hilfreiche Befehle sind `\section *` und `\hfill`.

Optional sind `\small` und `\textcolor {color}{text}`.

Aufgabe 1

(5 Punkte)

Aufgabe 3: Lösungs-Umgebung

Entwerfe eine Umgebung `loesung`, die den Text "**Lösung:**" als Präfix besitzt.

Der Präfix ist der einzige und optionale Parameter.


Hilfreiche Befehle:

- `\medskip` vertikaler Lücke
- `\noindent` keine Einrückung
- `\textbf {}` Fett geschrieben

Lösung:

So sieht die `loesung`-Umgebung in Verwendung aus.

Fonts

 live/02_fonts/

W_iR S_c *H* *r* *e* *B* *E* *n* ei *n* *ε* *N*
E_r *P* *R* *E* *G* *G* *e* *R* **b** *R* *I* *e* *F* *m* *I* *t* T *e* *x*

Abbildung: Mit Font Styles und Sizes freidrehen

<code>\tiny</code>	<code>\scriptsize</code>	<code>\footnotesize</code>	<code>\small</code>	<code>\normalsize</code>
Text	Text	Text	Text	Text
<code>\large</code>	<code>\Large</code>	<code>\LARGE</code>	<code>\huge</code>	<code>\Huge</code>
Text	Text	Text	Text	Text

Tabelle: Font Sizes in Tex

Beispiel (Verwendung von `\tiny`)

```
1 {\tiny So werden die Schriftgrößen verwendet!}
```

So werden die Schriftgrößen verwendet!

Mathe Text Style		Text Style	
\mathcal{A}	\mathcal{A}	Text	Text
\mathbb{A}	\mathbb{A}	<i>Text</i>	<i>Text</i>
\mathfrak{A}	\mathfrak{A}	Text	Text
A	A	Text	Text
\mathbf{A}	\mathbf{A}	Text	Text

Tabelle: Font Styles für Text im Mathe-Modus, sowie den Text-Modus

\uppercase {LaTeX}	\lowercase {LaTeX}
LATEX	latex

Tabelle: Alles groß-/kleinschreiben ist kein Problem

“ Schriftarten, die nicht „häufig“ vorkommen,
sind verdammt nervig.

MiKTeX, TeX Live werden mit einer Auswahl von Schriftarten ausgeliefert.

The L^AT_EX Font Catalogue

[FRONT PAGE] [SERIF FONTS] [SERIF FONTS, SUB-CATEGORISED] [SANS SERIF FONTS] [TYPEWRITER FONTS] [CALLIGRAPHICAL AND HANDWRITTEN FONTS] [UNCIAL FONTS] [BLACKLETTER FONTS] [OTHER FONTS]
[FONTS WITH MATH SUPPORT] [FONTS WITH OPENTYPE OR TRUETYPE SUPPORT] [ALL FONTS, BY CATEGORY] [ALL FONTS, ALPHABETICALLY] [ABOUT THE L^AT_EX FONT CATALOGUE] [PACKAGES THAT PROVIDE MATH SUPPORT]

Finding the right font

Fonts with math support

Serif Fonts

Sans Serif Fonts

Typewriter Fonts

Calligraphical and Handwritten Fonts

Uncial Fonts

Blackletter Fonts

Other Fonts

Fonts in upper case only

Decorative Initials

Other (mostly decorative) Fonts



[https://tug.org/
FontCatalogue/](https://tug.org/FontCatalogue/)

Definition

- 1 `\usepackage[T1]{fontenc}`
- 2 `\usepackage{fontname}`

Der Standard-Weg um **dokumentenweit** Fonts einzustellen (unter *pdf(La)TeX* und *MiKTeX*).

Xe(La)TeX/Lua(La)TeX verwenden das Package `fontspec` und sind iA. besser im Umgang mit Schriftarten.



<https://tug.org/FontCatalogue/>

Beispiel (Comic Sans-ish)


```
1 \usepackage[T1]{fontenc}
2 \usepackage[default]{comicneue}
```

Comic Neue noch mit **Horrorfärbung™**

Abbildung: Die Schriftart in Verwendung



[https://tug.org/
FontCatalogue/](https://tug.org/FontCatalogue/)

In **Word**  kann man einzelne Textschnipsel in verschiedenen Schriftarten schreiben. Geht das in TeX auch? **Ja, aber umständlicher.**

Beispiel (Inline)

- 1 Hier steht was.
- 2 `{\fontfamily{ComicNeue-TLF}\selectfont Hier Comic Neue.}`
- 3 Hier wieder nicht.

Hier steht was. Hier Comic Neue. Hier wieder nicht.

Abbildung: Umständliches Inline

Definition

```
1 \newcommand{\comicneue}{\fontfamily{ComicNeue-TLF}\selectfont}  
2 \DeclareTextFontCommand{\textcn}{\comicneue}
```

`\textcn` wird dann so verwendet wie `\textbf`, `\textrm`, `\textsc`, ...

Wie kommen wir überhaupt auf **ComicNeue-TLF**?

- 1 Gewünschte Schriftart als Standart setzen (`\usepackage{ ... }`)
- 2 Im Text `\familydefault`
- 3 Dieser String ist die gesuchte fontfamily

Definition

```
1 \newcommand{\comicneue}{\fontfamily{ComicNeue-TLF}\selectfont}  
2 \DeclareTextFontCommand{\textcn}{\comicneue}
```

Beispiel (\comicneue / \textcn)

```
1 {\comicneue Hier steht etwas auf diese Weise} \\  
2 \textcn{Hier steht etwas auf die andere Weise}
```

Hier steht etwas auf diese Weise

Hier steht etwas auf die andere Weise

Abbildung: Nützlicheres Inline

Aufgaben

Aufgabe 4: Neue Monospaced Schriftart

Suche dir im **TeXFont Catalogue**¹ eine neue Typewriter-Font aus, die die **Computer Modern** Monospace Schriftart ersetzt.
(Ich empfehle **Fira Mono** oder **DejaVu Sans Mono**)

Standart	Fira Mono	DejaVu Sans Mono
Test123	Test123	Test123

Tabelle: Vergleich der Schriftarten

¹<https://tug.org/FontCatalogue/>

Aufgabe 5: Awesome Fonts 😊

Neben den normalen Schriftarten gibts auch andere witzige Dinge.

Deine Aufgabe ist es folgende Sequenz von Symbolen anzugeben:




Abbildung: Die Symbolsequenz

Startpunkt



<https://www.ctan.org/pkg/fontawesome5>

Packages

 live/03_packages/

Warum der Spaß?

- Die Präambel läuft über / ist zu lang.
- Viele Dokumente mit der selben / ähnlichen Präambel (Übungsblätter)
- Befehle mit anderen teilen

Warum der Spaß?

- Die Präambel läuft über / ist zu lang.
- Viele Dokumente mit der selben / ähnlichen Präambel (Übungsblätter)
- Befehle mit anderen teilen

Idee

Präambel in eine `settings.tex` packen.

`\input {settings.tex}` anstatt der alten Präambel.

`\input` vs `\include`

1 `\input` {filename}

- Importiert filename.tex
- Als ob Code in aufrufender Datei stehen würde
- Befehle verwendbar

1 `\include` {filename}

- Importiert filename.tex
- In Kompilation eigene Datei
- Befehle nicht verwendbar
- nützlich bei großen Projekten mit Teildokumenten

Eine Sammlung von Befehlen und Umgebungen, die in anderen Dokumenten eingesetzt werden kann.

Nicht viel anders zur `settings.tex`-Lösung. Die Unterschiede sind:

1 Dateiendung `.sty`

2 Einbindung mittels `\usepackage{packagename}`

3 Angabe des "Headers"

1 `\NeedsTeXFormat{LaTeX2e}`

2 `\ProvidesPackage{packagename}[YYYY/MM/dd package description]`

4 `\usepackage` → `\RequirePackage`

Eine Sammlung von Befehlen und Umgebungen, die in anderen Dokumenten eingesetzt werden kann.

Nicht viel anders zur `settings.tex`-Lösung. Besonderheiten:

- `\newcommand` kann weiter verwendet werden
- `\renewcommand` kann weiter verwendet werden
- `\providecommand` definiert Befehl, falls nicht schon vorher vorhanden
- `\CheckCommand` genau wie `\newcommand`, falls Befehl bereits vorhanden und anders als in `\CheckCommand` definiert gibts **Error**

Aufgaben

Aufgabe 6: Alle Befehle sind schon da 🎵

Sind sie das? Funktionieren alle Befehle und Umgebungen in `uebungsblatt.sty` ohne Probleme in einer anderen Datei?

Kommentiere `\EXERCISES` in `main.tex` aus.

Sorge dafür, dass das keinen Fehler wirft. Selbiges für die anderen Übungsblatt spezifischen Befehle.

Zusatzaufgabe 7:

`\documentclass {uebungsblatt}`

Neben Packages gibt es noch Klassen. Benutzt und benötigt in jedem Dokument:

`\documentclass [a4paper]{scratcl}`

Schreibe `uebungsblatt.sty` zu `uebungsblatt.cls` um. Richte dich nach der **Overleaf Dokumentation**. Der Artikel zum Unterschied von Package und Class ist auch empfohlen.



https://www.overleaf.com/learn/latex/Writing_your_own_class

Markdown & pandoc

 `live/04_markdown_and_pandoc/`

 `https://pandoc.org`

 `https://github.github.com/gfm/`

Was ist Markdown? Das ist Markdown!

- Markdown ist eine `_simple_` Notation für Text, Notizen, usw.
- Wird von Git-hostern unterstützt (``.md``, ``.markdown``)
- Ist als "Source-Code" menschenlesbar
- [**Spezifikation**](<https://github.github.com/gfm/>)

Definition (pandoc --help)

```
pandoc [OPTIONS] [FILES]
-f FORMAT, -r FORMAT  --from=FORMAT, --read=FORMAT
-t FORMAT, -w FORMAT  --to=FORMAT, --write=FORMAT
-o FILE               --output=FILE
```

Beispiel (→)

```
>>> pandoc -f markdown -t html -o info.html info.md
```

Beispiel (→)

```
>>> pandoc -f markdown -t pdf \
>      --pdf-engine=xelatex \
>      -o default.pdf \
>      shownotes.md
```

LaTeX Advanced

Eigene Environments bis hin zu eigenen Packages

Alexander **Phi** Goetz info@phictional.de

14.04.2022

Inhalt

- Eigene Commands `\newcommand`, `\renewcommand`
 - `\mathbb{R}` \Rightarrow `R` \Leftarrow `\bbR`
- Eigene Umgebungen `\newenvironment`, `\renewenvironment`
 - `\textsc`, `\texttt`, ...
 - LaTeX Font Catalogue with Comic Neue
- Eigene Packages
- Markdown & pandoc

Einleitung

Im Verlauf des Workshops wird an der typischen Tübinger-Info TeX-Vorlage herumeditiert.

Commands

Commands ohne Argument:

`\leq` \Rightarrow \leq

Commands **mit** *[optionalen]* Argumenten:



`\sqrt{4}` \Rightarrow $\sqrt{4}$

`\sqrt[3]{4}` \Rightarrow $\sqrt[3]{4}$



Zum Benutzen gibt es ein **LATEX Mathematical Symbols** und zum verstehen der Befehle `\newcommand` und `\renewcommand` gibts die Overleaf Dokumentation¹.

¹Overleaf Dokumentation: Commands





Beispiel ( →  )

```
>>> pandoc -f markdown -t beamer -o slides.pdf slides.md
```

Beispiel ( →  HTML)


```
>>> pandoc \  
> -f markdown -t slidy \  
> -s --mathjax \  
> -o slidy.html slides.md
```

Beispiel ( →  HTML)

```
>>> pandoc \  
> -f markdown -t revealjs \  
> -s --mathjax \  
> -o revealjs.html slides.md
```

Danke fürs mitmachen

Wenn ihr uns mitteilen wollt, wie es euch gefallen hat:

 <https://phictional.de/tutor/feedback>



- **Material:** <https://phictional.de/tutor/latex/>