

Einführung in das Textsatzsystem \LaTeX

Komplexe Makros und Befehle

Maximilian Jalea

`max@mathphys.stura.uni-heidelberg.de`

Vorläufige Version

26. Januar 2017

1 Verschiedenes

Poster

Vorlesungsmitschriften

2 Makros in $\text{\LaTeX} 2_{\epsilon}$

newcommand, newenvironment & Co

def und let

Naming Conventions

3 $\text{\LaTeX} 3$

Makros in $\text{\LaTeX} 3$

4 Lua \LaTeX

Teil I

Verschiedenes

∃ diverse Klassen für Satz von (wissenschaftlichen) Postern:
`a0poster`, `sciposter`, `tikzposter`

In Overleaf ausprobieren:



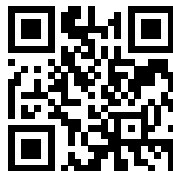
<http://polr.me/tex1201>

∃ diverse Klassen für Satz von (wissenschaftlichen) Postern:
[a0poster](#), [sciposter](#), [tikzposter](#)

Empfehlung: [tikzposter](#)

Nutzt TikZ um Objekte (Blocks, etc.) auf Poster zu platzieren.
Bedienung vergleichbar mit [beamer](#).

In Overleaf ausprobieren:



<http://polr.me/tex1201>

- in Vorlesungen oder Übungen mit \TeX manchmal nützlich
- entweder extrem hohe Tippgeschwindigkeit nötig
- oder durchdachte Befehlsdefinitionen
- *wichtig*: alle strukturelle Information muss vorhanden sein!
(auch, wenn es nicht gut aussieht)

- häufig nur stichpunktartiges Aufschreiben

⇒ `\obeylines`

- Aufzählungen abkürzen, z. B. mittels `\let•\item`
- ...

Teil II

L^AT_EX 2_ε

Zur Definition eigener Befehle in \LaTeX verfügbar:

`\newcommand`, `\renewcommand`, `\newenvironment`

Zur Definition eigener Befehle in \LaTeX verfügbar:

$\backslash\text{newcommand}$, $\backslash\text{renewcommand}$, $\backslash\text{newenvironment}$

```
 $\backslash(\text{re})\text{newcommand}\{\langle\text{Befehlsname}\rangle\}$ 
```

$[\langle\text{Anzahl der Argumente}\rangle]$

$[\langle\text{Default für erstes (optionales) Argument}\rangle]$

$\{\langle\text{Befehlsdefinition}\rangle\}$

```
 $\backslash\text{newenvironment}\{\langle\text{Umgebungsname}\rangle\}$ 
```

$[\langle\text{Anzahl der Argumente}\rangle]$

$[\langle\text{Default für erstes (optionales) Argument}\rangle]$

$\{\langle\text{Definition Code vor Umgebung}\rangle\}$

$\{\langle\text{Definition Code nach Umgebung}\rangle\}$

Zur Definition eigener Befehle in \LaTeX verfügbar:

$\backslash\text{newcommand}$, $\backslash\text{renewcommand}$, $\backslash\text{newenvironment}$

```
 $\backslash(\text{re})\text{newcommand}\{\langle\text{Befehlsname}\rangle\}$ 
```

$[\langle\text{Anzahl der Argumente}\rangle]$

$[\langle\text{Default für erstes (optionales) Argument}\rangle]$

$\{\langle\text{Befehlsdefinition}\rangle\}$

```
 $\backslash\text{newenvironment}\{\langle\text{Umgebungsname}\rangle\}$ 
```

$[\langle\text{Anzahl der Argumente}\rangle]$

$[\langle\text{Default für erstes (optionales) Argument}\rangle]$

$\{\langle\text{Definition Code vor Umgebung}\rangle\}$

$\{\langle\text{Definition Code nach Umgebung}\rangle\}$

Varianten mit Stern: $\backslash\text{newcommand}^*$ für zusätzliche Fehler-Checks, falls Argumente *keine* Umbrüche/Leerzeilen enthalten dürfen

T_EX bietet die Primitiven `\def` und `\let`

T_EX bietet die Primitiven `\def` und `\let`

`\def`*<Befehlsname>**<Argument(e)>*{*<Befehlsdefinition>*}

```
\def\mymakro#1#2{Makro mit zwei Argumenten #1 und #2}
```

T_EX bietet die Primitiven `\def` und `\let`

`\def`*<Befehlsname>**<Argument(e)>*{*<Befehlsdefinition>*}

```
\def\mymakro#1#2{Makro mit zwei Argumenten #1 und #2}
```

`\let`*<neuer Befehlsname>**<alter Befehlsname>*

```
\let\newmakro\oldmakro
```

- generiert `\newmakro` mit exakt den selben Eigenschaften wie `\oldmakro`
- wenn sich `\oldmakro` ändert, bleibt `\newmakro` erhalten

- `\def` und `\let` auch in L^AT_EX verfügbar
- High-Level Befehle wie `\newcommand` sind meist vorzuziehen
- `\let` manchmal praktisch
- nur benutzen, wenn man weiß was man tut

Naming Conventions

- `lowercase` Endnutzer-Befehle auf Dokumenten-Level
(braucht man ständig)
- `MixedCase` Befehle für spezielle Funktionen in Paketen oder Klassen
(braucht man selten)
- `with@sign` interne Befehle in Paketen oder im \LaTeX -Kernel
(braucht man *nie*)

Naming Conventions

- `lowercase` Endnutzer-Befehle auf Dokumenten-Level
(braucht man ständig)
- `MixedCase` Befehle für spezielle Funktionen in Paketen oder Klassen
(braucht man selten)
- `with@sign` interne Befehle in Paketen oder im \LaTeX -Kernel
(braucht man *nie*)

spezieller Schutzmechanismus

@-Zeichen hat anderen category code als normale Buchstaben, Befehle mit @ werden daher ignoriert.

Ausschalten: `\makeatletter`

wieder Einschalten: `\makeatother`

Naming Conventions

- `lowercase` Endnutzer-Befehle auf Dokumenten-Level
(braucht man ständig)
- `MixedCase` Befehle für spezielle Funktionen in Paketen oder Klassen
(braucht man selten)
- `with@sign` interne Befehle in Paketen oder im \LaTeX -Kernel
(braucht man *nie*)

spezieller Schutzmechanismus

@-Zeichen hat anderen category code als normale Buchstaben, Befehle mit @ werden daher ignoriert.

Ausschalten: `\makeatletter`

wieder Einschalten: `\makeatother`

\TeX -Primitiven sind – aus historischen Gründen – auch lowercase

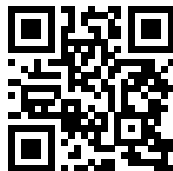
Teil III

L^AT_EX3

- Mit \LaTeX 3 wird alles besser:
 - Konsequente Unterscheidung zwischen Nutzer-, Design- und Programmierenebene
 - Namespaces für Pakete
 - sehr bequeme und flexible Befehlsdefinitionen
- \LaTeX 3-Syntax schon jetzt nutzbar:
 - Paket `expl3` für Entwickler
 - Paket `xparse` für Endnutzer



In Overleaf ausprobieren:



<http://polr.me/tex130>

Mit Paket `xparse` verfügbar:

`\NewDocumentCommand`, `\RenewDocumentCommand`,
`\NewDocumentEnvironment`, ...

```
\NewDocumentCommand{\Befehlsname}  
  {\Argumentstruktur}  
  {\Definition}
```

\Argumentstruktur beschreibt wie viele und welche Argumente der Befehl annimmt (sozusagen die Signatur)

mandatorische Argumente

m	klassisches mandatorisches Argument	$\{\langle \dots \rangle\}$
l	liest alles vor der nächsten Klammer	$\langle \dots \rangle \{$
r	$\langle t1 \rangle \langle t2 \rangle$ alles zwischen $\langle t1 \rangle$ und $\langle t2 \rangle$ z. B. $r\langle \rangle$	$\langle \langle \dots \rangle \rangle$
u	$\{ \langle t \rangle \}$ liest alles bis $\langle t \rangle$ z. B. $u\{\$ \&\}$	$\langle \dots \rangle \$ \&$
v	Verbatim-Input	$ \langle \dots \rangle $
	Eingabe wird nicht interpretiert	$\{ \langle \dots \rangle \}$

```
\NewDocumentCommand{\mycommand}{ m l m r ^ ° }  
  { (#1, #2, #3, #4) }  
\mycommand{eins}zwei{drei}^vier°
```

optionale Argumente

o	klassisches optionales Argument	[$\langle \dots \rangle$]
O{ $\langle df \rangle$ }	wie o mit Default-Wert z. B. O{default}	[$\langle \dots \rangle$]
d $\langle t1 \rangle \langle t2 \rangle$	alles zwischen $\langle t1 \rangle$ und $\langle t2 \rangle$ z. B. d:+	: $\langle \dots \rangle$ +
D $\langle t1 \rangle \langle t2 \rangle$ { $\langle df \rangle$ }	wie d mit Default-Wert z. B. d(){bla}	($\langle \dots \rangle$)
s	Stern, setzt \BooleanTrue, falls vorhanden	*

```
\NewDocumentCommand{\mycommand}  
{ d<| O{zwei} s D|>{vier} }  
{ (#1,#2,#4) \IfBooleanT{#3}{:-)} }  
\mycommand<eins|[2]*
```

Argument-Modifizier

+ \langle Arg-Kürzel \rangle erlaubt Eingabe von Umbrüchen innerhalb eines Arguments

z. B. +m

> $\{ \langle$ Prozessor $\rangle \}$ Argumente vor dem Auslesen bearbeiten

z. B. > { \ReverseBoolean} m

> { \TrimSpaces} o

```
\NewDocumentCommand{\mycommand}
{ >{ \ReverseBoolean} s o +m }
{ \IfBooleanTF{#1}{kein stern}{stern} #2 #3 }
\mycommand*{Text mit\\Umbruch}
```


Umgebungen

```
\NewDocumentEnvironment{⟨Umgebungsname⟩}  
  {⟨Argumentstruktur⟩}  
  {⟨Definition Code vor Umgebung⟩}  
  {⟨Definition Code nach Umgebung⟩}
```

```
\newDocumentEnvironment{myquote}{ o }  
  {\begin{quote}\sffamily\itshape}  
  {\end{quote}\IfNoValueF{#1}{Quelle:#1}}  
  
\begin{myquote}[Internet]  
  Bla bla, Chemtrails, Lügenpresse ...  
\end{myquote}
```

Erweiterte L^AT_EX3-Funktionalität für Entwickler mit [expl3](#) verfügbar

`\ExplSyntaxOn`

⟨Code⟩

`\ExplSyntaxOff`

Schaltet neue Syntax ein und aus

Teil IV

Lua^AT_EX

Innerhalb von TeX Lua-Code eingeben:

```
\directlua{⟨Lua-Code⟩}
```

Innerhalb von Lua-Code etwas an TeX ausgeben:

```
tex.print(⟨TeX-Ausgabe⟩)
```

Nutzung von Lua mit Lua \LaTeX

Innerhalb von \TeX Lua-Code eingeben:

```
\directlua{\Lua-Code}
```

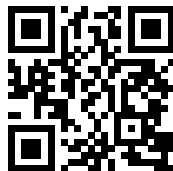
Innerhalb von Lua-Code etwas an \TeX ausgeben:

```
tex.print(\TeX-Ausgabe)
```

```
$\pi = \directlua{  
  tex.sprint(math.pi)  
}$
```

$$\pi = 3.1415926535898$$

In Overleaf ausprobieren:



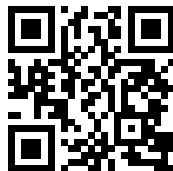
<http://polr.me/tex1303>

- `\directlua` macht manchmal Probleme
 - bei Umbrüchen
 - bei Lua-Kommentaren `---`
 - bei Sonderzeichen `_ ^ & $ \{ \} %`
- Paket `luacode` behebt viele dieser Probleme.:

```
\begin{luacode*}
  ⟨Lua-Code⟩
\end{luacode*}
```

- in Variante mit Stern sind keine \TeX -Befehle möglich
- in Variante ohne Stern werden \TeX -Makros expandiert

In Overleaf ausprobieren:



<http://polr.me/tex1303>



The \LaTeX 3 Project.

„The xparse package Document command parser“.

`texdoc xparse`



The \LaTeX 3 Project.

„The \LaTeX 3 Interfaces“.

`texdoc interface3`



The \LaTeX 3 Project.

„The expl3 package and \LaTeX 3 programming“.

`texdoc expl3`



Manuel Pégourié-Gonnard.

„Eine Einführung in Lua \LaTeX “.

`texdoc luatex-doc-de`



Manuel Pégourié-Gonnard.

„The luacode package“.

`texdoc luacode`