

Einführung in das Textsatzsystem \LaTeX

Komplexe Makros und Befehle

Maximilian Jalea
latexkurs@mnt1.de

21. Januar 2018

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 $\text{\LaTeX} 3$

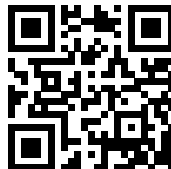
4 Lua \LaTeX

Teil I

Verschiedenes

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

In Overleaf ausprobieren:



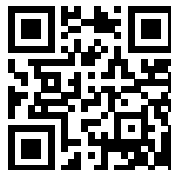
<http://qn3.de/tex1301>

∃ 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://qn3.de/tex1301>

- in Vorlesungen oder Übungen mit \TeX en 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 L^AT_EX verfügbar:

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

```
\(re)newcommand{⟨Befehlsname⟩}
  [⟨Anzahl der Argumente⟩]
  [⟨Default für erstes (optionales) Argument⟩]
  {⟨Befehlsdefinition⟩}
```

```
\newenvironment{⟨Umgebungsname⟩}
  [⟨Anzahl der Argumente⟩]
  [⟨Default für erstes (optionales) Argument⟩]
  {⟨Definition Code vor Umgebung⟩}
  {⟨Definition Code nach Umgebung⟩}
```

Zur Definition eigener Befehle in L^AT_EX verfügbar:

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

```
\(re)newcommand{⟨Befehlsname⟩}
  [⟨Anzahl der Argumente⟩]
  [⟨Default für erstes (optionales) Argument⟩]
  {⟨Befehlsdefinition⟩}
```

```
\newenvironment{⟨Umgebungsname⟩}
  [⟨Anzahl der Argumente⟩]
  [⟨Default für erstes (optionales) Argument⟩]
  {⟨Definition Code vor Umgebung⟩}
  {⟨Definition Code nach Umgebung⟩}
```

Varianten mit Stern: `\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

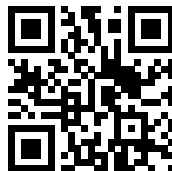
Teil III

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://qn3.de/tex1302>

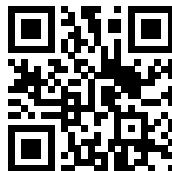
Mit Paket `xparse` verfügbar:

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

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

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

In Overleaf ausprobieren:



<http://qn3.de/tex1302>

mandatorische Argumente

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

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

optionale Argumente

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

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

Argument-Modifizier

+{*Arg-Kürzel*} erlaubt Eingabe von Umbrüchen innerhalb eines Arguments

z. B. +m

>{*Prozessor*} 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}
```



```
\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 \LaTeX 3-Funktionalität für Entwickler mit `expl3` verfügbar

```
\ExplSyntaxOn
```

```
  \langle Code \rangle
```

```
\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^AT_EX

Innerhalb von T_EX Lua-Code eingeben:

```
\directlua{⟨Lua-Code⟩}
```

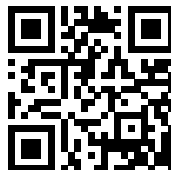
Innerhalb von Lua-Code etwas an T_EX ausgeben:

```
tex.print(⟨TeX-Ausgabe⟩)
```

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

$\pi = 3.1415926535898$

In Overleaf ausprobieren:



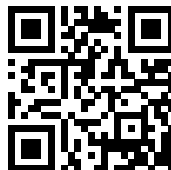
<http://qn3.de/tex1303>

- `\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 T_EX-Befehle möglich
- in Variante ohne Stern werden T_EX-Makros expandiert


In Overleaf ausprobieren:





<http://qn3.de/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`