Einführung in das Textsatzsystem LaTeX

Komplexe Makros und Befehle

Moritz Brinkmann
moritz.brinkmann@iwr.uni-heidelberg.de

Vorläufige Version

26. Januar 2017

Übersicht

Teil I

Verschiedenes

Poster

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

In Overleaf ausprobieren:

http://qn3.de/tex1301

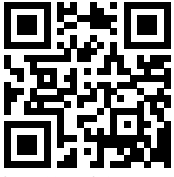
Poster

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

Empfehlung: `tikzposter`

Nutzt Ti*k*Z um Objekte (Blocks, etc.) auf Poster zu platzieren. Bedienung vergleichar mit `beamer`.

In Overleaf ausprobieren:



http://qn3.de/tex1301

Mitschreiben

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

Mitschreiben

- häufig nur stichpunktartiges Aufschreiben

$\Longrightarrow$ `\obeylines`

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

Teil II

LaTeX 2_ε

Makros in LaTeX

Zur Definition eigener Befehle in LaTeX verfügbar:
`\newcommand`, `\renewcommand`, `\newenvironment`

Makros in LaTeX

Zur Definition eigener Befehle in LaTeX 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⟩}
```

Makros in LaTeX

Zur Definition eigener Befehle in LaTeX 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

Makros in TEX

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

Makros in TEX

TEX bietet die Primitiven \def und \let

\def⟨*Befehlsname*⟩⟨*Argument(e)*⟩{⟨*Befehlsdefinition*⟩}

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

Makros in TEX

TEX 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

Makros in TEX

- `\def` und `\let` auch in LATEX 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

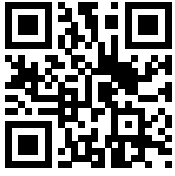
Teil III

LaTeX3

Makros in LaTeX3

- Mit LaTeX3 wird alles besser:
  - Konsequente Unterscheidung zwischen Nutzer-, Design- und Programmierebene
  - Namespaces für Pakete
  - sehr bequeme und flexible Befehlsdefinitionen
- LaTeX3-Syntax schon jetzt nutzbar:
  - Paket `expl3` für Entwickler
  - Paket `xparse` für Endnutzer

In Overleaf ausprobieren:



http://qn3.de/tex1302

Makros in LaTeX3

Mit Paket xparse verfügbar:
`\NewDocumentCommand`, `\RenewDocumentCommand`, `\NewDocumentEnvironment`, ...

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

⟨*Argumentstruktur*⟩ beschreibt wie viele und welche Argumente der Befehl annimmt (sozusagen die Signatur)

In Overleaf ausprobieren:

http://qn3.de/tex1302

Argumentstruktur

mandatorische Argumente

`m`	klassisches mandatorisches Argument	{⟨...⟩}
`l`	liest alles vor der nächsten Klammer	⟨...⟩{
`r`⟨*t1*⟩⟨*t2*⟩	alles zwischen ⟨*t1*⟩ und ⟨*t2*⟩ z. B. r<>	<⟨...⟩>
`u{`⟨*t*⟩`}`	liest alles bis ⟨*t*⟩ z. B. u{§&}	⟨...⟩§&
`v`	Verbatim-Input	|⟨...⟩|
	Eingabe wird nicht interpretiert	{⟨...⟩}

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

Argumentstruktur

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-Modifier

`+`⟨*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}
```

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}
```

expl3

Erweiterte LaTeX3-Funktionalität für Entwickler mit `expl3` verfügbar

```
\ExplSyntaxOn
  ⟨Code⟩
\ExplSyntaxOff
```

Schaltet neue Syntax ein und aus

Teil IV

LuaLaTeX

Nutzung von LuaLaTeX

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 LuaLaTeX

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://qn3.de/tex1303

Nutzung von LuaLaTeX

- `\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://qn3.de/tex1303

Weiterführende Literatur


The LaTeX3 Project.
„The xparse package Document command parser“.
`texdoc xparse`


The LaTeX3 Project.
„The LaTeX3 Interfaces“.
`texdoc interface3`


The LaTeX3 Project.
„The expl3 package and LaTeX3 programming“.
`texdoc expl3`


Manuel Pégourié-Gonnard.
„Eine Einführung in LuaLaTeX“.
`texdoc lualatex-doc-de`


Manuel Pégourié-Gonnard.
„The luacode package“.
`texdoc luacode`