

Programmierung mit \LaTeX ... und anderen Programmiersprachen

Wolfgang Dautermann

FH JOANNEUM

Chemnitzer Linxstage 2015

- 1 Programmierung in \LaTeX
 - nützliche Zusatzpakete
 - Named Parameter
 - Kontrollstrukturen
 - Mathematik
 - String-Manipulationen
- 2 Lua \LaTeX
- 3 Externe Programme
- 4 Python \TeX
- 5 Perl \TeX
- 6 weitere Möglichkeiten

(Eigene) Kommandos (um)definieren

... ist vermutlich bekannt?

```
\newcommand und \renewcommand
```

```
\newcommand{\meinbefehl}{Inhalt}
```

```
\renewcommand{\meinbefehl}{neuer Inhalt}
```

z.B.:

```
\newcommand{\CLT}{Chemnitzer Linxstage 2015}
```

```
\CLT
```

Chemnitzer Linxstage 2015

Kommandos mit Argumenten / „Funktionen“

1-9 notwendige Argumente

```
\renewcommand{\meinbefehl}[narg]{Inhalt #1 #2 ... #9}
```

narg... number of arguments

z.B.:

```
\newcommand{\fahrt}[2]{Ich fahre von #1 nach #2.}  
\fahrt{Graz}{Chemnitz}
```

Ich fahre von Graz nach Chemnitz.

Kommandos mit optionalem Argument

EIN optionales Argument ist möglich

Defaultwert für #1 angeben.

```
\(re)newcommand{\meinbefehl}[narg][Defaultwert #1]{...}
```

z.B.:

```
\renewcommand{\fahrt}[3][Zug]{#1fahrt von #2 nach #3.}  
\fahrt{Graz}{Chemnitz} \\  
\fahrt[Auto]{Graz}{Chemnitz}
```

Zugfahrt von Graz nach Chemnitz.

Autofahrt von Graz nach Chemnitz.

Eigene Umgebungen

`\newenvironment`, `\renewenvironment`

Analog zu `\(re)newcommand`:

```
\newenvironment{myenv}{<startbefehle>}{<endbefehle>}
```

stellt

```
\begin{myenv} ... \end{myenv}
```

zur Verfügung.

Argumente:

```
\newenvironment{myenv}[Anzahl][OptionalArg]{start}{end}
```

Variablen

Stringvariablen

... wurden grad behandelt. Makros.

Stringvariablen

```
% $mystring = "Abc";
```

```
\newcommand{\mystring}{Abc}
```

... und beim nächsten mal \renewcommand:

```
\renewcommand{\mystring}{Def} % $mystring = "Def";
```

Variablen

„Integervariablen“ / Counter

Counter

```
\newcounter{MeinCounter}      % Deklaration  
  
\setcounter{MeinCounter}{n}  % $i=n  
  
\stepcounter{MeinCounter}    % $i++  
  
\addtocounter{MeinCounter}{n} % $i=$i+n (auch negativ)  
  
\value{MeinCounter} % Wert (zum Rechnen, nicht zur Ausgabe)
```

Etliche Counter sind standardmässig definiert:

page, section, subsection, enumi, enumii, equation, ...

Variablen

„Integervariablen“ / Counter – Ausgabe

Counter – Ausgabe

<code>\theMeinCounter / \arabic{MeinCounter}</code>	1, 2, 3, ...
<code>\alph{MeinCounter}</code>	a, b, c, ...
<code>\Alph{MeinCounter}</code>	A, B, C, ...
<code>\roman{MeinCounter}</code>	i, ii, iii, ...
<code>\Roman{MeinCounter}</code>	I, II, III, ...

Variablen

„Float-variablen“ / Längen

Float-Variablen / Längen

```
\newlength{MeineLaenge}           % Deklaration  
  
\settolength{MeineLaenge}{f} % $i=f (mit Einheit! (mm, ...))  
  
\addtolength{MeineLaenge}{f} % $i=$i+f  
  
0.6\Meinelaenge                   % Multiplikation (0.6\textwidth)
```

Kommandos mit etlichen Argumenten

werden schnell unübersichtlich...

Beispiel-Befehlsdefinition

```
\newcommand{\meinbefehl}[9]{...}
```

```
\meinbefehl{...}{...}{...}{...}{...}{...}{...}{...}{...}
```

- An welcher Stelle muss ich jetzt XYZ eintragen?
War das Parameter 7 oder 8?
- Was war nochmal die Bedeutung von Parameter 6?

Kommandos mit etlichen Argumenten

Lösung: Key-Value-Systeme, named parameters

Bekannt von `graphicx`:

```
\includegraphics [width=XX,height=YY,angle=A,bb=...,clip=true]{bilddatei}
```

Etliche Pakete zur Lösung des Problems existieren

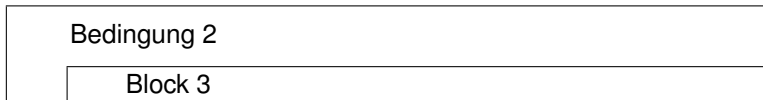
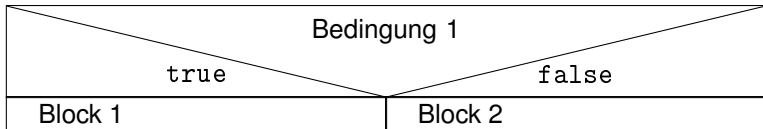
<http://www.ctan.org/topic/keyval>

Beispiel mit `keyval`:

```
\usepackage{keyval}
% keys definieren:
\define@key{<family>}{<key>}{<function>}
...
% Keys verwenden:
\newcommand{\meinbefehl}[1]{\setkeys{<family>}{#1}}
```

Live Demo...

Kontrollstrukturen



Wahrheitswerte und if-Abfragen – Paket `ifthen` (etwas älter)

Boolean-Variablen

```
\newboolean{myboolvar}           % Deklaration
\setboolean{myboolvar}{false}    % Zuweisung
\boolean{myboolvar}              % Abfrage des Werts

\ifthenelse{<test>}{<then-block>}{<else-block>}
```

Tests:

Operatoren: `\AND`, `\OR`, `\NOT`, `\(`, `\)`

Vergleiche: `x < y` / `x = y` / `x > y`

Funktionen: `\isodd{<x>}` `\isundefined{<kommando>}`

`\equal{<string1>}{<string2>}`

`\lengthtest{<dim1> < = > <dim2>}`

Weitere „Kontrollstrukturen“: `whiledo`

`whiledo` (Paket `ifthen`)

```
\newcounter{i}
\setcounter{i}{10}
\whiledo{\value{i} > -1}{
  i = \arabic{i} \\
  \addtocounter{i}{-1}
}
```

Umfangreicheres Beispiel: [99 bottles of beer](#) in \LaTeX .

TikZ / PGF: pgffor

foreach (Paket pgffor)

```
\usepackage{pgffor}
```

```
\foreach <variable> in {<liste>} {<kommandos>}
```

```
\foreach \x in {1,2,4,8,16} {\x, }
```

```
\foreach \x in {1,2,3,...,10} {\x, }
```


TkZ / PGF: „höhere“ Mathematik: pgfmath

pgfmathparse (Paket pgfmath)

```
\usepackage{pgfmath}
```

```
\pgfmathparse{ <ausdruck> }
```

```
Resultat in: \pgfmathresult
```

Etliche Operatoren und Funktionen sind vorhanden.

Live Demo...

stringstrings: String-Manipulationen

Beispiele

```
\usepackage{stringstrings}
\caseupper{Gross und Klein} = GROSS UND KLEIN
\stringlength{Chemnitzer Linxstage 2015} = 25
\whereischar{Chemnitzer Linxstage 2015}{x} = 16
\substring{Chemnitzer Linxstage 2015}{11}{16} = Linux
```

Etliche weitere Funktionen sind vorhanden – siehe Paketdokumentation...

Fragile Befehle

`\protect` hilft dagegen...

- Befehle mit „moving Arguments“
- z.B. `\footnote`
- werden zum Glück weniger...

Live Demo...

Umständlich. Gehts einfacher?

Ja. z.B. mit:

- Lisp
- Lua
- Shell & Co.
- Python
- Perl
- ...

Lisp

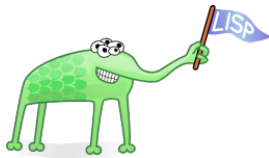
LISP on TeX — A LISP interpreter on TeX

Introduction

LISP on TeX is a LISP interpreter written only with TeX macros. It works as a style file of LaTeX.

LISP on TeX adopts static scoping, dynamic typing, and eager evaluation. We can program easily with LISP on TeX.

```
\usepackage{lisp-on-tex}  
\lispinterp{  
  LISP-CODE  
}
```

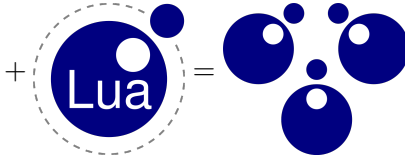


Beispiel: Fakultätsberechnung in Lisp

Viel umfangreicheres Beispiel in der Doku: Mandelbrot-Berechnung in Lisp.

Lua: Lua^AT_EX

... die Zukunft von ^AT_EX



(1)

- Übersetzen mit `luaAtex`
- Lua-code direkt einbinden mit: `\directlua`
- Achtung mit Lua-Kommentaren!
- → Lua-Files mit der Lua-Funktion `dofile()` einbinden

Lua^AT_EX-Beispiele

Live Demo

- 1 Wert von π mit Lua ausgeben
- 2 mehrere Lua-Befehle hintereinander. Problem: Lua-Kommentare (--)
- 3 Beispiel 2 mit eigener Lua-Datei
- 4 Kommandos mit Lua definieren

Live Demo...

Shell und Co

aus Sicherheitsgründen¹ normalerweise sehr eingeschränkt bzw. deaktiviert...



Ausführen von Programmen

- Liste der zulässigen Programme: `shell_escape_commands` in `/usr/share/texmf/web2c/texmf.cnf`
- Alles erlauben mit der Option `--shell-escape`
- `\input{|"./meinprogramm.sh"}`
- (oder umständlicher:
`\write18{./meinprogramm.sh > scriptoutput.tex}`
`\input{scriptoutput.tex}`)

¹Are Text-Only Data Formats Safe? Or, Use This \LaTeX Class File to Pwn Your Computer:
<http://cseweb.ucsd.edu/~hovav/dist/texhack.pdf>

L^AT_EX-Beispiele mit externen Programmen

- 1 Shell- und Perlskript aufrufen
Übersetzen mit: `pdflatex --shell-escape helloworld.tex`
- 2 Gleichungen und Integrale mit Maxima lösen und das Ergebnis ins Output-File automatisch übernehmen²
Übersetzen mit `pdflatex --shell-escape maxima.tex`
Maxima muss logischerweise installiert sein.

Live Demo...

²Warum soll ich selber rechnen? Dafür habe ich einen Rechner!

Python_TEX

+



- 1 Python für math. Berechnungen nutzen³
- 2 Kommandos mit Python definieren
- 3 Graphiken generieren und verwenden

³wozu selber rechnen?

PythonT_EX – Verwendung

Verwendung

```
\usepackage{pythontex}  
...  
\py{ PYTHON-EXPRESSION }  
  
\pyc{ PYTHON-CODE }  
  
\begin{pycode}  
...  
\end{pycode}
```

Python_TE_X als MIDDLEPROCESSOR

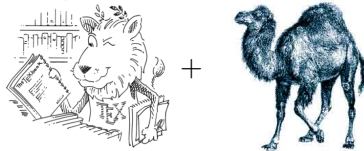


Aufruf

```
pdflatex datei.tex  
pythontex datei.tex  
pdflatex datei.tex
```

Live-Demo.

Python_TE_X unterstützt auch andere Programmiersprachen (Ruby, Julia, Octave)

PerLT_EX

- Kommandos und Umgebungen in Perl definieren
- Perl-Code einbinden

PerLT_EX

Verwendung

```
\usepackage{perltex}
...
\perlnewcommand{\BEFEHL}[#arg]{DEFINITION}
\perlnewenvironment{\ENVNAME}[#arg]{BEGINCODE}{ENDCODE}
% Analog: \perlrenewcommand / \perlrenewenvironment
\perldo{ Perl-Code }
```

Beispiel: `\substr{ }{ }{ }`

```
\perlnewcommand{\substr}[3]{substr $_[0], $_[1], $_[2]}
```

Live Demo...

PerlT_EX – Aufruf

Übersetzen mit:

```
perltex [--latex=pdflatex] perltex1.tex
```

Sicherheit

- Aufruf standardmässig in SANDBOX (keine Module möglich, ...)
- `--nosafe` ermöglicht vollen Zugriff.
- Features freischalten mit: `--permit=FEATUREa`

^a`perldoc Opcode`

PerL \TeX – noperltex

Dokumente ohne PerL \TeX weitergeben und übersetzen

- Übersetzen mit:
`perltex --makesty perltex1.tex`
- Stylefile „noperltex.sty“ wird erzeugt
(Dokumentenspezifisch, bei Änderung neu generieren!).
- `\usepackage{perltex}` \Rightarrow `\usepackage{noperltex}`
- Dokument ist NORMAL (pdf \LaTeX , latex, lua \LaTeX , x \LaTeX)
übersetzbar.

Weitere Möglichkeiten

Es gibt noch viele weitere interessante Pakete auf CTAN

- `arrayjobx`: Array data structures for (La)T_EX
- `datatool`: Tools to load and manipulate data
- `sagetex`: Embed Sage code and plots into L^AT_EX
- `boolexpr`: A boolean expression evaluator and a switch command
- `etoolbox`: Tool-box for L^AT_EX programmers using e-T_EX
- ...

Einfach mal auf [CTAN](#) suchen – es gibt Zusatzpakete für (fast) jedes Problem!

Vielen Dank

Fragen? (hoffentlich richtige...) Antworten!

Vielen Dank für Ihre Aufmerksamkeit

Wolfgang Dautermann

wolfgang.dautermann [AT] fh-joaanneum.at

Werbeeinschaltung ;-)



- FH Joanneum Graz
- 24.+25. April 2015
 - (24.04.: Workshops (Nachmittag))
 - (25.04.: ganztägiges Vortragsprogramm, Projektstände)