Wissenschaftliches Arbeiten mit LaTeX

Makros und Debugging

Daniel Borchmann

6. Dezember 2016

https://algebra20.de/dl16

Hochschulgruppe für
Freie Software und
Freies Wissen

https://fsfw-dresden.de

Ziele

Ziele

- Einen kleinen Einblick gewinnen, wie LaTeX (TeX) funktioniert

Ziele

- Einen kleinen Einblick gewinnen, wie LaTeX (TeX) funktioniert
- Lernen, eigene Makros und Umgebungen zu schreiben

Ziele

- Einen kleinen Einblick gewinnen, wie LaTeX (TeX) funktioniert
- Lernen, eigene Makros und Umgebungen zu schreiben
- LaTeX-Fehlermeldungen verstehen und beheben

Wie funktionieren LaTeX-Makros?

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Beispiel

Wenn TeX das Makro

```
\TeX
```

---

im Quelltext sieht, wird dieses *expandiert* durch

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Beispiel

Wenn TeX das Makro

```
\TeX
```

---

im Quelltext sieht, wird dieses *expandiert* durch

```
T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
```

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Beispiel

Wenn TeX das Makro

```
\TeX
```

im Quelltext sieht, wird dieses *expandiert* durch

```
T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
```

Expansion

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Beispiel

Wenn TeX das Makro

```
\TeX
```

im Quelltext sieht, wird dieses *expandiert* durch

```
T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
```

Expansion

- Expandiert werden nur *definierte* Makros.

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Beispiel

Wenn TeX das Makro

```
\TeX
```

im Quelltext sieht, wird dieses *expandiert* durch

```
T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
```

Expansion

- Expandiert werden nur *definierte* Makros.
- *Primitive* Makros werden direkt von TeX verarbeitet

Wie LaTeX den Quelltext sieht

TeX (und damit auch LaTeX) arbeiten mit Hilfe von *Makros*

Beispiel

Wenn TeX das Makro

```
\TeX
```

---

im Quelltext sieht, wird dieses *expandiert* durch

```
T\kern -.1667em\lower .5ex\hbox {E}\kern -.125emX.
```

Expansion

- Expandiert werden nur *definierte* Makros.
- *Primitive* Makros werden direkt von TeX verarbeitet
- Menge der primitiven Makros ist fest, definierte Makros können vom Benutzer hinzugefügt werden

Makrodefinitionen

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros
- `\renewcommand` zur Neudefinition bereits bestehender Makros

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros
- `\renewcommand` zur Neudefinition bereits bestehender Makros
- `\providecommand` zur Definition eines Makros, sofern dies noch nicht definiert ist

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros
- `\renewcommand` zur Neudefinition bereits bestehender Makros
- `\providecommand` zur Definition eines Makros, sofern dies noch nicht definiert ist
- *-Varianten, die Makros definieren, deren Argumente keine Absätze enthalten dürfen

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros
- `\renewcommand` zur Neudefinition bereits bestehender Makros
- `\providecommand` zur Definition eines Makros, sofern dies noch nicht definiert ist
- `*`-Varianten, die Makros definieren, deren Argumente keine Absätze enthalten dürfen
- `\DeclareRobustCommand` zur Definition *nicht-zerbrechlicher* Befehle

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros
- `\renewcommand` zur Neudefinition bereits bestehender Makros
- `\providecommand` zur Definition eines Makros, sofern dies noch nicht definiert ist
- `*`-Varianten, die Makros definieren, deren Argumente keine Absätze enthalten dürfen
- `\DeclareRobustCommand` zur Definition *nicht-zerbrechlicher* Befehle

Beispiel

```
\newcommand{\bsp}{beispielsweise}
```

Makrodefinitionen in LaTeX

Zur Definition neuer Makros stehen in LaTeX drei Arten von Makros bereit:

- `\newcommand` zur Definition neuer Makros
- `\renewcommand` zur Neudefinition bereits bestehender Makros
- `\providecommand` zur Definition eines Makros, sofern dies noch nicht definiert ist
- `*`-Varianten, die Makros definieren, deren Argumente keine Absätze enthalten dürfen
- `\DeclareRobustCommand` zur Definition *nicht-zerbrechlicher* Befehle

Beispiel

```
\newcommand{\bsp}{beispielsweise}
```

Hinweis

Makrodefinitionen sollten in der Präambel stehen.

Argumente

Makros können auch *Argumente* bekommen

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

- Argumente beginnen mit #

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

- Argumente beginnen mit #
- Erlaubt sind maximal 9 Argumente

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

- Argumente beginnen mit #
- Erlaubt sind maximal 9 Argumente
- Argumente können beliebig oft und in beliebiger Reihenfolge in der Makrodefinition verwendet werden

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

- Argumente beginnen mit #
- Erlaubt sind maximal 9 Argumente
- Argumente können beliebig oft und in beliebiger Reihenfolge in der Makrodefinition verwendet werden

Hinweis

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

- Argumente beginnen mit #
- Erlaubt sind maximal 9 Argumente
- Argumente können beliebig oft und in beliebiger Reihenfolge in der Makrodefinition verwendet werden

Hinweis

- Alle Argumente sind obligatorisch

Argumente

Makros können auch *Argumente* bekommen

Beispiel

```
\newcommand{\inBlau}[1]{\textcolor{blue}{#1}}
```

Allgemein gilt

- Argumente beginnen mit #
- Erlaubt sind maximal 9 Argumente
- Argumente können beliebig oft und in beliebiger Reihenfolge in der Makrodefinition verwendet werden

Hinweis

- Alle Argumente sind obligatorisch
- `\newcommand` kann auch Makros mit einem optionalen Argument definieren.

Andere Arten der Makrodefinition

Andere Arten der Makrodefinition

In TEX

Andere Arten der Makrodefinition

In TEX

- Mittels `\def`:

```
\def\inBlau#1{\textcolor{blue}{#1}}
```

Andere Arten der Makrodefinition

In TEX

- Mittels `\def`:

```
\def\inBlau#1{\textcolor{blue}{#1}}
```

- Definitionen flexibler, aber auch fehleranfälliger

Andere Arten der Makrodefinition

In TEX

- Mittels `\def`:

```
\def\inBlau#1{\textcolor{blue}{#1}}
```

- Definitionen flexibler, aber auch fehleranfälliger

In LATEX3

Andere Arten der Makrodefinition

In TEX

- Mittels `\def`:

```
\def\inBlau#1{\textcolor{blue}{#1}}
```

- Definitionen flexibler, aber auch fehleranfälliger

In LATEX3

- Mit dem Kommando `\DeclareDocumentCommand`

```
\DeclareDocumentCommand \chapter { s o m } {
  \IfBooleanTF {#1}
    { \typesetstarchapter {#3} }
    { \typesetnormalchapter {#2} {#3} } }
```

Andere Arten der Makrodefinition

In TEX

- Mittels `\def`:

```
\def\inBlau#1{\textcolor{blue}{#1}}
```

- Definitionen flexibler, aber auch fehleranfälliger

In LATEX3

- Mit dem Kommando `\DeclareDocumentCommand`

```
\DeclareDocumentCommand \chapter { s o m } {
  \IfBooleanTF {#1}
    { \typesetstarchapter {#3} }
    { \typesetnormalchapter {#2} {#3} } }
```

- Ermöglicht explizite Angabe der *Signatur* eines Makros

Andere Arten der Makrodefinition

In TEX

- Mittels `\def`:

```
\def\inBlau#1{\textcolor{blue}{#1}}
```

- Definitionen flexibler, aber auch fehleranfälliger

In LATEX3

- Mit dem Kommando `\DeclareDocumentCommand`

```
\DeclareDocumentCommand \chapter { s o m } {
  \IfBooleanTF {#1}
    { \typesetstarchapter {#3} }
    { \typesetnormalchapter {#2} {#3} } }
```

- Ermöglicht explizite Angabe der *Signatur* eines Makros
- Mit Paket `xparse` in LATEX verwendbar

Umgebungsdefinitionen

Neue Umgebungen können ebenfalls definiert werden

Umgebungsdefinitionen

Neue Umgebungen können ebenfalls definiert werden

Beispiel

```
\newenvironment{sketch}[1]
  {\begingroup\color{#1}}        % begin-Definition
  {\endgroup}                    % end-Definition
...
\begin{sketch}{gray}
  ...
\end{sketch}
```

Umgebungsdefinitionen

Neue Umgebungen können ebenfalls definiert werden

Beispiel

```
\newenvironment{sketch}[1]
  {\begingroup\color{#1}}        % begin-Definition
  {\endgroup}                    % end-Definition
...
\begin{sketch}{gray}
  ...
\end{sketch}
```

Argumente können nur im begin-Teil genutzt werden.

Ausblick: „Programmieren“ in TEX

PlainTEX

```
\newif\ifprime \newif\ifunknown                    % boolean variables
\newcount\n \newcount\p \newcount\d \newcount\a % integer variables
\def\primes#1{2,~3%                    % assume that #1 is at least 3
  \n=#1 \advance\n by -2               % n more to go
  \p=5                                 % odd primes starting with p
  \loop\ifnum\n>0 \printifprime\advance\p by 2\repeat}
\def\printp{,                  % we will invoke \printp if \p is prime
  \ifnum\n=1 and~\fi           % `and' precedes the last value
  \number\p \advance\n by -1 }
\def\printifprime{\testprimality \ifprime\printp\fi}
\def\testprimality{{\d=3 \global\primetrue
  \loop\trialdivision \ifunknown\advance\d by 2\repeat}}
\def\trialdivision{\a=\p \divide\a by \d
  \ifnum\a>\d \unknowntrue\else\unknownfalse\fi
  \multiply\a by \d
  \ifnum\a=\p \global\primefalse\unknownfalse\fi}

The first thirty prime numbers are \primes{30}. \bye
```

LuaLaTeX

```
\documentclass[tikz,border=10pt]{standalone}
\usepackage{pgfplots}
\usepackage{luacode}

\begin{luacode}
  function weierstrass(x0, x1, n, a, b, epsilon)
    local out = assert(io.open("tmp.data", "w"))
    ...
\end{luacode}

\begin{document}

\directlua{weierstrass(-2,2,500,0.3,5,1.e-12)}
\begin{tikzpicture}
  \begin{axis}[axis lines=middle, ymin=-1.5, ymax=1.75]
    \addplot[thick] table {tmp.data};
  \end{axis}
\end{tikzpicture}

\end{document}
```

Debugging

Fehlermeldungen in LaTeX

Problem ...

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind
- Fehlerbeschreibung ist deswegen meist *nicht hilfreich*

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind
- Fehlerbeschreibung ist deswegen meist *nicht hilfreich*

Allgemeine Lösungstrategien

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind
- Fehlerbeschreibung ist deswegen meist *nicht hilfreich*

Allgemeine Lösungstrategien

- Ordnung im TeX-Dokument

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind
- Fehlerbeschreibung ist deswegen meist *nicht hilfreich*

Allgemeine Lösungstrategien

- Ordnung im TeX-Dokument
- Fehlereinkreisung durch „binäre Suche“

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind
- Fehlerbeschreibung ist deswegen meist *nicht hilfreich*

Allgemeine Lösungstrategien

- Ordnung im TeX-Dokument
- Fehlereinkreisung durch „binäre Suche“
- `\RequirePackage{nag}` zum Auffinden veralteter Befehle

Fehlermeldungen in LaTeX

Problem ...

- Fehlermeldungen in LaTeX sind meist schwer verständlich
- Ursache: Fehler werden meist oft erst erkannt, *nachdem* alle Makros expandiert worden sind
- Fehlerbeschreibung ist deswegen meist *nicht hilfreich*

Allgemeine Lösungstrategien

- Ordnung im TeX-Dokument
- Fehlereinkreisung durch „binäre Suche“
- `\RequirePackage{nag}` zum Auffinden veralteter Befehle
- Verwendung von externen *Prüfprogrammen* wie `lacheck` oder `chktex`

„Klassische“ Fehlermeldungen

„Klassische“ Fehlermeldungen

- Schließende } ohne dazu passende, öffnende {

```
! Too many }'s.
1.6 \date December 2004}
```

„Klassische“ Fehlermeldungen

- Schließende } ohne dazu passende, öffnende {

```
! Too many }'s.
1.6 \date December 2004}
```

- Undefinierter Befehl (meistens vertippt)

```
! Undefined control sequence.
1.6 \dtae
{December 2004}
```

„Klassische“ Fehlermeldungen

- Schließende } ohne dazu passende, öffnende {

```
! Too many }'s.
1.6 \date December 2004}
```

- Undefinierter Befehl (meistens vertippt)

```
! Undefined control sequence.
1.6 \dtae
{December 2004}
```

- Mathematikbefehl außerhalb des Mathematikmodus’ benutzt

```
! Missing $ inserted
```

„Klassische“ Fehlermeldungen

„Klassische“ Fehlermeldungen

- Unerlaubter Absatz im Argument eines Makros

```
Runaway argument?
{December 2004 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
\par
l.8
```

„Klassische“ Fehlermeldungen

- Unerlaubter Absatz im Argument eines Makros

```
Runaway argument?
{December 2004 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
\par
l.8
```

- Fehlendes `\item` in Aufzählung

```
! LaTeX Error: Something's wrong--perhaps a missing \item.
...
l.37   \end{itemize}
```

„Klassische“ Fehlermeldungen

- Unerlaubter Absatz im Argument eines Makros

```
Runaway argument?
{December 2004 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
\par
l.8
```

- Fehlendes `\item` in Aufzählung

```
! LaTeX Error: Something's wrong--perhaps a missing \item.
...
l.37    \end{itemize}
```

Mehr Hilfe unter
`https://en.wikibooks.org/wiki/LaTeX/Errors_and_Warnings`

Mehr Informationen bei Fehlern

TEX (und damit auch LATEX) kann dazu gebracht werden, bei Fehlern mehr Informationen auszugeben.

Mehr Informationen bei Fehlern

TEX (und damit auch LATEX) kann dazu gebracht werden, bei Fehlern mehr Informationen auszugeben.

- `\errorcontextlines=5` im Dokument sorgt dafür, dass bei Fehlern die ersten 5 Expansionsstufen angezeigt werden.

Mehr Informationen bei Fehlern

TEX (und damit auch LATEX) kann dazu gebracht werden, bei Fehlern mehr Informationen auszugeben.

- `\errorcontextlines=5` im Dokument sorgt dafür, dass bei Fehlern die ersten 5 Expansionsstufen angezeigt werden.
- `\listfiles` in der Präambel zeigt die Versionen aller geladenen Pakete an

Mehr Informationen bei Fehlern

TEX (und damit auch LATEX) kann dazu gebracht werden, bei Fehlern mehr Informationen auszugeben.

- `\errorcontextlines=5` im Dokument sorgt dafür, dass bei Fehlern die ersten 5 Expansionsstufen angezeigt werden.
- `\listfiles` in der Präambel zeigt die Versionen aller geladenen Pakete an
- Viele *Tracing-Befehle* sind direkt in TEX eingebaut

```
\tracingmacros=1
\tracingcommands=1
\tracingall
```

(Siehe auch `https://tex.stackexchange.com/questions/60491/latex-tracing-commands-list`)

Mehr Informationen bei Fehlern

TEX (und damit auch LATEX) kann dazu gebracht werden, bei Fehlern mehr Informationen auszugeben.

- `\errorcontextlines=5` im Dokument sorgt dafür, dass bei Fehlern die ersten 5 Expansionsstufen angezeigt werden.
- `\listfiles` in der Präambel zeigt die Versionen aller geladenen Pakete an
- Viele *Tracing-Befehle* sind direkt in TEX eingebaut

```
\tracingmacros=1
\tracingcommands=1
\tracingall
```

(Siehe auch `https://tex.stackexchange.com/questions/60491/latex-tracing-commands-list`)

- `\usepackage{trace}`