(LA)TEX-Programmierung

Eine kurze Einführung

Erik Zöllner

20. 01. 2019

CTAN lion drawing by Duane Bibby; ctan.org/lion

Programmieren mit Paketen

- `ifthen`
- `pgffor`
- `xparse`

Kommando Herkunft

- TeX
- ε-TeX
- LaTeX
- Pakete

Kommando Herkunft

- TEX → The TEXbook
- TEX → TEX by Topic
- ε-TEX → The ε-TEX manual
- LATEX → macros2e
- LATEX → source2e
- Pakete → texdoc
- Pakete → CTAN

Kommando Herkunft

- TEX → The TEXbook
- TEX → TEX by Topic
- ε-TEX → The ε-TEX manual
- LATEX → macros2e
- LATEX → source2e
- Pakete → texdoc
- Pakete → CTAN

„ε-TEX has been specified by the LATEX team as the engine for the development of LATEX 2ε, in the immediate future; as a result, LATEX programmers may (in all current TEX distributions) assume ε-TEX functionality.“

[https://www.ctan.org/pkg/etex]

Gliederung

Verarbeitungsschritte & Makros

Wenn-Abfragen

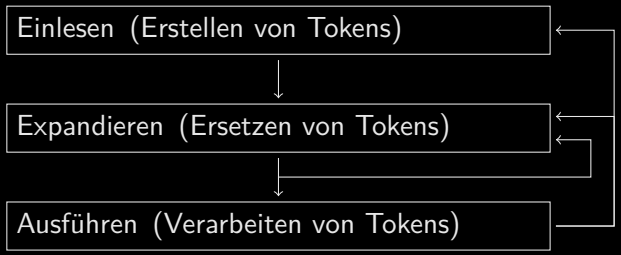
Weiteres

Verarbeitungsschritte

1. Einlesen (Erstellen von Tokens)
2. Expandieren (Ersetzen von Tokens)
3. Ausführen (Verarbeiten von Tokens)
4. Setzen (Positionieren der Ausgabe)

[*TEX by Topic* von Victor Eijkhout, Seite 21f]

Verarbeitungsschritte



Tokens

bestehen aus:

- Character Code
- Category Code (kurz: Catcode)

Catcodes

Catcode	Bedeutung	Zeichen
0	Escape character	\
1	Begin grouping	{
2	End grouping	}
3	Math shift	$
4	Alignment tab	&
5	End of line	<return>
6	Parameter	#
7	Superscript	^
8	Subscript	_
9	Ignored character	<null>
10	Space	<space> und <tab>
11	Letter	a–z und A–Z
12	Other	andere Zeichen
13	Active character	z. B. ~
14	Comment character	%
15	Invalid character	<delete>

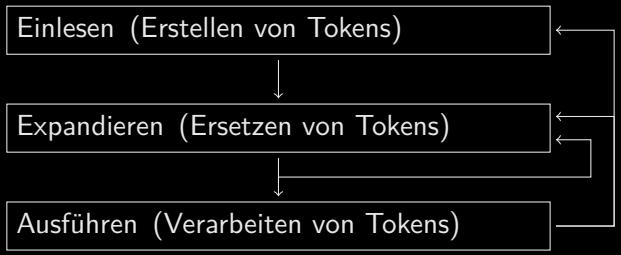
s. *TEX by Topic* von Victor Eijkhout, Seite 29

Catcodes, die nicht in Tokens vorkommen

Catcode	Bedeutung	Zeichen
0	Escape character	\
1	Begin grouping	{
2	End grouping	}
3	Math shift	$
4	Alignment tab	&
5	End of line	<return>
6	Parameter	#
7	Superscript	^
8	Subscript	_
9	Ignored character	<null>
10	Space	<space> und <tab>
11	Letter	a–z und A–Z
12	Other	andere Zeichen
13	Active character	z. B. ~
14	Comment character	%
15	Invalid character	<delete>

s. *TEX by Topic* von Victor Eijkhout, Seite 29

Verarbeitungsschritte



Makros und Verarbeitungsschritte

Livecoding

Definieren von Macros

TEX

```
\def\greeting#1{hello #1}
```

LATEX

```
\newcommand{\greeting}[1]{hello #1}
\renewcommand{\greeting}[1]{hello #1}
```

Definieren von Macros

TeX

```
\def\greeting#1{hello #1}
```

⚠ Überschreibt das zu definierende Makro kommentarlos, falls es bereits existiert.

LaTeX

```
\newcommand{\greeting}[1]{hello #1}
\renewcommand{\greeting}[1]{hello #1}
```

Makro ist `\long` (Argument darf `\par` enthalten).
Die Sternchen-Version definiert es als *nicht* `\long`.

Definieren von Macros

LaTeX	Plain TeX	
\newcommand \renewcommand \providecommand	\def	\gdef
\protected@edef	\edef	\xdef
	\let	\futurelet

etoolbox, suffix, xparse

Definieren von Macros

LaTeX	Plain TeX	
\newcommand	\def	\gdef
\renewcommand		
\providecommand		
\protected@edef	\edef	\xdef
	\let	\futurelet

etoolbox, suffix, xparse

Definieren von Macros

LaTeX	Plain TeX	
\newcommand	\def	\gdef
\renewcommand		
\providecommand		
\protected@edef	\edef	\xdef
	\let	\futurelet

etoolbox, suffix, xparse

Definieren von Macros

LaTeX	Plain TeX	
\newcommand	\def	\gdef
\renewcommand		
\providecommand		
\protected@edef	\edef	\xdef
	\let	\futurelet

etoolbox, suffix, xparse

Prefixes für \def und \edef

- \global
- \long (Standard in LaTeX)
- \outer
- \protected

Gliederung

Verarbeitungsschritte & Makros

Wenn-Abfragen

Weiteres

Wenn-Abfragen

- `\ifx`, `\if`, `\ifcat`
- `\ifnum`, `\ifodd`, `\ifcase`
- `\ifdim`
- `\ifvmode`, `\ifhmode`, `\ifmmode`, `\ifinner`
- `\ifvoid`, `\ifhbox`, `\ifvbox`
- `\ifeof`
- `\iftrue`, `\iffalse`
- `\ifdefined`, `\ifcsname...\endcsname`
- `\iffontchar`

`\newif`

`\unless`

s. *TEX by Topic*, Kapitel 13: *Conditionals*,

s. *The ε-TEX manual*, Abschnitt 3.12: *Expandable Commands*

Wenn-Abfragen

Livecoding

Leeres Argument?

Knuth-TEX

```
\ifx \relax #1\relax
    empty%
\else
    not empty%
\fi
```

ε-TEX

```
\if \relax \detokenize{#1}\relax
    empty%
\else
    not empty%
\fi
```

Leeres Argument?

Knuth-TEX

```
\ifx \relax #1\relax
    empty%
\else
    not empty%
\fi
```

⚠ Falsch, wenn #1 mit \relax beginnt.

ε-TEX

```
\if \relax \detokenize{#1}\relax
    empty%
\else
    not empty%
\fi
```

\if statt \ifx, damit \detokenize expandiert wird. \relax ist nicht expandierbar und daher nicht betroffen.

Delimited Arguments & Wenn-Abfragen

```
\documentclass{article}

\def\formatdate#1.#2.#3 {% #1: day, #2: month, #3: year
    (%
    \ifx \relax #3\relax
        % no year given
        \the\year
    \else\ifnum #3 < 100
        % year two digits abbreviation given
        19#3%
    \else
        % full year given
        #3%
    \fi \fi
    -#2-#1)
}

\begin{document}
    \formatdate 20.11.2001
    \formatdate 20.11.99
    \formatdate 20.11.
\end{document}
```

Gliederung

Verarbeitungsschritte & Makros

Wenn-Abfragen

Weiteres

Schleifen

- Rekursion

Schleifen

- Rekursion
- `\loop body 1\if...body 2\repeat`
- `\@for\i:=\commaSeparatedList\do{body}`
- `\@tfor\i:=tokens{or}{groups}\do{body}`
- `pgffor`
- ...

@: Other oder Letter

```
\makeatletter
```

```
\makeatother
```

@: Other oder Letter

```
\makeatletter
\newcommand{\mymacro}{%
    \@ifstar
        \@mymacro@starred
        \@mymacro@normal
}
...
\makeatother
```

Ändern von Catcodes im Argument

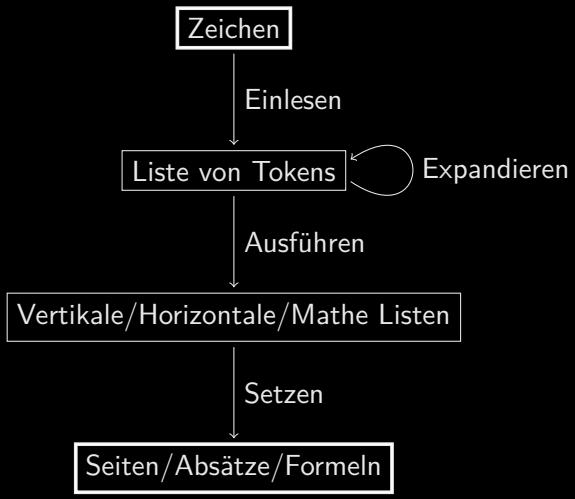
```
\newcommand{\filename}{%
    \begingroup
        \catcode`_=12
        \DoFilename
}
\newcommand{\DoFilename}[1]{%
    \endgroup
    \emph{#1}%
}
```

Ändern von Catcodes im Argument

```
\newcommand{\filename}{%
    \begingroup
        \catcode`_=12
        \DoFilename
}
\newcommand{\DoFilename}[1]{%
    \endgroup
    \emph{#1}%
}
```

⚠ Diese Funktionalität ist besser über ein Ersetzen von Tokens zu implementieren, damit das Makro auch in Argumenten (wie z. B. von `\caption`) verwendet werden kann. Siehe Aufgabe 1.

Wiederholung



Anhang

Ignorieren von Leerzeichen
Expandierbarkeit
Makro-/Kommandonamen
Umgebungen
Fragile & Geschützte/Robuste Kommandos
Zähler & Zahlen
Ändern von Catcodes
Vorausschauen aufs nächste Token
Der Nutzen Nichts zu tun
Generieren von Fehlern und Warnungen
Debugging
Weitere Informationen
Pakete

Ignorieren von Leerzeichen I

Leerzeichen werden ignoriert

- am Ende der Zeile

unabhängig von Catcodes!

Ignorieren von Leerzeichen II

Zeichen der Kategorie 10 (Spaces) werden ignoriert

- nach Control Words (aber *nicht* nach Control Characters oder Active Characters!)
- nach anderen Zeichen der Kategorie 10 (Spaces)
- am Anfang einer Zeile

beim Erstellen der Tokens

Ignorieren von Leerzeichen III

Bereits erstellte Spacetokens werden ignoriert

- vor Undelimited Arguments beim Expandieren von Makros
- nach `\kernel@ifnextchar` $\Longrightarrow$ vor optionalen Argumenten
- nach `\ignorespaces`
- außerhalb des H-Mode
- (*nicht* mehr als 1) vor und hinter dem optionalen Gleichheitszeichen bei Zuweisungen, z. B. bei `\let`

Expandierbarkeit

Nicht expandierbar:

- Definitionen/Zuweisungen: \def/\edef/\let/\catcode/...
- \uppercase/\lowercase
- \ignorespaces/\unskip
- \relax

Expandierbar (aber nicht unbedingt vollständig expandierbar):

- \if...\else \fi
- \input/\@@input
- \expandafter/\noexpand/\unexpanded (https://tex.stackexchange.com/a/497646/120953)
- \the
- \csname ...\endcsname
- Alle definierten Kommandos, die keine Primitive sind (mit Ausnahme von \protected Kommandos in gewissen Situationen)

Makro-/Kommandonamen

	Syntax	Beispiel
Control Words	`[0][11]+`	`\documentclass`
Control Character	`[0][^11]`	`\", \\, \_`
Active Character	`[13]`	`~`
Active Math Character	`$[11\|12]$`	`'`
	`\mathcode "8000`	

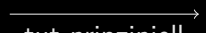
`\csname ...\endcsname`

Syntax Erläuterung:

`[`*n*`]`	Ein beliebiges Zeichen der Kategorie *n*
`[^`*n*`]`	Ein beliebiges Zeichen, dass *nicht* der Kategorie *n* angehört
`+`	Einmal oder mehrfach
`$[11\|12]$`	Ein Zeichen der Kategorie 11 oder 12 im Mathe-Modus

Umgebungen

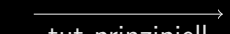
```
\begin{itemize}
    \item Abc
    \item Def
    \item Ghi
\end{itemize}
```




```
\begingroup
\itemize
    \item Abc
    \item Def
    \item Ghi
\enditemize
\endgroup
```

Umgebungen

```
\begin{itemize}
    \item Abc
    \item Def
    \item Ghi
\end{itemize}
```




```
\begingroup
\itemize
    \item Abc
    \item Def
    \item Ghi
\enditemize
\endgroup
```

s. \csname ...\endcsname

Fragile & Geschützte/Robuste Kommandos

https://tex.stackexchange.com/a/4747

Schützen von fragilen Kommandos:

- `\protect`: Präfix bei der Anwendung
- `\protected`: Präfix bei der Definition (robustes Kommando)
- `\robustify`: Ändern eines fragilen Kommandos in ein Robustes (`etoolbox`)

Definieren von robusten Kommandos mit `etoolbox`:

- `\newrobustcmd`
- `\renewrobustcmd`
- `\providerobustcmd`

Testen, ob in geschütztem Kontext:

- `\ifx \protect\relax`

Zähler

TeX (Wertänderungen lokal)

```
\newcount \mycount % global
\mycount = 42
\advance\mycount by  1
\advance\mycount by -1
\the \mycount
```

LaTeX (global)

```
\newcounter{mycounter}
\setcounter{mycounter}{42}
\stepcounter{mycounter}
\addtocounter{mycounter}{-1}
\themycounter
```

Scratch-Register: `https://tex.stackexchange.com/a/88261`

Zahlen

- Dezimal `[0-9]+`
- Oktal `'[0-7]+`
- Hexadezimal `"[0-9A-F]+` (keine Kleinbuchstaben)
- Character Code `` `. ``
- Counter `\mycount`
- Berechnungen `\numexpr <Berechnung>\relax`

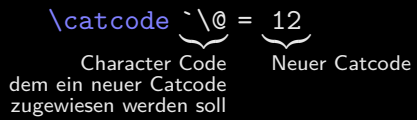
ACHTUNG: rundet mathematisch

Beispiele:

```
\number"2A
\newcount\mycount \mycount=42
\catcode`\@=\the\catcode`A
\char\numexpr `A - 23 \relax
\the\numexpr "8000 / 777 \relax % faster than \number\numexpr
```

Ändern von Catcodes

`\makeatletter`, `\makeatother`



`\@makeother`, `\dospecials`, `\@sanitize`, `\@firstofone`, `\active`

Vorausschauen aufs nächste Token

```
\newcommand\checknexttoken[1]{%
    \ifx \specialtoken #1%
        \expandafter
        ↪  \handleSpecialCase
    \else
        \expandafter #1%
    \fi
}
```

```
\newcommand\checknexttoken{\futurelet
↪  \nexttoken \dochecknexttoken}
\newcommand\dochecknexttoken{%
    \ifx \nexttoken \specialtoken
        \expandafter
        ↪  \handleSpecialCase
    \fi
}
```

Vorausschauen aufs nächste Token

```
\newcommand\checknexttoken[1]{%
    \ifx \specialtoken #1%
        \expandafter
        ↪  \handleSpecialCase
    \else
        \expandafter #1%
    \fi
}
```

- Ignoriert Leerzeichen
- Entfernt geschweifte Klammern
- Problematisch, wenn #1 durch mehr oder weniger als ein Token ersetzt wird
- Kann nicht über Dateiende hinauslesen ohne vorheriges \futurelet

```
\newcommand\checknexttoken{\futurelet
↪  \nexttoken \dochecknexttoken}
\newcommand\dochecknexttoken{%
    \ifx \nexttoken \specialtoken
        \expandafter
        ↪  \handleSpecialCase
    \fi
}
```

- Nicht expandierbar

Der Nutzen Nichts zu tun

\@firstofone, \@gobble, \@firstoftwo, \@secondoftwo

https://tex.stackexchange.com/questions/21262/what-do-firstoftwo-and-secondoftwo-do

https://tex.stackexchange.com/questions/315749/do-all-ifs-need-to-be-defined-before-a-nested-if

\relax

https://tex.stackexchange.com/questions/96501/what-does-relax-do/96505

Generieren von Fehlern und Warnungen

- `\PackageError{packagename}{errortext}{helptext}`
- `\PackageWarning{packagename}{warningtext}`
- `\PackageInfo{packagename}{infotext}`

clsguide, source2e

Debugging

- `\typeout{...}` schreibt das Argument in die log-Datei (nach einer vollständigen Expandierung). Wenn es mit einem Ausrufezeichen beginnt interpretiert TeXstudio die Zeile als Fehler und hebt sie entsprechend hervor.
- `\show`/`\meaning` zeigt Definitions Präfixe, Parametertext und Replacementtext eines Kommandos. (\show in der log-Datei, \meaning im pdf. \meaning sollte mit Fontencoding T1 oder in Typewriter-Schrift verwendet werden, damit die korrekten Zeichen ausgegeben werden.)
- `\showthe`/`\the` zeigt den aktuellen Wert einer Variable (beispielsweise eines Counters). (\showthe in der log-Datei, \the im pdf.)
- `\tracingmacros=1`/`2` schreibt alle Expandierungen mit den entsprechenden Argumenten in die log-Datei.
- `\tracingcommands=2`/`3` schreibt ausgeführte und expandierte Primitive sowie die Ergebnisse von if-Abfragen in die log-Datei.
- viele weitere tracing-Kommandos

Weitere Informationen

- Allgemein
  - `$ texdoc <paketname>`
  - `https://ctan.org/`
  - `https://tex.stackexchange.com/`
- Plain-TEX
  - *The TEXbook* by Donald E. Knuth
  - *TEX by Topic*
- LATEX
  - macros2e
  - source2e
- Schreiben von Paketen und Klassen
  - clsguide
  - TeXstudio auto completion
  - TeXstudio auto completion: Beispiel graphicx

Nützliche Pakete für Programmierung

- `pgffor` (For-Schleifen. Es gibt viele verschiedene Schleifen-Kommanods, oft ist auch das LaTeX-Kommando `\@for` oder das Plain TeX-Kommando `\loop` nützlich.)
- `pgfkeys` (`key=value` Syntax. Dies ist nicht das einfachste, aber ein sehr mächtigstes Paket für diesen Zweck. Alternativen s. `https://tex.stackexchange.com/q/26771`)
- `etoolbox` (Ändern von Kommandos, Additional Document Hooks, usw.)
- `suffix` (Definieren von Alternativen für ein Kommando (Starred-Version, Optional Arguments, ...))
- `(x)ifthen` (Wenn Abfragen. Ich bevorzuge meist das TeX-Primitiv `\ifx` stattdessen.)
- `calc` (Ermöglich die Verwendung von Grundrechenarten in den Argumenten gewisser Kommandos. Ich bevorzuge meist ein explizites `\numexpr` stattdessen, s. `https://tex.stackexchange.com/a/245663`.)
- `pgfmath` (Mathematische Berechnungen, inklusive Potenzierung, Logarithmierung und trigonometrischer Funktionen.)
- `xparse` (Definition von Kommandos mit flexibleren Optionen für Parameter als `\newcommand`)
- `datatool` (Datenbanken. Mögliche Anwendung: Laden von csv-Dateien und Anzeige in einer Tabelle.)

Andere Nützliche Pakete 1

- `babel`, `inputenc`, `fontenc`, `lmodern` (Für (deutsche) Sprache, s. Folie **??**)
- `hyperref` (Links; Metainformationen von pdf-Dateien; Automatische Verlinkung von Verweisen. Sollte zum Schluss geladen werden.)
- `csquotes` (Anführungszeichen. `\enquote`, `\textelp` und `\textins`. Mit `biblatex`: `\SetCiteCommand\autocite`; `\blockcquote` oder `\textcquote` und `displaycquote`.)
- `enumitem` (Formatierung von Listen)
- `fancyhdr` (Kopf- und Fußzeilen)
- `geometry` (Ändern der Seitenränder usw. Vor `fancyhdr` zu laden.)
- `microtype` (Automatische Optimierung von Abständen zwischen Buchstaben um besseren Textfluss zu erzeugen. Langsam. Meist nicht erforderlich, kann aber gegen overfull oder underfull boxes helfen. Funktioniert *nicht* mit XeLaTeX und LuaLaTeX.)
- `url` (`\url` Kommando, dass Zeilenumbrüche erlaubt. Falls `hyperrefs` `\url` Kommando overfull oder underfull boxes erzeugt.)
- `amsmath` oder `mathtools`, ggf. `autonum` (Mathematiksatz)
- `xcolor` (Farben)

Andere Nützliche Pakete 2

- `placeins` (Mehr Kontrolle über Floats)
- `graphicx` oder `graphbox` (Einfügen von Bildern)
- `tikz` (Erstellen von Zeichnungen, Pfeildiagrammen und vielem mehr)
- `booktabs` und `array` (für Tabellen)
- `tabularx` (Tabellen mit fester Breite)
- `longtable` (Tabellen, die über Seitenenden umgebrochen werden können)
- `siunitx` (Einheiten & Zahlen, Ausrichten von Zahlen an Dezimaltrennzeichen in Tabelle)
- `minted` (Einbinden von Quelltext mit Syntaxhighlighting)
- `tcolorbox` (fancy Boxen. Kann auch mit verschiedenen Paketen zur Darstellung von Quelltext zusammenarbeiten. Kann auch LaTeX-Code *und* Ausgabe nebeneinander anzeigen. s. a. `https://tex.stackexchange.com/q/135871/120953`)
- `cleveref` (Automatische Angabe des Typs einer Referenz und Sortierung/Formatierung von Mehrfachreferenzen)
- `biblatex` (Einfacher anpassbare Quellenverzeichnisse und Zitierungen mit `\autocite`)
- viele weitere, s. `https://ctan.org/topics/highscore`