Eigene Befehle, Umgebungen und Klassen

LaTeX-Kurs der Unix-AG

Johannes Kloos

19. Juli 2006

Mit freundlicher Unterstützung des AStAs der TU Kaiserslautern

Definition von Befehlen

- Eigene Befehle definiert man mit `\newcommand`, Beispiel:

```
\newcommand{\usw}{und so weiter...}
```

  Der Befehl `\usw` wird dann gesetzt als

  und so weiter...

- Kompliziertere Befehle sind auch möglich:

```
\usepackage{amsmath}
\newcomannd{\R}{\mathbbm{R}}
```

  definiert einen Befehl `\R`, der $\mathbb{R}$ setzt.
- Wenn es den Befehl schon gibt, meldet LaTeX einen Fehler. Befehle ersetzen: `\renewcommand`; Befehle nur definieren, wenn es sie noch nicht gibt: `\providecommand`.

Befehle mit Argumenten

- Argumente an Befehle übergeben:

```
\newcommand{\erzeugnis}[1]{\langle #1 \rangle}
```

  Der Aufruf `\erzeugnis{x_1,\ldots,x_n}` liefert $\langle x_1, \ldots, x_n \rangle$.
- Allgemeine Syntax von `\newcommand`:

```
\newcommand{\name}[Args][Default]{Definition}
```

  Hierbei steht **name** für den Namen des Befehls, **Args** für die Zahl der Argumente, **Default** für die Voreinstellungen des ersten Arguments und **Definition** für die eigentliche Befehlsdefinition.
  Argumente spricht man mit `#n` an (wobei **n** eine Zahl von 1 bis 9 ist), das erste darf optional sein und hat denn **Default** als Wert.

Definition von Umgebungen

- Analog zu Befehlsdefinitionen:

```
\newenvironment{name}[Args]{Anfang}{Ende}
```

  Hierbei steht **name** für den Namen der Umgebung, **Args** für die Zahl der Argumente, **Anfang** für die Befehle, die am Anfang der Umgebung stehen sollen, und **Ende** für die Befehle am Ende.
- Beispiel auf der nächsten Folie.
- Funktioniert im Prinzip wie `\newcommand`, entsprechend `\renewenvironment`.

Beispiel für eigene Umgebungen

Beispiel:

```
\newenvironment{dummerspruch}[1]{#1 sagt: \begin{
  quote}}{\end{quote}}
\begin{dummerspruch}[Gollum]
My preciousssss!
\end{dummerspruch}
```

liefert:
Gollum sagt:

My preciousssss!

Pakete und Klassen

- Wie definiert LaTeX seine Befehle? Der größte Teil wird in **Pakete** und **Klassen** festgelegt.
- Das grundlegende Layout und einige wesentliche Befehle werden von der Dokumentklasse ausgewählt, die mit `documentclass` ausgewählt wurde.
- Weitere Befehle, Einstellungen etc. werden in Paketen definiert, die mittels `usepackage` geladen werden.
- Pakete und Klassen können Optionen erhalten: `\usepackage[all]{xy}`.
- Klassen werden in Dateien gespeichert, die auf `.cls` enden, Pakete in solchen, die auf `.sty` enden.
- Dateizugriff basiert auf TeX-Mechanismen.

Pakete und Klassen: Suchpfade

- Wo sucht TeX die Dateien?
- Im **TeX-Suchpfad** – den kann man sich bei TeTeX und TexLive mittels `kpsepath` anzeigen lassen: `kpsepath tex`.
- Standardmäßig sind das aktuelle Verzeichnis und die TeX-Installation im TeX-Pfad, bei den von mir getesten Systemen u.a. das `texmf/tex`-Verzeichnis im Homeverzeichnis des Benutzers.
- Auch andere Dateien werden in entsprechenden Suchpfaden gefunden, z.B. Schriften, Bibtex-Dateien etc.
- Dateien im TeX-Bäumen findet man mit `kpsewhich`: `kpsewhich xy.sty`.

Pakete und Klassen: TeX-Baum

- Organisation der Dateien in einer verschachtelten Verzeichnisstruktur:

```
$ kpsewhich xy.sty
/usr/share/texmf-texlive/tex/generic/xypic/xy.sty
$ kpsewhich amsmath.sty
/usr/share/texmf-texlive/tex/latex/amslatex/...
```

  Unter `tex` finden sich Paket- und Klassendateien, dann unter `latex` solche für LaTeXetc.
- Wie werden Dateien hier gesucht? Dafür gibt es (bei tetex und TeXLive) die Datei `ls-R` im Wurzelverzeichnis eines TeX-Suchpfades.
- Anlegen/Aktualisieren dieser Datei: `mktexlsr`.

Eigene Pakete

- Im Prinzip enthält eine Paket-Datei nach einem einfachen Header normale LaTeX-Befehle
- Eine Paket-Datei kann irgendwo im TeX-Suchpfad abgelegt werden und sollte der üblichen Namenskonvention entsprechen (Dateiname endet auf .sty).
- Sonderbedeutung von @: In Paketen und Klassen kann dieses Zeichen wie ein gewöhnlicher Buchstabe verwendet werden, in Dokumenten ist es ein Sonderzeichen (das kann man aber mit `\makeatletter` und `\makeatother` umschalten).

Struktur einer Paket-Datei

- Grobe Struktur:
  Kenndaten
  Initialisierung
  Optionen
  Laden von Paketen
  Definitionen
- Kenndaten: Angabe der benötigten LaTeX-Version mit `NeedsTeXFormat`, und welche Klassen und Pakete bereitgestellt werden (mit `\ProvidesPackage` und `\ProvidesFile`).
- Normalerweise:

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{Name}[2006/07/19 Beschreibung]
```


Spezielle Befehle

- Laden von Paketen: Statt `\usepackage` sollte man `\RequirePackage` verwenden, da hier beim doppelten Laden eines Paketes eine angepasste Fehlerbehandlung erfolgt.
- `\AtEndOfPackage`, `\AtEndOfClass`, `\AtBeginDocument`, `AtEndDocument` führen ihr Argument an der angegebenen Stelle aus.
- `\IfFileExists` und `\InputIfFileExists` führen abhängig von der Existenz einer Datei Code aus.

Spezielle Befehle (2)

- Statt mit `\newcommand` kann man Befehle auch mit `\DeclareRobustCommand` und `\DeclareRobustCommand*` definieren. Unterschied: Die Befehle sind robust, d.h. sie können auch problemlos in Überschriften etc eingesetzt werden (Seiteneffekte treten nur einmal auf).
- `\CheckCommand` überprüft die Definition eines Befehls und erzeugt einen Fehler, wenn sie nicht stimmt.
- Man kann Befehle auch mit `\def` definieren (wird aber nicht empfohlen), das ist aber nützlich, um sich interne Befehle, Variablen etc. zu deklarieren.
- Wenn man die Definition eines Befehls speichern will, hilft `\let`.

Beispiel

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{Beispiel}[2006/07/19 Beispiel]
\RequirePackage{amssymb}
\def\@vorne{$\Box$}
\def\@hinten{$\Box$}
\newcommand{\markierung}[2]
  {\def\@vorne{#1} \def\@hinten{#1}}
\let\@oldEmph=\emph
\renewcommand{\emph}[1]
  {\@vorne \@oldEmph{#1} \@hinten}
```

Dieses Paket definiert den Befehl `\emph` um, so dass vor und hinter dem hervorgehobenen Ausdruck zwei wählbare Markierungen stehen.

Eigene Klassen

So ähnlich wie eigene Pakete. Unterschiede:

- `\LoadClass` lädt andere Klasse (auf die Art kann man eine gegebene Klasse erweitern).
- `\PassOptionsToClass` gibt Optionen an eine andere Klasse weiter.
- `\ProvidesClass` statt `\ProvidesPackage`.
- Empfehlenswert: Basis-Klasse aussuchen und erweitern, sonst hat man viel Arbeit.

Was man sonst noch braucht

- Boxen
- Längen
- Zähler
- Optionen
- ...

Zu Details sollte man in den LaTeX-Begleiter und in `www.tug.org/tex-archive/macros/latex/doc/clsguide.pdf` schauen.

Boxen

- TeX setzt eine Seite in Boxen mit Text/Grafik/usw.
- Drei Arten: LR (Text von links nach rechts, nur eine Zeile), Par (mehrere Zeilen), Rule (Linie).
- LR-Boxen: `\mbox` und `\fbox` setzen Text ohne bzw. mit Rahmen, bei `\makebox` und `\framebox` zusätzlich noch Größenangaben.
- Par-Boxen: `\parbox` setzt Text mit angegebener Breite, `\minipage` ist entsprechende Umgebung, die sich wie eine logische Seite verhält.
- Linien mit `\rule`.

Längen

- Konzept der Längenparameter: Feste Länge in einer Variable, ggf. mit Toleranzbereich, in dem sie gestreckt/gestaucht werden können (“Gummilänge”).
- Bestimmen Abstände etc. im Dokument.
- Angabe einer Länge: `1cm plus 2pt minus 3ex`.
- `\fill`: Beliebig dehnbare Länge, mindestens 0;
- `\stretch`: Dehnbare Länge nach Parameter.
- `\newlength` definiert eine neue Längenvariable.
- `\setlength` setzt eine Länge auf einen angegebenen Wert.
- `\addtolength`, `\settowidth`, `\settoheight`, `\settodepth`.
- `\hspace` setzt horizontalen Zwischenraum, `\hspace*` nicht-löschbaren, `\vspace` vertikalen, `\hfill` bis Zeilenende.

Zähler

- z.B. für Seitenzahlen, Aufzählungsnummern etc.
- Definition mit `\newcounter`.
- Setzen/ändern mit `\setcounter` und `addtocounter`
- `\value` liefert Wert, `\arabic`, `\roman` usw. zum Ausgeben.
- Normalerweise wird `\thezaehler` definiert für Ausgabe des Zählers.
- `\stepcounter` zählt eins weiter, `\refstepcounter` setzt zusätzlich Referenz.
- Vordefiniert: u.a. `section`, `subsection`, `page`, `enumi` bis `enumiv` usw.
- Zähler können von anderen abhängen und werden auf 0 gesetzt, wenn der Elternzähler erhöht wird.

Optionen

- `\DeclareOption` definiert Option und Code, der bei deren Auftreten ausgeführt wird.
- `\DeclareOption*` behandelt unbekannte Optionen.
- `\PassOptionsToPackage` übergibt Optionen an anderes Paket.
- `\CurrentOption`: Aktuelle Option (für `\DeclareOption*`), `\OptionNotUsed` markiert Option als unbenutzt.
- `\ExecuteOptions` führt Code für angegebene Optionen aus, `\ProcessOptions` führt übergebene Optionen aus.