Eigene Kommandos in LaTeX

Philipp Wendler

12. Juni 2008

Save boxes

- Die Variablen von LaTeX für Boxen.
- Deklarierung: \newsavebox{\mybox}
- Befüllen:
  - \sbox{\mybox}{*content*} wie \mbox
  - \savebox{\mybox}[*width*][*pos*]{*content*} wie \makebox
  - \begin{lrbox}{\mybox}
    content
    \end{lrbox} wie \sbox
- Ausgabe: \usebox{\mybox}

Save boxes, example

```
\newsavebox{\ex}

\savebox {\ex} [5cm] {\color{red}\textbf{Beispiel} }

\framebox{ \usebox{\ex} }     \usebox{\ex}
```

Beispiel **Beispiel**

Counter

- Die Variablen von LaTeX für Integer.
- Deklarierung: \newcounter{mycounter}[*oldcounter*]
  Initialisierung mit 0
- Untergeordnete Counter:
  \@addtoreset{*reset−counter*}{mycounter}
  \@removefromreset{*reset−counter*}{mycounter}
- Setzen:
  \setcounter{mycounter}{*val*}
  \addtocounter{mycounter}{*val*}
- Erhöhen um 1, resetted alle untergeordneten Counter:
  \stepcounter{mycounter}{*val*}
  \refstepcounter{mycounter}{*val*}
  Letzteres setzt \ref auf \themycounter

Counter-Ausgabe

- Verwendung, wo LaTeX eine Zahl erwartet:
  `\value{mycounter}`
- Ausgabe als Text:

8	viii	VIII
h	H	††

```
\arabic{ex}   \roman{ex}   \Roman{ex}
\alph{ex}     \Alph{ex}    \fnsymbol{ex}
```

- Standard-Ausgabe: `\themycounter`
  - Exisitiert für jeden Counter
  - wird z.B. durch `\ref` aufgerufen
  - kann mehrere Counter gleichzeitig ausgeben:

```
\renewcommand\theex %
    { \arabic{section}.\alph{ex}) }
\theex
```

3.h)

Befehlsklassen

- Befehle gedacht für Dokumente:
  kurze Namen und kleingeschrieben (z.B. `\section`)
- Befehle für Paket-Schreiber:
  CamelCase (z.B. `\RequirePackage`)
- Interne Befehle, Kernelbefehle:
  @ im Namen (z.B. `\@ifnextchar`)
  Benutzung möglichst vermeiden, Verhalten kann sich ändern!

Leider nicht vollständig durchhaltbar.

Verwendung von internen Befehlen

- können normal nicht verwendet werden
  wegen dem ungültigen Zeichen
- in der Dokument-Präambel:

  `\makeatletter` `\comm@nd` `\makeatother`
- Verwendung ungefährlich, man muss nur daran denken
- in Paketen und Klassen: nie \makeat∗ verwenden,
  Standardverhalten ist bereits wie nach `\makeatletter`

Namensbeschränkungen

Befehle:

- Nur Buchstaben (a-z und A-Z), kein _:-+ etc.!
- Einzig der Modifier * am Ende erlaubt (technisch kein Namensbestandteil).
- Ausnahme sind Namen aus nur einem Zeichen: alles (ja!) erlaubt: \ü \$ \%

Umgebungen:

- Theoretisch mehr Zeichen erlaubt, aber nicht offiziell (kann Probleme mit Paketen geben).
- Einzig sichere Namensbestandteile sind a-z, A-Z und *.

newcommand

\newcommand\mycmd[#*args*][*default*]{*command definition*}

- \mycmd: Name des Befehlsklassen
- [#*args*]: optional Anzahl der Parameter (≤ 9)
- [*default*]: optional Angabe eines Default-Werts für den 1. Parameter
- {*command definition*}: Inhalt des Befehls

  Auf die Parameter kann mit #1 bis #*args* zugegriffen werden.

  Bei verschachtelten Definitionen ##1 usw.

Argument-Beschränkungen

Einschränkungen für Befehls-Argumente:

- Geschweifte Klammern müssen ausgeglichen sein.
- \verb, die verbatim-Umgebung und ähnliches nicht erlaubt
- in optionalen Argumenten ist] nur innerhalb von Klammern erlaubt (z.B. \item[{a]}] geht, \item[a]] nicht)

falls man \newcommand* benutzt hat:

- Absätze (z.B. durch Leerzeilen oder \par erzeugt) sind nicht erlaubt.

Andere Befehlsdefinitionen

Alle Parameter wie bei \newcommand, ebenso existieren analoge Versionen mit *

- \renewcommand
  Redefiniert den Befehl. Änderung der Parameteranzahl sind möglich, aber keine Überladungen.
- \providecommand
  Definiert den Befehl nur, wenn er noch nicht existiert.
- \DeclareRobustCommand
  Definiert einen „robusten“Befehl.
- \CheckCommand
  Wirft einen Fehler, wenn die vorhanden Definition nicht exakt mit der angegebenen übereinstimmt.

newenvironment

\newenvironment{myenv}[#*args*][*default*]{*begcmd*}{*endcmd*}
\renewenvironment{myenv}[#*args*][*default*]{*begcmd*}{*endcmd*}

- myenv: Name der Umgebung
- [#*args*]: optional Anzahl der Parameter (≤ 9)
- [*default*]: optional Angabe eines Default-Werts für den 1. Parameter
- {*begcmd*}: Befehle die bei \begin{name} ausgeführt werden
- {*endcmd*}: Befehle die bei \end{name} ausgeführt werden

Zugriff auf Parameter nur in *begcmd* möglich!

Abhilfe: z.B. save boxes oder einen temporären Befehl definieren.

Beispiel

```
\newcommand\mycmd{}
\newenvironment{myenv}[2][Anfang]                    %
    {#1\renewcommand\mycmd[1]{#2 (##1)}}             %
    {\mycmd{Demo}}
```

```
\begin{myenv}{Ende}       und \end{myenv}

\begin{myenv}[begin]{end}
  aussen
  \begin{myenv} [[] {]} innen \end{myenv}
  aussen
\end{myenv}
```

Anfang und Ende (Demo)

begin aussen [innen] (Demo) aussen end (Demo)

Nützliche Pakete

calc

Erlaubt Arithmetik mit Längen und Zahlen (z.B. Countern).

ifthenelse

\ifthenelse{*test*}{*then*}{*else*}
\whiledo{*test*}{*command*}
String- und Längenvergleiche
boolsche Variablen und Arithmetik