

The `keycommand` package

Florent Chervet <florent.chervet@free.fr>

July 22, 2009

Abstract

`keycommand` provides an easy way to define commands or environments with optional keys. It provides `\newkeycommand` and its relative `\renewkeycommand`, `\newkeyenvironment`, `\renewkeyenvironment` and `\providekeycommand`. Moreover it is possible to define key-commands using `\def`, `\edef`, `\gdef` or `\xdef` via the `\keycmd` prefix. This package requires and is based on the package `kvsetkeys` by Heiko Oberdiek. It is designed to work with ϵ -TeX for the code uses the primitives `\unexpanded` and `\protected`.

Contents

1	Introduction	1
1.1	User interface	1
1.2	Error messages	2
2	Implementation	3
2.1	Identification	3
2.2	Requirements	3
2.3	Defining keys	3
2.4	The <code>\keycmd</code> prefix	4
2.5	new key-commands	5
2.6	new key-environments	6
2.7	<code>kvsetkeys</code> correction	6
3	Example	7
4	References	8
5	Index	8

1 Introduction

1.1 User interface

With `keycommand` it becomes very easy to define commands with optional keys. Just say:

```
\newkeycommand\CommandWithKeys[kOne=default,...][2]{%  
    definition with \commandkey{kOne} etc. #1 and #2}
```

As far as the keys are optional, it is not allowed to have another optional parameter in a key-command.

This documentation is produced with the `DocStrip` utility.

→ To get the documentation, run (thrice): `pdflatex keycommand.dtx`
→ To get the package, run: `etex keycommand.dtx`

keycommand enables us to define key-environments as well, and provides:

```

\newkeycommand      \renewkeycommand
\newkeyenvironment  \renewkeyenvironment
and: \providekeycommand

```

Moreover, if you need (or prefer) the syntax of `\def` (or `\gdef`, `\edef`, `\xdef`) you shall refer to the section [The `\keycmd` prefix](#) (in [Implementation](#)).

`\newkeycommand {<control sequence>} [<key-value list>] [<number of args>] {<definition>}`

keycommand allow L^AT_EX users to define commands with key in a easy way. Better is a small example than a long talking: let's define a command `\Rule` whose `width`, `thickness` and `raise` can be specified as keys.

With keycommand we just have to say:

```

\newkeycommand\Rule[raise=.4ex,width=1em,thick=.4pt][1]{%
  \rule[\commandkey{raise}]{\commandkey{width}}{\commandkey{thick}}
  #1%
  \rule[\commandkey{raise}]{\commandkey{width}}{\commandkey{thick}}}

```

which defines the keys `width`, `thick` and `raise` with their default values (if not specified): `1em`, `.4pt` and `.4ex`. Now `\Rule` can be used as follow:

```

1: \Rule[width=2em]{hello}      → width=2em,thick=.4pt,raise=.4ex
2: \Rule[thick=1pt,width=2em]{hello} → width=2em,thick=1pt,raise=.4ex
3: \Rule{hello}                 → width=1em,thick=.4pt,raise=.4ex
4: \Rule[thick=2pt,raise=1ex]{hello} → width=1em,thick=2pt,raise=1ex
  et cætera.

```

They will produce:

```

1:  —hello—
2:  —hello—
3:  —hello—
4:  ■hello■

```

`\newkeyenvironment {<envir name>} [<key-values pairs>] [<number of args>] {<begin>} {<end>}`

In the same way, you may define environments with optional keys as follow:

```

\newkeyenvironment{EnvirWithKeys}[kOne=default value,...][n]
{ commands at begin EnvirWithKeys }
{ commands at end EnvirWithKeys }

```

Where *n* is the number of mandatory other arguments (*ie* without keys), if any.

A example of a key-environment is left in the file: `keycommand-example.tex`.

1.2 Error messages

If you use the command `\Rule` (defined in [1.1](#)) with a key say: `height` that has not been declared at the definition of the key-command, you will get an error message like this:

```

There was no key ‘height’
in the keycommand \Rule!
see the definition of the keycommand.

```

However, if you use `\commandkey{height}` in the definition of `\Rule` you will not have any error message: `\commandkey{height}` will just be expanded into `\relax` at `\Rule` expansion time.

To be honest, when you redefine a key-command using `\renewkeycommand` or `\renewkeyenvironment` or `\keycmd\def` the keys defined before for the old command are undefined. This way you have the expected error message in all cases.

★ ★ ★

2 Implementation

2.1 Identification

This package is intended to use with L^AT_EX so we don't check if it is loaded twice.

```
1 (*package)
2 \NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
3 [2005/12/01]% LaTeX must be 2005/12/01 or younger (see kvsetkeys.dtx).
4 \ProvidesPackage{keycommand}
5 [2009/07/22 an easy way to define commands with keys]
```

2.2 Requirements

The package is based on `kvsetkeys`. `kvsetkeys` is more reliable than `keyval` as far as spaces and bracket (groups) are concerned. Please refer to the `kvsetkeys` documentation for more information.

As long as we use ε -T_EX primitives in `keycommand` we also load the `etex` package in order to get an error message if ε -T_EX is not running.

```
6 \RequirePackage{etex,kvsetkeys}
```

2.3 Defining keys

`\kcmd@keydef` To handle the case where the key-command was defined as `\global`, we have to define keys globally too. Therefore, we can't use the `\define@key` macro of the `keyval` package.

```
7 \def\kcmd@keydef#1#2#3#4#5{% #1=\global(eventually), #2=command, #3=family, #4=key, #5=def
8   #1\expandafter\edef\csname kcmd@keys\string#2\endcsname{%
9     \csname kcmd@keys\string#2\endcsname,#4}%
10  #1\@namedef{KV@#3@#4@default\expandafter}\expandafter{%
11    \csname KV@#3@#4\endcsname{#5}}%
12  #1\@namedef{KV@#3@#4}##1}
```

`\kcmd@definekey` In order to define keys, we will use the `\kv@parse` macro (`kvsetkeys`). Therefore, the only requirement is to define the *processor*.

```
13 \def\kcmd@definekey#1#2#3#4#5{%
14   \begingroup\edef\@tempa{\endgroup
15     \unexpanded{\kcmd@keydef{#1}{#2}{#3}{#4}{#5}}{\def
16       \expandafter\noexpand\csname #3@#4\endcsname{###1}}%
17   }\@tempa}
```

`\kcmd@undefinekeys` Now in case we redefine a key-command, we would like the old keys (*ie* the keys associated to the old definition of the command) to be cleared, undefined. That's the job of `\kcmd@undefinekeys`:

```
18 \def\kcmd@undefinekeys#1#2{%
19   \ifundefined{kcmd@keys\string#2}
20     \relax
21   {\expandafter\@for\expandafter\kcmp@temp
22     \expandafter:\expandafter=\csname kcmd@keys\string#2\endcsname
23     \do{#1\expandafter\let
```

```

24      \csname KV@kcmd@\expandafter\@gobble\string #2@\kcmp@temp @default\endcsname
25      \@undefined
26      #1\expandafter\let
27      \csname KV@kcmd@\expandafter\@gobble\string #2@\kcmp@temp\endcsname
28      \@undefined}}%
29      #1\@namedef{kcmd@keys\string#2}{\@gobble}}

```

2.4 The \keycmd prefix

\keycmd acts just like a (expandable) prefix for \def or \edef:

The syntax is:

\keycmd*	* optional
(\protected \long \outer \global)	optional
(\def \gdef \edef \xdef)	required
\CommandWithKeys	Name of the command to define
[key=value pairs]	keys and default values
<i>parameter string</i>	optional (same syntax as \def)
{ <i>replacement text</i> }	required

Without the star form, \long is assumed; but it can always be specified as \long after \keycmd. Example:

```
\keycmd\gdef\CommandWithKeys[kOne=defOne,kTwo=defTwo]#1#2{...}
```

\keycmd First we have to read the prefixes, if any:

```
30 \DeclareRobustCommand\keycmd{\@star@or@long\kcmd@prefix}
```

\kcmd@prefix This macro reads the prefixes one after another (including the \def word) and stores them in \kcmd@prfx. We open a group for all declarations will be local until the final definition of \CommandWithKeys.

```

31 \def\kcmd@prefix{\begingroup
32   \let\kcmd@gb1\@empty
33   \def\kcmd@prfx{\longrel@x}%
34   \futurelet\x\kcmd@pref@x}
35 \def\kcmd@pref@x{%
36   \ifx\x\@sptoken      \let\next\kcmd@pref@x@space
37   \else                \let\next\kcmd@addto@prfx
38     \ifx\x\long
39     \else\ifx\x\outer
40     \else\ifx\x\protected
41     \else\ifx\x\global  \def\kcmd@gb1{\global}%
42     \else
43       \def\kcmd@pref@x@next{\expandafter\key@cmd\noexpand}%
44       \ifx\x\def
45       \else\ifx\x\edef
46       \else\ifx\x\gdef  \def\kcmd@gb1{\global}%
47       \else\ifx\x\xdef  \def\kcmd@gb1{\global}%
48       \else \def\next{\kcmd@error{a \string\def\space
49         was expected after \string\keycmd.}}%
50       \fi\fi\fi\fi
51       \fi\fi\fi\fi
52   \fi\next}
53 \begingroup\def\:{\kcmd@pref@x@space}
54 \expandafter\gdef\:{\futurelet\x\kcmd@pref@x}
55 \endgroup
56 \def\kcmd@addto@prfx#1{\expandafter\def\expandafter
57   \kcmd@prfx\expandafter{\kcmd@prfx#1}\kcmd@pref@x@next}
58 \def\kcmd@pref@x@next{\futurelet\x\kcmd@pref@x}
59 \def\kcmd@error#1{\@latexerror{#1}\@ehd}

```

\key@cmd, \@keycmd \key@cmd will take the name of the command to be defined as its first argument and checks if there are keys-values placed between brackets just after. Then, \@keycmd

will check if the command is definable; if it is not, then the switch `\@tempswa` is set to false: the definition is processed nevertheless with a basic `\def`, but the group (opened in `\kcmd@prefix`) is closed just after the assignment, canceling everything out.

```

60 \def\key@cmd#1{\@testopt{\expandafter\@keycmd\noexpand#1}{}}
61 \def\@keycmd#1[#2]{\@tempswafalse\expandafter
62   \rc@ifdefinable\noexpand#1{\@tempswatrue}%
63   \if@tempswa
64     \let#1=\relax
65     \def\next{\kcmd@def#1{#2}}%
66   \else \def\next{\afterassignment\endgroup
67     \def\kcmd@notdefinable}%
68   \fi\next}

```

`\kcmd@relaxify` We temporarily assign the value `\relax` to some commands in order to avoid so many `\noexpand` during the expanded definition of `\kcmd@def`:

```

69 \def\kcmd@relaxify{%
70   \let\commandkey\relax
71   \let\kvsetkeys\relax
72   \let\kv@parse\relax
73   \let\@testopt\relax
74   \let\kv@set@family@handler\relax
75   \let\kcmd@undefinekeys\relax
76   \let\kcmd@keyerr\relax
77   \let\kcmd@definekey\relax
78   \def\###1{\expandafter\noexpand\csname ##1\endcsname}}

```

`\kcmd@def` `\kcmd@def` will define the keys and the command itself:

```

79 \def\kcmd@def#1#2{% #1=\Command, #2=key-values
80   \edef\kcmd@fam{\kcmd@expandafter\@gobble\string#1}%
81   \kcmd@relaxify
82   \edef\kcmd@def@##1{\endgroup
83     \kv@set@family@handler{\kcmd@fam}{\kcmd@keyerr{#1}{#####1}{#####2}}%
84     \kcmd@undefinekeys{\kcmd@gbl}{#1}%
85     \kv@parse{##1}{\kcmd@definekey{\kcmd@gbl}{#1}{\kcmd@fam}}%
86     \kcmd@gbl\protected\def#1{% entry point
87       \protected\def\commandkey#####1{\noexpand\csname\kcmd@fam @#####1\endcsname}%
88       \def" \kcmd\string#1"#####1[#####2]{%
89         \kvsetkeys{\kcmd@fam}{#####1,#####2}%
90         \string #1}%
91       \@testopt{" \kcmd\string#1"{##1}}{}}%
92   \let\commandkey\relax
93   \expandafter\expandafter\expandafter
94   \expandafter\expandafter\expandafter
95   \expandafter\kcmd@prfx"\string#1"%
96   }\kcmd@def@{#2}}

```

`\kcmd@keyerr` `\kcmd@keyerr` is the default handler for key-commands. It is called whenever the user wants to use a key that was not defined in the key-command:

```

97 \def\kcmd@keyerr#1#2#3{%
98   \let\wheremsg\empty
99   \ifdefined\trcg@where\trcg@where{#1}\fi
100   \@latexerror{There was no key "#2" \MessageBreak
101     in the keycommand \string#1!\MessageBreak
102     see the definition of the keycommand (or environment)\wheremsg}\@ehd}

```

2.5 new key-commands

`\newkeycommand` The `\expandafter... \noexpand` trick is there in case the command to (re-)define had been defined as `\outer` before...

```

103 \DeclareRobustCommand\newkeycommand{\@star@or@long
104   \expandafter\new@keycommand\noexpand}
105 \DeclareRobustCommand\renewkeycommand{\@star@or@long
106   \expandafter\renew@keycommand\noexpand}
107 \DeclareRobustCommand\providekeycommand{\@star@or@long
108   \expandafter\provide@keycommand\noexpand}

109 \def\new@keycommand#1{\@testopt{\expandafter\@newkeycommand\noexpand#1}{}}
110 \def\@newkeycommand#1[#2]{\begingroup
111   \@tempwafalse\expandafter
112   \@ifdefinable\noexpand#1{\@tempwattrue}%
113   \if@tempwa
114     \let#1=\relax
115     \let\kcmd@gb1\@empty
116     \def\kcmd@prfx##1{\unexpanded{\@testopt{\@argdef{##1}}0}}%
117     \def\next{\kcmd@def#1[#2]}%
118   \else \def\next{\afterassignment\endgroup
119     \def\kcmd@notdefinable}%
120   \fi\next}

121 \def\renew@keycommand#1{\begingroup
122   \escapechar\m@ne\edef\@gtempa{\string#1}}%
123   \expandafter\@ifundefined\@gtempa
124     {\endgroup\@latex@error{\noexpand#1undefined}\@ehc}
125   \endgroup
126   \let\@ifdefinable\@rc@ifdefinable
127   \expandafter\new@keycommand\noexpand#1}

128 \def\provide@keycommand#1{\begingroup
129   \escapechar\m@ne\edef\@gtempa{\string#1}}%
130   \expandafter\@ifundefined\@gtempa
131     {\endgroup\new@keycommand#1}
132     {\endgroup\let\kcmd@notdefinable\noexpand
133       \renew@keycommand\kcmd@notdefinable}}

```

2.6 new key-environments

```

134 \DeclareRobustCommand\newkeyenvironment{\@star@or@long\new@keyenvironment}
135 \DeclareRobustCommand\renewkeyenvironment{\@star@or@long\renew@keyenvironment}

136 \def\new@keyenvironment#1{\@testopt{\@newkeyenva{#1}}{}}
137 \def\@newkeyenva#1[#2]{%
138   \kernel@ifnextchar [{\@newkeyenvb{#1} [{#2}]} {\@newkeyenv{#1} [{#2}] [0]}}
139 \def\@newkeyenvb#1[#2] [#3]{\@newkeyenv{#1} [{#2}] [#3]}
140 \def\@newkeyenv#1#2#3#4{%
141   \@ifundefined{#1}%
142     {\expandafter\let\csname #1\expandafter\endcsname
143       \csname end#1\endcsname}%
144     \relax
145   \expandafter\@newkeycommand
146     \csname #1\endcsname#2{#3}%
147   \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}

148 \def\renew@keyenvironment#1{%
149   \@ifundefined{#1}%
150     {\@latex@error{Environment #1 undefined}\@ehc
151     }\relax
152   \expandafter\let\csname#1\endcsname\relax
153   \expandafter\let\csname\expandafter\string\csname #1\endcsname\endcsname\relax
154   \expandafter\let\csname end#1\endcsname\relax
155   \new@keyenvironment{#1}}

```

2.7 kvsetkeys correction

`\kv@normalize` inserts an unwilling blank space due to a small error in the code. Mr Oberdiek has been asked for correction...

```
156 \def\kv@normalize#1{%
157   \begingroup
158     \toks@{,#1,}%
159     \KVS@Comma
160     \KVS@SpaceComma{ }%
161     \KVS@CommaSpace
162     \KVS@CommaComma
163     \KVS@Equals
164     \KVS@SpaceEquals{ }%
165     \KVS@EqualsSpace% <original code>\KVS@EqualsSpace{ }</original code>
166     \xdef\KVS@Global{\the\toks@}%
167   \endgroup
168   \let\kv@list\KVS@Global
169 }

170 </package>
```

3 Example

```
171 <*example>
172 \ProvidesFile{keycommand-example}
173 \documentclass{article}
174 \usepackage[T1]{fontenc}
175 \usepackage[latin1]{inputenc}
176 \usepackage[american]{babel}
177 \usepackage{keycommand}
178 \usepackage{framed}
179 %
180 \makeatletter
181 \parindent\z@
182 \newkeycommand\Rule[raise=.4ex,width=1em,thick=.4pt][1]{%
183   \rule{\commandkey{raise}}{\commandkey{width}}{\commandkey{thick}}}%
184   #1%
185   \rule{\commandkey{raise}}{\commandkey{width}}{\commandkey{thick}}}%
186
187 \newkeycommand\charleads[sep=1][2]{%
188   \ifhmode\else\leavevmode\fi\setbox\@tempboxa\hbox{#2}\@tempdima=1.584\wd\@tempboxa%
189   \cleaders\hb@xt@\commandkey{sep}\@tempdima{\hss\box\@tempboxa\hss}#1%
190   \setbox\@tempboxa\box\voidb@x}
191 \newcommand\charfill[1][\charleads[{#1}]{\hfill\kern\z@}]
192 \newcommand\charfil[1][\charleads[{#1}]{\hfil\kern\z@}]
193 %
194 \newkeyenvironment{dblruled}[first=.4pt,second=.4pt,sep=1pt,left=\z@]{%
195   \def\FrameCommand{%
196     \vrule\@width\commandkey{first}%
197     \hskip\commandkey{sep}
198     \vrule\@width\commandkey{second}%
199     \hspace{\commandkey{left}}}%
200   \parindent\z@
201   \MakeFramed {\advance\hsize-\width \FrameRestore}}
202   {\endMakeFramed}
203 %
204 \makeatother
205 \begin{document}
206 \title{This is {\tt keycommand-example.tex}}
207 \author{Florent Chervet}
208 \date{July 22, 2009}
209 \maketitle
210
211 \section{Example of a keycommand : \texttt{\string\Rule}}
212
213 \begin{tabular*}{\textwidth}{rl}
214 \verb+\Rule[width=2em]{hello}+:&\Rule[width=2em]{hello}\cr
215 \verb+\Rule[thick=1pt,width=2em]{hello}+:&\Rule[thick=1pt,width=2em]{hello}\cr
216 \verb+\Rule{hello}+:&\Rule{hello}\cr
```

```

217 \verb+\Rule[thick=1pt,raise=1ex]{hello}+:&\Rule[thick=1pt,raise=1ex]{hello}
218 \end{tabular*}
219
220 \section{Example of a keycommand : \texttt{\string\charfill}}
221
222 \begin{tabular*}\textwidth{rp{.4\textwidth}}
223 \verb+\charfill{$\star$}+: & \charfill{$\star$}\cr
224 \verb+\charfill[sep=2]{$\star$}+: & \charfill[sep=2]{$\star$} \\
225 \verb+\charfill[sep=.7]{\textasteriskcentered}+: & \charfill[sep=.7]{\textasteriskcentered}
226 \end{tabular*}
227
228
229 \section{Example of a keyenvironment : \texttt{dblruled}}
230
231 \verb+\begin{dblruled}+\par
232 \verb+ test for dblruded key-environment\par+\par
233 \verb+ test for dblruded key-environment\par+\par
234 \verb+ test for dblruded key-environment+\par
235 \verb+\end{dblruded}+
236
237 \begin{dblruded}
238 test for dblruded key-environment\par
239 test for dblruded key-environment\par
240 test for dblruded key-environment
241 \end{dblruded}
242
243
244 \verb+\begin{dblruded}[first=4pt,sep=2pt,second=.6pt,left=.2em]+\par
245 \verb+ test for dblruded key-environment\par+\par
246 \verb+ test for dblruded key-environment\par+\par
247 \verb+ test for dblruded key-environment+\par
248 \verb+\end{dblruded}+
249
250 \begin{dblruded}[first=4pt,sep=2pt,second=.6pt,left=.2em]
251 test for dblruded key-environment\par
252 test for dblruded key-environment\par
253 test for dblruded key-environment
254 \end{dblruded}
255
256
257 \end{document}
258 \example

```

4 References

- [1] Heiko Oberdiek: *The kvsetkeys package*; 2007/09/29 v1.3; [CTAN:macros/latex/contrib/oberdiek/kvsetkeys.dtx](#).
- [2] David Carlisle: *The keyval package*; 1999/03/16 v1.13; [CTAN:macros/latex/required/graphics/keyval.dtx](#).

5 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols	
<code>\@argdef</code>	116
<code>\@ehc</code>	124, 150
<code>\@ehd</code>	59, 102
<code>\@ifdefinable</code>	112, 126
<code>\@ifundefined</code>	19, 123, 130, 141, 149
<code>\@newkeycommand</code>	109, 110, 145
<code>\@newkeyenv</code>	138, 139, 140
<code>\@newkeyenva</code>	136, 137
<code>\@newkeyenvb</code>	138, 139
<code>\@rc@ifdefinable</code>	62, 126
<code>\@sptoken</code>	36

<code>\@star@or@long</code>	30, 103, 105, 107, 134, 135	<code>\kcmd@relaxify</code>	69, 81
<code>\@undefined</code>	25, 28	<code>\kcmd@undefinekeys</code>	18, 75, 84
A			
<code>\afterassignment</code>	66, 118	<code>\kcmp@temp</code>	21, 24, 27
C			
<code>\charfil</code>	192	<code>\key@cmd</code>	43
<code>\charfill</code>	191, 220, 223, 224, 225	<code>\key@cmd, _@keycmd</code>	60
<code>\charleads</code>	187, 191, 192	<code>\keycmd</code>	30, 49
<code>\cleaders</code>	189	<code>\kv@list</code>	168
<code>\commandkey</code>	70, 87, 92, 183, 185, 189, 196, 197, 198, 199	<code>\kv@normalize</code>	156
D			
<code>\DeclareRobustCommand</code>	30, 103, 105, 107, 134, 135	<code>\kv@parse</code>	72, 85
F			
<code>\futurelet</code>	34, 54, 58	<code>\kv@set@family@handler</code>	74, 83
K			
<code>\kcmd@addto@prfx</code>	37, 56	<code>\kvsetkeys</code>	71, 89
<code>\kcmd@def</code>	65, 79, 117	L	
<code>\kcmd@def@</code>	82, 96	<code>\l@ngrel@x</code>	33, 147
<code>\kcmd@definekey</code>	13, 77, 85	N	
<code>\kcmd@error</code>	48, 59	<code>\new@keycommand</code>	104, 109, 127, 131
<code>\kcmd@fam</code>	80, 83, 85, 87, 89	<code>\new@keyenvironment</code>	134, 136, 155
<code>\kcmd@gbl</code>	32, 41, 46, 47, 84, 85, 86, 115	<code>\newkeycommand</code>	2, 103, 182, 187
<code>\kcmd@keydef</code>	7, 15	<code>\newkeyenvironment</code>	2, 134, 194
<code>\kcmd@keyerr</code>	76, 83, 97	P	
<code>\kcmd@notdefinable</code>	67, 119, 132, 133	<code>\protected</code>	40, 86, 87
<code>\kcmd@pref@x</code>	34, 35, 54, 58	<code>\provide@keycommand</code>	108, 128
<code>\kcmd@pref@x@next</code>	43, 57, 58	<code>\providekeycommand</code>	107
<code>\kcmd@pref@x@space</code>	36, 53	R	
<code>\kcmd@prefix</code>	30, 31	<code>\renew@keycommand</code>	106, 121, 133
<code>\kcmd@prfx</code>	33, 57, 95, 116	<code>\renew@keyenvironment</code>	135, 148
		<code>\renewkeycommand</code>	105
		<code>\renewkeyenvironment</code>	135
		<code>\Rule</code>	182, 211, 214, 215, 216, 217
		<code>\rule</code>	183, 185
		U	
		<code>\unexpanded</code>	15, 116