The **keycommand*** package

an easy way to define commands with optional keys.

Florent Chervet <florent.chervet@free.fr>

2009/08/26 – version 2.ζ

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 primives \unexpanded and \protected.

Contents

1 Introduction **2**
- 1.1 User interface . . . 2
- 1.2 Error messages . . . 3
- 1.3 Test if a key is defined . . . 3

2 Implementation **3**
- 2.1 Identification . . . 3
- 2.2 Requirements . . . 4
- 2.3 Syntactical enhancement . . . 4
- 2.4 Defining keys . . . 4
- 2.5 The **\keycmd** prefix . . . 5
- 2.6 new key-commands . . . 7
- 2.7 new key-environments . . . 7
- 2.8 Test if keys are blank . . . 8

3 Example **9**

4 History **10**
- [2009/08/26 v2.z] . . . 10
- [2009/08/04 v2.e-] . . . 10
- [2009/07/22 v1.0] . . . 10

5 References **10**

6 Index **11**

* keycommand: CTAN:macros/latex/contrib/keycommand
This documentation is produced with the DocStrip utility.

⟶ To get the documentation, run (thrice): `pdflatex keycommand.dtx`
To get the index, run: `makeindex -s gind.ist keycommand.idx`
⟶ To get the package, run: `etex keycommand.dtx`

The .dtx file is embedded into this pdf file thank to embedfile by H. Oberdiek.

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 keycommand.

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 LaTeX users to define commands with optional keys 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▬

Nota bene: it is also possible to give a key a default value which is the value of another key. For example:

```
\newkeycommand\CmdKey[alpha=hello, beta=\commandkey{alpha}]{...}
```

When called as: `\CmdKey[alpha=world]`, the key `beta` will then have the same value: `world`.

`\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.

1.3 Test if a key is defined

When you define a key command you may let the default value of a key empty. Then, you may wish to expand some commands only if the key has been given by the user (with a non empty value). This can be achieved using the macro `\ifcommandkey`:

`\ifcommandkey` {⟨*key name*⟩}{⟨*commands if key is blank*⟩}{⟨*commands if key is NOT blank*⟩}

★ ★ ★

2 Implementation

2.1 Identification

This package is intended to use with LaTeX 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 v2.e- an easy way to define commands with optional 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-TEX primitives in keycommand we also load the etex package in order to get an error message if ε-TEX is not running.

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

2.3 Syntactical enhancement

We will define a shortcut for `\expandafter\noexpand\csname...\endcsname` all along this package.

```
\edef\kcmd@AtEnd{\catcode34 \the\catcode34}% "
\catcode34 4
```

2.4 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.

```
\def\kcmd@keydef#1#2#3#4#5{% #1=\global, #2=command, #3=family, #4=key, #5=def
   #1\expandafter\edef\csname kcmd@keys\string#2\endcsname{%
      \csname kcmd@keys\string#2\endcsname,#4}%
   #1\@namedef{KV@#3@#4@default\expandafter}\expandafter{%
      \csname KV@#3@#4\endcsname{#5}}%
   #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*.

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

```
\def\kcmd@undefinekeys#1#2{%
   \@ifundefined{kcmd@keys\string#2}
      \relax
      {\expandafter\@for\expandafter\kcmp@temp
         \expandafter:\expandafter=\csname kcmd@keys\string#2\endcsname
      \do{#1\expandafter\let
            \csname KV@kcmd@\expandafter\@gobble\string #2@\kcmp@temp @default\endcsname
            \@undefined
         #1\expandafter\let
            \csname KV@kcmd@\expandafter\@gobble\string #2@\kcmp@temp\endcsname
            \@undefined}}%
   #1\@namedef{kcmd@keys\string#2}{\@gobble}}
```

2.5 The \keycmd prefix

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

The syntax is:

	\keycmd*␣	* optional
possibly	(\long␣|\global␣|\protected␣|\outer␣)	optional (zero or more)
in a macro	(\def|\edef|\gdef|\xdef)	required: see below
	control sequence	required
	[key=value pairs]	keys and default values
	Parameter string	optional
	⟨*Replacement text*⟩	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:

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

\kcmd@prefix This is the prefixes scanner: 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.

```
\def\kcmd@prefix{\begingroup
   \let\kcmd@gbl\@empty
   \def\kcmd@prfx{\l@ngrel@x}%
   \futurelet\x\kcmd@@prefix}
\def\kcmd@@prefix{%
   \let\kcmd@next@addto\kcmd@next@prefix
   \ifx\x\@sptoken          \let\next\kcmd@space@prefix
   \else                    \let\next\kcmd@addto@prfx
      \ifx\x\long
      \else\ifx\x\outer
      \else\ifx\x\protected
      \else\ifx\x\global    \def\kcmd@gbl{\global}%
      \else
                            \def\kcmd@next@addto{\expandafter\key@cmd\noexpand}%
         \ifx\x\def
         \else\ifx\x\edef
         \else\ifx\x\gdef   \def\kcmd@gbl{\global}%
         \else\ifx\x\xdef   \def\kcmd@gbl{\global}%
         \else              \let\kcmd@next@addto\kcmd@next@prefix
            \ifx\y\x\kcmd@error@prefix
            \else\let\y\x
            \fi
            \let\next\kcmd@expand@prefix
         \fi\fi\fi\fi
      \fi\fi\fi\fi
   \fi\next}
\def\kcmd@next@prefix{\futurelet\x\kcmd@@prefix}
\def\kcmd@expand@prefix{\expandafter\kcmd@next@prefix}
\def\kcmd@addto@prfx#1{\let\y\@undefined
   \expandafter\def\expandafter\kcmd@prfx
         \expandafter{\kcmd@prfx#1}\kcmd@next@addto}
\expandafter\def\expandafter\kcmd@space@prefix\space{\kcmd@next@prefix}
\def\kcmd@error@prefix{\@latex@error{A \string\def\space
```

```
   (or \string\gdef\space or \string\edef\space or \string\xdef)\MessageBreak
  was expected after \string\keycmd\MessageBreak
  I found a \meaning\x!\MessageBreak
  see keycommand documentation for more information}\@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.

```
\def\key@cmd#1{\@testopt{\expandafter\@keycmd\noexpand#1}{}}
\def\@keycmd#1[#2]{\@tempswafalse\expandafter
   \@rc@ifdefinable\noexpand#1{\@tempswatrue}%
   \if@tempswa
      \let#1=\relax
      \def\next{\kcmd@def#1{#2}}%
   \else \def\next{\afterassignment\endgroup
      \def\kcmd@notdefinable}%
   \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@:

```
\def\kcmd@relaxify{%
   \let\commandkey\relax
   \let\kvsetkeys\relax
   \let\kv@parse\relax
   \let\@testopt\relax
   \let\kv@set@family@handler\relax
   \let\kcmd@undefinekeys\relax
   \let\kcmd@keyerr\relax
   \let\kcmd@definekey\relax
   \def\"##1"{\expandafter\noexpand\csname ##1\endcsname}}
```

\kcmd@def

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

```
\def\kcmd@def#1#2{% #1=\Command, #2=key-values
   \edef\kcmd@fam{kcmd@\expandafter\@gobble\string#1}%
   \kcmd@relaxify
   \edef\kcmd@def@##1{\endgroup
      \kv@set@family@handler{\kcmd@fam}{\kcmd@keyerr{#1}{####1}{####2}}%
      \kcmd@undefinekeys{\kcmd@gbl}{#1}%
      \kv@parse{##1}{\kcmd@definekey{\kcmd@gbl}{#1}{\kcmd@fam}}%
      \kcmd@gbl\protected\def#1{% entry point
         \def\commandkey########1{\noexpand\csname\kcmd@fam @#########1\endcsname}%
         \def\"kcmd\string#1"########1[########2]{%
               \kvsetkeys{\kcmd@fam}{########1,########2}%
               \"\string #1"}%
         \@testopt{\"kcmd\string#1"{##1}}{}}%
      \let\commandkey\relax
      \expandafter\expandafter\expandafter
         \expandafter\expandafter\expandafter
            \expandafter\kcmd@prfx\"\string#1"%
   }\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:

```
\def\kcmd@keyerr#1#2#3{%
   \let\wheremsg\@empty
```

```
    \ifdefined\trcg@where\trcg@where{#1}\fi
    \@latex@error{There was no key "#2" \MessageBreak
       in the keycommand \string#1!\MessageBreak
       see the definition of the keycommand (or environment)\wheremsg}\@ehd}
```

2.6 new key-commands

\newkeycommand

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

```
\DeclareRobustCommand\newkeycommand{\@star@or@long
   {\expandafter\new@keycommand\noexpand}}
\DeclareRobustCommand\renewkeycommand{\@star@or@long
   {\expandafter\renew@keycommand\noexpand}}
\DeclareRobustCommand\providekeycommand{\@star@or@long
   {\expandafter\provide@keycommand\noexpand}}

\def\new@keycommand#1{\@testopt{\expandafter\@newkeycommand\noexpand#1}{}}
\def\@newkeycommand#1[#2]{\begingroup
   \@tempswafalse\expandafter
   \@ifdefinable\noexpand#1{\@tempswatrue}%
   \if@tempswa
      \let#1=\relax
      \let\kcmd@gbl\@empty
      \def\kcmd@prfx##1{\unexpanded{\@testopt{\@argdef{##1}}0}}%
      \def\next{\kcmd@def#1{#2}}%
   \else \def\next{\afterassignment\endgroup
      \def\kcmd@notdefinable}%
   \fi\next}

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

\def\provide@keycommand#1{\begingroup
   \escapechar\m@ne\edef\@gtempa{{\string#1}}%
   \expandafter\@ifundefined\@gtempa
      {\endgroup\new@keycommand#1}
      {\endgroup\let\kcmd@notdefinable\noexpand
      \renew@keycommand\kcmd@notdefinable}}
```

2.7 new key-environments

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

\def\new@keyenvironment#1{\@testopt{\@newkeyenva{#1}}{}}
\def\@newkeyenva#1[#2]{%
   \kernel@ifnextchar [{\@newkeyenvb{#1}[{#2}]}{\@newkeyenv{#1}{[{#2}][0]}}}
\def\@newkeyenvb#1[#2][#3]{\@newkeyenv{#1}{[{#2}][#3]}}
\def\@newkeyenv#1#2#3#4{%
   \@ifundefined{#1}%
      {\expandafter\let\csname #1\expandafter\endcsname
            \csname end#1\endcsname}%
      \relax
   \expandafter\@newkeycommand
      \csname #1\endcsname#2{#3}%
```

```
    \l@ngrel@x\expandafter\def\csname end#1\endcsname{#4}}

\def\renew@keyenvironment#1{%
  \@ifundefined{#1}%
     {\@latex@error{Environment #1 undefined}\@ehc
     }\relax
  \expandafter\let\csname#1\endcsname\relax
  \expandafter\let\csname\expandafter\string\csname #1\endcsname\endcsname\relax
  \expandafter\let\csname end#1\endcsname\relax
  \new@keyenvironment{#1}}
```

2.8 Test if keys are blank

First we need some helper macros:

```
\def\kcmd@afterelse#1\else#2\fi{\fi#1}
\def\kcmd@afterfi#1\fi{\fi#1}
```

`\expandnext` The following macros comes from the etextools[1] package (by F. Chervet):

```
\newcommand\kcmd@expandnext[2]{%
   \ifx#1\kcmd@expandnext
      \kcmd@afterelse\expandafter\expandafter\expandafter
            \expandafter\@kcmd@expandnext{#2}{\expandafter\expandafter\expandafter}%
   \else\kcmd@afterfi\expandafter#1\expandafter{#2}%
   \fi}
\long\def\@kcmd@expandnext#1#2#3{%
   \ifx#1\kcmd@expandnext
      \expandafter\kcmd@afterelse\expandafter\expandafter\expandafter
            \expandafter\@kcmd@expandnext{#3}{\expandafter#2#2}%
   \else
      \expandafter\kcmd@afterfi#2#1#2{#3}%
   \fi}
```

`\kcmd@expandonce` The following macro comes from the etoolbox[2] package (by P. Lehmann):

```
\def\kcmd@expandonce#1{\unexpanded\expandafter{#1}}
```

`\kcmd@ifblank` The following macro comes from the url[3] package:

```
\begingroup\catcode`\:=4\catcode`\&=4
\gdef\kcmd@ifblank#1{\kcmd@ifblank@#1&&\@secondoftwo\@firstoftwo:}
\gdef\kcmd@ifblank@#1#2&#3#4#5:{#4}
\endgroup
```

`\ifcommandkey`

```
\newcommand\ifcommandkey[3]{%
   \kcmd@expandnext\kcmd@expandnext\kcmd@expandnext\kcmd@expandnext\kcmd@expandnext
   \kcmd@expandnext\kcmd@expandnext\kcmd@expandnext\kcmd@expandnext\kcmd@expandnext
   \kcmd@expandnext\kcmd@expandnext\kcmd@expandnext\kcmd@ifblank{%
      \kcmd@expandnext\kcmd@expandnext\kcmd@expandonce{\commandkey{#1}}}
      {#3}
      {#2}}

\kcmd@AtEnd
⟨/package⟩
```

1. etextools: CTAN:macros/latex/contrib/etextools
2. etoolbox: CTAN:macros/latex/contrib/etoolbox
3. url: CTAN:macros/latex/contrib/misc/url.sty

3 Example

```
⟨*example⟩
\ProvidesFile{keycommand-example}
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[latin1]{inputenc}
\usepackage[american]{babel}
\usepackage{keycommand}
\usepackage{framed}
%
\makeatletter
\parindent\z@
\newkeycommand\Rule[raise=.4ex,width=1em,thick=.4pt][1]{%
   \rule[\commandkey{raise}]{\commandkey{width}}{\commandkey{thick}}%
   #1%
   \rule[\commandkey{raise}]{\commandkey{width}}{\commandkey{thick}}}

\newkeycommand\charleads[sep=1][2]{%
   \ifhmode\else\leavevmode\fi\setbox\@tempboxa\hbox{#2}\@tempdima=1.584\wd\@tempboxa%
   \cleaders\hb@xt@\commandkey{sep}\@tempdima{\hss\box\@tempboxa\hss}#1%
   \setbox\@tempboxa\box\voidb@x}
\newcommand\charfill[1][]{\charleads[{#1}]{\hfill\kern\z@}}
\newcommand\charfil[1][]{\charleads[{#1}]{\hfil\kern\z@}}
%
\newkeyenvironment{dblruled}[first=.4pt,second=.4pt,sep=1pt,left=\z@]{%
   \def\FrameCommand{%
      \vrule\@width\commandkey{first}%
      \hskip\commandkey{sep}
      \vrule\@width\commandkey{second}%
      \hspace{\commandkey{left}}}%
   \parindent\z@
   \MakeFramed {\advance\hsize-\width \FrameRestore}}
   {\endMakeFramed}
%
\makeatother
\begin{document}
\title{This is {\tt keycommand-example.tex}}
\author{Florent Chervet}
\date{July 22, 2009}
\maketitle

\section{Example of a keycommand : \texttt{\string\Rule}}

\begin{tabular*}\textwidth{rl}
\verb+\Rule[width=2em]{hello}+:&\Rule[width=2em]{hello}\cr
\verb+\Rule[thick=1pt,width=2em]{hello}+:&\Rule[thick=1pt,width=2em]{hello}\cr
\verb+\Rule{hello}+:&\Rule{hello}\cr
\verb+\Rule[thick=1pt,raise=1ex]{hello}+:&\Rule[thick=1pt,raise=1ex]{hello}
\end{tabular*}

\section{Example of a keycommand : \texttt{\string\charfill}}

\begin{tabular*}\textwidth{rp{.4\textwidth}}
\verb+\charfill{$\star$}+: & \charfill{$\star$}\cr
\verb+\charfill[sep=2]{$\star$}+: & \charfill[sep=2]{$\star$} \\
\verb+\charfill[sep=.7]{\textasteriskcentered}+: & \charfill[sep=.7]{\textasteriskcentered
\end{tabular*}


\section{Example of a keyenvironment : \texttt{dblruled}}

```

```
\verb+\begin{dblruled}+\par
\verb+   test for dblruled key-environment\par+\par
\verb+   test for dblruled key-environment\par+\par
\verb+   test for dblruled key-environment+\par
\verb+\end{dblruled}+

\begin{dblruled}
 test for dblruled key-environment\par
 test for dblruled key-environment\par
 test for dblruled key-environment
\end{dblruled}


\verb+\begin{dblruled}[first=4pt,sep=2pt,second=.6pt,left=.2em]+\par
\verb+   test for dblruled key-environment\par+\par
\verb+   test for dblruled key-environment\par+\par
\verb+   test for dblruled key-environment+\par
\verb+\end{dblruled}+

\begin{dblruled}[first=4pt,sep=2pt,second=.6pt,left=.2em]
 test for dblruled key-environment\par
 test for dblruled key-environment\par
 test for dblruled key-environment
\end{dblruled}


\end{document}
⟨/example⟩
```

4 History

[2009/08/26 v2.z]

-

[2009/08/04 v2.e-]

- Fix catcode of double quote (") in case user command had a double quote in its name...
- Add History to the documentation file
- Modify the prefixes scanner (it is now the same as the one of ltxnew[4]). Modify the documentation (KOMA-Script classe)

[2009/07/22 v1.0]

- First version.

5 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.

4. ltxnew: CTAN:macros/latex/contrib/ltxnew

6 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

\& 182
\@argdef 126
\@ehc 134, 160
\@ehd 69, 112
\@firstoftwo 183
\@ifdefinable 122, 136
\@ifundefined 21, 133, 140, 151, 159
\@kcmd@expandnext 171, 174, 177
\@newkeycommand 119, 120, 155
\@newkeyenv 148, 149, 150
\@newkeyenva 146, 147
\@newkeyenvb 148, 149
\@rc@ifdefinable 72, 136
\@secondoftwo 183
\@sptoken 39
\@star@or@long . . 32, 113, 115, 117, 144, 145
\@undefined 27, 30, 61

A

\afterassignment 76, 128

C

\catcode 7, 8, 182
\charfil 216
\charfill 215, 244, 247, 248, 249
\charleads 211, 215, 216
\cleaders 213
\commandkey 80, 97, 102, 190, 207, 209, 213, 220, 221, 222, 223

D

\DeclareRobustCommand 32, 113, 115, 117, 144, 145

E

\expandnext 168

F

\futurelet 36, 59

I

\ifcommandkey *3*, 186

K

\kcmd@@prefix 36, 37, 59
\kcmd@addto@prfx 40, 61
\kcmd@afterelse 166, 170, 176
\kcmd@afterfi 167, 172, 179
\kcmd@AtEnd 7, 193
\kcmd@def 75, 89, 127
\kcmd@def@ 92, 106
\kcmd@definekey 15, 87, 95
\kcmd@error@prefix 52, 65
\kcmd@expand@prefix 55, 60
\kcmd@expandnext 168, 169, 175, 187, 188, 189, 190
\kcmd@expandonce 181, 190
\kcmd@fam 90, 93, 95, 97, 99
\kcmd@gbl 34, 44, 49, 50, 94, 95, 96, 125
\kcmd@ifblank 182, 189
\kcmd@ifblank@ 183, 184
\kcmd@keydef 9, 17
\kcmd@keyerr 86, 93, 107
\kcmd@next@addto 38, 46, 51, 63
\kcmd@next@prefix 38, 51, 59, 60, 64
\kcmd@notdefinable 77, 129, 142, 143
\kcmd@prefix 32, 33
\kcmd@prfx 35, 62, 63, 105, 126
\kcmd@relaxify 79, 91
\kcmd@space@prefix 39, 64
\kcmd@undefinekeys 20, 85, 94
\kcmp@temp 23, 26, 29
\key@cmd 46
\key@cmd,␣\@keycmd 70
\keycmd 32, 67
\kv@parse 82, 95
\kv@set@family@handler 84, 93
\kvsetkeys 81, 99

L

\l@ngrel@x 35, 157

M

\meaning 68

N

\new@keycommand 114, 119, 137, 141
\new@keyenvironment 144, 146, 165
\newkeycommand *2*, 113, 206, 211
\newkeyenvironment *3*, 144, 218

P

\protected 43, 96
\provide@keycommand 118, 138
\providekeycommand 117

R

\renew@keycommand 116, 131, 143
\renew@keyenvironment 145, 158
\renewkeycommand 115
\renewkeyenvironment 145
\Rule 206, 235, 238, 239, 240, 241
\rule 207, 209

U

\unexpanded 17, 126, 181

Y

\y 52, 53, 61