The makecmds package*

Author: Peter Wilson, Herries Press
Maintainer: Will Robertson
`will dot robertson at latex-project dot org`

2009/09/03

Abstract

The makecmds package provides several additional commands along the lines of the traditional `\(re)newcommand` and friends.

Contents

1 Introduction

LaTeX provides commands like `\newcommand`, `\renewcommand` and `\providecommand`. Similarly there are the `\newenvironment` and `\renewenvironment` commands but no matching `\provideenvironment` command. This package provides additional command making commands like `\makecommand`, which is equivalent to the pair `\providecommand` and `\renewcommand`, or `\provideenvironment` to match with `\providecommand`.

This manual is typeset according to the conventions of the LaTeX DOCSTRIP utility which enables the automatic extraction of the LaTeX macro source files [GMS94].

Section 2 describes the usage of the makecmds package and commented source code is in Section 3.

---

*This file has version number v1.0a, last revised 2009/09/03.

Table 1: Macro making commands

Command	Action	Condition
`\new...{`⟨*name*⟩`}`⟨*...*⟩	Creates new macro definition for ⟨*name*⟩	⟨*name*⟩ must not have been previously defined
`\renew...{`⟨*name*⟩`}`⟨*...*⟩	Redefines the macro definition for ⟨*name*⟩	⟨*name*⟩ must have been previously defined
`\provide...{`⟨*name*⟩`}`⟨*...*⟩	If ⟨*name*⟩ has not been previously defined, creates new macro definition for ⟨*name*⟩ otherwise does nothing	
`\make...{`⟨*name*⟩`}`⟨*...*⟩	Creates or changes the macro definition for ⟨*name*⟩	

2 The makecmds package

Table 1 shows the general kinds of macro making commands that are available in LaTeX and the additional forms provided by the makecmds package.

This package extends the basic commands with `\make...` versions of `command`, `environment`, `length` and `counter` macros, and `\provide...` versions for `environment`, `length` and `counter` macros.

2.1 Options

Several commands of the form `\make...` are provided by the package. For example, `\makecommand{\foo}{...}` will define a new command `\foo` if it does not already exist, otherwise it will silently redefine `\foo`. The package takes a single option, namely warn. This option will generate a warning message whenever one of the `\make...` commands redefines an existing definition.

2.2 Commands

`\makecommand`

The `\makecommand` command takes the same arguments as the `\(re)newcommand` family does, and likewise there is also a starred version of the command. `\makecommand{\foo}...}` is equivalent to first calling `\providecommand{\foo}{}` and then `\renewcommand{\foo}...}`. That is, it defines `\foo` irrespective of whether or not `\foo` has been previously defined.

`\provideenvironment`
`\makeenvironment`

The `\provideenvironment` macro is like `\providecommand` except that it applies to an environment instead of a command. Similarly `\makeenvironment` is analagous to `\makecommand`. Both macros take the same arguments as the `\newenvironment` command.

`\providelength`
`\makelength`

These are `\provide...` and `\make...` versions of the `\newlength` command. They take the same argument as `\newlength`.

`\providecounter`
`\makecounter`

These are `\provide...` and `\make...` versions of the `\newcounter` command.

They take the same arguments as `\newcounter`.

3 The class code

To try and avoid name clashes, all the internal commands include the string `m@k`. I have used `\def` instead of `\newcommand` in the following code as I have previously coded variants of the new commands in some LaTeX classes and packages, and therefore want to overwrite those if this package happens to be used in conjunction with the pre-existing macro definitions.

3.1 Preliminaries

Announce the name and version of the package, which requires LaTeX 2_ε.

```
⟨*usc⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{makecmds}[2009/09/03 v1.0a extra command making commands]

```

`\ifm@kwarn` This is used for implementing the warn option.

```
\newif\ifm@kwarn
  \m@kwarnfalse
```

Now declare and process the options.

```
\DeclareOption{warn}{\m@kwarntrue}
\ProcessOptions\relax

```

3.2 The commands

Much of the code consists of copying and then making minor alterations to the code in the LaTeX kernel.

`\makecommand`
`\m@ke@command`
The `\make...` version of `\newcommand`, originally defined in `ltdefns.dtx`. `\m@ke@command` is the internal code that does the work.

```
\def\makecommand{\@star@or@long\m@ke@command}
\def\m@ke@command#1{%
  \ifx #1\undefined\else
    \ifm@kwarn
      \PackageWarning{makecmds}{Redefining command `\protect#1'}
    \fi
  \fi
  \let\@ifdefinable\@rc@ifdefinable
  \new@command#1}

```

`\provideenvironment`
`\m@kprovide@environment`

The environment version of `\providecommand`, and is based on the code in `ltdefns.dtx`. The internal version is `\m@kprovide@environment`. The kernel command `\new@environment` actually creates a new environment.

```
\def\provideenvironment{%
  \@star@or@long\m@kprovide@environment}
\def\m@kprovide@environment#1{%
  \@ifundefined{#1}{%
    \expandafter\let\csname#1\endcsname\relax
    \expandafter\let\csname end#1\endcsname\relax
    \new@environment{#1}}{\m@kdiscardenvargs{#1}}
}
```

`\m@kdiscardenvargs`
`\m@kenva`
`\m@kenvb`

`\m@kdiscardenvargs` gathers up the arguments to `\provideenvironment`, and then proceeds to discard them.

```
\def\m@kdiscardenvargs#1{%
  \@testopt{\m@kenva#1}0}
\def\m@kenva#1[#2]{%
  \@ifnextchar [{\m@kenvb#1[#2]}{\m@kthrowenv{#1}{[#2]}}}
\def\m@kenvb#1[#2][#3]{\m@kthrowenv{#1}{[#2][#3]}}
```

`\m@kthrowenv`

`\m@kthrowenv` takes all the possible (4) arguments to a `\(re)newenvironment` command and throws them away.

```
\def\m@kthrowenv#1#2#3#4{}

```

`\makeenvironment`
`\m@ke@environment`

The `\make...` version of `\newenvironment`, originally defined in `ltdefns.dtx`. `\m@ke@environment` is the internal code that does the work.

```
\def\makeenvironment{\@star@or@long\m@ke@environment}
\def\m@ke@environment#1{%
  \@ifundefined{#1}{\new@environment{#1}}{%
    \ifm@kwarn
      \PackageWarning{makecmds}{Redefining environment ‘#1’}
    \fi
    \renew@environment{#1}}
}

```

`\providelength`
`\makelength`

These are `\provide...` and `\make...` versions of `\newlength` (from `ltlength.dtx`).

```
\def\providelength#1{%
  \ifx #1\undefined
    \newlength{#1}
  \fi
}
\def\makelength#1{%
  \ifx #1\undefined
    \newlength{#1}
  \else
    \ifm@kwarn
      \PackageWarning{makecmds}{Redefining length ‘\protect#1’}
    \fi
    \newskip#1
  \fi
}

```

`\providecounter`
`\makecounter`

These are `\provide...` and `\make...` versions of `\newcounter` (from `ltcounts.dtx`).

```
\def\providecounter#1{%
  \@ifundefined{c@#1}{\newcounter{#1}}{%
    \@ifnextchar[{\m@k@gobbleendoptarg}{}}%
}
\def\makecounter#1{%
  \expandafter\ifx \csname c@#1\endcsname \undefined
  \else
    \ifm@kwarn
      \PackageWarning{makecmds}{Redefining counter `#1'}
    \fi
  \fi
  \@definecounter{#1}%
  \@ifnextchar[{\@newctr{#1}}{}
}

```

`\m@k@gobbleendoptarg`

A macro that discards an optional argument coming at the end of a list of (optional) arguments (i.e., the tokens ... `[optarg]`).

```
\def\m@k@gobbleendoptarg[#1]{}

```

The end of this package.

```
⟨/usc⟩
```

References

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, 1994.

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.