multidef: quick definition of multiple similar LaTeX macros

Nicolas Markey

2016/03/14

Abstract

multidef provides a succinct way of defining series of macros having similar definitions. While this can be achieved quite easily with a little of TeX programming, I found no package offering a command similar to the `\multidef` command defined in the present package.

1 Usage

The command `\multidef` can be used to quickly define several similar macros. For instance:

```
\multidef{\textit{#1}}{apple,banana,strb->strawberry}
```

After this single line, you can use commands `\apple`, `\banana` and `\strb` to write their names in italics: *apple*, *banana*, and *strawberry*.

The package has several features, such as

- adding prefix/suffix to all command names;
- raising errors and/or warnings if some commands are already defined;
- allowing commands with arguments.

For example, after writing

```
\multidef[arg=1]{\ensuremath{\mathsf{#1}(##1)}}{fst->first,lst->last}
```

so that you can write `\fst{w}` to write $\mathsf{first}(w)$.

2 Examples

I very often use the `\mathcal` command to get calligraphic-font letters in math mode. With multidef I now simply write

```
\multidef{\ensuremath{\mathcal{#1}}}{A-Z}
```

and write `\calG` to write $\mathcal{G}$. Here `A-Z` is a shorthand for the 26 letters of the basic Latin alphabet.

In the same way, I can define

```
\usepackage{dsfonts}
\let\mathbb\mathds
\makeatletter
\newcommand\optbb[1]{%
  \@ifnextchar+{\ensuremath{\mathbb{#1}_{\geq 0}}\@gobble}
    {\@ifnextchar*{\ensuremath{\mathbb{#1}_{>0}}\@gobble}
      {\ensuremath{\mathbb{#1}}}}}
\makeatother
\multidef[prefix=bb]{\optbb{#1}}{A-Z,one->1}
```

and then `\bbR+` writes $\mathbb{R}_{\geq 0}$, while `$\bbone_{S}$` outputs $\mathbb{1}_S$.

As a last example, we can use multidef to redefine all `\...name` (e.g. `\refname`, `\partname`, ...) commands succinctly. For this, we would deactivate the error and warning mechanisms, as we know we are redefining those macros:

```
\multidef[noerr,nowarn,suffix=name]{#1}{ref->R\'ef\'erences,
  part->Partie, appendix->Annexe,...}
```

Then `\refname` contains 'Références'.

3 The code

We use xkeyval to handle package and command options. The package has two options, `noerr` and `nowarn`. The former tells multidef not to raise an error when redefining a command (default to true). The latter tells not to raise a warning (defaults to false). Thus the default behaviour is to only raise a warning when redefining a command. Notice that the keys `noerr` and `nowarn` are also available as arguments of the `\multidef` command, to change the selected behaviour locally.

noerr
nowarn

```
\RequirePackage{xkeyval}
\define@boolkeys{mdef}{noerr,nowarn}[true]
\DeclareOptionX{noerr}[true]{\setkeys{mdef}{noerr=#1}}
\DeclareOptionX{nowarn}[true]{\setkeys{mdef}{nowarn=#1}}
\ExecuteOptionsX{noerr=false,nowarn=false}
\ProcessOptionsX
\ifKV@mdef@noerr
\presetkeys{mdef}{noerr=true}{}
\else
\presetkeys{mdef}{noerr=false}{}
\fi
\ifKV@mdef@nowarn
\presetkeys{mdef}{nowarn=true}{}
\else
```

```
\presetkeys{mdef}{nowarn=false}{}
\fi
```

We have five main other keys to be used by the `\multidef` command:

- `prefix` and `suffix` define the prefix and suffix to be used in the name of the command. These keys have equivalent shorthands `p` and `s`.
- `arg` (and the equivalent `args`) can be used to define the number of arguments of the series of commands to be defined.
- `long` and `global` can be used to define `\long` and `\global` macros.

prefix
suffix
arg
long
global

```
\define@key{mdef}{prefix}{\def\@mdprefix{#1}}
\define@key{mdef}{p}{\def\@mdprefix{#1}}
\define@key{mdef}{suffix}{\def\@mdsuffix{#1}}
\define@key{mdef}{s}{\def\@mdsuffix{#1}}
\define@key{mdef}{arg}{\def\@mdargs{#1}}
\define@key{mdef}{args}{\def\@mdargs{#1}}
\define@boolkeys{mdef}{long,global}[true]
\presetkeys{mdef}
          {p=,s=,prefix=,suffix=,long=false,global=false,arg=0,args=0}{}
```

We define shorthands for defining series of commands indexed by letters of the alphabet. Can be useful sometimes...

\@mdef@az
\@mdef@AZ
\@mdef@alphabet
\@mdef@Alphabet

```
\def\@mdef@az{a-z}
\def\@mdef@AZ{A-Z}
\def\@mdef@alphabet{a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z}
\def\@mdef@Alphabet{A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z}
```

We now define `\multidef`: it will first deal with option keys, store the definition of the commands being defined, and then call its friend `\@mdef`, whose role is to deal with each entry in the comma-separated list.

\multidef

```
\newcommand\multidef[3][]{%
  \setkeys{mdef}{#1}%
  \def\@mdef@com##1{#2}%
  \@mdef#3,\@end}
```

Command `\@mdef` takes the first item in the comma-separated list, and first checks if it is a shorthand `a-z` or `A-Z`. If not, it calls `\@@mdef` on the first item, and `\@mdef` on the remainder of the list.

\@mdef

```
\def\@mdef #1,#2\@end{\expandafter\def\expandafter\@arg
    \expandafter{\romannumeral-\expandafter`\expandafter\.#1}%
  \ifx\@arg\@mdef@az
```

```
    \expandafter\@mdef \@mdef@alphabet,\@end
  \else
    \ifx\@arg\@mdef@AZ
      \expandafter\@mdef \@mdef@Alphabet,\@end
    \else
      \expandafter\@@mdef\@arg->->->\@end
    \fi
  \fi
  \def\@arg{#2}%
  \ifx\@arg\@empty\else\@mdef #2\@end\fi}
```

Now, command `\@@mdef` checks if the command name already exists, and issues errors and warning if needed. It also calls `\@@@mdef` with two arguments: the first one is the string to be used in the name of the command, the second one is the string to be used in the definition. The latter might be the empty string in case both strings are supposed to be the same.

`\@@mdef`
`\@mdef@redeftok`
`\@mdef@comma`
`\@mdef@finalwarn`

```
\newtoks\@mdef@redeftok
\def\@mdef@comma{}
\def\@@mdef#1->#2->#3\@end{%
  \@ifundefined{\@mdprefix#1\@mdsuffix}
    {\@@@mdef{#1}{#2}}
    {\edef\@mdef@redef{\the\@mdef@redeftok\@mdef@comma
        \@backslashchar\@mdprefix#1\@mdsuffix}
     \def\@mdef@comma{, }
     \@mdef@redeftok=\expandafter{\@mdef@redef}
     \ifKV@mdef@noerr
       \@@@mdef{#1}{#2}%
       \ifKV@mdef@nowarn\else
         \PackageWarning{multidef}
           {command \expandafter\noexpand\csname\@mdprefix#1\@mdsuffix\endcsname
             redefined}
       \fi
     \else
       \PackageError{multidef}
         {command \expandafter\noexpand\csname\@mdprefix#1\@mdsuffix\endcsname
           already defined}\@ehc
     \fi
     \ifKV@mdef@nowarn\else
       \@ifundefined{@mdwarnonce}
         {\def\@mdwarnonce{}%
          \@mdef@finalwarn}
         {}
     \fi}
}
\def\@mdef@finalwarn{%
  \AtEndDocument{\PackageWarningNoLine{multidef}{There were
    redefined commands (\the\@mdef@redeftok)}}}
```

Finally, `\@@@mdef` calls `\@mdef@def` with the appropriate arguments. This is where the commands are really defined. The definition uses `\@yargd@f`, following the definition of `\newcommand` in LaTeX.

`\@@@mdef`
`\@mdef@def`

```
\def\@@@mdef#1#2{\def\@arg@{#2}%
  \ifx\@arg@\@empty
    \@mdef@def{#1}{#1}%
  \else
    \@mdef@def{#1}{#2}%
  \fi}
\def\@mdef@def#1#2{%
    \let\reserved@b\@gobble
    \ifKV@mdef@global\let\@mdglobal\global\else\let\@mdglobal\relax\fi
    \ifKV@mdef@long\let\@mdlong\long\else\let\@mdlong\relax\fi
    \def\l@ngrel@x{\@mdlong\@mdglobal}
  \expandafter\expandafter\expandafter\@yargd@f\expandafter\@mdargs\csname
  \@mdprefix#1\@mdsuffix\expandafter\endcsname\expandafter{\@mdef@com{#2}}
}
```