The rgltxdoc package*

Richard Grewe
r-g+tex@posteo.net

January 12, 2019

1 Introduction

This package combines several other packages and defines additional macros and environments for the purpose of documenting LaTeX code. The package mainly serves the purpose of combining the preferences used in the author's package documentations. However, others can use the package as well. Compatibility between versions cannot be guaranteed, however.

2 Basic Dependencies

Generally, the documentation can be compiled with pdfLaTeX and with LuaLaTeX. Other processors are untested. There is no assertion as to how close the pdfLaTeX and LuaLaTeX results are to each other.

```
\RequirePackage{ifluatex}
```

The etoolbox package is used to simplify some of the package's code.

```
\RequirePackage{etoolbox}
```

3 Documentation Input

The documentation is expected to be written in UTF-8 and in US-english language.

```
\ifbool{luatex}{
  \RequirePackage[utf8]{luainputenc}
  \RequirePackage{polyglossia}
  \setmainlanguage[variant=american]{english}
}{
  \RequirePackage[utf8]{inputenc}
  \RequirePackage[english]{babel}
}
```

*This document corresponds to rgltxdoc v1, dated 2019/01/05. The package is available online at http://www.ctan.org/pkg/rgltxdoc and https://github.com/Ri-Ga/rgltxdoc.

4 General Appearance

Code in this section determines the general appearance of documentation text and is not specific to documenting LaTeX code.

4.1 Page Layout

For the page layout, A4 is used for the paper size. Border correction established wider left margins for typesetting long macro names. The DIV value is tuned to make the lines wide enough to support at least 72 characters in the package documentation code.

```
\RequirePackage[a4paper,twoside=false]{geometry}
\RequirePackage[DIV=9,BCOR=2.25cm]{typearea}
```

4.2 Fonts

For the font, Latin Modern is used. Particularly, a light version of the typewriter font is used, such that highlighting in listings is possible via a bold font series.

```
\ifbool{luatex}{
  \RequirePackage{fontspec}
  \setmainfont[SmallCapsFont={* Caps}]{Latin Modern Roman}
  \setsansfont{Latin Modern Sans}
  \setmonofont[Scale=MatchLowercase,
               SmallCapsFont={Latin Modern Mono Caps}]
    {Latin Modern Mono Light}
}{
  \RequirePackage[T1]{fontenc}
  \RequirePackage[lighttt]{lmodern}
```

With just the above code, a construct like `\cs{foo\meta{bar}}` for documenting parameter-dependent macro names fails due to missing fonts. The following two lines fix this. The first line ensures that the typewriter font is loaded (via an `\hbox` with typewriter text that is not actually displayed) and the second line declares the required font shape (see https://tex.stackexchange.com/questions/234003/italic-font-in-lmodern-lighttt).

```
  \bgroup\setbox\z@\hbox{\ttfamily ignore}\egroup
  \DeclareFontShape{T1}{lmtt}{m}{it}{<->sub*lmtt/m/sl}{}
}
```

Finally, microtype is used for small font improvements.

```
\RequirePackage{microtype}
```

We simplify quoting names through the csquotes package and register `"` to produce double opening/closing quotation marks.

```
\RequirePackage{csquotes}
\MakeOuterQuote{"}
```

4.3 Document Structure

For the most part, documentations are structured as usual, through a title as well as sections and sub-sections and so forth. The following two packages improve

the possibilities for using lists in documentation and visually improve the index through a two-column layout.

```
\RequirePackage{enumitem}
\RequirePackage[columns=2]{idxlayout}
```

The cleveref and varioref packages shall be used for referencing structural entities, such as sections and figures. Hyperlinks are enabled through hypdoc.

```
\RequirePackage{varioref}
\RequirePackage{hypdoc}
\RequirePackage[capitalise,noabbrev,nameinlink]{cleveref}
```

5 Documenting Things

This package builds on the doc package for several documentation macros, such as `\marg`, `\oarg`, and `\meta`.

```
\RequirePackage{doc}
```

5.1 Macros and Environments

The main macros here, `\NiceDescribeMacro` and `\NiceDescribeEnv`, are references to `\DescribeMacro` and `\DescribeEnv` of the doc package, with which they share the purpose. The main difference is the appearance in that the "nice" macros include the argument list.

`\NiceDescribeMacro`[⟨*idx*⟩]{⟨*macro*⟩}{⟨*parameters*⟩}
`\NiceDescribeMacros`{⟨*n*⟩}[⟨*idx*$_1$⟩]{⟨*macro*$_1$⟩}{⟨*params*$_1$⟩}…[⟨*idx*$_n$⟩]{⟨*macro*$_n$⟩}{⟨*params*$_n$⟩}

These macros produce a description header for a single macro or, respectively, for multiple macros. The above two lines are an example of such a description header, which is produced by the following code:

```
\NiceDescribeMacros{2}
  {\NiceDescribeMacro}{\oarg{idx}\marg{macro}\marg{parameters}}
  {\NiceDescribeMacros}{\marg{n}
    \oarg{idx$_1$}\marg{macro$_1$}\marg{params$_1$}\ldots
    \oarg{idx$_n$}\marg{macro$_n$}\marg{params$_n$}}
```

The arguments to the macro are described below:

⟨*n*⟩	This argument specifies the number of macros to be described.
⟨*macro*⟩, ⟨*macro*$_1$⟩,…, ⟨*macro*$_n$⟩	These arguments specify the macros for which a description header shall be produced.
⟨*parameters*⟩, ⟨*params*$_1$⟩,…, ⟨*params*$_n$⟩	These arguments take the sequence that specifies all optional and mandatory arguments of the respective ⟨*macro*⟩. Typically, these would be sequences of `\oarg` and `\marg` instances.
⟨*idx*⟩, ⟨*idx*$_1$⟩,…, ⟨*idx*$_N$⟩	These optional arguments arguments specify which index entries shall be documented, if they differ from the respective ⟨*macro*⟩ parameters. This, for instance allows ⟨*macro*⟩ to be "`\foo(*)`" whereas the ⟨*idx*⟩ parameter could be "`\foo,\foo*`".

\NiceDescribeEnv[⟨*idx*⟩]{⟨*environment*⟩}{⟨*parameters*⟩}
\NiceDescribeEnvs{⟨*n*⟩}[⟨*idx*$_1$⟩]{⟨*env*$_1$⟩}{⟨*params*$_1$⟩}...[⟨*idx*$_n$⟩]{⟨*env*$_n$⟩}{⟨*params*$_n$⟩}

These macros are the counterparts of \NiceDescribeMacro and, respectively, \NiceDescribeMacros when it comes to LaTeX environments. The ⟨*environment*⟩ (resp. ⟨*env*$_1$⟩ to ⟨*env*$_n$⟩) parameters are the names of the respective environments. A usage example can be found in the implementation part of Section 5.2 on page 6.

\NiceDescribeCounte[⟨*idx*⟩]{⟨*counter*⟩}{⟨*qualifiers*⟩}
\NiceDescribeCounters{⟨*n*⟩}[⟨*idx*$_1$⟩]{⟨*ctr*$_1$⟩}{⟨*qual*$_1$⟩}...[⟨*idx*$_n$⟩]{⟨*ctr*$_n$⟩}{⟨*qual*$_n$⟩}

These macros are analogous to the above macros, but aimed for documenting LaTeX counters.

\NewNiceDescription{⟨*type*⟩}{⟨*efmt*⟩}{⟨*afmt*⟩}{⟨*icmd*⟩}

This macro is used internally for defining the above macros and can be used for defining new types of entity descriptions. The following table describes the arguments of the macro.

⟨*type*⟩	The ⟨*type*⟩ argument is the name of the type of entities.
⟨*efmt*⟩	The ⟨*efmt*⟩ argument is TeX code that formats the entities in the margin. It can – and should – reference the positional parameter #1, through which it is passed the name of the entities.
⟨*afmt*⟩	The ⟨*afmt*⟩ argument is TeX code that formats the arguments or qualifiers of the entities in the body of the documentation. Analogous to ⟨*efmt*⟩, also ⟨*afmt*⟩ receives the arguments/qualifiers through the positional parameter #1.
⟨*icmd*⟩	The ⟨*icmd*⟩ argument is TeX code that adds a usage entry for the entity to the index. It takes one argument, through which ⟨*icmd*⟩ is passed the entity name.

A usage example for \NewNiceDescription can be found in the implementation below.

Implementation

\NewNiceDescription The \NewNiceDescription{⟨*type*⟩}{⟨*efmt*⟩}{⟨*afmt*⟩}{⟨*icmd*⟩} macro defines the \NiceDescribe⟨*type*⟩ and \NiceDescribe⟨*type*⟩s macros and saves the ⟨*efmt*⟩ and ⟨*afmt*⟩ parameters for use by \NiceDescribe⟨*type*⟩s.

```
\newcommand\NewNiceDescription[4]{%
  \expandafter\newcommand\csname NiceDescribe#1\endcsname{%
    \csname NiceDescribe#1s\endcsname{1}}%
  \expandafter\newcommand\csname NiceDescribe#1s\endcsname{%
    \rgltxdoc@Desc
      {\csuse{rgltxdoc@@efmt@#1}}
      {\csuse{rgltxdoc@@afmt@#1}}
      {#4}}%
  \csdef{rgltxdoc@@efmt@#1}##1{#2}%
  \csdef{rgltxdoc@@afmt@#1}##1{#3}}
```

\NiceDescribeMacro
\NiceDescribeMacros

Macro names are formatted detokenized through \string. Arguments are formatted as is. For the index, \SpecialUsageIndex of the doc package is used.

```
\NewNiceDescription{Macro}{\string#1}{#1}{\SpecialUsageIndex}
```

\NiceDescribeEnv
\NiceDescribeEnvs

Environment names are formatted with a gray \begin and \end. The arguments of environments are formatted as is. For the index, \SpecialEnvIndex of the doc package is used.

```
\NewNiceDescription{Env}
  {\textcolor{gray}{\cs{begin}}\cmarg{#1}\\
   \textcolor{gray}{\cs{end}}\cmarg{#1}}
  {#1}{\SpecialEnvIndex}
```

\NiceDescribeCounter
\NiceDescribeCounters

Counter names are formatted as is. Arguments or qualifiers should usually not be present for counters, but if provided, they would be formatted as is. The index entry is produced through \SpecialOtherIndex (see its documentation below).

```
\NewNiceDescription{Counter}{#1}{#1}
                    {\SpecialOtherIndex{counter}{counters}}
```

\SpecialOtherIndex

The \SpecialOtherIndex{⟨*type*⟩}{⟨*types*⟩}{⟨*name*⟩} macro adds an index entry of the given ⟨*type*⟩ (with plural form ⟨*types*⟩) and given ⟨*name*⟩. The macro is a straightforward generalization of \SpecialEnvIndex.

```
\newcommand\SpecialOtherIndex[3]{\@bsphack
  \index{#3\actualchar{\protect\ttfamily#3}
    (#1)\encapchar usage}%
  \index{#2:\levelchar#3\actualchar
    {\protect\ttfamily#3}\encapchar usage}\@esphack}
```

\rgltxdoc@DescRec

The \rgltxdoc@Desc{⟨*efmt*⟩}{⟨*afmt*⟩}{⟨*icmd*⟩}{⟨*n*⟩}[⟨*idx*⟩]{⟨*entity*⟩}{⟨*args*⟩} macro formats a description header for ⟨*n*⟩ entities, of which the first are specified through ⟨*idx*⟩, ⟨*entity*⟩, and ⟨*args*⟩. The margin parts are formatted through the ⟨*efmt*⟩{⟨*entity*⟩} macro, the parts in the text body through the ⟨*afmt*⟩{⟨*args*⟩} macro. The index entries are created through the ⟨*icmd*⟩{⟨*idx*⟩} macro. In its implementation, \rgltxdoc@Desc builds on \pbox from the pbox package. It uses \rgltxdoc@DescRec and \rgltxdoc@DescRec@i (both with the same argument lists) for the parsing of arguments and for recursively grabbing the arguments for the ⟨*n*⟩ entities. At first, \rgltxdoc@Desc creates some vertical space above a list of description headers. Afterwards it starts the recursion.

```
\RequirePackage{pbox}
\newcommand\rgltxdoc@Desc{\medskip\par\noindent\rgltxdoc@DescRec}
\newcommand\rgltxdoc@DescRec[4]{%
  \@ifnextchar[%]
    {\rgltxdoc@DescRec@i{#1}{#2}{#3}{#4}}%
    {\rgltxdoc@DescRec@i{#1}{#2}{#3}{#4}[]}}
\def\rgltxdoc@DescRec@i#1#2#3#4[#5]#6#7{%
```

The following code creates the “margin” text (more precisely, a box to the left of the text) and the ⟨*args*⟩ next to it.

```
  \leavevmode\null\hbox to\z@{\hss%
    \pbox[t]{3\marginparwidth}{\ttfamily #1{#6}}%
```

If there is no ⟨*args*⟩, then the margin part is moved towards the left by a \quad.

```
    \ifstrempty{#7}{\quad}{}%
  #2{#7}\relax
```

Next, the index entries are created, through the comma-separated ⟨*idx*⟩ if this optional argument is given.

```
  \ifstrempty{#5}%
    {#3{#6}}%
    {\forcsvlist{#3}{#5}}%
```

Next, we check whether ⟨*n*⟩ > 1 and recurse, after a line break, if this is satisfied.

```
  \ifnumgreater{#4}{1}%
    {\\\rgltxdoc@DescRec{#1}{#2}{#3}{#4-1}}%
```

Finally, the following code ends a list of description headers, taking into account that an empty ⟨*args*⟩ allows the documentation text to already start in the same line as the "margin" text.

```
    {\ifstrempty{#7}{}{\smallskip\par\noindent}\ignorespaces}}
```

5.2 Arguments, Keys, and Values

Longer descriptions of macro/environment arguments as well as of keys (in key-value lists) and special values can be typeset in tables. For a common appearance, the keyvaltable package is used.

`\begin{KeyValTable}{KeyDesc}`
`\end{KeyValTable}`

This table is used for describing keys in key-value lists. It has three columns: `key`, `desc`, and `default`. The former two have the obvious meaning. The latter allows for specifying a default value for the key that is used when the key is not provided.

`\begin{KeyValTable}{ValDesc}`
`\end{KeyValTable}`

This table is used for describing special values (constants). It has two columns, `val` and `desc`, with their obvious meaning.

`\begin{KeyValTable}{ArgDesc}`
`\end{KeyValTable}`

This table is used for describing arguments of macros and environments in a structured fashion. It has two columns, `arg` and `desc`. Examples of this kind of table can be found in Section 5.1.

Implementation The keyvaltable package is used for creating the tables that document keys, values etc.

```
\RequirePackage{keyvaltable}
\kvtSet{headbg=black!10,rowbg=white..black!5}
```

The following code defines the table types. The code should be self-explanatory in terms of which columns exist and what their alignment and purpose is.

```
\NewKeyValTable{KeyDesc}{%
  key:  align=l,    format=\texttt, head=\textbf{Key};
```

```
  desc: align=X,    head=\textbf{Description and Possible Values};
  default: align=l, format=\texttt, head=\textbf{Default};
}
\NewKeyValTable[showhead=false]{ValDesc}{%
  val:  align=l,    format=\texttt, head=\textbf{Value};
  desc: align=X,                    head=\textbf{Description};
}
\NewKeyValTable[showhead=false]{ArgDesc}{%
  arg:  align=l, head=\textbf{Argument};
  desc: align=X, head=\textbf{Description};
}
```

5.3 Individual Entities

\env The \env{⟨*environment*⟩} macro is the counterpart of \cs for environment names instead of command names.

```
\newcommand\env[1]{\texttt{#1}}
```

\pkgname The \pkgname⟨*package-name*⟩ macro typesets package names in a uniform font (sans-serif). Moreover, the package checks whether the package actually exists, in order to identify embarrassing typos in the package name.

```
\newrobustcmd\pkgname[1]{%
  \IfFileExists{#1.sty}
    {\textsf{#1}}
    {\rgltxdoc@err{Package `#1' not found. Spelling?}}}
```

\cmarg \coarg The \cmarg{⟨*const-arg*⟩} and \coarg{⟨*const-arg*⟩} macros are counterparts for \marg and \oarg. They format constant argument values, though.

```
\newcommand\cmarg[1]{\mbox{\texttt{\string{#1\string}}}}
\newcommand\coarg[1]{\mbox{\texttt{[#1]}}}
```

The following enables references to various LaTeX tools in the common formatting of their names.

```
\RequirePackage{hologo}
```

6 Typesetting Examples

For typesetting examples, the showexpl package is used. Some specific settings for the appearance of the example listings are defined and some auxiliary macros simplify special examples.

Generally, code examples shall be typeset in one of two ways:

1. through `lstlisting` environments, if only code shall be displayed but no visualization of the code's output;
2. through `LTXexample` environments, if the code as well as its output shall be displayed.

Below follows an example of `LTXexample` that uses some of the features provided by rgltxdoc on top of showexpl: Labels/references and sections.

```
\section{Test}
\label{sec:test}
\cref{sec:test} has number~\ref{sec:test}.
```

1 Test

Section 1 has number 1.

The following code performs the setup for both (because `LTXexample` builds on `lstlisting`).

```
\RequirePackage{showexpl}
\lstset{%
  gobble=2,
  frame=trbl,
  backgroundcolor=\color{black!5!white},
  explpreset={%
    numbers=none, columns=flexible, basicstyle=\footnotesize\ttfamily},
  numbers=none, columns=flexible, basicstyle=\footnotesize\ttfamily,
  preset={\rgltxdoc@ExampleFix\rgltxdoc@SaveSecs\small\sffamily},
  overhang=2cm,
  pos=r,
  captionpos=b}
```

The following enables references to `LTXexample` and `lstlisting` environments through `\cref` and `\vref`.

```
\crefname{lstlisting}{Listing}{Listings}
```

The following adds the `morepreset` key to `listing` environments, to allow for extending `preset` code rather than overwriting it.

```
\lst@Key{morepreset}\relax{\appto\SX@preset{#1}}
```

`\rgltxdoc@ExampleFix` The `\rgltxdoc@ExampleFix` macro performs some setup to enable, to some extent, functionality that showexpl disables or does not implement. Concretely,

- the macro simulates labels and references, as long as labels are only referenced after they have been defined (in `LTXexample` environments, the normal label and ref mechanism is otherwise disabled);
- the macro re-enables the default `\marginpar` macro, which is disabled by `LTXexample` presumably due to its suboptimal appearance; for the cases in which the appearance can be justified, the macro is enabled.

```
\newcommand\rgltxdoc@ExampleFix{%
```

The fake `\label[`⟨*type*⟩`]{`⟨*label*⟩`}` macro takes the optional ⟨*type*⟩, as the cleveref package defines it. The macro first saves the current label value in a global macro for basic `\refs`. Then, for `\crefs`, the macro also stores the label's type, either from ⟨*type*⟩ or from cleveref's routines.

```
  \renewcommand\label[2][]{%
    \global\csletcs{rgltxdoc@@lbl@##2}{@currentlabel}%
    \ifstrempty{##1}
      {\csxdef{rgltxdoc@@lbltype@##2}{\rgltxdoc@curlbltype}}
      {\csgdef{rgltxdoc@@lbltype@##2}{##1}}}%
```

The `\ref` and `\cref` macros simply use the values stored by `\label`. Note that the multitude of further cleveref and varioref macros, e.g., `\crefrange` are currently

not implemented. They would need to be defined when there is actual demand for them.

```
  \def\ref##1{\csuse{rgltxdoc@@lbl@##1}}%
  \def\cref##1{%
    \csuse{cref@\csuse{rgltxdoc@@lbltype@##1}@name}~\ref{##1}}%
  \let\marginpar=\rgltxdoc@@marginpar
}
\let\rgltxdoc@@marginpar=\marginpar
```

The `\rgltxdoc@curlbltype` and `\rgltxdoc@curlbltype@i` macros are auxiliary macros for parsing the content of `\cref@currentlabel`, as set by the cleveref package. The first bracketet value in the content is the label type we're interested in here. If there's no current label type, we silently use the empty string.

```
\def\rgltxdoc@curlbltype{%
  \@ifundefined{cref@currentlabel}{}
    {\expandafter\rgltxdoc@curlbltype@i\cref@currentlabel\@nil}}
\def\rgltxdoc@curlbltype@i[#1][#2][#3]#4\@nil{#1}
```

`\rgltxdoc@SaveSecs` `\rgltxdoc@RestoreSecs`

The `\rgltxdoc@SaveSecs` macro saves the section counters and the `\rgltxdoc@RestoreSecs` macro restores the values of the section counters. This allows one to use sectioning commands in code examples without interfering with the section numbering in the documentation.

```
\newcommand\rgltxdoc@SaveSecs{%
  \@for\SC:=chapter,section,subsection,subsubsection\do{%
    \@ifundefined{c@\SC}{}
      {\csedef{rgltx@@ctr@\SC}{\the\value{\SC}}%
       \setcounter{\SC}{0}}}}
\newcommand\rgltxdoc@RestoreSecs{%
  \@for\SC:=chapter,section,subsection,subsubsection\do{%
    \@ifundefined{c@\SC}{}
      {\setcounter{\SC}{\csuse{rgltx@@ctr@\SC}}}}}
\patchcmd{\SX@resultInput}{\par}{\rgltxdoc@RestoreSecs\par}
  {}
  {\rgltxdoc@warn{Could not patch showexpl to reset section counters.}}
```

7 Shared Internal Code

`\rgltxdoc@err` `\rgltxdoc@warn`

The `\rgltxdoc@err{`⟨*error*⟩`}` macro raises the given ⟨*error*⟩. The `\rgltxdoc@warn{`⟨*warning*⟩`}` macro raises the given ⟨*warning*⟩.

```
\newcommand\rgltxdoc@err[1]{%
  \PackageError{rgltxdoc}{#1}{}}
\newcommand\rgltxdoc@warn[1]{%
  \PackageWarning{rgltxdoc}{#1}{}}
```

Change History

v1

General: Initial version 1

Index