The rhxterm Package
Version 1.12

Ralf Hemmecke

2005/11/12

1 Introduction

The rhxterm package consist of two parts. One part defines a framework to define special commands that hyperlink and index text parts. It is descriped in Section 4. The other part uses the framework in order to actually define such commands like \defineterm and \useterm. This is described in Section 2.

When reading new texts one sometimes comes across new notions that might have been defined some pages earlier. It is always a pain and actually time consuming to browse through the text in order to find the place where the notion was defined. It arose the desire to generate hyperlinks to the definition of such notions and also generate an index of the places in text where the notion is defined or used.

In short, we want the following behaviour.

- `\defineterm{TLI}`: Print `TLI` in text and index and use it as a label.
- `\defineterm{L@TI}`: Print `TI` in text and index, but use `L` for sorting the index. Furthermore `L` is used as label intended to be put into \label.
- `\defineterm[T]{LI}` Print `T` in text and use `LI` for the index and the label.
- `\defineterm[T]{L@I}`: Print `T` in text and use `I` for the index. The label part is made of `L`.

Similarly \useterm should be used. It is intended that \defineterm highlights the notion in some way and \useterm generates a hyperlink to this definition.

By the implementation given below, in particular the implementation of \rhxtermSplitAtOther, the argument inside the braces must not contain any `&` character.

2 The Definition of \defineterm and \useterm

Our initial idea was that a notions that is newly defined should appear in a special font in the text. Additionally it should also appear in the index. Furthermore, whenever this new notion is used in other places it should be hyperlinked to the place where it is defined. So we wanted to have two commands

\defineterm and \useterm that do this with a nice and easy-to-remember syntax. The syntax is very similar to the syntax of the \index command, but the semantics is a little different, since we did not want to repeat information in the text. Basically, we define a command \rhxtermNewCommand which takes two arguments. Take as an example the actual definitions of \defineterm and \useterm.

```
⟨Definition of defineterm and useterm⟩≡
  \rhxtermNewCommand{\defineterm}{\rhxDefineTerm}
  \rhxtermNewCommand{\useterm}{\rhxUseTerm}
```

It defines the first argument in terms of the second with the following semantics that we demonstrate by the \defineterm command.

Whenever \defineterm is used in the text, its arguments are translated to three arguments and \rhxDefineTerm is applied to them. So we have the following equivalences.

`\defineterm{TLI}`	`\rhxDefineTerm{TLI}{TLI}{TLI}`
`\defineterm{L@TI}`	`\rhxDefineTerm{L}{TI}{L@TI}`
`\defineterm[T]{LI}`	`\rhxDefineTerm{LI}{T}{LI}`
`\defineterm[T]{L@I}`	`\rhxDefineTerm{L}{T}{L@I}`

This translation makes it easy to actually define \rhxDefineTerm and \rhxUseTerm since these commands always take 3 arguments where the first argument is a label, the second argument is to be typeset in text and the third argument should simply be handed over to an \index command.

The actual definitions can be found below.

```
⟨Definition of rhxDefineTerm and rhxUseTerm⟩≡
  \newcommand{\rhxDefineTerm}[3]{%
    \hypertarget{dt:#1}{\defineTermStyle{#2}}%
    \label{!#1}%
    \index{#3|defineTermPage}%
  }
  \newcommand{\rhxUseTerm}[3]{%
    \@ifundefined{r@!#1}% corresponds to \label{!#1} above
      {\useTermStyle{#2}}%
      {\hyperlink{dt:#1}{\useTermStyle{#2}}}%
    \index{#3}%
  }
```

As can be seen from above, there is some style information that can be redefined by a user of the rhxterm package.

```
⟨Definition of defineTermStyle and useTermStyle⟩≡
  \newcommand{\defineTermPage}[1]{\defineTermStyle{\hyperpage{#1}}}
  \newcommand{\defineTermStyle}[1]{\textbf{#1}}
  \newcommand{\useTermStyle}[1]{\emph{#1}}
```

3 Relation to the hyperref Package

The above definitions rely on the hyperref package. In order to make the rhxterm package also working without the hyperref package we provide dummy commands that will be overridden if hyperref is loaded after rhxterm.

⟨*dummy hyperref commands*⟩≡
```
\@ifundefined{hyperlink}{\def\hyperlink#1#2{#2}}{}
\@ifundefined{hypertarget}{\def\hypertarget#1#2{#2}}{}
\@ifundefined{hyperpage}{\def\hyperpage#1{#1}}{}
```

4 The General Framework

Commands defined by \rhxtermNewCommand take an optional and a mandatory argument which are translated to three arguments (see example of \defineterm in the table above).

If the optional argument is not present, we let it be equal to the second.

⟨*rhxtermNewCommand*⟩≡
```
\def\rhxtermarg{\rhxtermNoOptionalArgument}
\def\rhxtermNewCommand#1#2{%
  \newcommand{#1}[2][\rhxtermNoOptionalArgument]{%
    \def\rhxtermgiven{##1}%
    \ifx\rhxtermgiven\rhxtermarg \rhxtermTwoToThree{##2}{##2}{#2}%
    \else                        \rhxtermTwoToThree{##1}{##2}{#2}%
    \fi}}
```

Now to translate the two arguments to three arguments in the appropriate form, we can assume that \rhxtermTwoToThree is called with the following pattern of arguments.

```
\rhxtermTwoToThree{C}{C}{\somecommand}
\rhxtermTwoToThree{A@C}{A@C}{\somecommand}
\rhxtermTwoToThree{C}{B}{\somecommand}
\rhxtermTwoToThree{C}{A@B}{\somecommand}
```

according to the table above.

For simplicity we assume that we are given

```
\rhxtermTwoToThree{R@S}{U@V}{\somecommand}
```

This must be transformed into

```
\somecommand{U}{S}{U@V}
```

Therefore we define \rhxtermTwoToThree as follows.

```
⟨make 3 arguments out of 2⟩≡
  \def\rhxtermTwoToThree#1#2#3{%
    #3{\rhxtermAtSplit{#2}{\rhxfirstoftwo}}%
      {\rhxtermAtSplit{#1}{\rhxsecondoftwo}}%
      {#2}}
  \def\rhxfirstoftwo#1#2{#1}
  \def\rhxsecondoftwo#1#2{#2}
```

It takes a command \rhxtermAtSplit which basically splits its first argument at the `@` sign or doubles its first argument if there is no `@` sign in it. The second argument is applied to the resulting two tokens. Thus we have the following translations.

$$\begin{array}{lcl} \texttt{\textbackslash rhxtermAtSplit\{A@B\}\{\textbackslash somecommand\}} & \longrightarrow & \texttt{\textbackslash somecommand\{A\}\{B\}} \\ \texttt{\textbackslash rhxtermAtSplit\{A\}\{\textbackslash somecommand\}} & \longrightarrow & \texttt{\textbackslash somecommand\{A\}\{A\}} \end{array}$$

The command itself is using an auxiliary construction to detect the presence of an `@` sign. In order to allow the splitting in both cases (if `@` has category code 11 (letter) or 12 (other)), we basically branch at that decision. The way it is implemented below is a bit different, though.

First we treat the case when `@` has category code 12. It is clear that the first and second argument of \rhxtermSplitAtOther are exactly what we want. However, if the first argument of \rhxtermAtSplit has no `@` sign in it or there is an `@` sign but it is of category code 11 (letter) then the first and second argument of \rhxtermSplitAtOther are equal and the third argument is empty. In that case the \if command in the code chunk below compares `&` with `&` and we fall into the *true* branch. Otherwise the third argument starts with something that is not an `@` sign, namely the first letter of the first argument. For this reason the following code breaks if the first argument of \rhxtermAtSplit starts with an `&` sign or contains an `&` sign after the second `@`.

If the first argument of \rhxtermAtSplit contains two `@` signs, everything after and including the second `@` sign is ignored.

```
⟨split at @ sign⟩≡
  \makeatother
  \def\rhxtermAtSplit#1{\rhxtermSplitAtOther#1@#1@\rhxtermAtEnd}
  \def\rhxtermSplitAtOther#1@#2@#3\rhxtermAtEnd{%
    \if &#3&\expandafter\rhxtermAtSplitLetter
    \else   \expandafter\rhxtermApplyThird\fi
    {#1}{#2}}
```

In the *true* case, we must test whether there is an `@` sign of category code 11 in the argument and split at it.

```
⟨split at @ sign⟩+≡
  \makeatletter
  \def\rhxtermAtSplitLetter#1#2{\rhxtermSplitAtLetter#1@#1@\rhxtermAtEnd}
  \def\rhxtermSplitAtLetter#1@#2@#3\rhxtermAtEnd{\rhxtermApplyThird{#1}{#2}}
```

Finally, the tokens are put in the right order.

```
⟨split at @ sign⟩+≡
  \def\rhxtermApplyThird#1#2#3{#3{#1}{#2}}
```

5 Putting Everything Together

```
⟨general framework⟩≡
  ⟨rhxtermNewCommand⟩
  ⟨make 3 arguments out of 2⟩
  ⟨split at @ sign⟩
```

I like to keep the version of the package equal to the CVS version of the file.

```
⟨version information⟩≡
  \def\rhxtermCVSID $#1Id: #2,v #3 #4 #5 #6 Exp ${%
    \gdef\rhxtermcvsversion{#3}%
    \gdef\rhxtermcvsdate{#4}%
  }
  \rhxtermCVSID $Id: rhxterm.sty.nw,v 1.12 2005/11/12 19:10:21 hemmecke Exp $
  \ProvidesPackage{rhxterm}[\rhxtermcvsdate{} v\rhxtermcvsversion{}
  Notion definition and use (rhx)]
```

```
⟨*⟩≡
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  %
  % rhxterm (C) 2005 Ralf Hemmecke (ralf@hemmecke.de)
  %
  % $Id: rhxterm.sty.nw,v 1.12 2005/11/12 19:10:21 hemmecke Exp $
  %
  % rhxterm defines several commands for the definition and usage
  % of concepts. It should help to get an interactive dvi when used
  % together with the hyperref package.
  %
  % Documentation for this package can be found in the corresponding
  % noweb file. (http://www.hemmecke.de/ralf)
  %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  \NeedsTeXFormat{LaTeX2e}
  ⟨version information⟩
  ⟨general framework⟩
  ⟨Definition of rhxDefineTerm and rhxUseTerm⟩
  ⟨Definition of defineterm and useterm⟩
  ⟨Definition of defineTermStyle and useTermStyle⟩
  ⟨dummy hyperref commands⟩
  ⟨table of contents workaround⟩
```

6 Known Problems

Using \defineterm and \useterm as defined above in the table of contents, i. e., in \section commands might produce hyperlinks in the line of the table of contents that do not point to the corresponding section but to the corresponding item of \useterm. In order to prevent this misbehavior one must switch off the hyperlink part of those commands inside the table of contents. It can be done with the following commands where basically only the style survives.

```
⟨table of contents workaround⟩≡
  \let\rhxtermtableofcontents\tableofcontents
  \def\tableofcontents{{%
      \renewcommand{\rhxDefineTerm}[3]{\defineTermStyle{##2}}%
      \renewcommand{\rhxUseTerm}[3]{\useTermStyle{##2}}%
      \rhxtermtableofcontents}}
```