

The **ydoc** Class and Packages

Martin Scharrer

martin@scharrer-online.de

<http://latex.scharrer-online.de/ydoc/>

CTAN: <http://tug.ctan.org/pkg/ydoc>

<http://www.ctan.org/pkg/ydoc>

Version 0.5alpha

2011/03/20

Abstract

This package bundle is currently under development. All functionality, settings and macro as well as file names can change in later versions and may be incomplete! It is not ready yet to be used for other packages.

The `ydoc` class and packages provide macros to document the functionality and implementation of \LaTeX classes and packages. It is similar to the `ltxdoc` class with the `doc` package, but uses more modern features/packages by default (e.g. `xcolor`, `hyperref`, `listings`). However, some of the features like code indexing is not yet included.

1 Introduction

The `ydoc` packages allow the documentation of \LaTeX packages and classes. The name stands for “Yet another *Documentation Package*” and is a pun on the fact that there are several documentation packages written by package developers to document their own packages. All these packages didn’t suited the author and therefore he, take a guess, wrote his own documentation package. It (will) support(s) all macros and environments (but not necessary with full/identical features) provided by the `doc` package to allow the fast adaption of existing `.dtx` files.

This documentation uses the `ydoc` packages itself and therefore also acts as a live example.

1.1 `ydoc` Files

The `ydoc` bundle consists (at the moment, subject to change) of the `ydoc` class and the packages `ydoc`, `ydoc-code`, `ydoc-desc`, `ydoc-expl` and `ydoc-doc`. The `ydoc` class and package allow the user the freedom to use the functionality with other classes if wanted. The class will load the package. The `ydoc` package loads the packages `ydoc-code`, `ydoc-desc`, `ydoc-expl` and `ydoc-doc`, which provide the functionality to document \LaTeX code implementation, describe the user-level macro, include live code examples and provide replacements for the macros of the `doc`

package, respectively. These packages can be loaded on their own in other kind of \TeX documents if required.

1.2 Similar Packages

Other documentation related classes and packages are `ltxdoc`, `doc`, `dox`, `xdoc`, `gmdoc`, `pauldoc`, `hypdoc`, `codedoc`, `nicetext` and `tkz-doc`.

2 Usage

(section incomplete)

2.1 Code Documentation Environments

```
\begin{macro}{macro}[# of args]{arg 1 description}...{arg n description}  
  macro documentation  
  \begin{macrocode}  
    macro code  
  \end{macrocode}  
  ...  
\end{macro}
```

The implementation of macros can be documented using this environment. The actual *macro code* must be placed in a `macrocode` environment. Longer macro definition can be split using multiple `macrocode` environments with interleaved documentation texts.

The `ydoc` definition of the macro environment has an additional feature compare to `doc`. The arguments of the macro (`#1`, `#2`, ...) can be documented in a vertical list. The environment has an optional argument to declare the *number of arguments* the macro implementation has. The descriptions of this macro arguments are read from the next arguments of the environment. If the *number of arguments* is not given or zero (or less) no further arguments are read by the macro environment.

```
\begin{macrocode}  
  macro code  
\end{macrocode}
```

This environment wraps around any \TeX code and types it verbatim. The environment end is read verbatim as well and must be written on a separate line beginning with a percent followed by exactly four spaces: `%\end{macrocode}`.

```

\begin{environment}{\name} [# of args] {\arg 1 description} . . . {\arg n description}
  \environment documentation
\begin{macrocode}
  \macro code
\end{macrocode}
. . .
\end{environment}

```

This environment provides the same functionality as the macro environment above, but for environments instead.

2.2 Description Macros and Environments

```
\DescribeMacro\macro\macro arguments
```

The `\DescribeMacro` is used to describe macros included their arguments. It takes the to be described `\macro` as first argument (can also be enclosed in `{ }`). The macro name can include `'@'`. Any number of `\macro arguments` (in a broad sense, see Table ??) following it are formatted as arguments of this macro. Any following non-argument token (normal text, macro, etc.) will make `\DescribeMacro` stop collecting arguments. For example, if a `TEX` group should be started using `{ }` direct after `\DescribeMacro` a `\relax` (or a similar macro) should be inserted between them, otherwise the group will be taken as mandatory argument of the described macro.

Multiple `\DescribeMacro` in a row will automatically stacked inside one framed box. If this is not wanted simply separate them with `\relax` or any other macro or token. See also the `DescribeMacros` environment below.

Examples:

`\DescribeMacro\mymacro* [<optional>] {<meta text>}` will result in `\mymacro* [<optional>] {<meta text>}` (inside a framed box).

The above syntax description of `\DescribeMacro` itself was typeset with `\DescribeMacro\DescribeMacro<\textbackslash macro><macro arguments>`.

Special macros with have a partner macro as end marker can be typeset like this: `\DescribeMacro\csname<text>\AlsoMacro\endcsname`, which will result in `\csname<text>\endcsname`.

```
\Macro\macro\macro arguments
```

This macro is like an in-text version of `\DescribeMacro`. The macro description stays as part of the surrounding text and is not placed inside a framed box. The description can be broken between lines. This can be avoided by placing it inside a `\mbox{}`. `\Macro` is equivalent to `\MacroArgs\AlsoMacro`.

```
\MacroArgs\macro arguments
```

This macro formats the `\macro arguments` the same way as `\DescribeMacro` and `\Macro` but without a macro name. Like `\Macro` the description is placed in-text.

`\AlsoMacro⟨\macro⟩⟨further macro arguments⟩`

This macro can only be used inside the *⟨macro arguments⟩* of the above macros and typesets an additional macro as part of the syntax of the described macro. The additional macro is normally an end- or other marker of some kind. Further macro arguments may follow. Macros which are not part of the syntax but normal arguments should be written as `<\textbackslash name>` (yielding *⟨\name⟩*) instead. The ‘|’ character is an abbreviation of `\AlsoMacro`, but only at places where this can appear.

Examples:

```
\Macro\@for<\textbackslash var> ’:=’ <list> \AlsoMacro\do {<code>}
\@for⟨\var⟩:=⟨list⟩\do{⟨code⟩}
```

```
\Macro\pgfkeys{<key1>’=’<value1>’,’<key2>’/.code=’{<code>}}
\pgfkeys{⟨key1⟩=⟨value1⟩,⟨key2⟩/.code={⟨code⟩}}
```

`\MakeShortMacroArgs*⟨char⟩`

This macro is similar to `\MakeShortVerb` from the `shortvrb` package. It can be used to globally define one character to act like `\MacroArgs` till the same character is discovered again. Special characters must be escaped with an backslash for the definition. One additional benefit beside the shorter size is that the argument list is automatically terminated. For example `\MakeShortMacroArgs{\}` will make ‘“<arg>{<arg>}”’ act like ‘`\MacroArgs<arg>{<arg>}\relax`’. One side-effect is that should the argument list be terminated, e.g. by an unknown element or macro, then the rest of the text till the end-character is typeset as normal, but inside a group.

The starred version will define the character equal to `\Macro` instead.

`\DeleteShortMacroArgs{⟨char⟩}`

Globally removes the special meaning from *⟨char⟩* given to him by `\MakeShortMacroArgs`.

Note that special characters like ‘ are best defined `\AtBeginDocument` and deleted again `\AtEndDocument` to avoid issues if they are written to the aux file by some package.

```
\begin{DescribeMacros}
  \Macro⟨\name⟩⟨arguments⟩
  \Macro⟨\name⟩⟨arguments⟩
  ...
\end{DescribeMacros}
```

This environment can be used to place multiple macro description into the same framed box. The macros are described using `\Macro`, which has a slightly different definition than outside of this environment, to place the description into a `\hbox`. The environment stacks these `\hboxes` in a `\vbox`. The macros can also be placed freely using anything which produces a `\hbox`, e.g. `\hbox{\Macro\A ~~~ \Macro\B}` or using a `tabular` (see also `DescribeMacrosTab`).

```
\begin{DescribeMacrosTab}{<tabular column definition>}
  <tabular content>
\end{DescribeMacrosTab}
```

This is a special version of the DescribeMacros environment which adds a tabular environment around the content. This is useful if a large set of small macros should be described at once. Placing them all below each other would result in a very bad page layout. The environment has one argument which is passed to tabular as the column definition. A ‘@{’ is added before and after to remove any margins.

```
\begin{DescribeEnv}{<name>}{<arguments>}
  <body content> \\  

  <more body content>
\end{DescribeEnv}
```

```
\DescribeEnv [<body content>] {<name>}{<arguments>}
```

The DescribeEnv can be used to describe environments in the same way the \DescribeMacro macro describes macros. Supported *<arguments>* are shown in Table ???. Potential *<body content>* can be placed between the begin and end of the environment description to explain the user what kind of material should be placed inside it. The environment also exists in macro form as \DescribeEnv, which allows to provide small *<body content>* as an optional argument. Please note that for this optional argument a \MacroArgs is automatically inserted, but not for the \DescribeEnv environment content.

The body content is placed into a indented \hbox{} stacked inside a \vbox{} also holding the environment begin and end line. The \\
 macro is redefined to create a new indented \hbox acting as new code line. Therefore this environment is similar to a one-column tabular: all macros placed into a line are only valid up to the next line end.

```
\DescribeLength<name>{<default value>}
```

This macro can be used to describe L^AT_EX lengths also known as dimensions. Multiple \DescribeLength macros in a row will automatically be grouped.

2.3 Format Macros

```
\cs{<macro name>}      \env{<environment name>}
\pkg{<package name>}   \cls{<class name>}
```

This macros can be used to format names of macros, environments, packages and classes, respectively. At the moment they simply use \texttt.

```
\bslash  \percent  \braceleft  \braceright
```

This macros define expandable backslash (\backslash_{12}), percent char ($\%_{12}$), and left ($\{_{12}$) and right ($\}_{12}$) braces with catcode 12 (other), respectively. They should only be used with text-typet font when used in text, because other fonts might not have the correct characters. The macros must be protected when used in a moving argument.

Table 1: Supported ‘arguments’ for `\DescribeMacro/\DescribeEnv/\MacroArgs`.

Description	Syntax	Result	Macro ^a
Meta text	<code><text></code>	<i><text></i>	<code>\meta{<text>}</code>
Mandatory Argument	<code>{args}</code>	<code>{args}</code>	
–, with meta text	<code>{<text>}</code>	<i>{<text>}</i>	<code>\marg{<text>}</code>
Optional Argument	<code>[args]</code>	<code>[args]</code>	
–, with meta text	<code>[<text>]</code>	<i>[<text>]</i>	<code>\oarg{<text>}</code>
Picture Argument	<code>(args)</code>	<code>(args)</code>	
–, with meta text	<code>(<text>)</code>	<i>(<text>)</i>	<code>\parg{<text>}</code>
Beamer Overlay Argument	<code><<args>></code>	<code><args></code>	
–, with meta text	<code><< <text> >></code>	<i><<text></i>	<code>\aarg{<text>}</code>
Star	<code>*</code>	<code>*</code>	
Verbatim content	<code>'\$&^%_#&\$\'</code>	<code>\$&^%_#&\$\</code>	
–, produce ‘ char	<code>’</code>	<code>’</code>	
Insert any TeX code	<code>!\fbox{T}!</code>	<code>T</code>	
Unbreakable Space	<code>~</code>		
Space (explicit macro)	<code>\space</code>		
Second macro (e.g. endmarker)	<code>\AlsoMacro\macro</code>	<code>\macro</code>	
short version:	<code> \macro</code>	<code>\macro</code>	

^a) As alternative to be used inside normal text.

Note that ‘args’ can itself be further macro arguments except true verbatim.

<code>\meta{<meta text>}</code>	<code>\marg{<argument text>}</code>
<code>\oarg{<argument text>}</code>	<code>\parg{<argument text>}</code>
<code>\aarg{<argument text>}</code>	<code>\sarg</code>

This macros allow to typeset meta text and mandatory, optional, picture and beamer overlay arguments as well as a star symbol. They are used internally by `\MacroArgs` and friends. See Table ?? for examples.

<code>\metastyle</code>	<code>\margstyle</code>
<code>\oargstyle</code>	<code>\pargstyle</code>
<code>\aargstyle</code>	<code>\sargstyle</code>

This macros are used to define the style in which the corresponding macros above are being formatted. They are used like `{\stylemacro}{<material>}` to allow the styles to use macros like `\ttfamily` or `\texttt{<material>}`. By default the optional argument and the also optional star are printed in the color ‘optional’ which is a 65% gray.

2.4 Settings

The following macro and dimensions can be redefined by the user to adjust the layout of the package documentation.

<code>\descindent</code>	(Default: -20pt)
<code>\beforedescskip</code>	(Default: 12pt plus 4pt minus 4pt)
<code>\afterdescskip</code>	(Default: 6pt plus 2pt minus 2pt)

These length define the indentation and vertical distances before and after a `\Describe...` macro or environment, respectively.

<code>\descsep</code>	(Default: 1em in tt font = 10.5pt)
-----------------------	------------------------------------

This macro defines the space on the left and right side between the description text and the framed box.

2.5 Macros and Environments to include LaTeX Code Examples

<code>\begin{example}</code>
<code>\end{example}</code>

<code>\begin{examplecode}(to be written)</code>
<code>\end{examplecode}</code>

2.6 Class File

```

1 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
2 \RequirePackage{svn-prov}[2010/04/03]
3 \ProvidesClassSVN
4   {$Id: ydoc_cls.dtx 2336 2011-03-18 00:56:07Z ✓
   martin $}
5 %<!VERSION>
6 %<*DRIVER>
7   [develop]
8 %</DRIVER>
9   [ydoc class: document LaTeX class and packages]
```

At the moment simply load article class with `a4paper` option and load the `ydoc` package.

```

10 \PassOptionsToClass{a4paper}{article}
11 \DeclareOption*{\expandafter\PassOptionsToClass\✓
   expandafter{\CurrentOption}{article}}
12 \ProcessOptions\relax
13 \LoadClass{article}
14 \RequirePackage{ydoc}
```

2.7 Macros and Environments to document Implementations

```

15 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
16 \RequirePackage{svn-prov}[2010/04/03]
17 \ProvidesPackageSVN
18     {$Id: ydoc_code_sty.dtx 2339 2011-03-18 20:18:09Z
    martin $}
19 %<!VERSION>
20 %<*DRIVER>
21     [develop]
22 %</DRIVER>
23     [ydoc package to document macro code]

24 \RequirePackage{hyperref}
25 \hypersetup{colorlinks=true,pdfborder=0 0 0,
    pdfborderstyle={}}

26 \IfFileExists{needspace.sty}{%
27     \RequirePackage{needspace}
28 }{%
29     \def\Needspace{\@ifstar\@gobble\@gobble}
30 }

```

2.7.1 Color and style definitions

```

31 \RequirePackage{xcolor}
32 \definecolor{macroimpl}{rgb}{0.0,0.0,0.4}

```

2.7.2 General Macros

`\ydocwrite`

```

33 \@ifundefined{ydocwrite}{%
34     \newwrite\ydocwrite
35 }{}

```

`\ydocfname`

```

36 \@ifundefined{ydocfname}{%
37     \def\ydocfname{\jobname.cod}%
38 }{}

```

`\ydoc@catcodes`


```

39 \def\ydoc@catcodes{%
40   \let\do\@makeother
41   \dospecials
42   \catcode '\=\active
43   \catcode '\^M=\active
44   \catcode '\ =\active
45 }

```

2.7.3 Handling Macrocode

`macrocode`

```

46 \def\macrocode{%
47   \par\noindent
48   \begingroup
49   \ydoc@catcodes
50   \macro@code
51 }
52 \def\endmacrocode{}

```

`\macro@code`

#1: verbatim macro code

```

53 \begingroup
54 \endlinechar\m@ne
55 \@firstofone{%
56 \catcode '\|=0\relax
57 \catcode '\(=1\relax
58 \catcode '\)=2\relax
59 \catcode '\*=14\relax
60 \catcode '\{=12\relax
61 \catcode '\}=12\relax
62 \catcode '\ =12\relax
63 \catcode '\%=12\relax
64 \catcode '\\=\active
65 \catcode '\^M=\active
66 \catcode '\ =\active
67 }*
68 |gdef|macro@code#1^M%      \end{macrocode}(*
69 |endgroup|expandafter|macro@@code|expandafter(|\
   ydoc@removeline#1|noexpand|lastlinemacro)*
70 )*
71 |gdef|ydoc@removeline#1^M(|noexpand|firstlinemacro)*
72 |gdef|ydoc@defspecialmacros(*
73 |def^M(|noexpand|newlinemacro)*
74 |def (|noexpand|spacemacro)*
75 |def\(|noexpand|bslashmacro)*

```

```

76 )*
77 |gdef|ydoc@defrevspecialmacros(*
78 |def|newlinemacro(|noexpand^^M)*
79 |def|spacemacro(|noexpand )*
80 |def|bslashmacro(|noexpand\)*
81 )*
82 |endgroup

```

\macro@@code

#1: verbatim macro code

```

83 \def\macro@@code#1{%
84   {\ydoc@defspecialmacros
85    \xdef\themacrocode{#1}}%
86   \PrintMacroCode
87   \end{macrocode}%
88 }

```

\linenumberbox

```

89 \def\newlinemacro{\\\null}
90 \def\spacemacro{\ }
91 \def\bslashmacro{\char92}
92 \def\lastlinemacro{}
93 \def\firstlinemacro{\linenumberbox}
94 \def\newlinemacro{\\\linenumberbox}
95 \newcounter{linenumber}
96 \def\linenumberbox{%
97   \hbox to 1.25em{}%
98   \llap{%
99     \stepcounter{linenumber}%
100    {\footnotesize\color{gray}\thelinenumber~}%
101   }%
102 }

```

\PrintMacroCode

```

103 \def\PrintMacroCode{%
104   \begingroup
105   \ttfamily
106   \noindent\themacrocode
107   \endgroup
108 }

```

\PrintMacroCode

```
109 \RequirePackage{listings}

110 \def\PrintMacroCode{%
111   \begingroup
112   \let\firstlinemacro\empty
113   \let\lastlinemacro\empty
114   \def\newlinemacro{^^J}%
115   \let\bslashmacro\bslash
116   \let\spacemacro\space
117   \immediate\openout\ydocwrite=\ydocfname\relax
118   \immediate\write\ydocwrite{\themacrocode}%
119   \immediate\closeout\ydocwrite
120   \@nameuse{ydoc@countbslashes}%
121   \ydoclistingssettings
122   \let\input\@input
123   \lstinputlisting{\ydocfname}%
124   \endgroup
125 }
```

\ydoclistingssettings

```
126 \def\ydoclistingssettings{%
127   \lstset{%
128     language=[latex]tex,basicstyle=\ttfamily,
129     numbers=left,numberstyle=\tiny\color{gray},✓
130     firstnumber=last,
131     breaklines,prebreak={\mbox{\tiny$\swarrow$}},
132     commentstyle=\color{black!60},
133   }%
134 \ydoclistingssettings
```

\macro@impl@args

#1: number of macro arguments

```
135 \def\macro@impl@args [#1]{%
136   \begingroup
137   \parindent=10pt\relax
138   \let\macro@impl@argcnt\@tempcnta
139   \let\macro@impl@curarg\@tempcntb
140   \macro@impl@argcnt=#1\relax
141   \macro@impl@curarg=0\relax
142   \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
143     \expandafter\macro@impl@arg
```

```

144   \else
145     \expandafter\macro@impl@endargs
146   \fi
147 }

```

`\macro@impl@endargs`

```

148 \def\macro@impl@endargs{%
149   \endgroup
150   \unskip\par\noindent\ignorespaces
151 }

```

`\macro@impl@argline`

#1: argument number
#2: argument description

```

152 \def\macro@impl@argline#1#2{%
153   \par{\texttt{\##1}:~#2\strut}%
154 }

```

`\macro@impl@arg`

#1: argument description

```

155 \def\macro@impl@arg#1{%
156   \advance\macro@impl@curarg by\@ne\relax
157   \macro@impl@argline{\the\macro@impl@curarg}{#1}%
158   \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
159     \expandafter\macro@impl@arg
160   \else
161     \expandafter\macro@impl@endargs
162   \fi
163 }

```

`macro`

#1: implemented macro

```

164 \def\macro#1{%
165   \PrintMacroImpl{#1}%
166   \@ifnextchar[%]
167     {\macro@impl@args}%
168     {}%
169 }
170 \def\endmacro{}

```

environment

#1: environment name

```

171 \def\environment#1{%
172   \PrintEnvImplName{#1}%
173   \@ifnextchar[%]
174     {\macro@impl@args}%
175     {}%
176 }
177 \def\endenvironment{}

```

\PrintMacroImpl

#1: macro (token)

```

178 \def\PrintMacroImpl#1{%
179   \par\bigskip\noindent
180   \Needspace*{3\baselineskip}%
181   \hbox{%
182     \edef\name{\expandafter\@gobble\string#1}%
183     \global\@namedef{href@impl@\name}{}%
184     \immediate\write\@mainaux{%
185       \global\noexpand\@namedef{href@impl@\name}{}%
186     }%
187     \raisebox{4ex}[4ex]{\hypertarget{impl:\name}{}%
188     \hspace*{\descindent}\fbox{%
189       \hspace*{\descsep}%
190       \@ifundefined{href@desc@\name}{}{\hyperlink{
191         desc:\name}}}%
192       {\PrintMacroImplName{#1}}%
193       \hspace*{\descsep}%
194     }%
195   }%
196 }

```

\PrintMacroImplName

#1: macro (token)

```

197 \def\PrintMacroImplName#1{%
198   \implstyle{\string#1\strut}%
199 }

```

\PrintEnvImplName#1: environment name
test

```

200 \def\PrintEnvImplName#1{%
201   \par\bigskip\noindent
202   \hbox{\hspace*{\descindent}\fbox{{\implstyle{#1}}}}✓
203   %
204   \par\medskip
205 }

```

`\implstyle`

```

205 \def\implstyle{\ttfamily\bfseries\color{macroimpl}}

```

`\bslash`

Defines an expandable backslash with catcode 12: ‘\₁₂’. The \@firstofone trick is used to read the \gdef\bslash code before changing the catcode.

```

206 {%
207 \@firstofone{%
208   \catcode'\=12
209   \gdef\bslash
210 }{\}
211 }%}

```

2.8 Description Macros and Environments

```

212 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
213 \RequirePackage{svn-prov}[2010/04/03]
214 \ProvidesPackageSVN[ydoc-desc]
215   {$Id: ydoc_desc_sty.dtx 2369 2011-03-20 00:04:45Z✓
216   martin $}
216 %<!VERSION>
217 %<*DRIVER>
218   [develop]
219 %</DRIVER>
220   [ydoc package to describe macros, environments, ✓
221   options etc.]
221 \IfFileExists{needspace.sty}{%
222   \RequirePackage{needspace}
223 }{%
224   \def\Needspace{\@ifstar@gobble@gobble}
225 }

```

The short verbatim code is required for the similar macros provided here.

```

226 \RequirePackage{shortvrb}

```

The etoolbox package is used mainly for \newrobustcmd.

```

227 \RequirePackage{etoolbox}

```

2.8.1 Color and style definitions

```
228 \RequirePackage{xcolor}
229 \definecolor{macrodesc}{rgb}{0,0.2,0.6}
230 \definecolor{macroimpl}{rgb}{0,0.1,0.3}
231 \definecolor{meta}{rgb}{0,0.25,0.75}
232 \definecolor{scriptcolor}{rgb}{0.2,0.6,0.2}
233 \definecolor{optioncolor}{rgb}{0.3,0.2,0}
234 \colorlet{optional}{black!65!white}
235 \colorlet{metaoptional}{optional!50!meta}
236 \providecolor{urlcolor}{named}{blue}
237 \providecolor{linkcolor}{named}{blue}
238 \providecolor{filecolor}{named}{blue}
239 \providecolor{citecolor}{named}{blue}
240 \providecolor{anchorcolor}{named}{blue}
241 \providecolor{menucolor}{named}{blue}
242 \providecolor{runcolor}{named}{blue}

244 \RequirePackage{hyperref}
245 \hypersetup{%
246     colorlinks=true,
247     pdfborder=0 0 0,
248     pdfborderstyle={},
249     urlcolor=urlcolor,
250     linkcolor=linkcolor,
251     filecolor=filecolor,
252     citecolor=citecolor,
253     anchorcolor=anchorcolor,
254     menucolor=menucolor,
255     runcolor=runcolor,
256 }
```

2.8.2 Text Formatting Macros

<code>\meta</code>

Prints *[meta text](#)*.

```
258 \newrobustcmd*\meta[1]{%
259     {\metastyle{%
260         \ensuremath\langle
261             #1\/%
262         \ensuremath\rangle
263     }}%
264 }
```

`\marg`

Sets style and adds braces. The text is formatted as separate set of macro arguments.

```
265 \newrobustcmd*{\marg}[1]{%
266   {\margstyle{%
267     {\ttfamily\braceleft}%
268     \meta{#1}%
269     {\ttfamily\braceright}%
270   }}%
271 }
```

`\oarg`

Sets style and adds brackets. The text is formatted as separate set of macro arguments.

```
272 \newrobustcmd*{\oarg}[1]{%
273   {\oargstyle{%
274     {\ttfamily[]}%
275     \meta{#1}%
276     {\ttfamily[]}%
277   }}%
278 }
```

`\parg`

Sets style and adds parentheses.

```
279 \newrobustcmd*{\parg}[1]{%
280   {\pargstyle{%
281     {\ttfamily{}}%
282     \meta{#1}%
283     {\ttfamily}}%
284   }}%
285 }
```

`\aarg`

Sets style and adds angles.

```
286 \newrobustcmd*{\aarg}[1]{%
287   {\aargstyle{%
288     {\ttfamily<}%
289     \meta{#1}%
290     {\ttfamily>}%
291   }}%
292 }
```


`\sarg`

Prints star with given style.

```
293 \newrobustcmd*{\sarg}{\sargstyle{*}}
```

`\pkg`

`\cls`

`\lib`

`\env`

`\opt`

`\file`

```
294 \newrobustcmd*\pkg [1]{\pkgstyle{#1}}
295 \newrobustcmd*\cls [1]{\clsstyle{#1}}
296 \newrobustcmd*\lib [1]{\libstyle{#1}}
297 \newrobustcmd*\env [1]{\envstyle{#1}}
298
299 \newrobustcmd*\opt{\@ifstar\ys@opt\y@opt}
300 \def\y@opt#1{\optstyle{#1}}
301 \def\ys@opt#1{\optstyle{#1}\optpar{#1}}
302 \newrobustcmd*\optpar [1]{\marginpar{\hbox to \
marginparwidth{\hss\y@opt{#1}}}}
303
304 \newrobustcmd*\file [1]{\filestyle{#1}}
305 \newcommand*\pkgstyle [1]{\texttt{\textcolor{pkg}
}{#1}}
306 \newcommand*\clsstyle [1]{\texttt{\textcolor{cls}
}{#1}}
307 \newcommand*\libstyle [1]{\texttt{\textcolor{lib}
}{#1}}
308 \newcommand*\envstyle [1]{\texttt{\textcolor{env}
}{#1}}
309 \newcommand*\optstyle [1]{\textsf{\textcolor{opt}
}{#1}}
```

```

310 \newcommand*\filestyle [1]{\texttt{\textcolor{file
    }{#1}}}
311 \colorlet{cls}{black}
312 \colorlet{lib}{black}
313 \colorlet{env}{black}
314 \colorlet{file}{black}
315 \colorlet{pkg}{black}
316 \definecolor{opt}{rgb}{0.5,0.16666,0}

```

`\cs`

`\cmd`

```

317 \newrobustcmd*\cs [1]{\texttt{\textbackslash #1}}
318 \newrobustcmd*\cmd [1]{\texttt{\{\escapechar=92\string
    #1}}}

```

2.8.3 Text Formatting Styles

`\macrodescstyle`

Style of described macro names.

```

319 \def\macrodescstyle{\ttfamily\bfseries\color{
    macrodesc}}

```

`\macroargsstyle`

Default style for macro arguments (e.g. `\MacroArgs`).

```

320 \def\macroargsstyle{\ttfamily}

```

`\envcodestyle`

Default style for code body content in described environments.

```

321 \def\envcodestyle{\ttfamily}

```

`\verbstyle`

Style for verbatim text inside macro argument list.

```

322 \def\verbstyle{\verbatim@font}

```

`\metastyle`

Meta text style. Because `\macroargsstyle` might be also active a `\normalfont` reset the font.

```
323 \def\metastyle{\normalfont\itshape\color{meta}}
```

`\margstyle`

Style for `\marg`.

```
324 \def\margstyle{}
```

`\Optional`

`\optional`

`\optionalstyle`

```
325 \protected\def\Optional{\optionalon\optional}
326 \def\optionalstyle{\blendcolors*{!60!white}\color{✓
  black!75}}
```

`\optionalon`

`\optionaloff`

```
327 \def\optionalon{\protected\def\optional{\✓
  optionalstyle}}
328 \def\optionaloff{\let\optional\relax}
329 \optionalon
```

`\oargstyle`

Style for `\oarg`. A special color is set to show the ‘optional’ status.

```
330 \def\oargstyle{\optional}
```

`\pargstyle`

Style for `\parg`.

```
331 \def\pargstyle{}
```

`\aargstyle`

Style for `\aarg`.

```
332 \def\aargstyle{}
```

`\sargstyle`

Style for `\sarg`. A special color is set to show the ‘optional’ status.

```
333 \def\sargstyle{\ttfamily\color{optional}}
```

2.8.4 Dimension Registers

`\descindent`

```
334 \newdimen\descindent
```

```
335 \descindent=-\parindent
```

`\beforedescskip`

```
336 \newdimen\beforedescskip
```

```
337 \beforedescskip=\bigskipamount
```

`\afterdescskip`

```
338 \newdimen\afterdescskip
```

```
339 \afterdescskip=\medskipamount
```

`\descsep`

Set to 1em in tt font.

```
340 \newdimen\descsep
```

```
341 \begingroup
```

```
342 \ttfamily
```

```
343 \global\descsep=1em\relax
```

```
344 \endgroup
```

2.8.5 Macro Argument Reading Mechanism

`\read@Macro@arg`

Reads next token and calls second macro.

```
345 \def\read@Macro@arg{%
346   \futurelet\@let@token\handle@Macro@arg
347 }
```

`\AlsoMacro`

Reads argument while @ is a letter, prints the macro name and reads further arguments.

```
348 \newcommand*\AlsoMacro{%
349   \begingroup\makeatletter
350   \AlsoMacro@
351 }
352 \def\AlsoMacro@#1{%
353   \endgroup
354   %<*DEBUG>
355   \typeout{DEBUG: Macro: \string#1}%
356   %</DEBUG>
357   \PrintMacroName{#1}%
358   \read@Macro@arg
359 }
```

`\ydoc@short@AlsoMacro`

Makes & an alias for \AlsoMacro.

```
360 \begingroup
361 \catcode'\|\active
362 \gdef\ydoc@short@AlsoMacro{%
363   \catcode'\|\active
364   \let|\AlsoMacro
365 }
366 \endgroup
```

`\ydoc@macrocatcodes`

Sets the catcodes inside for read@Macro@arg material.

```
367 \def\ydoc@macrocatcodes{%
368   \ydoc@short@AlsoMacro
369   \@makeother\'%
370   \@makeother\!%
371   \@makeother\[%
372   \@makeother\]%
373   \@makeother\(%
374   \@makeother\)%
375 }
```

`\handle@Macro@arg`

Checks if next token is the begin of a valid macro argument and calls the appropriate read macro or the end macro otherwise.

```
376 \def\handle@Macro@arg{%
377   \expandafter\let\expandafter\handler\csname
      handle@Macro@token@meaning@\let@token@endcsname
378   \ifx\handler\relax
379     \def\handler{\ifhmode\unskip\fi\end@Macro@args}%
380   %<*DEBUG>
381     \typeout{DEBUG: Stopped at: \expandafter\meaning\
      csname @let@token@endcsname}%
382     \typeout{}%
383   \else
384     \expandafter\ifx\csname @let@token@endcsname\
      AlsoMacro
385     \typeout{DEBUG: TOKEN: \string\AlsoMacro}%
386   \else
387     \typeout{DEBUG: TOKEN: \expandafter\meaning\
      csname @let@token@endcsname}%
388   \fi
389 %</DEBUG>
390   \fi
391   \handler
392 }
393 \def\define@Macro@handler{%
394   \begingroup
395   \ydoc@macrocatcodes
396   \define@Macro@handler@
397 }
398 \def\define@Macro@handler@#1{%
399   \endgroup
400   \@namedef{handle@Macro@token@meaning#1}%
401 }
```

`\end@Macro@args`

Closes box as calls hook. Might be locally redefined by some macros calling `\read@Macro@arg`.

```
402 \def\end@Macro@args{%
403   \y@egroup
404   \after@Macro@args
405 }
```

`\after@Macro@args`

Hook to add additional commands in certain situations.

```

406 \def\after@Macro@args{%
407 }

```

Macro argument reading macros

This macros read the macro arguments and call the appropriate format macros.

`\read@Macro@marg`

```

408 \define@Macro@handler{\bgroup}{%
409   \begingroup
410     \afterassignment\read@Macro@marg@
411     \let\@let@token=%
412 }
413 \def\read@Macro@marg@{%
414   \bgroup
415   \margstyle{}%
416   \let\end@Macro@args\empty%
417   {\ttfamily\braceleft}%
418   \aftergroup\read@Macro@marg@@
419   \read@Macro@marg
420 }
421 \def\read@Macro@marg@@{%
422   {\ttfamily\braceright}%
423   \endgroup
424   \read@Macro@marg
425 }

```

`\read@Macro@oarg`

```

426 \define@Macro@handler{[]}{%
427   \begingroup
428     \let\read@Macro@oarg@end\read@Macro@oarg@@end
429     \let\end@Macro@args\read@Macro@oarg@end
430     \oargstyle{}%
431     {\ttfamily[]}%
432     \read@Macro@marg
433 }
434 \define@Macro@handler{[]}{%
435   \read@Macro@oarg@end
436 }
437 \def\read@Macro@oarg@@end#1{%
438   #1%
439   {\ttfamily}}%
440 \endgroup
441 \read@Macro@marg
442 }

```

```

443 \def\read@Macro@oarg@end{\end@Macro@args}
444 \let\read@Macro@aarg@end\read@Macro@oarg@end
445 \let\read@Macro@parg@end\read@Macro@oarg@end

```

\read@Macro@parg

```

446 \define@Macro@handler{()}{%
447     \begingroup
448         \let\read@Macro@parg@end\read@Macro@parg@@end
449         \let\end@Macro@args\read@Macro@parg@end
450         \pargstyle{}%
451         {\ttfamily{}}%
452         \read@Macro@arg
453     }
454 \define@Macro@handler{}{}{%
455     \read@Macro@parg@end
456 }
457 \def\read@Macro@parg@@end#1){%
458     #1%
459     {\ttfamily)}%
460 \endgroup
461 \read@Macro@arg
462 }

```

\read@Macro@aarg

```

463 \def\read@Macro@aarg<{%
464     \begingroup
465         \let\read@Macro@aarg@end\read@Macro@aarg@@end
466         \let\end@Macro@args\read@Macro@aarg@end
467         \aargstyle{}%
468         {\ttfamily<}%
469         \read@Macro@arg
470     }
471 \define@Macro@handler{>}{%
472     \read@Macro@aarg@end
473 }
474 \def\read@Macro@aarg@@end#1>>{%
475     #1%
476     {\ttfamily>}%
477 \endgroup
478 \read@Macro@arg
479 }

```


`\read@Macro@angle`

```
480 \define@Macro@handler{<}<{%  
481   \futurelet\@let@token\read@Macro@angle@  
482 }
```

`\read@Macro@angle@`

```
483 \def\read@Macro@angle@{%  
484   \ifx\@let@token<%  
485     \expandafter\read@Macro@aarg  
486   \else  
487     \expandafter\read@Macro@meta  
488   \fi  
489 }
```

`\read@Macro@meta`

```
490 \def\read@Macro@meta#1>{%  
491   \meta{#1}\read@Macro@arg  
492 }
```

`\read@Macro@sarg`

```
493 \define@Macro@handler**{%  
494   \sarg\read@Macro@arg  
495 }
```

`\read@Macro@verb`

Sets up verbatim mode calls second macro.

```
496 \define@Macro@handler{'}'{%  
497   \begingroup  
498   \let\do\@makeother  
499   \dospecials  
500   \@noligs  
501   \@makeother\'%  
502   \obeyspaces  
503   \read@Macro@verb@  
504 }
```

`\read@Macro@verb@`

Closes verbatim mode and formats text. If #1 is empty (') than a single ' is printed.

```
505 \begingroup
506 \@makeother\'%
507 \gdef\read@Macro@verb@#1' {%
508   \endgroup
509   \ifx\relax#1\relax
510     {\verbstyle{\string'}}%
511   \else
512     {%
513       \frenchspacing
514       \@noligs\verbstyle{#1}}%
515   \fi
516   \read@Macro@arg
517 }
518 \endgroup
```

`\read@Macro@cmds`

Simply executes given code.

```
519 \define@Macro@handler!!#1! {%
520   #1\relax
521   \read@Macro@arg
522 }
```

`\read@Macro@rmspace`

Removes space. The \@firstofone is used to preserve the space in the macro definition.

```
523 \define@Macro@handler{\@sptoken} {%
524   \read@Macro@arg
525 }
```

`\read@Macro@addtoken`

Takes token over from input to output 'stream'. This is used for \space and ~.

```
526 \define@Macro@handler{~}#1 {%
527   #1\read@Macro@arg
528 }
529 \AtBeginDocument {%
530 \define@Macro@handler{~}#1 {%
531   #1\read@Macro@arg
532 }
```

```

533 }
534 \define@Macro@handler{\space}#1{%
535   #1\read@Macro@arg
536 }

```

2.8.6 Description Macros

For Macros

\DescribeMacro

```

537 \@ifundefined{DescribeMacro}{}{%
538   \PackageInfo{ydoc-desc}{Redefining \string\
539     DescribeMacro}{}%
540 }

```

A `\DescribeMacro` places itself in a `DescribeMacros` environment. Multiple `\DescribeMacro` macros will stack themselves inside this environment. For this to work `\DescribeMacros` is locally defined to `\y@egroup` to close the `\hbox` from the previous `\DescribeMacro`.

```

540 \def\DescribeMacro{%
541   \DescribeMacros
542   \let\DescribeMacros\y@egroup
543   \optionalon
544   \def\after@Macro@args{\endDescribeMacros}%
545   \begingroup\makeatletter
546   \Describe@Macro
547 }

```

\DescribeScript

```

548 \def\DescribeScript#1{%
549   \DescribeMacros
550   \let\DescribeMacros\y@egroup
551   \optionalon
552   \def\after@Macro@args{\endDescribeMacros}%
553   \hbox\y@bgroup
554   \texttt{#1}%
555   \ydoc@macrocatcodes
556   \macroargsstyle
557   \read@Macro@arg~%
558 }

```

`\Describe@Macro`

```
559 \def\Describe@Macro#1{%
560   \endgroup
561   \edef\name{\expandafter\@gobble\string#1}%
562   \global\@namedef{href@desc@\name}{}%
563   \immediate\write\@mainaux{%
564     \global\noexpand\@namedef{href@desc@\name}{}%
565   }%
566   \hbox\y@bgroup
567   \@ifundefined{href@impl@\name}{}{\hyperlink{impl:\↵
     name}}%
568   {%
569   \hbox{\vbox to 0pt{\vss\hbox{\raisebox{4ex}{\↵
     hypertarget{desc:\name}}}}}%
570   \PrintMacroName{#1}}%
571   }%
572   \ydoc@macrocatcodes
573   \macroargsstyle
574   \read@Macro@arg
575 }
```

`\MakeShortMacroArgs`

Defines the given character as short version for `\MacroArgs`. It is first define to be a short verbatim character to take advantage of the house-keeping (save & restore of the original catcode and definition) of `shortvrb`.

The starred version define the character to act like `\Macro` instead.

```
576 \newcommand*\MakeShortMacroArgs{%
577   \@ifstar
578     {\@MakeShortMacroArgs\Macro}%
579     {\@MakeShortMacroArgs\MacroArgs}%
580 }
581 \def\@MakeShortMacroArgs#1#2{%
582   \MakeShortVerb{#2}
583   \catcode'#2\active
584   \begingroup
585   \catcode'\~\active
586   \lccode'\~'#2\relax
587   \lowercase{\endgroup\gdef~{\bgroup\let~\egroup#1}}%
588 }
```

`\DeleteShortMacroArgs`

```
589 \newcommand*\DeleteShortMacroArgs [1]{%
590   \DeleteShortVerb{#1}%
591 }
```

`\Macro`

Simply uses the two macros below.

```
592 \newcommand*\Macro{\MacroArgs\AlsoMacro}
```

`\@Macro`

Alternative definition of `\Macro` inside `DescribeMacros` environments.

```
593 \def\@Macro{%
594   \begingroup\makeatletter
595   \Describe@Macro
596 }

597 \define@Macro@handler\AlsoMacro{}
598 \define@Macro@handler\DescribeMacro{}
```

`\MacroArgs`

Uses the normal macro argument reading mechanism from `\DescribeMacro`. Instead of a box a simple group is added.

```
599 \newcommand*\MacroArgs{%
600   \begingroup
601   \def\end@Macro@args{\endgroup\xspace}%
602   \ydoc@macrocatcodes
603   \macroargsstyle
604   %< *DEBUG >
605   \typeout{}%
606   \typeout{DEBUG: Start MacroArgs}%
607   %< /DEBUG >
608   \read@Macro@arg
609 }
610 \RequirePackage{xspace}
```

`\DescribeMacros`

```
611 \def\DescribeMacros{%
612   \begingroup
613   \let\Macro\@Macro
614   \parindent=0pt\relax
615   \setbox\descbox\vbox\y@bgroup
616 }
```

`\endDescribeMacros`

```
617 \def\endDescribeMacros{%
618   \y@egroup
619   \PrintMacros
620   \endgroup
621 }
```

`\DescribeMacrosTabcolsep`

```
622 \def\DescribeMacrosTabcolsep{\tabcolsep}
```

`\DescribeMacrosTab`

```
623 \def\DescribeMacrosTab{%
624   \DescribeMacros
625   \hbox\y@bgroup
626   \tabcolsep=\DescribeMacrosTabcolsep\relax
627   \DescribeMacrosTab@
628 }
629 \def\DescribeMacrosTab@#1{\tabular{@{}#1@{}}}
```

`\endDescribeMacrosTab`

```
630 \def\endDescribeMacrosTab{%
631   \endtabular\y@egroup
632   \endDescribeMacros
633 }
```

For Lengths

`\DescribeLength`

```
634 \newcommand*\DescribeLength{%
635   \begingroup
636   \let\DescribeLength\Describe@Length
637   \setbox\descbox\hbox\y@bgroup
638   \tabular{@{}l@{\hspace{2em}}l@{}}%
639   \Describe@Length
640 }
```

`\Describe@Length`

```
641 \newcommand*\Describe@Length [2]{%
642   \PrintLengthName{#1}&
643   (Default: {\macroargsstyle#2\unskip})%
644   \@ifnextchar\DescribeLength
645     {\}%
646     {%
647       \endtabular
648       \y@egroup
649       \PrintLength
650       \endgroup
651     }%
652 }
```

For Environments

`\DescribeEnv`

```
653 \@ifundefined{DescribeEnv}{%
654   \PackageInfo{ydoc-desc}{Redefining \string\
655   DescribeEnv}{%
656 }
```

```
657 \newcommand*\DescribeEnv [2] []{%
658   \begingroup
659   \def\DescribeEnv@name{#2}%
660   \let\\\DescribeEnv@newline
```

Sets after-macro-arguments hook. First checks if the environment or macro version was used. The environment starts a new line only if the next token isn't `\end`, which is taken as end of the environment.

```
661   \ifx\@currenvir\DescribeEnv@string
662     \def\after@Macro@args{%
663       \let\after@Macro@args\empty
664       \setbox\@tempboxa\hbox\y@bgroup
665       \@ifnextchar\end{%
666         {\DescribeEnv@newline}%
667         #1%
668       }%

```

The macro version adds the optional argument as content line if given.

```
669   \else
670     \ifx\relax#1\relax
671       \def\after@Macro@args{%
672         \y@bgroup
```

```

673         \endDescribeEnv
674     }%
675     \else
676         \def\after@Macro@args{%
677             \setbox\@tempboxa\hbox\y@bgroup
678             \DescribeEnv@newline\MacroArgs#1%
679             \endDescribeEnv
680         }%
681     \fi
682 \fi

Start \vbox and adds first line.

683 \setbox\descbox\vbox\y@bgroup
684 \envcodestyle
685 \let\PrintEnv\PrintSubEnv
686 \hbox\y@bgroup
687 \PrintEnvName{\begin}{\DescribeEnv@name}%
688 \ydoc@macrocatcodes
689 \macroargsstyle
690 \read@Macro@arg
691 }

```

`\DescribeEnv@newline`

Closes existing and starts a new horizontal box representing a indented line. The optional argument allows to add extra space between lines like the normal `\`. Negative values are not supported.

```

692 \newcommand*\DescribeEnv@newline [1] [0pt] {%
693     \strut\y@egroup
694     {\vskip#1}%
695     \hbox\y@bgroup\strut
696     \hspace*{\descsep}%
697     \ignorespaces
698 }%

```

`\DescribeEnv@string`

Holds the environment name for comparison.

```

699 \def\DescribeEnv@string{DescribeEnv}

```

`\descbox`

Save box to store description content.

```

700 \newbox\descbox

```


`\endDescribeEnv`

```
701 \def\endDescribeEnv{%
702   \y@egroup
703   \begingroup
704   \setbox\@tempboxa\lastbox
705   \ifcase0%
706     \ifdim\wd\@tempboxa>\descsep1\fi
707     \ifdim\ht\@tempboxa>\ht\strutbox1\fi
708     \ifdim\dp\@tempboxa>\dp\strutbox1\fi
709   \else
710     \box\@tempboxa
711   \fi
712   \endgroup
713   \hbox\y@bgroup
714     \PrintEnvName{\end}{\DescribeEnv@name}
715   \y@egroup
716   \y@egroup
717   \PrintEnv
718   \endgroup
719 }
```

2.8.7 Print Macros

`\PrintMacroName`

Formats macro name. The backslash is forced to tt font.

```
720 \def\PrintMacroName#1{%
721   {\macrodescstyle{\strut
722     \texttt{\char92}}%
723     \escapechar\m@ne
724     \string#1\strut}}%
725 }
```

`\PrintLengthName`

Formats length register name.

```
726 \let\PrintLengthName\PrintMacroName
```

`\PrintEnvName`

#1 = ‘\begin’ or ‘\end’, #2 = env name.

```

727 \def\PrintEnvName#1#2{%
728   \strut
729   \string#1\braceleft
730   {\macrodescstyle#2\strut}%
731   \braceright
732 }

```

\PrintMacros

Prints macros described using \DescribeMacros. The actual content was stored inside \descbox. If it is wider than the line width it is centered.

```

733 \def\PrintMacros{%
734   \par\vspace\beforedescskip
735   \begingroup
736   \sbox\@tempboxa{\descframe{\usebox{\descbox}}}%
737   \Needspace*{\dimexpr\ht\@tempboxa+2\baselineskip\relax}%
738   \par\noindent
739   \ifdim\wd\@tempboxa>\dimexpr\linewidth-2\descindent\relax
740     \makebox[\linewidth][c]{\usebox\@tempboxa}%
741   \else
742     \hspace*{\descindent}%
743     \usebox\@tempboxa
744   \fi
745   \endgroup
746   \par
747   \vspace\afterdescskip
748   \par\noindent
749 }
750 \def\descframe#1{%
751   \fbox{\hspace*{\descsep}#1\hspace*{\descsep}}%
752 }

```

\PrintLength

Prints lengths registers described using one or multiple \DescribeLength.

```

753 \let\PrintLength\PrintMacros

```

\PrintEnv

Prints DescribeEnv environments. The actual content was stored inside \descbox.

```

754 \let\PrintEnv\PrintMacros

```

`\PrintSubEnv`

Prints sub environments, i.e. `DescribeEnv` environments inside the body of another `DescribeEnv`. The actual content was stored inside `\descbox`.

```
755 \def\PrintSubEnv{%  
756   \hbox{\hbox{\usebox{\descbox}}}%  
757 }
```

2.8.8 Special Character Macros

`\bslash`

Defines an expandable backslash with catcode 12: `'\12'`. The `\@firstofone` trick is used to read the `\gdef\bslash` code before changing the catcode.

```
758 {%  
759 \@firstofone{%  
760   \catcode'\=12  
761   \gdef\bslash  
762 }{\}  
763 }%}
```

`\percent`

Defines an expandable percent character with catcode 12: `'%12'`.

```
764 \begingroup  
765 \catcode'\%=12  
766 \gdef\percent{%}  
767 \endgroup
```

`\braceleft`

`\braceright`

Defines expandable left and right braces with catcode 12: `'{12}'` `'}12}'`.

```
768 \begingroup  
769 \catcode'\<=1  
770 \catcode'\>=2  
771 \catcode'\{=12  
772 \catcode'\}=12  
773 \gdef\braceleft <{>  
774 \gdef\braceright <}>  
775 \endgroup
```

2.8.9 Other Macros

`\y@bgroup`

`\y@egroup`

These macros are used to begin and end `\vbox/\hbox-es`.

```
776 \def\y@bgroup{\bgroup\color@setgroup}
777 \def\y@egroup{\color@endgroup\egroup}
```

`\codeline`

```
778 \newcommand*{\codeline}[1][c]{%
779   \codelinebefore
780   \hbox to \hsize\bgroup
781   \ifx i#1\hspace*{\leftmargin}\else
782     \ifx l#1\else\hss\fi
783   \fi
784   \let\xspace\relax
785   \hbox\bgroup
786   \aftergroup\codeline@end
787   \aftergroup#1%
788   \afterassignment\MacroArgs
789   \let\@let@token=%
790 }
791 \def\codeline@end#1{%
792   \ifx r#1\else\hss\fi
793   \egroup
794   \codelineafter
795 }
796 \newcommand*\codelinebefore{\par\smallskip\noindent}
797 \newcommand*\codelineafter {\par\smallskip\noindent}
```

`codequote`

```
798 \newenvironment{codequote}{%
799   \def\{\{\newline\relax\MacroArgs}%
800   \par\smallskip\bgroup\leftskip=\leftmargin\✓
      rightskip=\rightmargin\noindent\MacroArgs}
801   {\par\egroup\smallskip\noindent\✓
      ignorespacesafterend}
```

`macroquote`

```

802 \newenvironment{macroquote}{%
803   \def\{\{\newline\relax\Macro}%
804   \par\smallskip\bgroup\leftskip=\leftmargin\✓
      rightskip=\rightmargin\noindent\Macro}
805   {\par\egroup\smallskip\noindent\✓
      ignorespacesafterend}

```

2.9 Provide doc macros

```

806 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
807 \RequirePackage{svn-prov}[2010/04/03]
808 \ProvidesPackageSVN[ydoc]
809   {$Id: ydoc_doc_sty.dtx 2378 2011-03-20 20:34:27Z ✓
      martin $}
810 %<!VERSION>
811 %<*DRIVER>
812   [develop]
813 %</DRIVER>
814   [ydoc package to provide 'doc' macros]

```

`\ydoc@countbslashes`

Reads the macro code into a temp box. The backslashes are defined to increase a counter.

```

815 \newcount\ydoc@bslashcnt
816 \def\ydoc@countbslashes{%
817   \begingroup
818     \let\firstlinemacro\empty
819     \let\lastlinemacro\empty
820     \let\newlinemacro\empty
821     \let\spacemacro\empty
822     \def\bslashmacro{\global\advance\ydoc@bslashcnt ✓
      by\@ne}%
823     \setbox\@tempboxa\hbox{\themacrocode}%
824   \endgroup
825 }

```

`\Checksum`

```

826 \def\Checksum#1{%
827   \gdef\ydoc@checksum{#1}%
828 }
829 \let\ydoc@checksum\m@ne

```

`\AlsoImplementation`

`\OnlyDescription`

`\StopEventually`

`\Finale`

The first two macros modify the `\StopEventually` macro which either stores its argument in `\Final` or executes it itself.

```
830 \def\AlsoImplementation{%
831   \gdef\StopEventually##1{%
832     \@bsphack
833     \gdef\Finale{##1\ydoc@checkchecksum}%
834     \@esphack
835   }%
836 }
837 \AlsoImplementation
838 \def\OnlyDescription{%
839   \@bsphack
840   \long\gdef\StopEventually##1{##1\endinput}%
841   \@esphack
842 }
843 \let\Finale\relax
```

`\MakePercentComment`

`\MakePercentIgnore`

```
844 \def\MakePercentIgnore{\catcode'\%9\relax}
845 \def\MakePercentComment{\catcode'\%14\relax}
```

`\DocInput`

```
846 \def\DocInput#1{\MakePercentIgnore\input{#1}\
      MakePercentComment}
```

\CharacterTable

```
847 \providecommand*\CharacterTable{%
848     \begingroup
849     \CharTableChanges
850     \@CharacterTable
851 }
852 \def\@CharacterTable#1{%
853     \def\ydoc@used@CharacterTable{#1}%
854     \@onelevel@sanitize\ydoc@used@CharacterTable
855     \ifx\ydoc@used@CharacterTable\
856         ydoc@correct@CharacterTable
857         \typeout{*****}%
858         \typeout{* Character table correct *}%
859         \typeout{*****}%
860     \else
861         \PackageError{ydoc}{Character table
862             corrupted}
863             {\the\wrong@table}
864         \show\ydoc@used@CharacterTable
865         \show\ydoc@correct@CharacterTable
866     \fi
867     \endgroup
868 }
869 \newhelp\wrong@table{Some of the ASCII characters are
    corrupted.^^J
    I now \string\show\space you both tables
    for comparison.}
870 \newcommand*\CharTableChanges{}
```

\ydoc@correct@CharacterTable

```
870 \def\ydoc@correct@CharacterTable
871 {Upper-case \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
872 Lower-case \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
873 Digits \0\1\2\3\4\5\6\7\8\9
874 Exclamation \! Double quote \" Hash (\
    number) \#
875 Dollar \$ Percent \% \
    Ampersand \&
876 Acute accent \' Left paren \( Right \
    paren\)
877 Asterisk * Plus + Comma ,
878 Minus - Point . Solidus /
    \/}
```

```

879 Colon          \:      Semicolon      \;      Less ✓
      than        \<
880 Equals         \=      Greater than  \>      Question ✓
      mark \?
881 Commercial at \@    Left bracket  \[      ✓
      Backslash    \\
882 Right bracket  \]      Circumflex   \^      ✓
      Underscore   \_
883 Grave accent  \`      Left brace   \{      Vertical ✓
      bar \|
884 Right brace   \}      Tilde        \~}
885 \@onelevel@sanitize\ydoc@correct@CharacterTable
886 %

```

`\DoNotIndex`

```

887 \providecommand*\DoNotIndex[1]{%
888   \PackageWarning{ydoc}{Ignoring DoNotIndex - not ✓
      implemented yet!}}{}%
889 }

```

`\changes`

```

890 \providecommand*\changes[3]{%
891   \PackageWarning{ydoc}{Ignoring changes - not ✓
      implemented yet!}}{}%
892 }

```

`\RecordChanges`

```

893 \providecommand*\RecordChanges{%
894   \PackageWarning{ydoc}{List of changes not ✓
      implemented yet!}}{}%
895 }

```

`\PrintChanges`

```

896 \providecommand*\PrintChanges{%
897   \PackageWarning{ydoc}{List of changes not ✓
      implemented yet!}}{}%
898 }

```


`\PrintIndex`

```
899 \providecommand*\PrintIndex{%  
900   \PackageWarning{ydoc}{Code index not implemented ✓  
      yet!}{}{}%  
901 }
```

`\CodelineIndex`

```
902 \providecommand*\CodelineIndex{%  
903   \PackageWarning{ydoc}{Code line index not ✓  
      implemented yet!}{}{}%  
904 }
```

`\EnableCrossrefs`

```
905 \providecommand*\EnableCrossrefs{%  
906   \PackageWarning{ydoc}{Cross references not ✓  
      implemented yet!}{}{}%  
907 }
```

`\GetFileInfo`

Current implementation taken from doc package.

```
908 \providecommand*\GetFileInfo [1]{%  
909   \def\filename{#1}%  
910   \def\@tempb##1 ##2 ##3\relax##4\relax{%  
911     \def\filedate{##1}%  
912     \def\fileversion{##2}%  
913     \def\fileinfo{##3}}%  
914   \edef\@tempa{\csname ver@#1\endcsname}%  
915   \expandafter\@tempb\@tempa\relax? ? \relax\relax  
916 }
```

`\ydoc@checkchecksum`

```
917 \def\ydoc@checkchecksum{%  
918   \ifnum\ydoc@checksum=\m@ne  
919     \message{^^J}%  
920     \message{*****^^J}%  
921     \message{* No checksum found! *^^J}%  
922     \message{*****^^J}%
```

```

923     \GenericWarning{No checksum found}{Correct ✓
          checksum is \the\ydoc@bslashcnt^^J}{}}%
924 \else
925 \ifnum\ydoc@checksum=\z@
926   \message{^^J}%
927   \message{*****^^J}%
928   \message{* Checksum disabled *^^J}%
929   \message{*****^^J}%
930   \GenericWarning{Checksum disabled}{Correct ✓
          checksum is \the\ydoc@bslashcnt^^J}{}}%
931 \else
932 \ifnum\ydoc@checksum=\ydoc@bslashcnt
933   \message{^^J}%
934   \message{*****^^J}%
935   \message{* Checksum passed *^^J}%
936   \message{*****^^J}%
937 \else
938   \message{^^J}%
939   \message{*****^^J}%
940   \message{* Checksum wrong (\ydoc@checksum<>\the\
ydoc@bslashcnt) ^^J}%
941   \message{*****^^J}%
942   \GenericError{Checksum wrong}{Correct checksum is ✓
          \the\ydoc@bslashcnt^^J}{}}%
943 \fi
944 \fi
945 \fi
946 }

947 \RequirePackage{shortvrb}
948 \AtBeginDocument{\MakeShortVerb{\|}}

949 \RequirePackage{url}
950
951 \def\package{\def\@package}
952 \package{\jobname}
953
954 \def\ctanlocation{\def\@ctanlocation##1}
955 \ctanlocation{http://www.ctan.org/pkg/#1}
956
957 \date{Version \fileversion\space -- \filedate}
958
959 \def\@homepage{%
960   \begingroup
961   \edef\@tempa{%
962     \endgroup
963     \noexpand\url
964     {\@ctanlocation{\@package}}}%
965   }%
966   \@tempa

```

```

967 }
968
969 \protected\def\homepage{\urldef\@homepage\url}
970 \protected\def\email{\hyper@normalise\email@}
971 \def\email@#1{\def\@plainemail{#1}\def\@email{\%
    hyper@linkurl{\Hurl{#1}}{mailto:#1}}
972
973 \let\@email\empty
974
975 \let\@plainemail\empty
976 \title{The \texorpdfstring{\pkgtitle{\@package}}{\%
    @package} Package}
977
978 \protected\def\pkgtitle#1{%
979     \texorpdfstring{\textsf{#1}}{#1}%
980 }
981
982 \def\@maketitle{%
983     \newpage
984     \null\vskip 2em
985     \begin{center}%
986         \let\footnote\thanks
987         {\LARGE \@title \par }\vskip 1.5em%
988         {\large \lineskip .5em%
989         \begin{tabular}[t]{c}%
990             \@author
991         \end{tabular}}%
992         \par}%
993         \ifx\@plainemail\empty\else
994             {\large \lineskip .5em%
995             \begin{tabular}[t]{c}%
996                 \@email
997             \end{tabular}}%
998             \par}%
999         \fi
1000         \vskip 1em
1001         {\large \lineskip .5em%
1002         \begin{tabular}[t]{c}%
1003             \@homepage
1004         \end{tabular}}%
1005         \par}%
1006         \vskip 1em
1007         {\large \@date }%
1008     \end{center}%
1009     \par\vskip 1.5em
1010     \aftergroup\ydocpdfsettings
1011 }
1012
1013 \ifpdf
1014 \def\ydocpdfsettings{%

```

```

1015     \hypersetup{%
1016         pdfauthor    = {\@author\space<\@plainemail>},
1017         pdftitle     = {\@title},
1018         pdfsubject   = {Documentation of LaTeX package✓
                        \@package},
1019         pdfkeywords = {\@package, LaTeX, TeX}
1020     }%
1021 }
1022 \else
1023 \let\ydocpdfsettings\empty
1024 \fi
1025
1026 \let\orig@maketitle\maketitle
1027 \def\maketitle{%
1028     \ydocpdfsettings
1029     \orig@maketitle
1030     \let\orig@maketitle\relax
1031 }

```

2.10 Include Code Examples

```

1032 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
1033 \RequirePackage{svn-prov}[2010/04/03]
1034 \ProvidesPackageSVN[ydoc]
1035     {$Id: ydoc_expl_sty.dtx 2332 2011-03-17 17:12:35Z✓
        martin $}
1036 %<!VERSION>
1037 %<*DRIVER>
1038     [develop]
1039 %</DRIVER>
1040     [ydoc package to insert live examples of LaTeX ✓
        code]
1041 \RequirePackage{listings}
1042 \lst@RequireAspects{writefile}
1043 \def\ydoc@exafile{\jobname.exa}

```

\exampleprintsettings

```

1044 \def\exampleprintsettings{numbers=left,numberstyle=\✓
        tiny\color{gray}\sffamily,numbersep=5pt}%
1045 \newbox\examplecodebox
1046 \newbox\exampleresultbox

```

\BoxExample

```
1047 \def\BoxExample{%
1048   \setbox\examplecodebox\hbox{\color@setgroup
1049     \expandafter\expandafter\expandafter\
1050       lstinputlisting
1051     \expandafter\expandafter\expandafter [%
1052       \expandafter\exampleprintsettings\expandafter ,\
1053         thisexampleprintsettings]%
1054     {\ydoc@exafile}%
1055   \unskip\color@endgroup}%
1056 \setbox\exampleresultbox\hbox{\color@setgroup
1057   \@@input\ydoc@exafile\relax
1058   \unskip\color@endgroup}%
1059 }
```

\PrintExample

```
1058 %<DISABLED>
1059 \RequirePackage{showexpl}
1060 \def\PrintExample{%
1061   \begingroup
1062   \lstset{basicstyle=\ttfamily}%
1063   \MakePercentComment
1064   \LTXinputExample [varwidth]{\ydoc@exafile}%
1065   \endgroup
1066 }
1067 %</DISABLED>
```

\PrintExample

```
1068 \def\PrintExample{%
1069   \begingroup
1070   \BoxExample
1071   \@tempdima=\textwidth
1072   \advance\@tempdima by -\wd\examplecodebox\relax
1073   \advance\@tempdima by -\wd\exampleresultbox\relax
1074   \advance\@tempdima by -15pt\relax
1075   \ifdim\@tempdima>\bigskipamount
1076     \hbox to \textwidth{%
1077       \null\hss
1078       \minipage [c]{\wd\exampleresultbox}\fbox{\usebox\
1079         exampleresultbox}\endminipage
1080     }
1081   \hfill\hfill\hskip\bigskipamount\hskip15pt\hfill\
1082   \hfill
1083 }
```

```

1080     \minipage[c]{\wd\examplecodebox}\usebox\✓
        examplecodebox\endminipage
1081     \hss\null
1082   }%
1083   \else
1084     \vbox{%
1085       \centerline{\fbox{\usebox\exampleresultbox}}%
1086       \vspace{\bigskipamount}%
1087       \centerline{\usebox\examplecodebox}%
1088     }%
1089   \fi
1090   \endgroup
1091 }

1092 \def\examplecodesettings{gobble=4}

```

examplecode

```

1093 \lstnewenvironment{examplecode}[1][\]{%
1094   \def\thisexampleprintsettings{#1}%
1095   \expandafter\lstset\expandafter{\✓
        examplecodesettings,#1}%
1096   \setbox\@tempboxa\hbox\bgroup
1097   \lst@BeginWriteFile{\ydoc@exafile}%
1098 }
1099 {%
1100   \lst@EndWriteFile
1101   \egroup
1102   \begingroup
1103   \MakePercentComment
1104   \catcode'\^^M=5\relax
1105   \PrintExample
1106   \endgroup
1107 }

1108 \RequirePackage{float}

```

example

```

1109 \floatstyle{plain}
1110 \newfloat{example}{tbhp}{loe}
1111 \floatname{example}{\examplename}
1112 \def\examplename{Example}

```

exampletable

```
1113 \newenvironment{exampletable}{%  
1114   \floatstyle{plaintop}%  
1115   \restylefloat{example}%  
1116   \example  
1117 }{\endexample}
```