

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

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. This packages can be loaded on their own in other kind of \LaTeX 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: “%`UUUU`\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 including 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 1) 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 T_EX group should be started using `{ }` directly 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 be 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 which 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.

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

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

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

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

```
\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 1. 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.

<code>\DescribeLength<\name>{<default value>}</code>
--

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

2.3 Format Macros

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

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

<code>\bslash</code>	<code>\percent</code>	<code>\braceleft</code>	<code>\braceright</code>
----------------------	-----------------------	-------------------------	--------------------------

This macros define expandable backslash (\backslash), percent char ($\%$), and left ($\{$) and right ($\}$) braces with catcode 12 (other), respectively. They should only be used with text-typewriter 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.

<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 1 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.

`\descsep` (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

```
\begin{example}  
\end{example}
```

```
\begin{examplecode}  
\end{examplecode}
```

(to be written)

3 Implementation

3.1 Class File

```
1 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
2 \ProvidesClass{ydoc}[%
3 %<!DATE>
4 %<!VERSION>
5 %<*DRIVER>
6     2011/08/11 develop
7 %</DRIVER>
8     ydoc class: document LaTeX class and packages]
9
10 At the moment simply load article class with a4paper option and load the ydoc
11 package.
12
13 \PassOptionsToClass{a4paper}{article}
14 \DeclareOption*{\expandafter\PassOptionsToClass\relax
15     \expandafter{\CurrentOption}{article}}
16 \ProcessOptions\relax
17 \LoadClass{article}
18 \RequirePackage{ydoc}
```

3.2 Package File

```
14 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
15 \ProvidesPackage{ydoc}[%
16 %<!DATE>
17 %<!VERSION>
18 %<*DRIVER>
19     2011/08/11 develop
20 %</DRIVER>
21     ydoc package: document LaTeX class and packages]
22
23 \RequirePackage{svn-prov}[2010/04/03]
24 \RequirePackage{ydoc-code}
25 \RequirePackage{ydoc-expl}
26 \RequirePackage{ydoc-desc}
27 \RequirePackage{ydoc-doc}
28
29 \RequirePackage{newverbs}
30 \MakeSpecialShortVerb{\qverb}{\"}
31 \AtBeginDocument{\catcode'\^A=14\relax}
32
33 \input{ydoc.cfg}
```

3.3 Config File


```

33 %% Please delete the following line on manual changes/
34 :
35 \ProvidesFile{ydoc.cfg}[%
36 %<!DATE>
37 %<!VERSION>
38 %<*DRIVER>
39 2011/08/11 develop
40 %</DRIVER>
41 Default config file for ydoc]

42 \usepackage[T1]{fontenc}
43 \IfFileExists{fourier.sty}{%
44 \usepackage{fourier}
45 }{}

46 Use 'lmodern' only for the 'tt' font if fourier is installed.
47 \IfFileExists{lmodern.sty}{
48 \IfFileExists{fourier.sty}{
49 \renewcommand{\ttdefault}{lmtt}
50 }{
51 \usepackage{lmodern}
52 }
53 }{}

54 \urlstyle{sf}

55 Use micro-typesetting if pdftex is used:
56 \usepackage{ifpdf}
57 \ifpdf
58 \usepackage{microtype}
59 \fi

60 \usepackage{array}
61 \usepackage{booktabs}
62 \usepackage{multicol}
63 \usepackage{xcolor}
64 \usepackage{listings}
65 \usepackage{booktabs}
66 \usepackage{hyperref}

67 \reversemarginpar

```

3.4 Macros and Environments to document Implementations

```

65 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
66 \ProvidesPackage{ydoc-code}[%
67 %<!DATE>
68 %<!VERSION>
69 %<*DRIVER>

```

```

70      2011/08/11 develop
71 %</DRIVER>
72      ydoc package to document macro code]

73 \RequirePackage{hyperref}
74 \hypersetup{colorlinks=true,pdfborder=0 0 0,/
      pdfborderstyle={}}

75 \IfFileExists{needspace.sty}{%
76     \RequirePackage{needspace}
77 }{%
78     \def\Needspace{\@ifstar\@gobble\@gobble}
79 }

```

3.4.1 Color and style definitions

```

80 \RequirePackage{xcolor}
81 \definecolor{macroimpl}{rgb}{0.0,0.0,0.4}

```

3.4.2 General Macros

`\ydocwrite`

```

82 \@ifundefined{ydocwrite}{%
83     \newwrite\ydocwrite
84 }{}

```

`\ydocfname`

```

85 \@ifundefined{ydocfname}{%
86     \def\ydocfname{\jobname.cod}%
87 }{}

```

`\ydoc@catcodes`

```

88 \def\ydoc@catcodes{%
89     \let\do\@makeother
90     \dospecials
91     \catcode'\=\active
92     \catcode'\^M=\active
93     \catcode'\ =\active
94 }

```

3.4.3 Handling Macrocode

macrocode

```
95 \def\macrocode{%
96   \par\noindent
97   \begingroup
98   \ydoc@catcodes
99   \macro@code
100 }
101 \def\endmacrocode{}
```

\macro@code

#1: verbatim macro code

```
102 \begingroup
103 \endlinechar\m@ne
104 \@firstofone{%
105 \catcode'\|=0\relax
106 \catcode'\(=1\relax
107 \catcode'\)=2\relax
108 \catcode'\*=14\relax
109 \catcode'\{=12\relax
110 \catcode'\}=12\relax
111 \catcode'\ =12\relax
112 \catcode'\%=12\relax
113 \catcode'\\=\active
114 \catcode'\^M=\active
115 \catcode'\ =\active
116 }*
117 |gdef|macro@code#1^~M%      \end{macrocode}(*
118 |endgroup|expandafter|macro@@code|expandafter(|/
   ydoc@removeline#1|noexpand|lastlinemacro)*
119 )*
120 |gdef|ydoc@removeline#1^~M(|noexpand|firstlinemacro)*
121 |gdef|ydoc@defspecialmacros(*
122 |def^~M(|noexpand|newlinemacro)*
123 |def (|noexpand|spacemacro)*
124 |def\(|noexpand|bslashmacro)*
125 )*
126 |gdef|ydoc@defrevspecialmacros(*
127 |def|newlinemacro(|noexpand^~M)*
128 |def|spacemacro(|noexpand )*
129 |def|bslashmacro(|noexpand\)*
130 )*
131 |endgroup
```

`\macro@@code`

#1: verbatim macro code

```
132 \def\macro@@code#1{%
133     {\ydoc@defspecialmacros
134     \xdef\thetmacrocode{#1}}}%
135     \PrintMacroCode
136     \end{macrocode}%
137 }
```

`\linenumberbox`

```
138 \def\newlinemacro{\\\null}
139 \def\spacemacro{\ }
140 \def\bslashmacro{\char92}
141 \def\lastlinemacro{}
142 \def\firstlinemacro{\linenumberbox}
143 \def\newlinemacro{\\\linenumberbox}
144 \newcounter{linenumber}
145 \def\linenumberbox{%
146     \hbox to 1.25em{}%
147     \llap{%
148         \stepcounter{linenumber}%
149         {\footnotesize\color{gray}\thelinenumber~}%
150     }%
151 }
```

`\PrintMacroCode`

```
152 \def\PrintMacroCode{%
153     \begingroup
154     \ttfamily
155     \noindent\thetmacrocode
156     \endgroup
157 }
```

`\PrintMacroCode`

```
158 \RequirePackage{listings}

159 \def\PrintMacroCode{%
160     \begingroup
161     \let\firstlinemacro\empty
162     \let\lastlinemacro\empty
```

```

163 \def\newlinemacro{^^J}%
164 \let\bslashmacro\bslash
165 \let\spacemacro\space
166 \immediate\openout\ydocwrite=\ydocfname\relax
167 \immediate\write\ydocwrite{\themacrocode}%
168 \immediate\closeout\ydocwrite
169 \@nameuse{ydoc@countbslashes}%
170 \ydoclistingssettings
171 \let\input\@input
172 \lstinputlisting{\ydocfname}%
173 \endgroup
174 }

```

\ydoclistingssettings

```

175 \def\ydoclistingssettings{%
176   \lstset{%
177     language=[latex]tex,basicstyle=\ttfamily,
178     numbers=left,numberstyle=\tiny\color{gray},/
179     firstnumber=last,
180     breaklines,prebreak={\mbox{\tiny$\swarrow$}},
181     commentstyle=\color{black!60},
182   }%
183 \ydoclistingssettings

```

\macro@impl@args

#1: number of macro arguments

```

184 \def\macro@impl@args[#1]{%
185   \begingroup
186   \parindent=10pt\relax
187   \let\macro@impl@argcnt\@tempcnta
188   \let\macro@impl@curarg\@tempcntb
189   \macro@impl@argcnt=#1\relax
190   \macro@impl@curarg=0\relax
191   \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
192     \expandafter\macro@impl@arg
193   \else
194     \expandafter\macro@impl@endargs
195   \fi
196 }

```

\macro@impl@endargs

```
197 \def\macro@impl@endargs{%
198   \endgroup
199   \unskip\par\noindent\ignorespaces
200 }
```

\macro@impl@argline

#1: argument number
#2: argument description

```
201 \def\macro@impl@argline#1#2{%
202   \par{\texttt{\char"00#1}:~#2\strut}%
203 }
```

\macro@impl@arg

#1: argument description

```
204 \def\macro@impl@arg#1{%
205   \advance\macro@impl@curarg by\@ne\relax
206   \macro@impl@argline{\the\macro@impl@curarg}{#1}%
207   \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
208     \expandafter\macro@impl@arg
209   \else
210     \expandafter\macro@impl@endargs
211   \fi
212 }
```

macro

#1: implemented macro

```
213 \def\macro#1{%
214   \PrintMacroImpl{#1}%
215   \@ifnextchar[%]
216     {\macro@impl@args}%
217     {}%
218 }
219 \def\endmacro{}
```

key

#1: key family
#2: key name

```

220 \def\key#1#2{%
221   \PrintMacroImpl{KV@#1@#2}%
222   \@ifnextchar[%]
223     {\macro@impl@args}%
224     {}%
225 }
226 \def\endkey{}

```

environment

#1: environment name

```

227 \def\environment#1{%
228   \PrintEnvImplName{#1}%
229   \@ifnextchar[%]
230     {\macro@impl@args}%
231     {}%
232 }
233 \def\endenvironment{}

```

style

#1: style name

```

234 \def\style#1{%
235   \PrintStyleImplName{#1}%
236   \@ifnextchar[%]
237     {\macro@impl@args}%
238     {}%
239 }
240 \def\endstyle{}
241 \def\PrintStyleImplName{\PrintEnvImplName}

```

\PrintMacroImpl

#1: macro (token)

```

242 \def\PrintMacroImpl#1{%
243   \par\bigskip\noindent
244   \Needspace*{3\baselineskip}%
245   \hbox{%
246     \edef\name{\expandafter\@gobble\string#1}%
247     \global\@namedef{href@impl@\name}{}%
248     \immediate\write\@mainaux{%
249       \global\noexpand\@namedef{href@impl@\name}{}%
250     }%
251     \raisebox{4ex}[4ex]{\hypertarget{impl:\name}{}%
252     \hspace*{\descindent}\fbox{%
253       \hspace*{\descsep}%

```

```

254     \@ifundefined{href@desc@name}{\hyperlink{
        desc:name}}%
255     {\PrintMacroImplName{#1}}%
256     \hspace*{\descsep}%
257   }%
258 }%
259 \par\medskip\noindent
260 }

```

`\PrintMacroImplName`

#1: macro (token)

```

261 \def\PrintMacroImplName#1{%
262   \implstyle{\string#1\strut}%
263 }

```

`\PrintEnvImplName`

#1: environment name

test

```

264 \def\PrintEnvImplName#1{%
265   \par\bigskip\noindent
266   \hbox{\hspace*{\descindent}\fbox{\implstyle{#1}}}%
      %
267   \par\medskip
268 }

```

`\implstyle`

```

269 \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.

```

270 {%
271 \@firstofone{%
272   \catcode'\_12
273   \gdef\bslash
274 }{\_12}
275 }%

```


3.5 Provide doc macros

```
276 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
277 \ProvidesPackage{ydoc-doc}[%
278 %<!DATE>
279 %<!VERSION>
280 %<*DRIVER>
281     2011/08/11 develop
282 %</DRIVER>
283     ydoc package to provide 'doc' macros]
```

`\ydoc@countbslashes`

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

```
284 \newcount\ydoc@bslashcnt
285 \def\ydoc@countbslashes{%
286     \begingroup
287     \let\firstlinemacro\empty
288     \let\lastlinemacro\empty
289     \let\newlinemacro\empty
290     \let\spacemacro\empty
291     \def\bslashmacro{\global\advance\ydoc@bslashcnt /
292         by\@ne}%
293     \setbox\@tempboxa\hbox{\themacrocode}%
294     \endgroup
295 }
```

`\Checksum`

```
295 \def\Checksum#1{%
296     \gdef\ydoc@checksum{#1}%
297 }
298 \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.

```
299 \def\AlsoImplementation{%
300   \gdef\StopEventually##1{%
301     \@bsphack
302     \gdef\Finale{##1\ydoc@checkchecksum}%
303     \@esphack
304   }%
305 }
306 \AlsoImplementation
307 \def\OnlyDescription{%
308   \@bsphack
309   \long\gdef\StopEventually##1{##1\endinput}%
310   \@esphack
311 }
312 \let\Finale\relax
```

`\MakePercentComment`

`\MakePercentIgnore`

```
313 \def\MakePercentIgnore{\catcode'\%9\relax}
314 \def\MakePercentComment{\catcode'\%14\relax}
```

`\DocInput`

```
315 \def\DocInput#1{\MakePercentIgnore\input{#1}\%
    MakePercentComment}
```

`\CharacterTable`

```
316 \providecommand*\CharacterTable{%
317   \begingroup
318   \CharTableChanges
319   \@CharacterTable
320 }
321 \def\@CharacterTable#1{%
322   \def\ydoc@used@CharacterTable{#1}%
323   \@onelevel@sanitize\ydoc@used@CharacterTable
324   \ifx\ydoc@used@CharacterTable\%
     ydoc@correct@CharacterTable
```

```

325         \typeout{*****}%
326         \typeout{* Character table correct *}%
327         \typeout{*****}%
328     \else
329         \PackageError{ydoc}{Character table /
            corrupted}
330             {\the\wrong@table}
331         \show\ydoc@used@CharacterTable
332         \show\ydoc@correct@CharacterTable
333     \fi
334 \endgroup
335 }
336 \newhelp\wrong@table{Some of the ASCII characters are/
    corrupted.^^J
337     I now \string\show\space you both tables /
        for comparison.}
338 \newcommand*\CharTableChanges{}

```

\ydoc@correct@CharacterTable

```

339 \def\ydoc@correct@CharacterTable
340 {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
341 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
342 Digits \0\1\2\3\4\5\6\7\8\9
343 Exclamation \! Double quote \" Hash (/
    number) \#
344 Dollar \$ Percent \% /
    Ampersand \&
345 Acute accent \' Left paren \(( Right /
    paren \)
346 Asterisk \* Plus \+ Comma /
    \,
347 Minus \- Point \. Solidus /
    \/
348 Colon \: Semicolon \; Less /
    than \<
349 Equals \= Greater than \> Question/
    mark \?
350 Commercial at \@ Left bracket \[ /
    Backslash \\
351 Right bracket \] Circumflex \^ /
    Underscore \_
352 Grave accent \' Left brace \{ Vertical/
    bar \|
353 Right brace \} Tilde \~}
354 \@onelevel@sanitize\ydoc@correct@CharacterTable

```

355 %

\DoNotIndex

```
356 \providecommand*\DoNotIndex[1]{%
357   \PackageWarning{ydoc}{Ignoring DoNotIndex - not /
      implemented yet!}{}{}%
358 }
```

\changes

```
359 \providecommand*\changes[3]{%
360   \PackageWarning{ydoc}{Ignoring changes - not /
      implemented yet!}{}{}%
361 }
```

\RecordChanges

```
362 \providecommand*\RecordChanges{%
363   \PackageWarning{ydoc}{List of changes not /
      implemented yet!}{}{}%
364 }
```

\PrintChanges

```
365 \providecommand*\PrintChanges{%
366   \PackageWarning{ydoc}{List of changes not /
      implemented yet!}{}{}%
367 }
```

\PrintIndex

```
368 \providecommand*\PrintIndex{%
369   \PackageWarning{ydoc}{Code index not implemented /
      yet!}{}{}%
370 }
```

\CodelineIndex

```
371 \providecommand*\CodelineIndex{%
372   \PackageWarning{ydoc}{Code line index not /
      implemented yet!}{}{}%
373 }
```

\EnableCrossrefs

```
374 \providecommand*\EnableCrossrefs{%
375   \PackageWarning{ydoc}{Cross references not
      implemented yet!}{\}\}%
376 }
```

\GetFileInfo

Current implementation taken from doc package.

```
377 \providecommand*\GetFileInfo[1]{%
378   \def\filename{#1}%
379   \def\@tempb##1 ##2 ##3\relax##4\relax{%
380     \def\filedate{##1}%
381     \def\fileversion{##2}%
382     \def\fileinfo{##3}}%
383   \edef\@tempa{\csname ver@#1\endcsname}%
384   \expandafter\@tempb\@tempa\relax? ? \relax\relax
385 }
```

\ydoc@checkchecksum

```
386 \def\ydoc@checkchecksum{%
387   \ifnum\ydoc@checksum=\m@ne
388     \message{^^J}%
389     \message{*****^^J}%
390     \message{* No checksum found! *^^J}%
391     \message{*****^^J}%
392     \GenericWarning{No checksum found}{Correct
      checksum is \the\ydoc@bslashcnt^^J}{\}\}%
393   \else
394     \ifnum\ydoc@checksum=\z@
395       \message{^^J}%
396       \message{*****^^J}%
397       \message{* Checksum disabled *^^J}%
398       \message{*****^^J}%
399       \GenericWarning{Checksum disabled}{Correct
      checksum is \the\ydoc@bslashcnt^^J}{\}\}%
400     \else
401       \ifnum\ydoc@checksum=\ydoc@bslashcnt
402         \message{^^J}%
403         \message{*****^^J}%
404         \message{* Checksum passed *^^J}%
405         \message{*****^^J}%
406       \else
407         \message{^^J}%
```

```

408 \message{*****~J}%
409 \message{* Checksum wrong (\ydoc@checksum<>\the\
ydoc@bslashcnt) ~J}%
410 \message{*****~J}%
411 \GenericError{Checksum wrong}{Correct checksum is\
\the\ydoc@bslashcnt~J}{}{}%
412 \fi
413 \fi
414 \fi
415 }

416 \RequirePackage{shortvrb}
417 \AtBeginDocument{\MakeShortVerb{\|}}

418 \RequirePackage{url}
419
420 \def\package{\def\@package}
421 \package{\jobname}
422
423 \def\ctanlocation{\def\@ctanlocation##1}
424 \ctanlocation{http://www.ctan.org/pkg/#1}
425
426 \date{Version \fileversion\space -- \filedate}
427
428 \def\@homepage{%
429 \begingroup
430 \edef\@tempa{%
431 \endgroup
432 \noexpand\url
433 {\@ctanlocation{\@package}}}%
434 }%
435 \@tempa
436 }
437
438 \protected\def\homepage{\urldef\@homepage\url}
439 \protected\def\email{\hyper@normalise\email@}
440 \def\email@#1{\def\@plainemail{#1}\def\@email{\
hyper@linkurl{\Hurl{#1}}{mailto:#1}}}
441
442 \let\@email\empty
443
444 \let\@plainemail\empty
445 \title{The \texorpdfstring{\pkgtitle{\@package}}{\
@package} Package}
446
447 \protected\def\pkgtitle#1{%
448 \texorpdfstring{\textsf{#1}}{#1}%
449 }
450
451 \def\@maketitle{%

```

```

452 \newpage
453 \null\vskip 2em
454 \begin{center}%
455     \let\footnote\thanks
456     {\LARGE \@title \par }\vskip 1.5em%
457     {\large \lineskip .5em%
458     \begin{tabular}[t]{c}%
459         \@author
460     \end{tabular}%
461     \par}%
462     \ifx\@plainemail\empty\else
463         {\large \lineskip .5em%
464         \begin{tabular}[t]{c}%
465             \@email
466         \end{tabular}%
467         \par}%
468     \fi
469     \vskip 1em
470     {\large \lineskip .5em%
471     \begin{tabular}[t]{c}%
472         \@homepage
473     \end{tabular}%
474     \par}%
475     \vskip 1em
476     {\large \@date }%
477 \end{center}%
478 \par\vskip 1.5em
479 \aftergroup\ydocpdfsettings
480 }
481
482 \ifpdf
483 \def\ydocpdfsettings{%
484     \hypersetup{%
485         pdfauthor    = {\@author\space<\@plainemail>},
486         pdftitle     = {\@title},
487         pdfsubject   = {Documentation of LaTeX package/
488             \@package},
489         pdfkeywords  = {\@package, LaTeX, TeX}
490     }%
491 }
492 \else
493 \let\ydocpdfsettings\empty
494 \fi
495 \let\orig@maketitle\maketitle
496 \def\maketitle{%
497     \ydocpdfsettings
498     \orig@maketitle
499     \let\orig@maketitle\relax
500 }

```

3.6 Description Macros and Environments

```
501 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
502 \ProvidesPackage{ydoc-desc}[%
503 %<!DATE>
504 %<!VERSION>
505 %<*DRIVER>
506     2011/08/11 develop
507 %</DRIVER>
508     ydoc package to describe macros, environments, /
        options etc.]

509 \IfFileExists{needspace.sty}{%
510     \RequirePackage{needspace}
511 }{%
512     \def\Needspace{\@ifstar\@gobble\@gobble}
513 }
```

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

```
514 \RequirePackage{shortvrb}
```

The etoolbox package is used mainly for \newrobustcmd.

```
515 \RequirePackage{etoolbox}
```

3.6.1 Color and style definitions

```
516 \RequirePackage{xcolor}
517 \definecolor{macrodesc}{rgb}{0,0.2,0.6}
518 \definecolor{keydesc}{rgb}{0,0.4,0.9}
519 \definecolor{macroimpl}{rgb}{0,0.1,0.3}
520 \definecolor{meta}{rgb}{0,0.25,0.75}
521 \definecolor{scriptcolor}{rgb}{0.2,0.6,0.2}
522 \definecolor{optioncolor}{rgb}{0.3,0.2,0}
523 \colorlet{optional}{black!65!white}
524 \colorlet{metaoptional}{optional!50!meta}
525 \providecolor{urlcolor}{named}{blue}
526 \providecolor{linkcolor}{named}{blue}
527 \providecolor{filecolor}{named}{blue}
528 \providecolor{citecolor}{named}{blue}
529 \providecolor{anchorcolor}{named}{blue}
530 \providecolor{menucolor}{named}{blue}
531 \providecolor{runcolor}{named}{blue}

533 \RequirePackage{hyperref}
534 \hypersetup{%
535     colorlinks=true,
536     pdfborder=0 0 0,
537     pdfborderstyle={},
538     urlcolor=urlcolor,
```



```

539     linkcolor=linkcolor ,
540     filecolor=filecolor ,
541     citecolor=citecolor ,
542     anchorcolor=anchorcolor ,
543     menucolor=menucolor ,
544     runcolor=runcolor ,
545 }

```

3.6.2 Text Formatting Macros

\meta

Prints *meta text*.

```

547 \newrobustcmd*\meta[1]{%
548   {\metastyle{%
549     \ensuremath\langle
550     #1\/%
551     \ensuremath\rangle
552   }}%
553 }

```

\marg

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

```

554 \newrobustcmd*\marg[1]{%
555   {\margstyle{%
556     {\ttfamily\braceleft}%
557     \meta{#1}%
558     {\ttfamily\braceright}%
559   }}%
560 }

```

\oarg

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

```

561 \newrobustcmd*\oarg[1]{%
562   {\oargstyle{%
563     {\ttfamily[]}%
564     \meta{#1}%
565     {\ttfamily[]}%
566   }}%
567 }

```

`\parg`

Sets style and adds parentheses.

```
568 \newrobustcmd*{\parg}[1]{%  
569   {\pargstyle{%  
570     {\ttfamily}%  
571     \meta{#1}%  
572     {\ttfamily}}}%  
573   }}%  
574 }
```

`\aarg`

Sets style and adds angles.

```
575 \newrobustcmd*{\aarg}[1]{%  
576   {\aargstyle{%  
577     {\ttfamily<%  
578     \meta{#1}%  
579     {\ttfamily>%  
580     }}%  
581   }}
```

`\sarg`

Prints star with given style.

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

`\pkg`

`\cls`

`\lib`

`\env`

`\opt`

`\file`

```
583 \newrobustcmd*\pkg[1]{\pkgstyle{#1}}
584 \newrobustcmd*\cls[1]{\clsstyle{#1}}
585 \newrobustcmd*\lib[1]{\libstyle{#1}}
586 \newrobustcmd*\env[1]{\envstyle{#1}}
587
588 \newrobustcmd*\opt{\@ifstar\ys@opt\y@opt}
589 \def\y@opt#1{\optstyle{#1}}
590 \def\ys@opt#1{\optstyle{#1}\optpar{#1}}
591 \newrobustcmd*\optpar[1]{\marginpar{\hbox to \marginparwidth{\hss\y@opt{#1}}}}
592
593 \newrobustcmd*\file[1]{\filestyle{#1}}
594 \newcommand*\pkgstyle[1]{\texttt{\textcolor{pkg/}{#1}}}
595 \newcommand*\clsstyle[1]{\texttt{\textcolor{cls/}{#1}}}
596 \newcommand*\libstyle[1]{\texttt{\textcolor{lib/}{#1}}}
597 \newcommand*\envstyle[1]{\texttt{\textcolor{env/}{#1}}}
598 \newcommand*\optstyle[1]{\textsf{\textcolor{opt/}{#1}}}
599 \newcommand*\filestyle[1]{\texttt{\textcolor{file/}{#1}}}
600 \colorlet{cls}{black}
601 \colorlet{lib}{black}
602 \colorlet{env}{black}
603 \colorlet{file}{black}
604 \colorlet{pkg}{black}
605 \definecolor{opt}{rgb}{0.5,0.16666,0}
```

`\cs`

`\cmd`

```
606 \newrobustcmd*\cs[1]{\texttt{\textbackslash #1}}
607 \newrobustcmd*\cmd[1]{\texttt{\escapechar=92\string/
#1}}
```

`\Key`

```
608 \newrobustcmd*\Key[1]{\PrintKeyName{#1}\MacroArgs}
```

3.6.3 Text Formatting Styles

`\macrodescstyle`

Style of described macro names.

```
609 \def\macrodescstyle{\ttfamily\bfseries\color{/  
macrodesc}}
```

`\macrodescstyle`

Style of described macro names.

```
610 \def\keydescstyle{\ttfamily\bfseries\color{keydesc}}
```

`\macroargsstyle`

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

```
611 \def\macroargsstyle{\ttfamily}
```

`\envcodestyle`

Default style for code body content in described environments.

```
612 \def\envcodestyle{\ttfamily}
```

`\verbstyle`

Style for verbatim text inside macro argument list.

```
613 \def\verbstyle{\verbatim@font}
```

`\metastyle`

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

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

`\margstyle`

Style for `\marg`.

```
615 \def\margstyle{}
```

`\Optional`

`\optional`

`\optionalstyle`

```
616 \protected\def\Optional{\optionalon\optional}  
617 \def\optionalstyle{\blendcolors*{!60!white}\color{/  
    black!75}}
```

`\optionalon`

`\optionaloff`

```
618 \def\optionalon{\protected\def\optional{\/  
    optionalstyle}}  
619 \def\optionaloff{\let\optional\relax}  
620 \optionalon
```

`\oargstyle`

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

```
621 \def\oargstyle{\optional}
```

`\pargstyle`

Style for `\parg`.

```
622 \def\pargstyle{}
```

`\aargstyle`

Style for `\aarg`.

```
623 \def\aargstyle{}
```

`\sargstyle`

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

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

3.6.4 Dimension Registers

`\descindent`

```
625 \newdimen\descindent
626 \descindent=-\parindent
```

`\beforedescskip`

```
627 \newdimen\beforedescskip
628 \beforedescskip=\bigskipamount
```

`\afterdescskip`

```
629 \newdimen\afterdescskip
630 \afterdescskip=\medskipamount
```

`\descsep`

Set to 1em in tt font.

```
631 \newdimen\descsep
632 \begingroup
633 \ttfamily
634 \global\descsep=1em\relax
635 \endgroup
```

3.6.5 Macro Argument Reading Mechanism

`\read@Macro@arg`

Reads next token and calls second macro.

```
636 \def\read@Macro@arg{%
637   \futurelet\@let@token\handle@Macro@arg
638 }
```

`\AlsoMacro`

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

```
639 \newcommand*\AlsoMacro{%
640   \begingroup\makeatletter
641   \AlsoMacro@
642 }
643 \def\AlsoMacro@#1{%
644   \endgroup
645   %<*DEBUG>
646   %\typeout{DEBUG: Macro: \string#1}%
647   %</DEBUG>
648   \PrintMacroName{#1}%
649   \read@Macro@arg
650 }
```

`\ydoc@short@AlsoMacro`

Makes & an alias for \AlsoMacro.

```
651 \begingroup
652 \catcode'\active
653 \gdef\ydoc@short@AlsoMacro{%
654   \catcode'\active
655   \let|\AlsoMacro
656 }
657 \endgroup
```

`\ydoc@macrocatcodes`

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

```
658 \def\ydoc@macrocatcodes{%
659   \ydoc@short@AlsoMacro
660   \@makeother\'%
661   \@makeother\!%
662   \@makeother\[%
663   \@makeother\]%
664   \@makeother\(%
665   \@makeother\) %
666 }
```

`\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.

```

667 \def\handle@Macro@arg{%
668   \expandafter\let\expandafter\handler\csname /
        handle@Macro@token@\meaning\@let@token\endcsname
669   \ifx\handler\relax
670     \def\handler{\ifhmode\unskip\fi\end@Macro@args}%
671   %<*DEBUG>
672   % \typeout{DEBUG: Stopped at: \expandafter\meaning\
        csname @let@token\endcsname}%
673   % \typeout{}%
674   %\else
675   %\expandafter\ifx\csname @let@token\endcsname\
        AlsoMacro
676   % \typeout{DEBUG: TOKEN: \string\AlsoMacro}%
677   %\else
678   % \typeout{DEBUG: TOKEN: \expandafter\meaning\
        csname @let@token\endcsname}%
679   %\fi
680   %</DEBUG>
681   \fi
682   \handler
683 }
684 \def\define@Macro@handler{%
685   \begingroup
686   \ydoc@macrocatcodes
687   \define@Macro@handler@
688 }
689 \def\define@Macro@handler@#1{%
690   \endgroup
691   \@namedef{handle@Macro@token@\meaning#1}%
692 }

```

\end@Macro@args

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

```

693 \def\end@Macro@args{%
694   \y@egroup
695   \after@Macro@args
696 }

```

\after@Macro@args

Hook to add additional commands in certain situations.

```

697 \def\after@Macro@args{%
698 }

```


Macro argument reading macros

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

`\read@Macro@marg`

```
699 \define@Macro@handler{\bgroup}{%
700     \begingroup
701     \afterassignment\read@Macro@marg@
702     \let\@let@token=%
703 }
704 \def\read@Macro@marg@{%
705     \bgroup
706     \margstyle{}%
707     \let\end@Macro@args\empty%
708     {\ttfamily\braceleft}%
709     \aftergroup\read@Macro@marg@@
710     \read@Macro@marg
711 }
712 \def\read@Macro@marg@@{%
713     {\ttfamily\braceright}%
714     \endgroup
715     \read@Macro@marg
716 }
```

`\read@Macro@oarg`

```
717 \define@Macro@handler{[]}{%
718     \begingroup
719     \let\read@Macro@oarg@end\read@Macro@oarg@@end
720     \let\end@Macro@args\read@Macro@oarg@end
721     \oargstyle{}%
722     {\ttfamily[]}%
723     \read@Macro@marg
724 }
725 \define@Macro@handler{[]}{%
726     \read@Macro@oarg@end
727 }
728 \def\read@Macro@oarg@@end#1{%
729     #1%
730     {\ttfamily[]}%
731     \endgroup
732     \read@Macro@marg
733 }
734 \def\read@Macro@oarg@end{\end@Macro@args}
735 \let\read@Macro@aarg@end\read@Macro@oarg@end
736 \let\read@Macro@parg@end\read@Macro@oarg@end
```

`\read@Macro@parg`

```
737 \define@Macro@handler{()}{%
738     \begingroup
739     \let\read@Macro@parg@end\read@Macro@parg@@end
740     \let\end@Macro@args\read@Macro@parg@end
741     \pargstyle{}%
742     {\ttfamily{}}%
743     \read@Macro@arg
744 }
745 \define@Macro@handler{}{}{%
746     \read@Macro@parg@end
747 }
748 \def\read@Macro@parg@@end#1){%
749     #1%
750     {\ttfamily{}}%
751     \endgroup
752     \read@Macro@arg
753 }
```

`\read@Macro@aarg`

```
754 \def\read@Macro@aarg<{%
755     \begingroup
756     \let\read@Macro@aarg@end\read@Macro@aarg@@end
757     \let\end@Macro@args\read@Macro@aarg@end
758     \aargstyle{}%
759     {\ttfamily<}%
760     \read@Macro@arg
761 }
762 \define@Macro@handler{>}{%
763     \read@Macro@aarg@end
764 }
765 \def\read@Macro@aarg@@end#1>>{%
766     #1%
767     {\ttfamily>}%
768     \endgroup
769     \read@Macro@arg
770 }
```

`\read@Macro@angle`

```
771 \define@Macro@handler{<}<{%
772     \futurelet\@let@token\read@Macro@angle@
773 }
```

`\read@Macro@angle@`

```
774 \def\read@Macro@angle@{%  
775   \ifx\@let@token<%  
776     \expandafter\read@Macro@aarg  
777   \else  
778     \expandafter\read@Macro@meta  
779   \fi  
780 }
```

`\read@Macro@meta`

```
781 \def\read@Macro@meta#1>{%  
782   \meta{#1}\read@Macro@arg  
783 }
```

`\read@Macro@sarg`

```
784 \define@Macro@handler**{%  
785   \sarg\read@Macro@arg  
786 }
```

`\read@Macro@verb`

Sets up verbatim mode calls second macro.

```
787 \define@Macro@handler{'}'{%  
788   \begingroup  
789   \let\do\@makeother  
790   \dospecials  
791   \@noligs  
792   \@makeother\'%  
793   \obeyspaces  
794   \read@Macro@verb@  
795 }
```

`\read@Macro@verb@`

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

```
796 \begingroup  
797 \@makeother\'%  
798 \gdef\read@Macro@verb@#1' {%  
799   \endgroup  
800   \ifx\relax#1\relax
```

```

801     {\verbstyle{\string'}}}%
802   \else
803     {%
804       \frenchspacing
805       \@noligs\verbstyle{#1}}%
806   \fi
807   \read@Macro@arg
808 }
809 \endgroup

```

\read@Macro@cmds

Simply executes given code.

```

810 \define@Macro@handler!!#1!{%
811   #1\relax
812   \read@Macro@arg
813 }

```

\read@Macro@rmspace

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

```

814 \define@Macro@handler{\@sptoken} {%
815   \read@Macro@arg
816 }

```

\read@Macro@addtoken

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

```

817 \define@Macro@handler{~}#1{%
818   #1\read@Macro@arg
819 }
820 \AtBeginDocument{%
821   \define@Macro@handler{~}#1{%
822     #1\read@Macro@arg
823   }
824 }
825 \define@Macro@handler{\space}#1{%
826   #1\read@Macro@arg
827 }

```

3.6.6 Description Macros

For Macros

`\DescribeMacro`

```
828 \@ifundefined{DescribeMacro}{%
829   \PackageInfo{ydoc-desc}{Redefining \string\
      DescribeMacro}{}%
830 }
```

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`.

```
831 \def\DescribeMacro{%
832   \DescribeMacros
833   \let\DescribeMacros\y@egroup
834   \optionalon
835   \def\after@Macro@args{\endDescribeMacros}%
836   \begingroup\makeatletter
837   \Describe@Macro
838 }
```

`\DescribeScript`

```
839 \def\DescribeScript#1{%
840   \DescribeMacros
841   \let\DescribeMacros\y@egroup
842   \optionalon
843   \def\after@Macro@args{\endDescribeMacros}%
844   \hbox\y@bgroup
845   \texttt{#1}%
846   \ydoc@macrocatcodes
847   \macroargsstyle
848   \read@Macro@arg~%
849 }
```

`\DescribeKey`

```
850 \def\DescribeKey{%
851   \DescribeKeys
852   \let\DescribeKeys\y@egroup
853   \optionalon
854   \def\after@Macro@args{\endDescribeKeys}%
855   \begingroup\makeatletter
856   \Describe@Macro
857 }
```

\Describe@Macro

```
858 \def\Describe@Macro#1{%
859   \endgroup
860   \edef\name{\expandafter\@gobble\string#1}%
861   \global\@namedef{href@desc@\name}{}%
862   \immediate\write\@mainaux{%
863     \global\noexpand\@namedef{href@desc@\name}{}%
864   }%
865   \hbox\y@bgroup
866   \@ifundefined{href@impl@\name}{}{\hyperlink{impl:\./
      name}}}%
867   {%
868     \hbox{\vbox to 0pt{\vss\hbox{\raisebox{4ex}{\./
      hypertarget{desc:\name}}}}}%
869     \PrintMacroName{#1}}%
870   }%
871   \ydoc@macrocatcodes
872   \macroargsstyle
873   \read@Macro@arg
874 }
```

\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.

```
875 \newcommand*\MakeShortMacroArgs{%
876   \@ifstar
877     {\@MakeShortMacroArgs\Macro}%
878     {\@MakeShortMacroArgs\MacroArgs}%
879 }
880 \def\@MakeShortMacroArgs#1#2{%
881   \MakeShortVerb{#2}
882   \catcode'#2\active
883   \begingroup
884   \catcode'\~\active
885   \lccode'\~'#2\relax
886   \lowercase{\endgroup\gdef~{\bgroup\let~\egroup#1}}%
887 }
```

\DeleteShortMacroArgs

```
888 \newcommand*\DeleteShortMacroArgs[1]{%
889   \DeleteShortVerb{#1}%
890 }
```

`\Macro`

Simply uses the two macros below.

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

`\@Macro`

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

```
892 \def\@Macro{%
893   \begingroup\makeatletter
894   \Describe@Macro
895 }

896 \define@Macro@handler\AlsoMacro{}
897 \define@Macro@handler\DescribeMacro{}
898 \define@Macro@handler\DescribeKey{}
```

`\MacroArgs`

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

```
899 \newcommand*\MacroArgs{%
900   \begingroup
901   \def\end@Macro@args{\endgroup\xspace}%
902   \ydoc@macrocatcodes
903   \macroargsstyle
904   %< *DEBUG>
905   %\typeout{}%
906   %\typeout{DEBUG: Start MacroArgs}%
907   %< /DEBUG>
908   \read@Macro@arg
909 }
910 \RequirePackage{xspace}
```

`\DescribeMacros`

```
911 \def\DescribeMacros{%
912   \begingroup
913   \let\Macro\@Macro
914   \parindent=0pt\relax
915   \setbox\descbox\vbox\y@bgroup
916 }
```

\endDescribeMacros

```
917 \def\endDescribeMacros{%  
918   \y@egroup  
919   \PrintMacros  
920   \endgroup  
921 }
```

\DescribeKeys

```
922 \def\DescribeKeys{%  
923   \begingroup  
924   \let\PrintMacroName\PrintKeyName  
925   \let\Key\@Macro  
926   \parindent=0pt\relax  
927   \setbox\descbox\vbox\y@bgroup  
928 }
```

\endDescribeKeys

```
929 \def\endDescribeKeys{%  
930   \y@egroup  
931   \PrintKeys  
932   \endgroup  
933 }  
934 \def\PrintKeys{\PrintMacros}
```

\DescribeMacroTabcolsep

```
935 \def\DescribeMacroTabcolsep{\tabcolsep}
```

\DescribeMacroTab

```
936 \def\DescribeMacroTab{%  
937   \DescribeMacros  
938   \hbox\y@bgroup  
939   \tabcolsep=\DescribeMacroTabcolsep\relax  
940   \DescribeMacroTab@  
941 }  
942 \def\DescribeMacroTab@#1{\tabular{@{}}#1@{}}}
```


`\endDescribeMacrosTab`

```

943 \def\endDescribeMacrosTab{%
944     \endtabular\y@egroup
945     \endDescribeMacros
946 }

```

For Lengths

`\DescribeLength`

```

947 \newcommand*\DescribeLength{%
948     \begingroup
949     \let\DescribeLength\Describe@Length
950     \setbox\descbox\hbox\y@bgroup
951     \tabular{@{}l@{\hspace{2em}}l@{}}%
952     \Describe@Length
953 }

```

`\Describe@Length`

```

954 \newcommand*\Describe@Length[2]{%
955     \PrintLengthName{#1}&
956     (Default: {\macroargsstyle#2\unskip})%
957     \@ifnextchar\DescribeLength
958     {\}%
959     {%
960         \endtabular
961         \y@egroup
962         \PrintLength
963         \endgroup
964     }%
965 }

```

For Environments

`\DescribeEnv`

```

966 \@ifundefined{DescribeEnv}{}{%
967     \PackageInfo{ydoc-desc}{Redefining \string\
968         DescribeEnv}{}%
969 }
969 \let\DescribeEnv\relax

```

```

970 \newcommand*\DescribeEnv[2][]{%
971   \begingroup
972   \def\DescribeEnv@name{#2}%
973   \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.

```

974   \ifx\@currenvir\DescribeEnv@string
975     \def\after@Macro@args{%
976       \let\after@Macro@args\empty
977       \setbox\@tempboxa\hbox\y@bgroup
978       \@ifnextchar\end{}}%
979       {\DescribeEnv@newline}%
980     #1%
981   }%

```

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

```

982   \else
983     \ifx\relax#1\relax
984       \def\after@Macro@args{%
985         \y@bgroup
986         \endDescribeEnv
987       }%
988     \else
989       \def\after@Macro@args{%
990         \setbox\@tempboxa\hbox\y@bgroup
991         \DescribeEnv@newline\MacroArgs#1%
992         \endDescribeEnv
993       }%
994     \fi
995   \fi

```

Start \vbox and adds first line.

```

996   \setbox\descbox\vbox\y@bgroup
997   \envcodestyle
998   \let\PrintEnv\PrintSubEnv
999   \hbox\y@bgroup
1000   \PrintEnvName{\begin}{\DescribeEnv@name}%
1001   \ydoc@macrocatcodes
1002   \macroargsstyle
1003   \read@Macro@arg
1004 }

```

\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.

```

1005 \newcommand*\DescribeEnv@newline[1][0pt]{%
1006   \strut\y@egroup
1007   {\vskip#1}%
1008   \hbox\y@bgroup\strut
1009   \hspace*{\descsep}%
1010   \ignorespaces
1011 }%

```

`\DescribeEnv@string`

Holds the environment name for comparison.

```

1012 \def\DescribeEnv@string{\DescribeEnv}

```

`\descbox`

Save box to store description content.

```

1013 \newbox\descbox

```

`\endDescribeEnv`

```

1014 \def\endDescribeEnv{%
1015   \y@egroup
1016   \begingroup
1017   \setbox\@tempboxa\lastbox
1018   \ifcase0%
1019     \ifdim\wd\@tempboxa>\descsep1\fi
1020     \ifdim\ht\@tempboxa>\ht\strutbox1\fi
1021     \ifdim\dp\@tempboxa>\dp\strutbox1\fi
1022   \else
1023     \box\@tempboxa
1024   \fi
1025   \endgroup
1026   \hbox\y@bgroup
1027     \PrintEnvName{\end}{\DescribeEnv@name}
1028   \y@egroup
1029   \y@egroup
1030   \PrintEnv
1031   \endgroup
1032 }

```

3.6.7 Print Macros

`\PrintMacroName`

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

```
1033 \def\PrintMacroName#1{%  
1034     {\macrodescstyle{\strut  
1035         \texttt{\char92}%  
1036         \escapechar\m@ne  
1037         \string#1\strut}}%  
1038 }
```

`\PrintKeyName`

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

```
1039 \def\PrintKeyName#1{%  
1040     {\keydescstyle{\strut  
1041         #1\strut}}%  
1042 }
```

`\PrintLengthName`

Formats length register name.

```
1043 \let\PrintLengthName\PrintMacroName
```

`\PrintEnvName`

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

```
1044 \def\PrintEnvName#1#2{%  
1045     \strut  
1046     \string#1\braceleft  
1047     {\macrodescstyle#2\strut}%  
1048     \braceright  
1049 }
```

`\PrintMacros`

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

```
1050 \def\PrintMacros{%  
1051     \par\vspace\beforedescskip  
1052     \begingroup  
1053     \sbox\@tempboxa{\descframe{\usebox{\descbox}}}%  
1054     \Needspace*{\dimexpr\ht\@tempboxa+3\baselineskip\/  
        relax}%
```

```

1055 \par\noindent
1056 \ifdim\wd\@tempboxa>\dimexpr\linewidth-2\descindent/
      \relax
1057 \makebox[\linewidth][c]{\usebox\@tempboxa}%
1058 \else
1059 \hspace*{\descindent}%
1060 \usebox\@tempboxa
1061 \fi
1062 \endgroup
1063 \par
1064 \vspace\afterdescskip
1065 \par\noindent
1066 }
1067 \def\descframe#1{%
1068 \fbox{\hspace*{\descsep}#1\hspace*{\descsep}}%
1069 }

```

\PrintLength

Prints lengths registers described using one or multiple \DescribeLength.

```

1070 \let\PrintLength\PrintMacros

```

\PrintEnv

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

```

1071 \let\PrintEnv\PrintMacros

```

\PrintSubEnv

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

```

1072 \def\PrintSubEnv{%
1073 \hbox{\hbox{\usebox{\descbox}}}%
1074 }

```

3.6.8 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.

```

1075 {%
1076 \@firstofone{%
1077   \catcode'\=12
1078   \gdef\bslash
1079   }{\}
1080   }%}

```

`\percent`

Defines an expandable percent character with catcode 12: ‘%₁₂’.

```

1081 \begingroup
1082 \catcode'\%=12
1083 \gdef\percent{%}
1084 \endgroup

```

`\braceleft`

`\braceright`

Defines expandable left and right braces with catcode 12: ‘{₁₂’ ‘}₁₂’.

```

1085 \begingroup
1086 \catcode'\<=1
1087 \catcode'\>=2
1088 \catcode'\{=12
1089 \catcode'\}=12
1090 \gdef\braceleft <{>
1091 \gdef\braceright <}>
1092 \endgroup

```

3.6.9 Other Macros

`\y@bgroup`

`\y@egroup`

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

```

1093 \def\y@bgroup{\bgroup\color@setgroup}
1094 \def\y@egroup{\color@endgroup\egroup}

```

`\codeline`

```
1095 \newcommand*{\codeline}[1][c]{%
1096   \codelinebefore
1097   \hbox to \hsize\bgroup
1098   \ifx i#1\hspace*{\leftmargin}\else
1099     \ifx l#1\else\hss\fi
1100   \fi
1101   \let\xspace\relax
1102   \hbox\bgroup
1103   \aftergroup\codeline@end
1104   \aftergroup#1%
1105   \afterassignment\MacroArgs
1106   \let\@let@token=%
1107 }
1108 \def\codeline@end#1{%
1109   \ifx r#1\else\hss\fi
1110   \egroup
1111   \codelineafter
1112 }
1113 \newcommand*\codelinebefore{\par\smallskip\noindent}
1114 \newcommand*\codelineafter {\par\smallskip\noindent}
```

`codequote`

```
1115 \newenvironment{codequote}{%
1116   \def\{\{\newline\relax\MacroArgs}%
1117   \par\smallskip\bgroup\leftskip=\leftmargin\
1118     \rightskip=\rightmargin\noindent\MacroArgs}
1119   {\par\egroup\smallskip\noindent\
1120     ignorespacesafterend}
```

`macroquote`

```
1119 \newenvironment{macroquote}{%
1120   \def\{\{\newline\relax\Macro}%
1121   \par\smallskip\bgroup\leftskip=\leftmargin\
1122     \rightskip=\rightmargin\noindent\Macro}
1123   {\par\egroup\smallskip\noindent\
1124     ignorespacesafterend}
```

3.7 Include Code Examples

```
1123 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
1124 \ProvidesPackage{ydoc-expl}[%
1125 %<!DATE>
```

```

1126 %<!VERSION>
1127 %<*DRIVER>
1128     2011/08/11 develop
1129 %</DRIVER>
1130     ydoc package to insert live examples of LaTeX /
        code]

1131 \RequirePackage{listings}
1132 \lst@RequireAspects{writefile}
1133 \def\ydoc@exafile{\jobname.exa}

```

examplecode

```

1134 \lstdefinestyle{examplecode}{numbers=left,firstnumber/
        =1,numberstyle=\tiny\color{gray}\sffamily,/
        numbersep=5pt}%

```

exampleextract

```

1135 \lstdefinestyle{exampleextract}{gobble=4}%

1136 \newbox\examplecodebox
1137 \newbox\exampleresultbox

```

\BoxExample

```

1138 \def\BoxExample{%
1139     \setbox\examplecodebox\hbox{\color@setgroup
1140         \lstinputlisting[style=examplecode,style=/
            thisexampleprint]%
1141         {\ydoc@exafile}}%
1142     \unskip\color@endgroup}%
1143     \setbox\exampleresultbox\hbox{\color@setgroup
1144         \@input\ydoc@exafile\relax
1145         \unskip\color@endgroup}%
1146 }

```

\PrintExample

```

1147 %<*DISABLED>
1148 \RequirePackage{showexpl}
1149 \def\PrintExample{%
1150     \begingroup
1151     \lstset{style=examplecode}%
1152     \MakePercentComment

```



```

1153 \LTxinputExample[varwidth]{\ydoc@exafile}%
1154 \endgroup
1155 }
1156 %</DISABLED>

```

\PrintExample

```

1157 \def\PrintExample{%
1158   \begingroup
1159   \BoxExample
1160   \@tempdima=\textwidth
1161   \advance\@tempdima by -\wd\examplecodebox\relax
1162   \advance\@tempdima by -\wd\exampleresultbox\relax
1163   \advance\@tempdima by -15pt\relax
1164   \ifdim\@tempdima>\bigskipamount
1165     \hbox to \textwidth{%
1166       \null\hss
1167       \minipage[c]{\wd\exampleresultbox}\fbox{\usebox\
1168         exampleresultbox}\endminipage
1169       \hfill\hfill\hskip\bigskipamount\hskip15pt\hfill\
1170       \hfill
1171       \minipage[c]{\wd\examplecodebox}\usebox\
1172         examplecodebox\endminipage
1173       \hss\null
1174     }%
1175   \else
1176     \vbox{%
1177       \centerline{\fbox{\usebox\exampleresultbox}}%
1178       \vspace{\bigskipamount}%
1179       \centerline{\usebox\examplecodebox}%
1180     }%
1181   \fi
1182 \endgroup
1183 }

```

examplecode

```

1181 \lstnewenvironment{examplecode}[1][\{%
1182   \lstdefinestyle{thisexampleprint}{#1}%
1183   \lstset{style=exampleextract,#1}%
1184   \setbox\@tempboxa\hbox\bgroup
1185   \lst@BeginWriteFile{\ydoc@exafile}%
1186 }
1187 {%
1188   \lst@EndWriteFile
1189   \egroup
1190 \begingroup

```

```

1191 \MakePercentComment
1192 \catcode'\^M=5\relax
1193 \PrintExample
1194 \endgroup
1195 }

```

```

1196 \RequirePackage{float}

```

example

```

1197 \floatstyle{plain}
1198 \newfloat{example}{tbhp}{loe}
1199 \floatname{example}{\examplename}
1200 \def\examplename{Example}

```

exampletable

```

1201 \newenvironment{exampletable}{%
1202 \floatstyle{plaintop}%
1203 \restylefloat{example}%
1204 \example
1205 }{\endexample}

```