

The ydoc Class and Packages

Martin Scharrer
martin@scharrer.de

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

VC: https://bitbucket.org/martin_scharrer/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: “%`\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 \TeX 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 `'l'` 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	<code><text></code>	<code><text></code>	<code>\meta{<text>}</code>
Mandatory Argument	<code>{args}</code>	<code>{args}</code>	
—, with meta text	<code><text></code>	<code>{<text>}</code>	<code>\marg{<text>}</code>
Optional Argument	<code>[args]</code>	<code>[args]</code>	
—, with meta text	<code><text></code>	<code>[<text>]</code>	<code>\oarg{<text>}</code>
Picture Argument	<code>(args)</code>	<code>(args)</code>	
—, with meta text	<code><text></code>	<code>(<text>)</code>	<code>\parg{<text>}</code>
Beamer Overlay Argument	<code><<args>></code>	<code><args></code>	
—, with meta text	<code><< <text> >></code>	<code><< <text> >></code>	<code>\aarg{<text>}</code>
Star	<code>*</code>	<code>*</code>	
Verbatim content	<code>'\$&^%_#&\$\'</code>	<code>\$&^%_#&\$\</code>	
—, produce ‘ char	<code>''</code>	<code>,</code>	
Insert any T _E X 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.

```
\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.

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

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

```
\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-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.

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

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.

```
\metastyle  \margstyle  
\oargstyle  \pargstyle  
\aargstyle  \sargstyle
```

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.

```
\descindent      (Default: -20pt)  
\beforedescskip  (Default: 12pt plus 4pt minus 4pt)  
\afterdescskip   (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]
```

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

```
9 \PassOptionsToClass{a4paper}{article}
10 \DeclareOption*{\expandafter\PassOptionsToClass\
    expandafter{\CurrentOption}{article}}
11 \ProcessOptions\relax
12 \LoadClass{article}
13 \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 \RequirePackage{svn-prov}[2010/04/03]
23 \RequirePackage{ydoc-code}
24 \RequirePackage{ydoc-expl}
25 \RequirePackage{ydoc-desc}
26 \RequirePackage{ydoc-doc}
27
28 \RequirePackage{newverbs}
29 \MakeSpecialShortVerb{\qverb}{\"}
30 \AtBeginDocument{\catcode'\^A=14\relax}
31
32 \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
43 \usepackage[T1]{fontenc}
44 \IfFileExists{fourier.sty}{%
45     \usepackage{fourier}
46 }{}
47
48     Use 'lmodern' only for the 'tt' font if fourier is installed.
49
50 \IfFileExists{lmodern.sty}{
51     \IfFileExists{fourier.sty}{
52         \renewcommand{\ttdefault}{lmtt}
53     }{
54         \usepackage{lmodern}
55     }
56 }{}
57
58 \urlstyle{sf}
59
60     Use micro-typesetting if pdftex is used:
61
62 \usepackage{ifpdf}
63 \ifpdf
64 \usepackage{microtype}
65 \fi
66
67 \usepackage{array}
68 \usepackage{booktabs}
69 \usepackage{multicol}
70 \usepackage{xcolor}
71 \usepackage{listings}
72 \usepackage{booktabs}
73 \usepackage{hyperref}
74
75 \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\themacrocode{#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\themacrocode
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 }

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

```

`\ydoclistingssettings`

```

181 \def\ydoclistingssettings{%
182     \lstset{style=ydoccode}%
183 }

```

`\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{\##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   }{\}
275 }%}

```


3.5 Provide doc macros

```
276 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
277 \ProvidesPackage{ydoc-doc}[%
278 %<!DATE>
279 %<!VERSION>
280 %<*DRIVER>
281     2099/01/01 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 }
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\bundle{\def\@bundle}
424 \let\@bundle\@empty
425
426
427 \def\ctanlocation{\def\@ctanlocation##1}
428 \ctanlocation{http://www.ctan.org/pkg/#1}
429
430 \date{Version \fileversion\space -- \filedate}
431
432 \def\@homepage{%
433     \begingroup
434     \edef\@tempa{%
435         \endgroup
436         CTAN:
437         \noexpand\url
438         {\@ctanlocation{\ifx\@bundle\@empty\@package\
else\@bundle\fi}}}%
439     }%
440     \@tempa
441 }
442
443 \let\@repository\@empty
444 \protected\def\repository{\urldef\@repository\url}
445 \protected\def\homepage{\urldef\@homepage\url}
446 \protected\def\email{\hyper@normalise\email@}
447 \def\email@#1{\def\@plainemail{#1}\def\@email{\
hyper@linkurl{\Hurl{#1}}{mailto:#1}}}
448
449 \let\@email\empty
450
451 \let\@plainemail\empty

```

```

452 \title{The \texorpdfstring{\pkgtitle{\@package}}{\@package} Package}
453 \def\@bundlesubtitle{Part of the \texorpdfstring{\pkgtitle{\@bundle}}{\@bundle} bundle}
454
455 \protected\def\pkgtitle#1{%
456     \texorpdfstring{\textsf{#1}}{#1}%
457 }
458
459
460
461 \def\@maketitle{%
462     \newpage
463     \null\vskip 2em
464     \begin{center}%
465         \let\footnote\thanks
466         {\LARGE \@title \par }\vskip 1.5em%
467         \ifx\@bundle\@empty\else
468             {\large \@bundlesubtitle \par }\vskip 1.5em%
469         \fi
470         {\large \lineskip .5em%
471         \begin{tabular}[t]{c}%
472             \@author
473         \end{tabular}}%
474         \par}%
475         \ifx\@plainemail\@empty\else
476             {\large \lineskip .5em%
477             \begin{tabular}[t]{c}%
478                 \@email
479             \end{tabular}}%
480             \par}%
481         \fi
482         \vskip 1em
483         {\large \lineskip .5em%
484         \begin{tabular}[t]{c}%
485             \@homepage
486         \end{tabular}}%
487         \par}%
488         \vskip 1em
489         \ifx\@repository\@empty\else
490             {\large \lineskip .5em%
491             \begin{tabular}[t]{c}%
492                 VC: \@repository
493             \end{tabular}}%
494             \par}%
495         \fi
496         \vskip 1em
497         {\large \@date }%
498     \end{center}%
499     \par\vskip 1.5em

```

```

500     \aftergroup\ydocpdfsettings
501 }
502
503 \ifpdf
504 \def\ydocpdfsettings{%
505     \hypersetup{%
506         pdfauthor    = {\@author\space<\@plainemail>},
507         pdftitle     = {\@title},
508         pdfsubject   = {Documentation of LaTeX package/
509             \@package},
510         pdfkeywords  = {\@package, LaTeX, TeX}
511     }%
512 }
513 \else
514 \let\ydocpdfsettings\empty
515 \fi
516
517 \let\orig@maketitle\maketitle
518 \def\maketitle{%
519     \ydocpdfsettings
520     \orig@maketitle
521     \let\orig@maketitle\relax
522 }

```

3.6 Description Macros and Environments

```

522 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
523 \ProvidesPackage{ydoc-desc}[%
524 %<!DATE>
525 %<!VERSION>
526 %<*DRIVER>
527     2099/01/01 develop
528 %</DRIVER>
529     ydoc package to describe macros, environments, /
530     options etc.]
531
532 \IfFileExists{needspace.sty}{%
533     \RequirePackage{needspace}
534 }{%
535     \def\Needspace{\@ifstar\@gobble\@gobble}
536 }

```

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

```

535 \RequirePackage{shortvrb}

```

The etoolbox package is used mainly for `\newrobustcmd`.

```

536 \RequirePackage{etoolbox}

```


3.6.1 Color and style definitions

```
537 \RequirePackage{xcolor}

    Define special no-op 'none' color which does not change the color. This is not yet
    tested and may break output files, but seems to work fine with PDF

538 \expandafter\def\csname\string\color@none\endcsname{%
539     \xcolor@ {}{}{}{}}
540 }

541 \definecolor{macrodesc}{rgb}{0,0.2,0.6}
542 \definecolor{keydesc}{rgb}{0,0.4,0.9}
543 \definecolor{macroimpl}{rgb}{0,0.1,0.3}
544 \definecolor{meta}{rgb}{0,0.25,0.75}
545 \definecolor{scriptcolor}{rgb}{0.2,0.6,0.2}
546 \definecolor{optioncolor}{rgb}{0.3,0.2,0}
547 \colorlet{optional}{black!65!white}
548 \colorlet{metaoptional}{optional!50!meta}
549 \providecolor{urlcolor}{named}{blue}
550 \providecolor{linkcolor}{named}{blue}
551 \providecolor{filecolor}{named}{blue}
552 \providecolor{citecolor}{named}{blue}
553 \providecolor{anchorcolor}{named}{blue}
554 \providecolor{menucolor}{named}{blue}
555 \providecolor{runcolor}{named}{blue}

557 \RequirePackage{hyperref}
558 \hypersetup{%
559     colorlinks=true,
560     pdfborder=0 0 0,
561     pdfborderstyle={},
562     urlcolor=urlcolor,
563     linkcolor=linkcolor,
564     filecolor=filecolor,
565     citecolor=citecolor,
566     anchorcolor=anchorcolor,
567     menucolor=menucolor,
568     runcolor=runcolor,
569 }
```

3.6.2 Text Formatting Macros

`\meta`

Prints *[meta text](#)*.

```
571 \newrobustcmd*\meta[1]{%
572     {\metastyle{%
573     \ensuremath\langle
```

```

574     #1\/%
575     \ensuremath\angle
576     }}%
577 }

```

`\marg`

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

```

578 \newrobustcmd*\marg}[1]{%
579   {\margstyle{%
580     {\ttfamily\braceleft}%
581     \meta{#1}%
582     {\ttfamily\braceright}%
583   }}%
584 }

```

`\oarg`

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

```

585 \newrobustcmd*\oarg}[1]{%
586   {\oargstyle{%
587     {\ttfamily[]}%
588     \meta{#1}%
589     {\ttfamily[]}%
590   }}%
591 }

```

`\parg`

Sets style and adds parentheses.

```

592 \newrobustcmd*\parg}[1]{%
593   {\pargstyle{%
594     {\ttfamily{}}%
595     \meta{#1}%
596     {\ttfamily})}%
597   }}%
598 }

```

`\aarg`

Sets style and adds angles.

```

599 \newrobustcmd*{\aarg}[1]{%
600   {\aargstyle{%
601     {\ttfamily<}%
602     \meta{#1}%
603     {\ttfamily>}%
604   }}%
605 }

```

`\sarg`

Prints star with given style.

```

606 \newrobustcmd*{\sarg}{\sargstyle{*}}

```

`\pkg`

`\cls`

`\lib`

`\env`

`\opt`

`\file`

```

607 \newrobustcmd*\pkg[1]{\pkgstyle{#1}}
608 \newrobustcmd*\cls[1]{\clsstyle{#1}}
609 \newrobustcmd*\lib[1]{\libstyle{#1}}
610 \newrobustcmd*\env[1]{\envstyle{#1}}
611
612 \newrobustcmd*\opt{\@ifstar\ys@opt\y@opt}
613 \def\y@opt#1{\optstyle{#1}}
614 \def\ys@opt#1{\optstyle{#1}}\optpar{#1}}
615 \newrobustcmd*\optpar[1]{\marginpar{\hbox to \
        marginparwidth{\hss\y@opt{#1}}}}
616
617 \newrobustcmd*\file[1]{\filestyle{#1}}
618 \newcommand*\pkgstyle[1]{\texttt{\textcolor{pkg/
        }{#1}}}

```

```

619 \newcommand*\clsstyle [1]{\texttt{\textcolor{cls/
    }{#1}}}
620 \newcommand*\libstyle [1]{\texttt{\textcolor{lib/
    }{#1}}}
621 \newcommand*\envstyle [1]{\texttt{\textcolor{env/
    }{#1}}}
622 \newcommand*\optstyle [1]{\textsf{\textcolor{opt/
    }{#1}}}
623 \newcommand*\filestyle [1]{\texttt{\textcolor{file/
    }{#1}}}
624 \colorlet{cls}{none}
625 \colorlet{lib}{none}
626 \colorlet{env}{none}
627 \colorlet{file}{none}
628 \colorlet{pkg}{none}
629 \definecolor{opt}{rgb}{0.5,0.16666,0}

```

`\cs`

`\cmd`

```

630 \newrobustcmd*\cs [1]{\texttt{\textbackslash #1}}
631 \newrobustcmd*\cmd [1]{\texttt{\escapechar=92\string/
    #1}}

```

`\Key`

```

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

```

3.6.3 Text Formatting Styles

`\macrodescstyle`

Style of described macro names.

```

633 \def\macrodescstyle{\ttfamily\bfseries\color{/
    macrodesc}}

```

`\macrodescstyle`

Style of described macro names.

```

634 \def\keydescstyle{\ttfamily\bfseries\color{keydesc}}

```

`\macroargsstyle`

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

```
635 \def\macroargsstyle{\ttfamily}
```

`\envcodestyle`

Default style for code body content in described environments.

```
636 \def\envcodestyle{\ttfamily}
```

`\verbstyle`

Style for verbatim text inside macro argument list.

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

`\metastyle`

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

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

`\margstyle`

Style for `\marg`.

```
639 \def\margstyle{}
```

`\Optional`

`\optional`

`\optionalstyle`

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

`\optionalon`

`\optionaloff`

```
642 \def\optionalon{\protected\def\optional{\/  
    optionalstyle}}  
643 \def\optionaloff{\let\optional\relax}  
644 \optionalon
```

`\oargstyle`

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

```
645 \def\oargstyle{\optional}
```

`\pargstyle`

Style for `\parg`.

```
646 \def\pargstyle{}
```

`\aargstyle`

Style for `\aarg`.

```
647 \def\aargstyle{}
```

`\sargstyle`

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

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

3.6.4 Dimension Registers

`\descindent`

```
649 \newdimen\descindent  
650 \descindent=-\parindent
```

`\beforedescskip`

```
651 \newdimen\beforedescskip
652 \beforedescskip=\bigskipamount
```

`\afterdescskip`

```
653 \newdimen\afterdescskip
654 \afterdescskip=\medskipamount
```

`\descsep`

Set to 1em in tt font.

```
655 \newdimen\descsep
656 \begingroup
657 \ttfamily
658 \global\descsep=1em\relax
659 \endgroup
```

3.6.5 Macro Argument Reading Mechanism

`\read@Macro@arg`

Reads next token and calls second macro.

```
660 \def\read@Macro@arg{%
661   \futurelet\@let@token\handle@Macro@arg
662 }
```

`\AlsoMacro`

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

```
663 \newcommand*\AlsoMacro{%
664   \begingroup\makeatletter
665   \AlsoMacro@
666 }
667 \def\AlsoMacro@#1{%
668   \endgroup
669   %<*DEBUG>
670   %\typeout{DEBUG: Macro: \string#1}%
671   %</DEBUG>
672   \PrintMacroName{#1}%
673   \read@Macro@arg
674 }
```

`\ydoc@short@AlsoMacro`

Makes & an alias for `\AlsoMacro`.

```
675 \begingroup
676 \catcode '\|\active
677 \gdef\ydoc@short@AlsoMacro{%
678   \catcode '\|\active
679   \let|\AlsoMacro
680 }
681 \endgroup
```

`\ydoc@macrocatcodes`

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

```
682 \def\ydoc@macrocatcodes{%
683   \ydoc@short@AlsoMacro
684   \@makeother\'%
685   \@makeother\!%
686   \@makeother\[%
687   \@makeother\]%
688   \@makeother\(%
689   \@makeother\)%
690 }
```

`\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.

```
691 \def\handle@Macro@arg{%
692   \expandafter\let\expandafter\handler\csname /
693     handle@Macro@token\meaning\@let@token\endcsname
694   \ifx\handler\relax
695     \def\handler{\ifhmode\unskip\fi\end@Macro@args}%
696   %<*DEBUG>
697   % \typeout{DEBUG: Stopped at: \expandafter\meaning\ /
698     csname @let@token\endcsname}%
699   % \typeout{}%
700   %\else
701   %\expandafter\ifx\csname @let@token\endcsname\ /
702     AlsoMacro
703   % \typeout{DEBUG: TOKEN: \string\AlsoMacro}%
704   %\else
705   % \typeout{DEBUG: TOKEN: \expandafter\meaning\ /
706     csname @let@token\endcsname}%
707   %\fi
708 %</DEBUG>
```



```

705     \fi
706     \handler
707 }
708 \def\define@Macro@handler{%
709     \begingroup
710     \ydoc@macroatcodes
711     \define@Macro@handler@
712 }
713 \def\define@Macro@handler@#1{%
714     \endgroup
715     \@namedef{handle@Macro@token@\meaning#1}%
716 }

```

`\end@Macro@args`

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

```

717 \def\end@Macro@args{%
718     \y@egroup
719     \after@Macro@args
720 }

```

`\after@Macro@args`

Hook to add additional commands in certain situations.

```

721 \def\after@Macro@args{%
722 }

```

Macro argument reading macros

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

`\read@Macro@marg`

```

723 \define@Macro@handler{\bgroup}{%
724     \begingroup
725     \afterassignment\read@Macro@marg@
726     \let\@let@token=%
727 }
728 \def\read@Macro@marg@{%
729     \bgroup
730     \margstyle{}%
731     \let\end@Macro@args\empty%
732     {\ttfamily\braceleft}%
733     \aftergroup\read@Macro@marg@@
734     \read@Macro@marg

```

```

735 }
736 \def\read@Macro@marg@@{%
737     {\ttfamily\braceright}%
738     \endgroup
739     \read@Macro@arg
740 }

```

\read@Macro@oarg

```

741 \define@Macro@handler{[]}{%
742     \begingroup
743     \let\read@Macro@oarg@end\read@Macro@oarg@@end
744     \let\end@Macro@args\read@Macro@oarg@end
745     \oargstyle{}%
746     {\ttfamily []}%
747     \read@Macro@arg
748 }
749 \define@Macro@handler{[]}{%
750     \read@Macro@oarg@end
751 }
752 \def\read@Macro@oarg@@end#1{%
753     #1%
754     {\ttfamily []}%
755     \endgroup
756     \read@Macro@arg
757 }
758 \def\read@Macro@oarg@end{\end@Macro@args}
759 \let\read@Macro@aarg@end\read@Macro@oarg@end
760 \let\read@Macro@parg@end\read@Macro@oarg@end

```

\read@Macro@parg

```

761 \define@Macro@handler{()}{%
762     \begingroup
763     \let\read@Macro@parg@end\read@Macro@parg@@end
764     \let\end@Macro@args\read@Macro@parg@end
765     \pargstyle{}%
766     {\ttfamily ()}%
767     \read@Macro@arg
768 }
769 \define@Macro@handler{)}{%
770     \read@Macro@parg@end
771 }
772 \def\read@Macro@parg@@end#1{%
773     #1%
774     {\ttfamily ()}%
775     \endgroup

```

```

776     \read@Macro@arg
777 }

```

\read@Macro@aarg

```

778 \def\read@Macro@aarg<{%
779     \begingroup
780     \let\read@Macro@aarg@end\read@Macro@aarg@@end
781     \let\end@Macro@args\read@Macro@aarg@end
782     \aargstyle{}%
783     {\ttfamily<}%
784     \read@Macro@arg
785 }
786 \define@Macro@handler{>}{%
787     \read@Macro@aarg@end
788 }
789 \def\read@Macro@aarg@@end#1>>{%
790     #1%
791     {\ttfamily>}%
792     \endgroup
793     \read@Macro@arg
794 }

```

\read@Macro@angle

```

795 \define@Macro@handler{<<}{%
796     \futurelet\@let@token\read@Macro@angle@
797 }

```

\read@Macro@angle@

```

798 \def\read@Macro@angle@{%
799     \ifx\@let@token<%
800     \expandafter\read@Macro@aarg
801     \else
802     \expandafter\read@Macro@meta
803     \fi
804 }

```

\read@Macro@meta

```

805 \def\read@Macro@meta#1>{%
806     \meta{#1}\read@Macro@arg
807 }

```

`\read@Macro@sarg`

```
808 \define@Macro@handler**{%
809   \sarg\read@Macro@arg
810 }
```

Allows '=' to be used directly without switching to verbatim mode. This is especially useful for keys.

```
811 \define@Macro@handler{=}{%
812   =\read@Macro@arg
813 }
```

`\read@Macro@verb`

Sets up verbatim mode calls second macro.

```
814 \define@Macro@handler{'}' {%
815   \begingroup
816   \let\do\@makeother
817   \dospecials
818   \@noligs
819   \@makeother\'%
820   \obeyspaces
821   \read@Macro@verb@
822 }
```

`\read@Macro@verb@`

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

```
823 \begingroup
824 \@makeother\'%
825 \gdef\read@Macro@verb@#1' {%
826   \endgroup
827   \ifx\relax#1\relax
828     {\verbstyle{\string'}}%
829   \else
830     {%
831       \frenchspacing
832       \@noligs\verbstyle{#1}}%
833   \fi
834   \read@Macro@arg
835 }
836 \endgroup
```

`\read@Macro@cmds`

Simply executes given code.

```
837 \define@Macro@handler!!#1!{%  
838   #1\relax  
839   \read@Macro@arg  
840 }
```

`\read@Macro@rmspace`

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

```
841 \define@Macro@handler{\@sptoken} {%  
842   \read@Macro@arg  
843 }
```

`\read@Macro@addtoken`

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

```
844 \define@Macro@handler{~}#1{%  
845   #1\read@Macro@arg  
846 }  
847 \AtBeginDocument{%  
848   \define@Macro@handler{~}#1{%  
849     #1\read@Macro@arg  
850   }  
851 }  
852 \define@Macro@handler{\space}#1{%  
853   #1\read@Macro@arg  
854 }
```

3.6.6 Description Macros

For Macros

`\DescribeMacro`

```
855 \@ifundefined{DescribeMacro}{}{%  
856   \PackageInfo{ydoc-desc}{Redefining \string\  
     DescribeMacro}{}%  
857 }
```

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`.

```

858 \def\DescribeMacro{%
859   \DescribeMacros
860   \let\DescribeMacros\y@egroup
861   \optionalon
862   \def\after@Macro@args{\endDescribeMacros}%
863   \begingroup\makeatletter
864   \Describe@Macro
865 }

```

\DescribeScript

```

866 \def\DescribeScript#1{%
867   \DescribeMacros
868   \let\DescribeMacros\y@egroup
869   \optionalon
870   \def\after@Macro@args{\endDescribeMacros}%
871   \hbox\y@bgroup
872   \texttt{#1}%
873   \ydoc@macrocatcodes
874   \macroargsstyle
875   \read@Macro@arg~%
876 }

```

\DescribeKey

```

877 \def\DescribeKey{%
878   \DescribeKeys
879   \let\DescribeKeys\y@egroup
880   \optionalon
881   \def\after@Macro@args{\endDescribeKeys}%
882   \begingroup\makeatletter
883   \Describe@Macro
884 }

```

\Describe@Macro

```

885 \def\Describe@Macro#1{%
886   \endgroup
887   \edef\name{\expandafter\@gobble\string#1}%
888   \global\@namedef{href@desc@name}{}%
889   \immediate\write\@mainaux{%
890     \global\noexpand\@namedef{href@desc@name}{}%
891   }%
892   \hbox\y@bgroup

```

```

893 \@ifundefined{href@impl@\name}{}{\hyperlink{impl:\/
      name}}%
894 {%
895 \hbox{\vbox to 0pt{\vss\hbox{\raisebox{4ex}{\/
      hypertarget{desc:\name}}}}}%
896 \PrintMacroName{#1}}%
897 }%
898 \ydoc@macrocatcodes
899 \macroargsstyle
900 \read@Macro@arg
901 }

```

\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.

```

902 \newcommand*\MakeShortMacroArgs{%
903 \@ifstar
904   {\@MakeShortMacroArgs\Macro}%
905   {\@MakeShortMacroArgs\MacroArgs}%
906 }
907 \def\@MakeShortMacroArgs#1#2{%
908 \MakeShortVerb{#2}
909 \catcode'#2\active
910 \begingroup
911 \catcode'\~\active
912 \lccode'\~'#2\relax
913 \lowercase{\endgroup\gdef~{\bgroup\let~\egroup#1}}%
914 }

```

\DeleteShortMacroArgs

```

915 \newcommand*\DeleteShortMacroArgs[1]{%
916 \DeleteShortVerb{#1}%
917 }

```

\Macro

Simply uses the two macros below.

```

918 \newcommand*\Macro{\MacroArgs\AlsoMacro}

```

`\@Macro`

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

```
919 \def \@Macro {%
920   \begingroup \makeatletter
921   \Describe@Macro
922 }

923 \define@Macro@handler \AlsoMacro {}
924 \define@Macro@handler \DescribeMacro {}
925 \define@Macro@handler \DescribeKey {}
926 \define@Macro@handler \DescribeScript {}
```

`\MacroArgs`

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

```
927 \newcommand* \MacroArgs {%
928   \begingroup
929   \def \end@Macro@args {\endgroup \xspace}%
930   \ydoc@macrocatcodes
931   \macroargsstyle
932   %< *DEBUG >
933   %\typeout {}%
934   %\typeout {DEBUG: Start MacroArgs}%
935   %< /DEBUG >
936   \read@Macro@arg
937 }
938 \RequirePackage {xspace}
```

`\DescribeMacros`

```
939 \def \DescribeMacros {%
940   \begingroup
941   \let \Macro \@Macro
942   \parindent =0pt \relax
943   \setbox \descbox \vbox \y@bgroup
944 }
```

`\endDescribeMacros`

```
945 \def \endDescribeMacros {%
946   \y@egroup
947   \PrintMacros
948   \endgroup
949 }
```


`\DescribeKeys`

```
950 \def\DescribeKeys{%
951   \begingroup
952   \let\PrintMacroName\PrintKeyName
953   \let\Key\@Macro
954   \parindent=0pt\relax
955   \setbox\descbox\vbox\y@bgroup
956 }
```

`\endDescribeKeys`

```
957 \def\endDescribeKeys{%
958   \y@egroup
959   \PrintKeys
960   \endgroup
961 }
962 \def\PrintKeys{\PrintMacros}
```

`\DescribeMacrosTabcolsep`

```
963 \def\DescribeMacrosTabcolsep{\tabcolsep}
```

`\DescribeMacrosTab`

```
964 \def\DescribeMacrosTab{%
965   \DescribeMacros
966   \hbox\y@bgroup
967   \tabcolsep=\DescribeMacrosTabcolsep\relax
968   \DescribeMacrosTab@
969 }
970 \def\DescribeMacrosTab@#1{\tabular{@{}#1@{}}}
```

`\endDescribeMacrosTab`

```
971 \def\endDescribeMacrosTab{%
972   \endtabular\y@egroup
973   \endDescribeMacros
974 }
```

For Lengths

`\DescribeLength`

```
975 \newcommand*\DescribeLength{%
976   \begingroup
977   \let\DescribeLength\Describe@Length
978   \setbox\descbox\hbox\y@bgroup
979   \tabular{@{}l@{\hspace{2em}}l@{}}%
980   \Describe@Length
981 }
```

`\Describe@Length`

```
982 \newcommand*\Describe@Length[2]{%
983   \PrintLengthName{#1}&
984   (Default: {\macroargsstyle#2\unskip})%
985   \@ifnextchar\DescribeLength
986   {\}%
987   {%
988     \endtabular
989     \y@egroup
990     \PrintLength
991     \endgroup
992   }%
993 }
```

For Environments

`\DescribeEnv`

```
994 \@ifundefined{DescribeEnv}{}{%
995   \PackageInfo{ydoc-desc}{Redefining \string\
996     DescribeEnv}{}%
997 }
998 \let\DescribeEnv\relax
999
1000 \newcommand*\DescribeEnv[2][ ]{%
1001   \begingroup
1002   \def\DescribeEnv@name{#2}%
1003   \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.

```

1002 \ifx\@currenvir\DescribeEnv@string
1003 \def\after@Macro@args{%
1004 \let\after@Macro@args\empty
1005 \setbox\@tempboxa\hbox\y@bgroup
1006 \@ifnextchar\end{%
1007 {\DescribeEnv@newline}%
1008 #1%
1009 }%

```

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

```

1010 \else
1011 \ifx\relax#1\relax
1012 \def\after@Macro@args{%
1013 \y@bgroup
1014 \endDescribeEnv
1015 }%
1016 \else
1017 \def\after@Macro@args{%
1018 \setbox\@tempboxa\hbox\y@bgroup
1019 \DescribeEnv@newline\MacroArgs#1%
1020 \endDescribeEnv
1021 }%
1022 \fi
1023 \fi

```

Start \vbox and adds first line.

```

1024 \setbox\descbox\vbox\y@bgroup
1025 \envcodestyle
1026 \let\PrintEnv\PrintSubEnv
1027 \hbox\y@bgroup
1028 \PrintEnvName{\begin}{\DescribeEnv@name}%
1029 \ydoc@macrocatcodes
1030 \macroargsstyle
1031 \read@Macro@arg
1032 }

```

\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.

```

1033 \newcommand*\DescribeEnv@newline [1] [Opt] {%
1034 \strut\y@egroup
1035 {\vskip#1}%
1036 \hbox\y@bgroup\strut
1037 \hspace*{\descsep}%
1038 \ignorespaces
1039 }%

```

`\DescribeEnv@string`

Holds the environment name for comparison.

```
1040 \def\DescribeEnv@string{DescribeEnv}
```

`\descbox`

Save box to store description content.

```
1041 \newbox\descbox
```

`\endDescribeEnv`

```
1042 \def\endDescribeEnv{%
1043   \y@egroup
1044   \begingroup
1045   \setbox\@tempboxa\lastbox
1046   \ifcase0%
1047     \ifdim\wd\@tempboxa>\descsep1\fi
1048     \ifdim\ht\@tempboxa>\ht\strutbox1\fi
1049     \ifdim\dp\@tempboxa>\dp\strutbox1\fi
1050   \else
1051     \box\@tempboxa
1052   \fi
1053 \endgroup
1054 \hbox\y@bgroup
1055   \PrintEnvName{\end}{\DescribeEnv@name}
1056 \y@egroup
1057 \y@egroup
1058 \PrintEnv
1059 \endgroup
1060 }
```

3.6.7 Print Macros

`\PrintMacroName`

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

```
1061 \def\PrintMacroName#1{%
1062   {\macrodescstyle{\strut
1063     \texttt{\char92}}%
1064     \escapechar\m@ne
1065     \string#1\strut}}%
1066 }
```

`\PrintKeyName`

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

```
1067 \def\PrintKeyName#1{%
1068     {\keydescstyle{\strut
1069     #1\strut}}}%
1070 }
```

`\PrintLengthName`

Formats length register name.

```
1071 \let\PrintLengthName\PrintMacroName
```

`\PrintEnvName`

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

```
1072 \def\PrintEnvName#1#2{%
1073     \strut
1074     \string#1\braceleft
1075     {\macrodescstyle#2\strut}%
1076     \braceright
1077 }
```

`\PrintMacros`

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

```
1078 \def\PrintMacros{%
1079     \par\vspace\beforedescskip
1080     \begingroup
1081     \sbox\@tempboxa{\descframe{\usebox{\descbox}}}%
1082     \Needspace*{\dimexpr\ht\@tempboxa+3\baselineskip\relax}%
1083     \par\noindent
1084     \ifdim\wd\@tempboxa>\dimexpr\linewidth-2\descindent\relax
1085         \makebox[\linewidth][c]{\usebox\@tempboxa}%
1086     \else
1087         \hspace*{\descindent}%
1088         \usebox\@tempboxa
1089     \fi
1090 \endgroup
1091 \par
1092 \vspace\afterdescskip
```

```

1093   \par\noindent
1094 }
1095 \def\descframe#1{%
1096   \fbox{\hspace*{\descsep}#1\hspace*{\descsep}}%
1097 }

```

`\PrintLength`

Prints lengths registers described using one or multiple `\DescribeLength`.

```

1098 \let\PrintLength\PrintMacros

```

`\PrintEnv`

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

```

1099 \let\PrintEnv\PrintMacros

```

`\PrintSubEnv`

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

```

1100 \def\PrintSubEnv{%
1101   \hbox{\hbox{\usebox{\descbox}}}%
1102 }

```

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.

```

1103 {%
1104 \@firstofone{%
1105   \catcode'\=12
1106   \gdef\bslash
1107 }{\}
1108 }%}

```

`\percent`

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

```

1109 \begingroup
1110 \catcode'\%=12
1111 \gdef\percent{%}
1112 \endgroup

```

`\braceleft`

`\braceright`

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

```
1113 \begingroup
1114 \catcode '\<=1
1115 \catcode '\>=2
1116 \catcode '\{=12
1117 \catcode '\}=12
1118 \gdef\braceleft <{>
1119 \gdef\braceright <}>
1120 \endgroup
```

3.6.9 Other Macros

`\y@bgroup`

`\y@egroup`

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

```
1121 \def\y@bgroup{\bgroup\color@setgroup}
1122 \def\y@egroup{\color@endgroup\egroup}
```

`\codeline`

```
1123 \newcommand*{\codeline}[1][c]{%
1124   \codelinebefore
1125   \hbox to \hsize\bgroup
1126   \ifx i#1\hspace*{\leftmargin}\else
1127     \ifx l#1\else\hss\fi
1128   \fi
1129   \let\xspace\relax
1130   \hbox\bgroup
1131   \aftergroup\codeline@end
1132   \aftergroup#1%
1133   \afterassignment\MacroArgs
1134   \let\@let@token=%
1135 }
1136 \def\codeline@end#1{%
1137   \ifx r#1\else\hss\fi
1138   \egroup
```

```

1139     \codelineafter
1140 }
1141 \newcommand*\codelinebefore{\par\smallskip\noindent}
1142 \newcommand*\codelineafter {\par\smallskip\noindent}

```

codequote

```

1143 \newenvironment{codequote}{%
1144     \def\{\{\newline\relax\MacroArgs}%
1145     \par\smallskip\bgroup\leftskip=\leftmargin\
1146         \rightskip=\rightmargin\noindent\MacroArgs}
1147     {\par\egroup\smallskip\noindent\
1148         ignorespacesafterend}

```

macroquote

```

1147 \newenvironment{macroquote}{%
1148     \def\{\{\newline\relax\Macro}%
1149     \par\smallskip\bgroup\leftskip=\leftmargin\
1150         \rightskip=\rightmargin\noindent\Macro}
1151     {\par\egroup\smallskip\noindent\
1152         ignorespacesafterend}

```

3.7 Include Code Examples

```

1151 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
1152 \ProvidesPackage{ydoc-expl}[%
1153 %<!DATE>
1154 %<!VERSION>
1155 %<*DRIVER>
1156     2011/08/11 develop
1157 %</DRIVER>
1158     ydoc package to insert live examples of LaTeX /
1159     code]
1159 \RequirePackage{listings}
1160 \lst@RequireAspects{writefile}
1161 \def\ydoc@exafile{\jobname.exe}

```

examplecode

```

1162 \lstdefinestyle{examplecode}{%
1163     language=[latex]tex,
1164     basicstyle=\ttfamily,
1165     columns=fullflexible,
1166     numbers=left,

```



```

1167     firstnumber=1,
1168     numberstyle=\tiny\color{gray}\sffamily,
1169     numbersep=5pt,
1170     breaklines,prebreak={\mbox{\tiny$\swarrow$}},
1171     commentstyle=\color{black!60},
1172 }%

```

exampleresult

```

1173 \lstdefinestyle{exampleresult}{%
1174     firstnumber=1,
1175     gobble=0,
1176     basicstyle=\ttfamily,
1177     columns=fullflexible,
1178     commentstyle=\color{black!60},
1179 }

```

exampleextract

```

1180 \lstdefinestyle{exampleextract}{gobble=4}%
1181 \newbox\examplecodebox
1182 \newbox\exampleresultbox

```

\BoxExample

```

1183 \def\BoxExample{%
1184     \setbox\examplecodebox\hbox{\color@setgroup
1185         \lstinputlisting[style=examplecode,style=/  

1186             thisexampleprint]%
1187         {\ydoc@exafile}}%
1188     \unskip\color@endgroup}%
1189     \setbox\exampleresultbox\hbox{\color@setgroup
1190         \lstset{style=exampleresult}%
1191         @@input\ydoc@exafile\relax
1192     \unskip\color@endgroup}%

```

\PrintExample

```

1193 %< *DISABLED >
1194 \RequirePackage{showexpl}
1195 \def\PrintExample{%
1196     \begingroup
1197     \lstset{style=examplecode}%

```

```

1198 \MakePercentComment
1199 \LTXinputExample[varwidth]{\ydoc@exafile}%
1200 \endgroup
1201 }
1202 %</DISABLED>

```

\PrintExample

```

1203 \def\PrintExample{%
1204 \begingroup
1205 \BoxExample
1206 \@tempdima=\textwidth
1207 \advance\@tempdima by -\wd\examplecodebox\relax
1208 \advance\@tempdima by -\wd\exampleresultbox\relax
1209 \advance\@tempdima by -15pt\relax
1210 \ifdim\@tempdima>\bigskipamount
1211 \hbox to \textwidth{%
1212 \null\hss
1213 \minipage[c]{\wd\exampleresultbox}\fbox{\usebox\
1214 exampleresultbox}\endminipage
1215 \hfill\hfill\hskip\bigskipamount\hskip15pt\hfill\
1216 \hfill
1217 \minipage[c]{\wd\examplecodebox}\usebox\
1218 examplecodebox\endminipage
1219 \hss\null
1220 }%
1221 \else
1222 \vbox{%
1223 \centerline{\fbox{\usebox\exampleresultbox}}%
1224 \vspace{\bigskipamount}%
1225 \centerline{\usebox\examplecodebox}%
1226 }%
1227 \fi
1228 \endgroup
1229 }

```

examplecode

```

1227 \lstnewenvironment{examplecode}[1][\]{%
1228 \lstdefinestyle{thisexampleprint}{#1}%
1229 \setbox\@tempboxa\hbox\bgroup
1230 \lstset{style=exampleextract,#1}%
1231 \lst@BeginWriteFile{\ydoc@exafile}%
1232 }
1233 {%
1234 \lst@EndWriteFile
1235 \egroup

```

```

1236 \begingroup
1237 \MakePercentComment
1238 \catcode'\^M=5\relax
1239 \PrintExample
1240 \endgroup
1241 }

```

```

1242 \RequirePackage{float}

```

example

```

1243 \floatstyle{plain}
1244 \newfloat{example}{tbhp}{loe}
1245 \floatname{example}{\examplename}
1246 \def\examplename{Example}

```

exampletable

```

1247 \newenvironment{exampletable}{%
1248 \floatstyle{plaintop}%
1249 \restylefloat{example}%
1250 \example
1251 }{\endexample}

1252 \expandafter\ifx\csname ydocinclversion\endcsname\
1253 relax\else
1254 \endinput
1255 \fi

1256 \chardef\ydocinclversion=1

1257

1258 \newread\inFile
1259 \newread\subFile
1260 \newwrite\outFile
1261 \newif\ifContinue
1262 \newlinechar='^J

1263

1264 \def\makeOther#1{\catcode'#1=12\relax}
1265

1266 \let\inLine\relax
1267 \let\lastLine\relax

1268

1269 \def\includefiles#1#2{%
1270 \begingroup
1271 \immediate\openin\inFile#1\relax
1272 \immediate\openout\outFile#2\relax
1273 \makeOther\@%
1274 \makeOther\ \makeOther\\\makeOther\$%

```

```

1275 \makeOther\#\makeOther\^\makeOther\^^K%
1276 \makeOther\_ \makeOther\^^A\makeOther\%%
1277 \makeOther\~\makeOther\{\makeOther\}\makeOther\&%
1278 \endlinechar-1\relax
1279 \Continuetrue
1280 \loop
1281 \let\lastLine\inLine
1282 \read\inFile to\inLine
1283 \ifeof\inFile
1284 \Continuefalse
1285 \else
1286 \expandafter\checkLine\inLine\empty\empty\
empty\endLine
1287 \fi
1288 \ifContinue
1289 \repeat
1290 \immediate\closein\inFile
1291 \immediate\closeout\outFile
1292 \endgroup
1293 \end
1294 }
1295
1296 \def\copyline{%
1297 \immediate\write\outFile{\inLine}%
1298 }
1299
1300 \chardef\percentcharnum='\%
1301
1302 \begingroup
1303 \makeOther\%\makeOther\@\relax
1304 \gdef\SubFileOptionString{%\<@}\relax
1305 \gdef\CommentChar{%}\relax
1306 \catcode '\|=0
1307 \makeOther\ \makeOther\|\relax
1308 |gdef|IfFalseString{% \iffalse}|relax
1309 |gdef|FiString{% \fi}|relax
1310 |endgroup
1311
1312 \def\checkLine#1#2#3#4\endLine{%
1313 \def\firstthree{#1#2#3}%
1314 \ifx\firstthree\SubFileOptionString
1315 \readSubFile#4\endLine
1316 \else
1317 \copyline
1318 \fi
1319 }
1320
1321 \def\readSubFile#1>#2\endLine{%
1322 \immediate\openin\subFile=#1\relax
1323 \ifeof\subFile

```

```

1324         % File not found
1325     \else
1326         \message{^^JIncluding subfile '#1'^^J}%
1327         \immediate\write\outFile{\CommentChar<#1>}%
1328         \ifx\lastLine\IfFalseString
1329             \immediate\write\outFile{\FiString}%
1330         \fi
1331         \copySubFile
1332         \ifx\lastLine\IfFalseString
1333             \immediate\write\outFile{\IfFalseString}%
1334         \fi
1335         \immediate\write\outFile{\CommentChar</#1>}%
1336     \fi
1337     \immediate\closein\subFile
1338 }
1339
1340 \def\copySubFile{%
1341     \read\subFile to\subLine
1342     \ifeof\subFile\else
1343         \immediate\write\outFile{\subLine}%
1344         \expandafter\copySubFile
1345     \fi
1346 }

```