

The `ydoc` Class and Packages

Martin Scharrer

martin@scharrer-online.de

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

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

Version 0.4alpha

2011/01/09

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 L^AT_EX 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 L^AT_EX 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}[\langle number of arguments \rangle]{\langle arg 1 description \rangle}...{\langle arg n description \rangle}
  \langle macro documentation \rangle
  \begin{macrocode}
    \langle macro code \rangle
  \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}
  \langle macro code \rangle
\end{macrocode}
```

This environment wraps around any T_EX 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}{\langle environment name \rangle}[\langle number of arguments \rangle]{\langle arg 1 description \rangle}...{\langle arg n description \rangle}
\begin{macrocode}
\langle macro code \rangle
\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\langle macro \rangle\langle macro arguments \rangle

```

The `\DescribeMacro` is used to describe macros including their arguments. It takes the to be described `\langle macro \rangle` as first argument (can also be enclosed in `{ }`). The macro name can include ‘@’. Any number of `\langle macro arguments \rangle` (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 `{ }` direct after `\DescribeMacro` a `\relax` (or a similar macro) should be inserted between them, otherwise the group will be taken as mandatory argument of the described macro.

Multiple `\DescribeMacro` in a row will automatically 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*[\langle optional \rangle]{\langle meta text \rangle}` will result in `\mymacro*[\langle optional \rangle]{\langle meta text \rangle}` (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\langle macro \rangle\langle macro arguments \rangle

```

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.

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>}}
```

```
\begin{DescribeMacros}
  \Macro<\name><arguments>
  \Macro<\name><arguments>
  ...
\end{DescribeMacros}
```

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

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

This is a special version of the `DescribeMacros` environment which adds a tabular environment around the content. This is useful if a large set of small macros should be described at once. Placing them all below each other would

result in a very bad page layout. The environment has one argument which is passed to `tabular` as the column definition. A ‘`@{}`’ is added before and after to remove any margins.

```
\begin{DescribeEnv}{\langle name \rangle}{\langle arguments \rangle}
  \langle body content \rangle \\  

  \langle more body content \rangle
\end{DescribeEnv}
```

```
\DescribeEnv[\langle body content \rangle]{\langle name \rangle}{\langle arguments \rangle}
```

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{\langle name \rangle}{\langle default value \rangle}
```

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

2.3 Format Macros

```
\cs{\langle macro name \rangle}      \env{\langle environment name \rangle}
\pkg{\langle package name \rangle}  \cls{\langle class name \rangle}
```

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

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

Description	Syntax	Result	Macro ^a
Meta text	<code><text></code>	$\langle text \rangle$	<code>\meta{$\langle text \rangle$}</code>
Mandatory Argument	<code>{args}</code>	<code>{args}</code>	
—, with meta text	<code>{<text>}</code>	<code>{$\langle text \rangle$}</code>	<code>\marg{$\langle text \rangle$}</code>
Optional Argument	<code>[args]</code>	<code>[args]</code>	
—, with meta text	<code>[<text>]</code>	<code>[$\langle text \rangle$]</code>	<code>\oarg{$\langle text \rangle$}</code>
Picture Argument	<code>(args)</code>	<code>(args)</code>	
—, with meta text	<code>(<text>)</code>	<code>($\langle text \rangle$)</code>	<code>\parg{$\langle text \rangle$}</code>
Beamer Overlay Argument	<code><<args>></code>	<code><args></code>	
—, with meta text	<code><< <text> >></code>	<code><$\langle text \rangle$></code>	<code>\aarg{$\langle text \rangle$}</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>	\boxed{T}	
Unbreakable Space	<code>~</code>		
Space (explicit macro)	<code>\space</code>		
Second macro (e.g. endmarker)	<code>\AlsoMacro\macro</code>	<code>\macro</code>	

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

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

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

These 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>

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

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

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

2.5 Macros and Environments to include LaTeX Code Examples

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

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

3 Implementation

3.1 Class File

```
1 \LoadClassWithOptions{article}
2 %%\RequirePackage{doc}
3 \RequirePackage{ydoc}
```

3.2 Package File

```
4 \RequirePackage{ydoc-code}
5 \RequirePackage{ydoc-expl}
6 \RequirePackage{ydoc-desc}
7 \RequirePackage{ydoc-doc}
8
9 \RequirePackage{newverbs}
10 \MakeSpecialShortVerb{\qverb}{\"}
11 \AtBeginDocument{\catcode'\^A=14\relax}
```

3.3 Macros and Environments to document Implementations

```
12 \RequirePackage{hyperref}
13 \hypersetup{colorlinks=true,pdfborder=0 0 0,✓
  pdfborderstyle={}}
```

3.3.1 Color and style definitions

```
14 \RequirePackage{xcolor}
15 \definecolor{macroimpl}{rgb}{0.0,0.0,0.4}
```

3.3.2 General Macros

`\ydocwrite`

```
16 \@ifundefined{ydocwrite}{%
17   \newwrite\ydocwrite
18 }{}
```

`\ydocfname`


```

19 \@ifundefined{ydocfname}{%
20   \def\ydocfname{\jobname.cod}%
21 }{}

```

`\ydoc@catcodes`

```

22 \def\ydoc@catcodes{%
23   \let\do\@makeother
24   \dospecials
25   \catcode'\=\active
26   \catcode'\^M=\active
27   \catcode'\ =\active
28 }

```

3.3.3 Handling Macrocode

`macrocode`

```

29 \def\macrocode{%
30   \par\noindent
31   \begingroup
32   \ydoc@catcodes
33   \macro@code
34 }
35 \def\endmacrocode{}

```

`\macro@code`

#1: verbatim macro code

```

36 \begingroup
37 \endlinechar\m@ne
38 \@firstofone{%
39   \catcode'\|=0\relax
40   \catcode'\(=1\relax
41   \catcode'\)=2\relax
42   \catcode'\*=14\relax
43   \catcode'\{=12\relax
44   \catcode'\}=12\relax
45   \catcode'\ =12\relax
46   \catcode'\%=12\relax
47   \catcode'\=\active

```

```

48 \catcode'\^M=\active
49 \catcode'\ =\active
50 }*
51 \gdef\macro@code#1^M% \end{macrocode}{*
52 \endgroup\expandafter\macro@@code\expandafter(\✓
    ydoc@removeline#1\noexpand\lastlinemacro)*
53 )*
54 \gdef\ydoc@removeline#1^M(\noexpand\firstlinemacro)*
55 \gdef\ydoc@defspecialmacros(*
56 \def^M(\noexpand\newlinemacro)*
57 \def (\noexpand\spacemacro)*
58 \def\(\noexpand\bslashmacro)*
59 )*
60 \gdef\ydoc@defrevspecialmacros(*
61 \def\newlinemacro(\noexpand^M)*
62 \def\spacemacro(\noexpand )*
63 \def\bslashmacro(\noexpand\)*
64 )*
65 \endgroup

```

`\macro@@code`

#1: verbatim macro code

```

66 \def\macro@@code#1{%
67   {\ydoc@defspecialmacros
68     \xdef\themacrocode{#1}}%
69   \PrintMacroCode
70   \end{macrocode}%
71 }

```

`\linenumberbox`

```

72 \def\newlinemacro{\\\null}
73 \def\spacemacro{\ }
74 \def\bslashmacro{\char92}
75 \def\lastlinemacro{}
76 \def\firstlinemacro{\linenumberbox}
77 \def\newlinemacro{\\\linenumberbox}
78 \newcounter{linenumber}
79 \def\linenumberbox{%
80   \hbox to 1.25em{}%
81   \llap{%
82     \stepcounter{linenumber}%

```

```

83     {\footnotesize\color{gray}\thelinenumber~}%
84   }%
85 }

```

`\PrintMacroCode`

```

86 \def\PrintMacroCode{%
87   \begingroup
88   \ttfamily
89   \noindent\themacrocode
90   \endgroup
91 }

```

`\PrintMacroCode`

```

92 \RequirePackage{listings}

93 \def\PrintMacroCode{%
94   \begingroup
95   \let\firstlinemacro\empty
96   \let\lastlinemacro\empty
97   \def\newlinemacro{^^J}%
98   \let\bslashmacro\bslash
99   \let\spacemacro\space
100   \immediate\openout\ydocwrite=\ydocfname\relax
101   \immediate\write\ydocwrite{\themacrocode}%
102   \immediate\closeout\ydocwrite
103   \@nameuse{ydoc@counttbslashes}%
104   \ydoclistingssettings
105   \let\input\@input
106   \lstinputlisting{\ydocfname}%
107   \endgroup
108 }

```

`\ydoclistingssettings`

```

109 \def\ydoclistingssettings{%
110   \lstset{%
111     language=[latex]tex,basicstyle=\ttfamily,
112     numbers=left,numberstyle=\tiny\color{gray},↵
       firstnumber=last,

```

```

113     breaklines ,prebreak={\mbox{\tiny$\swarrow$}}%
114   }%
115 }

```

`\macro@impl@args`

#1: number of macro arguments

```

116 \def\macro@impl@args[#1]{%
117   \begingroup
118   \parindent=10pt\relax
119   \let\macro@impl@argcnt\@tempcnta
120   \let\macro@impl@curarg\@tempcntb
121   \macro@impl@argcnt=#1\relax
122   \macro@impl@curarg=0\relax
123   \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
124     \expandafter\macro@impl@arg
125   \else
126     \expandafter\macro@impl@endargs
127   \fi
128 }

```

`\macro@impl@endargs`

```

129 \def\macro@impl@endargs{%
130   \endgroup
131   \unskip\par\noindent\ignorespaces
132 }

```

`\macro@impl@argline`

#1: argument number
#2: argument description

```

133 \def\macro@impl@argline#1#2{%
134   \par{\texttt{\#\#1}:~#2\strut}%
135 }

```

`\macro@impl@arg`

#1: argument description

```

136 \def\macro@impl@arg#1{%
137   \advance\macro@impl@curarg by\@ne\relax
138   \macro@impl@argline{\the\macro@impl@curarg}{#1}%
139   \ifnum\macro@impl@curarg<\macro@impl@argcnt\relax
140     \expandafter\macro@impl@arg
141   \else
142     \expandafter\macro@impl@endargs
143   \fi
144 }

```

macro

#1: implemented macro

```

145 \def\macro#1{%
146   \PrintMacroImpl{#1}%
147   \@ifnextchar[%]
148     {\macro@impl@args}%
149     {}%
150 }
151 \def\endmacro{}

```

environment

#1: environment name

```

152 \def\environment#1{%
153   \PrintEnvImplName{#1}%
154   \@ifnextchar[%]
155     {\macro@impl@args}%
156     {}%
157 }
158 \def\endenvironment{}

```

\PrintMacroImpl

#1: macro (token)

```

159 \def\PrintMacroImpl#1{%
160   \par\bigskip\noindent
161   \hbox{%
162     \edef\name{\expandafter\@gobble\string#1}%
163     \global\@namedef{href@impl@\name}{}%
164     \immediate\write\@mainaux{%
165       \global\noexpand\@namedef{href@impl@\name}{}%
166     }%

```

```

167     \raisebox{4ex}[4ex]{\hypertarget{impl:\name}{}}%
168     \hspace*{\descindent}\fbox{%
169         \hspace*{\descsep}%
170         \@ifundefined{href@desc@\name}{}{\hyperlink{✓
            desc:\name}}}%
171         {\PrintMacroImplName{#1}}}%
172     \hspace*{\descsep}%
173 }%
174 }%
175 \par\medskip\noindent
176 }

```

\PrintMacroImplName

#1: macro (token)

```

177 \def\PrintMacroImplName#1{%
178     \implstyle{\string#1\strut}%
179 }

```

\PrintEnvImplName

#1: environment name

test

```

180 \def\PrintEnvImplName#1{%
181     \par\bigskip\noindent
182     \hbox{\hspace*{\descindent}\fbox{{\implstyle{#1}}}}✓
        %
183     \par\medskip
184 }

```

\implstyle

```

185 \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.

```

186 {%
187 \@firstofone{%
188   \catcode'\=12
189   \gdef\bslash
190   }{\}
191 }%}

```

3.4 Provide doc macros

\ydoc@countbslashes

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

```

192 \newcount\ydoc@bslashcnt
193 \def\ydoc@countbslashes{%
194   \begingroup
195     \let\firstlinemacro\empty
196     \let\lastlinemacro\empty
197     \let\newlinemacro\empty
198     \let\spacemacro\empty
199     \def\bslashmacro{\global\advance\ydoc@bslashcnt ✓
200       by\@ne}%
201     \setbox\@tempboxa\hbox{\themacrocode}%
202   \endgroup
203 }

```

\Checksum

```

203 \def\Checksum#1{%
204   \gdef\ydoc@checksum{#1}%
205 }
206 \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.

```
207 \def\AlsoImplementation{%
208   \gdef\StopEventually##1{%
209     \@bsphack
210     \gdef\Finale{##1\ydoc@checkchecksum}%
211     \@esphack
212   }%
213 }
214 \AlsoImplementation
215 \def\OnlyDescription{%
216   \@bsphack
217   \long\gdef\StopEventually##1{##1\endinput}%
218   \@esphack
219 }
220 \let\Finale\relax
```

`\MakePercentComment`

`\MakePercentIgnore`

```
221 \def\MakePercentIgnore{\catcode'\%9\relax}
222 \def\MakePercentComment{\catcode'\%14\relax}
```

`\DocInput`

```
223 \def\DocInput#1{\MakePercentIgnore\input{#1}\relax
    \MakePercentComment}
```

`\CharacterTable`

```
224 \providecommand*\CharacterTable{%
225   \begingroup
226   \CharTableChanges
227   \@CharacterTable
228 }
229 \def\@CharacterTable#1{%
```



```

230 \def\ydoc@used@CharacterTable{#1}%
231 \@onelevel@sanitize\ydoc@used@CharacterTable
232 \ifx\ydoc@used@CharacterTable\✓
      ydoc@correct@CharacterTable
233       \typeout{*****}%
234       \typeout{* Character table correct *}%
235       \typeout{*****}%
236 \else
237       \PackageError{ydoc}{Character table ✓
          corrupted}
238               {\the\wrong@table}
239       \show\ydoc@used@CharacterTable
240       \show\ydoc@correct@CharacterTable
241 \fi
242 \endgroup
243 }
244 \newhelp\wrong@table{Some of the ASCII characters are✓
      corrupted.^^J
245       I now \string\show\space you both tables ✓
          for comparison.}
246 \newcommand*\CharTableChanges{}

```

\ydoc@correct@CharacterTable

```

247 \def\ydoc@correct@CharacterTable
248 {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
249 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
250 Digits \0\1\2\3\4\5\6\7\8\9
251 Exclamation \! Double quote \" Hash (✓
      number) \#
252 Dollar \$ Percent \% ✓
      Ampersand \&
253 Acute accent \' Left paren \( Right ✓
      paren\)
254 Asterisk \* Plus \+ Comma ✓
      \,
255 Minus \- Point \. Solidus ✓
      \/
256 Colon \: Semicolon \; Less ✓
      than \<
257 Equals \= Greater than \> Question✓
      mark \?

```

```

258      Commercial at \@      Left bracket  \[      ✓
          Backslash          \\
259      Right bracket \]      Circumflex    \^      ✓
          Underscore         \_
260      Grave accent  \`      Left brace   \{      Vertical✓
          bar          \|
261      Right brace    \}      Tilde         \~}
262 \@onelevel@sanitize\ydoc@correct@CharacterTable
263 %

```

`\DoNotIndex`

```

264 \providecommand*\DoNotIndex[1]{%
265   \PackageWarning{ydoc}{Ignoring DoNotIndex - not ✓
     implemented yet!}{}{}}%
266 }

```

`\changes`

```

267 \providecommand*\changes[3]{%
268   \PackageWarning{ydoc}{Ignoring changes - not ✓
     implemented yet!}{}{}}%
269 }

```

`\RecordChanges`

```

270 \providecommand*\RecordChanges{%
271   \PackageWarning{ydoc}{List of changes not ✓
     implemented yet!}{}{}}%
272 }

```

`\PrintChanges`

```

273 \providecommand*\PrintChanges{%
274   \PackageWarning{ydoc}{List of changes not ✓
     implemented yet!}{}{}}%
275 }

```

`\PrintIndex`

```
276 \providecommand*\PrintIndex{%
277   \PackageWarning{ydoc}{Code index not implemented ✓
      yet!}{}{}%
278 }
```

`\CodelineIndex`

```
279 \providecommand*\CodelineIndex{%
280   \PackageWarning{ydoc}{Code line index not ✓
      implemented yet!}{}{}%
281 }
```

`\EnableCrossrefs`

```
282 \providecommand*\EnableCrossrefs{%
283   \PackageWarning{ydoc}{Cross references not ✓
      implemented yet!}{}{}%
284 }
```

`\GetFileInfo`

Current implementation taken from doc package.

```
285 \providecommand*\GetFileInfo[1]{%
286   \def\filename{#1}%
287   \def\@tempb##1 ##2 ##3\relax##4\relax{%
288     \def\filedate{##1}%
289     \def\fileversion{##2}%
290     \def\fileinfo{##3}}%
291   \edef\@tempa{\csname ver@#1\endcsname}%
292   \expandafter\@tempb\@tempa\relax? ? \relax\relax
293 }
```

`\ydoc@checkchecksum`

```

294 \def\ydoc@checkchecksum{%
295   \ifnum\ydoc@checksum=\m@ne
296     \message{^^J}%
297     \message{*****^^J}%
298     \message{* No checksum found! *^^J}%
299     \message{*****^^J}%
300     \GenericWarning{No checksum found}{Correct ✓
        checksum is \the\ydoc@bslashcnt^^J}{}}%
301   \else
302     \ifnum\ydoc@checksum=\z@
303       \message{^^J}%
304       \message{*****^^J}%
305       \message{* Checksum disabled *^^J}%
306       \message{*****^^J}%
307       \GenericWarning{Checksum disabled}{Correct ✓
        checksum is \the\ydoc@bslashcnt^^J}{}}%
308     \else
309       \ifnum\ydoc@checksum=\ydoc@bslashcnt
310         \message{^^J}%
311         \message{*****^^J}%
312         \message{* Checksum passed *^^J}%
313         \message{*****^^J}%
314       \else
315         \message{^^J}%
316         \message{*****^^J}%
317         \message{* Checksum wrong (\ydoc@checksum<>\the\✓
            ydoc@bslashcnt) ^^J}%
318         \message{*****^^J}%
319         \GenericError{Checksum wrong}{Correct checksum is ✓
            \the\ydoc@bslashcnt^^J}{}}%
320     \fi
321   \fi
322 \fi
323 }

324 \RequirePackage{shortvrb}
325 \AtBeginDocument{\MakeShortVerb{\|}}

```

3.5 Description Macros and Environments

```

326 \RequirePackage{hyperref}
327 \hypersetup{colorlinks=true,pdfborder=0 0 0,✓
    pdfborderstyle={}}

```

3.5.1 Color and style definitions

```
328 \RequirePackage{xcolor}
329 \definecolor{macrodesc}{rgb}{0.0,0.0,0.8}
330 \definecolor{macroimpl}{rgb}{0.0,0.0,0.4}
331 \definecolor{meta}{rgb}{0.0,0.4,0.4}
332 \definecolor{scriptcolor}{rgb}{0.2,0.6,0.2}
333 \definecolor{optioncolor}{rgb}{0.3,0.2,0}
334 \colorlet{optional}{black!65!white}
335 \colorlet{metaoptional}{optional!50!meta}
```

3.5.2 Text Formatting Macros

`\meta`

Prints $\langle meta\ text \rangle$.

```
336 \def\meta#1{%
337   \ensuremath\langle
338   {\metastyle{#1}\rangle}%
339   \ensuremath\rangle
340 }
```

`\marg`

Calls `\marg` with angles added to force meta format.

```
341 \def\marg#1{\@marg{<#1>}}
```

`\oarg`

Calls `\oarg` with angles added to force meta format.

```
342 \def\oarg#1{\@oarg{<#1>}}
```

`\parg`

Calls `\parg` with angles added to force meta format.

```
343 \def\parg#1{\@parg{<#1>}}
```

`\aarg`

Calls `\aarg` with angles added to force meta format.

```
344 \def\@arg#1{\@arg{<#1>}}
```

`\@marg`

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

```
345 \def\@marg#1{%
346   {\margstyle{%
347     {\ttfamily\braceleft}%
348     {\def\end@Macro@args{}\read@Macro@arg#1}%
349     {\ttfamily\braceright}%
350   }}%
351 }
```

`\@oarg`

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

```
352 \def\@oarg#1{%
353   {\oargstyle{%
354     {\ttfamily[]}%
355     {\def\end@Macro@args{}\read@Macro@arg#1}%
356     {\ttfamily[]}%
357   }}%
358 }
```

`\@parg`

Sets style and adds parentheses.

```
359 \def\@parg#1{%
360   {\pargstyle{%
361     {\ttfamily{}}%
362     {\def\end@Macro@args{}\read@Macro@arg#1}%
363     {\ttfamily})}%
364   }}%
365 }
```

`\@aarg`

Sets style and adds angles.

```

366 \def\@aarg#1{%
367   {\aargstyle{%
368     {\ttfamily<}%
369     {\def\end@Macro@args{\read@Macro@arg#1}%
370     {\ttfamily>}%
371   }}%
372 }

```

`\sarg`

Prints star with given style.

```

373 \def\sarg{{\sargstyle{*}}}

```

`\pkg`

`\cls`

`\env`

`\opt`

```

374 \RequirePackage{etoolbox}
375 \newrobustcmd*\pkg{\texttt}
376 \newrobustcmd*\cls{\texttt}
377 \newrobustcmd*\env{\texttt}
378 \newrobustcmd*\opt{\textsf}

```

`\cs`

`\cmd`

```

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

```

3.5.3 Text Formatting Styles

`\macrodescstyle`

Style of described macro names.

```
381 \def\macrodescstyle{\ttfamily\bfseries\color{✓  
macrodesc}}
```

`\macroargsstyle`

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

```
382 \def\macroargsstyle{\ttfamily}
```

`\envcodestyle`

Default style for code body content in described environments.

```
383 \def\envcodestyle{\ttfamily}
```

`\verbstyle`

Style for verbatim text inside macro argument list.

```
384 \def\verbstyle{\ttfamily}
```

`\metastyle`

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

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

`\margstyle`

Style for `\marg`.

```
386 \def\margstyle{}
```

`\oargstyle`

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


```
387 \def\oargstyle{\color{optional}\colorlet{meta}{\color{metaoptional}}}
```

\pargstyle

Style for \parg.

```
388 \def\pargstyle{}
```

\aargstyle

Style for \aarg.

```
389 \def\aargstyle{}
```

\sargstyle

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

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

3.5.4 Dimension Registers

\descindent

```
391 \newdimen\descindent
392 \descindent=-\parindent
```

\beforedescskip

```
393 \newdimen\beforedescskip
394 \beforedescskip=\bigskipamount
```

\afterdescskip

```
395 \newdimen\afterdescskip
396 \afterdescskip=\medskipamount
```

`\descsep`

Set to 1em in tt font.

```
397 \newdimen\descsep
398 \begingroup
399 \ttfamily
400 \global\descsep=1em\relax
401 \endgroup
```

3.5.5 Macro Argument Reading Mechanism

`\read@Macro@arg`

Reads next token and calls second macro.

```
402 \def\read@Macro@arg{%
403   \futurelet\@let@token\handle@Macro@arg
404 }
```

`\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.

```
405 \def\handle@Macro@arg{%
406   \ifcase0%
407     \ifx\@let@token\bgroup1\else
408     \ifx\@let@token[\empty2\else
409     \ifx\@let@token(\empty3\else
410     \ifx\@let@token<\empty4\else
411     \ifx\@let@token*\empty5\else
412     \ifx\@let@token'\empty6\else
413     \ifx\@let@token!\empty7\else
414     \ifx\@let@token\@sptoken8\else
415     \ifx\@let@token\space9\else
416     \ifx\@let@token~9\else
417     \ifx\@let@token\AlsoMacro10\else
418     \ifx\@let@token\DescribeMacro11\fi
419     \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
420   \relax
421   \unskip
422   \expandafter\end@Macro@args%0
423   \or\expandafter\read@Macro@marg%1
```

```

424     \or\expandafter\read@Macro@oarg%2
425     \or\expandafter\read@Macro@parg%3
426     \or\expandafter\read@Macro@angle%4
427     \or\expandafter\read@Macro@sarg%5
428     \or\expandafter\read@Macro@verb%6
429     \or\expandafter\read@Macro@cmds%7
430     \or\expandafter\read@Macro@rmSPACE%8
431     \or\expandafter\read@Macro@addtoken%9
432     \else%10-
433     \fi
434 }

```

`\end@Macro@args`

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

```

435 \def\end@Macro@args{%
436     \y@egroup
437     \after@Macro@args
438 }

```

`\after@Macro@args`

Hook to add additional commands in certain situations.

```

439 \def\after@Macro@args{%
440 }

```

Macro argument reading macros

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

`\read@Macro@marg`

```

441 \def\read@Macro@marg#1{%
442     \@marg{#1}\read@Macro@arg
443 }

```

`\read@Macro@oarg`

```

444 \def\read@Macro@oarg[#1]{%
445   \@oarg{#1}\read@Macro@arg
446 }

```

`\read@Macro@parg`

```

447 \def\read@Macro@parg(#1){%
448   \@parg{#1}\read@Macro@arg
449 }

```

`\read@Macro@aarg`

```

450 \def\read@Macro@aarg<#1>>{%
451   \@aarg{#1}\read@Macro@arg
452 }

```

`\read@Macro@angle`

```

453 \def\read@Macro@angle<{%
454   \futurelet\@let@token\read@Macro@angle@
455 }

```

`\read@Macro@angle@`

```

456 \def\read@Macro@angle@{%
457   \ifx\@let@token<%
458     \expandafter\read@Macro@aarg
459   \else
460     \expandafter\read@Macro@meta
461   \fi
462 }

```

`\read@Macro@meta`

```

463 \def\read@Macro@meta#1>{%
464   \meta{#1}\read@Macro@arg
465 }

```

`\read@Macro@sarg`

```
466 \def\read@Macro@sarg#1{%  
467   \sarg\read@Macro@arg  
468 }
```

`\read@Macro@verb`

Sets up verbatim mode calls second macro.

```
469 \def\read@Macro@verb{%  
470   \begingroup  
471   \let\do\@makeother  
472   \dospecials  
473   \read@Macro@verb@  
474 }
```

`\read@Macro@verb@`

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

```
475 \def\read@Macro@verb@'#1' {%  
476   \endgroup  
477   \ifx\relax#1\relax  
478     {\verbstyle{\string'}}}%  
479   \else  
480     {\verbstyle{#1}}}%  
481   \fi  
482   \read@Macro@arg  
483 }
```

`\read@Macro@cmds`

Simply executes given code.

```
484 \def\read@Macro@cmds!#1!{%  
485   #1\relax  
486   \read@Macro@arg  
487 }
```

`\read@Macro@rmSPACE`

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

```

488 \@firstofone{\def\read@Macro@rmspace} {%
489 \read@Macro@arg
490 }

```

`\read@Macro@addtoken`

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

```

491 \def\read@Macro@addtoken#1{%
492 #1\read@Macro@arg
493 }

```

3.5.6 Description Macros

For Macros

`\DescribeMacro`

```

494 \@ifundefined{DescribeMacro}{}{%
495 \PackageInfo{ydoc-desc}{Redefining \string\
DescribeMacro}{}%
496 }

```

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`.

```

497 \def\DescribeMacro{%
498 \DescribeMacros
499 \let\DescribeMacros\y@egroup
500 \def\after@Macro@args{\endDescribeMacros}%
501 \begingroup\makeatletter
502 \Describe@Macro
503 }

```

`\Describe@Macro`

```

504 \def\Describe@Macro#1{%
505 \endgroup
506 \edef\name{\expandafter\@gobble\string#1}%
507 \global\@namedef{href@desc@\name}{}%
508 \immediate\write\@mainaux{%

```

```

509     \global\noexpand\@namedef{href@desc@\name}{}%
510 }%
511 \hbox\y@bgroup
512 \@ifundefined{href@impl@\name}{}{\hyperlink{impl:\name}%
    name}}%
513 {\hypertarget{desc:\name}{\PrintMacroName{#1}}}%
514 \macroargsstyle
515 \read@Macro@arg
516 }

```

\Macro

Simply uses the two macros below.

```

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

```

\@Macro

Alternative definition of \Macro inside DescribeMacros environments.

```

518 \def\@Macro{%
519     \begingroup\makeatletter
520     \DescribeMacro
521 }

```

\AlsoMacro

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

```

522 \newcommand*\AlsoMacro{%
523     \begingroup\makeatletter
524     \AlsoMacro@
525 }
526 \def\AlsoMacro@#1{%
527     \endgroup
528     \PrintMacroName{#1}%
529     \read@Macro@arg
530 }

```

\MacroArgs

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

```

531 \newcommand*\MacroArgs{%
532   \begingroup
533   \def\end@Macro@args{\endgroup\xspace}%
534   \macroargsstyle
535   \read@Macro@arg
536 }
537 \RequirePackage{xspace}

```

`\DescribeMacros`

```

538 \def\DescribeMacros{%
539   \begingroup
540   \let\Macro\@Macro
541   \parindent=0pt\relax
542   \setbox\descbox\vbox\y@bgroup
543 }

```

`\endDescribeMacros`

```

544 \def\endDescribeMacros{%
545   \y@egroup
546   \PrintMacros
547   \endgroup
548 }

```

`\DescribeMacrosTabcolsep`

```

549 \def\DescribeMacrosTabcolsep{\tabcolsep}

```

`\DescribeMacrosTab`

```

550 \def\DescribeMacrosTab{%
551   \DescribeMacros
552   \hbox\y@bgroup
553   \tabcolsep=\DescribeMacrosTabcolsep\relax
554   \DescribeMacrosTab@
555 }
556 \def\DescribeMacrosTab@#1{\tabular{@{}#1@{}}}

```


`\endDescribeMacroTab`

```
557 \def\endDescribeMacroTab{%  
558   \endtabular\y@egroup  
559   \endDescribeMacros  
560 }
```

For Lengths

`\DescribeLength`

```
561 \newcommand*\DescribeLength{%  
562   \begingroup  
563   \let\DescribeLength\Describe@Length  
564   \setbox\descbox\hbox\y@bgroup  
565   \tabular{@{}l@{\hspace{2em}}l@{}}%  
566   \Describe@Length  
567 }
```

`\Describe@Length`

```
568 \newcommand*\Describe@Length[2]{%  
569   \PrintLengthName{#1}&  
570   (Default: {\macroargsstyle#2\unskip})%  
571   \@ifnextchar\DescribeLength  
572   {\}%  
573   {%  
574     \endtabular  
575     \y@egroup  
576     \PrintLength  
577     \endgroup  
578   }%  
579 }
```

For Environments

`\DescribeEnv`

```

580 \@ifundefined{DescribeEnv}{%
581   \PackageInfo{ydoc-desc}{Redefining \string\
        DescribeEnv}{}%
582 }
583 \let\DescribeEnv\relax

584 \newcommand*\DescribeEnv[2][]{%
585   \begingroup
586   \def\DescribeEnv@name{#2}%
587   \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.

```

588   \ifx\@currentvir\DescribeEnv@string
589     \def\after@Macro@args{%
590       \let\after@Macro@args\empty
591       \setbox\@tempboxa\hbox\y@bgroup
592       \@ifnextchar\end{%
593         {\DescribeEnv@newline}%
594         #1%
595       }%

```

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

```

596   \else
597     \ifx\relax#1\relax
598       \def\after@Macro@args{%
599         \y@bgroup
600         \endDescribeEnv
601       }%
602     \else
603       \def\after@Macro@args{%
604         \setbox\@tempboxa\hbox\y@bgroup
605         \DescribeEnv@newline\MacroArgs#1%
606         \endDescribeEnv
607       }%
608     \fi
609   \fi

```

Start `\vbox` and adds first line.

```

610   \setbox\descbox\vbox\y@bgroup
611   \envcodestyle
612   \let\PrintEnv\PrintSubEnv
613   \hbox\y@bgroup
614   \PrintEnvName{\begin}{\DescribeEnv@name}%
615   \macroargsstyle

```

```

616 \read@Macro@arg
617 }

```

\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.

```

618 \newcommand*\DescribeEnv@newline[1][0pt]{%
619 \strut\y@egroup
620 {\vskip#1}%
621 \hbox\y@bgroup\strut
622 \hspace*{\descsep}%
623 \ignorespaces
624 }%

```

\DescribeEnv@string

Holds the environment name for comparison.

```

625 \def\DescribeEnv@string{\DescribeEnv}

```

\descbox

Save box to store description content.

```

626 \newbox\descbox

```

\endDescribeEnv

```

627 \def\endDescribeEnv{%
628 \y@egroup
629 \begingroup
630 \setbox\@tempboxa\lastbox
631 \ifcase0%
632 \ifdim\wd\@tempboxa>\descsep1\fi
633 \ifdim\ht\@tempboxa>\ht\strutbox1\fi
634 \ifdim\dp\@tempboxa>\dp\strutbox1\fi
635 \else
636 \box\@tempboxa
637 \fi
638 \endgroup

```

```

639 \hbox\y@bgroup
640 \PrintEnvName{\end}{\DescribeEnv@name}
641 \y@egroup
642 \y@egroup
643 \PrintEnv
644 \endgroup
645 }

```

3.5.7 Print Macros

`\PrintMacroName`

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

```

646 \def\PrintMacroName#1{%
647   {\macrodescstyle{\strut
648     \texttt{\char92}%
649     \escapechar\m@ne
650     \string#1}}%
651 }

```

`\PrintLengthName`

Formats length register name.

```

652 \let\PrintLengthName\PrintMacroName

```

`\PrintEnvName`

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

```

653 \def\PrintEnvName#1#2{%
654   \strut
655   \string#1\braceleft
656   {\macrodescstyle#2\strut}%
657   \braceright
658 }

```

`\PrintMacros`

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

```

659 \def\PrintMacros{%
660   \par\vspace\beforedescskip
661   \noindent\hspace*{\descindent}%
662   \ifdim\wd\descbox>\linewidth
663     \makebox[\linewidth][c]{\fbox{\hspace*{\descsep}\↵
        usebox{\descbox}\hspace*{\descsep}}}%
664   \else
665     \fbox{\hspace*{\descsep}\usebox{\descbox}\hspace↵
        *{\descsep}}%
666   \fi
667   \par\vspace\afterdescskip
668 }

```

\PrintLength

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

```

669 \let\PrintLength\PrintMacros

```

\PrintEnv

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

```

670 \def\PrintEnv{%
671   \par\vspace\beforedescskip
672   \noindent\hspace*{\descindent}%
673   \fbox{\hspace*{\descsep}\usebox{\descbox}\hspace*{\↵
        descsep}}%
674   \par\vspace\afterdescskip
675 }

```

\PrintSubEnv

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

```

676 \def\PrintSubEnv{%
677   \hbox{\hbox{\usebox{\descbox}}}%
678 }

```

3.5.8 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.

```
679 {%
680 \@firstofone{%
681   \catcode'\=12
682   \gdef\bslash
683 }{\}
684 }%}
```

`\percent`

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

```
685 \begingroup
686 \catcode'\%=12
687 \gdef\percent{%}
688 \endgroup
```

`\braceleft`

`\braceright`

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

```
689 \begingroup
690 \catcode'\<=1
691 \catcode'\>=2
692 \catcode'\{=12
693 \catcode'\}=12
694 \gdef\braceleft <{>
695 \gdef\braceright <}>
696 \endgroup
```

3.5.9 Other Macros

`\y@bgroup`

`\y@egroup`

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

```
697 \def\y@bgroup{\bgroup\color@setgroup}
698 \def\y@egroup{\color@endgroup\egroup}
```

3.6 Include Code Examples

```
699 \RequirePackage{listings}
700 \lst@RequireAspects{writefile}
701 \def\ydoc@exafile{\jobname.exe}
```

`\exampleprintsettings`

```
702 \def\exampleprintsettings{numbers=left,numberstyle=\tiny\color{gray}\sffamily,numbersep=5pt}%
703 \newbox\examplecodebox
704 \newbox\exampleresultbox
```

`\BoxExample`

```
705 \def\BoxExample{%
706   \setbox\examplecodebox\hbox{\color@setgroup
707     \expandafter\expandafter\expandafter\
708       \lstinputlisting
709     \expandafter\expandafter\expandafter[%
710       \expandafter\exampleprintsettings\expandafter,\
711       thisexampleprintsettings]%
712     {\ydoc@exafile}%
713   \unskip\color@endgroup}%
714   \setbox\exampleresultbox\hbox{\color@setgroup
715     @@input\ydoc@exafile\relax
716   \unskip\color@endgroup}%
717 }
```

`\PrintExample`

```

716 %<*DISABLED>
717 \RequirePackage{showexpl}
718 \def\PrintExample{%
719   \begingroup
720   \lstset{basicstyle=\ttfamily}%
721   \MakePercentComment
722   \LTinputExample[varwidth]{\ydoc@exafile}%
723   \endgroup
724 }
725 %</DISABLED>

```

`\PrintExample`

```

726 \def\PrintExample{%
727   \begingroup
728   \BoxExample
729   \@tempdima=\textwidth
730   \advance\@tempdima by -\wd\examplecodebox\relax
731   \advance\@tempdima by -\wd\exampleresultbox\relax
732   \advance\@tempdima by -15pt\relax
733   \ifdim\@tempdima>\bigskipamount
734     \hbox to \textwidth{%
735       \null\hss
736       \minipage[c]{\wd\exampleresultbox}\fbox{\usebox\✓
        exampleresultbox}\endminipage
737       \hfill\hfill\hskip\bigskipamount\hskip15pt\hfill\✓
        \hfill
738       \minipage[c]{\wd\examplecodebox}\usebox\✓
        examplecodebox\endminipage
739       \hss\null
740     }%
741   \else
742     \vbox{%
743       \centerline{\fbox{\usebox\exampleresultbox}}%
744       \vspace{\bigskipamount}%
745       \centerline{\usebox\examplecodebox}%
746     }%
747   \fi
748   \endgroup
749 }

```



```
750 \def\examplecodesettings{gobble=4}
```

examplecode

```
751 \lstnewenvironment{examplecode}[1][\%
752 \def\thisexampleprintsettings{#1}%
753 \expandafter\lstset\expandafter{\%
    examplecodesettings,#1}%
754 \setbox\@tempboxa\hbox\bgroup
755 \lst@BeginWriteFile{\ydoc@exafile}%
756 }
757 {%
758 \lst@EndWriteFile
759 \egroup
760 \bgroup
761 \MakePercentComment
762 \catcode'\^^M=5\relax
763 \PrintExample
764 \endgroup
765 }

766 \RequirePackage{float}
```

example

```
767 \floatstyle{plain}
768 \newfloat{example}{tbhp}{loe}
769 \floatname{example}{\examplename}
770 \def\examplename{Example}
```

examletable

```
771 \newenvironment{examletable}{\%
772 \floatstyle{plaintop}%
773 \restylefloat{example}%
774 \example
775 }{\endexample}
```