

# The ydoc Class and Packages

Martin Scharrer  
[martin@scharrer.de](mailto:martin@scharrer.de)

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

VC: [https://bitbucket.org/martin\\_scharrer/ydoc/](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: “%`UUUU\end{macrocode}`”.

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

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

## 2.2 Description Macros and Environments

`\DescribeMacro⟨\macro⟩⟨macro arguments⟩`

The `\DescribeMacro` is used to describe macros including their arguments. It takes the to be described `⟨\macro⟩` as first argument (can also be enclosed in `{ }`). The macro name can include `'@'`. Any number of `⟨macro arguments⟩` (in a broad sense, see Table 1) following it are formatted as arguments of this macro. Any following non-argument token (normal text, macro, etc.) will make `\DescribeMacro` stop collecting arguments. For example, if a T<sub>E</sub>X group should be started using `{ }` directly after `\DescribeMacro` a `\relax` (or a similar macro) should be inserted between them, otherwise the group will be taken as mandatory argument of the described macro.

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

### Examples:

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

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

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

`\Macro⟨\macro⟩⟨macro arguments⟩`

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

`\MacroArgs⟨macro arguments⟩`

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

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

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

### Examples:

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

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

```
\MakeShortMacroArgs*{<char>}
```

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

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

```
\DeleteShortMacroArgs{<char>}
```

Globally removes the special meaning from `<char>` given to him by `\MakeShortMacroArgs`.

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

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

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

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

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

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

Description	Syntax	Result	Macro <sup>a</sup>
Meta text	<code>&lt;text&gt;</code>	<code>&lt;text&gt;</code>	<code>\meta{&lt;text&gt;}</code>
Mandatory Argument	<code>{args}</code>	<code>{args}</code>	
—, with meta text	<code>{&lt;text&gt;}</code>	<code>{&lt;text&gt;}</code>	<code>\marg{&lt;text&gt;}</code>
Optional Argument	<code>[args]</code>	<code>[args]</code>	
—, with meta text	<code>[&lt;text&gt;]</code>	<code>[&lt;text&gt;]</code>	<code>\oarg{&lt;text&gt;}</code>
Picture Argument	<code>(args)</code>	<code>(args)</code>	
—, with meta text	<code>(&lt;text&gt;)</code>	<code>(&lt;text&gt;)</code>	<code>\parg{&lt;text&gt;}</code>
Beamer Overlay Argument	<code>&lt;&lt;args&gt;&gt;</code>	<code>&lt;args&gt;</code>	
—, with meta text	<code>&lt;&lt; &lt;text&gt; &gt;&gt;</code>	<code>&lt;&lt; &lt;text&gt; &gt;&gt;</code>	<code>\aarg{&lt;text&gt;}</code>
Star	<code>*</code>	<code>*</code>	
Verbatim content	<code>'\$&amp;^%_#\$_\'</code>	<code>\$&amp;^%_#\$_\</code>	
—, produce ‘ char	<code>‘</code>	<code>‘</code>	
Insert any T <sub>E</sub> 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>	

<sup>a</sup>) As alternative to be used inside normal text.

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

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

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

The `DescribeEnv` can be used to describe environments in the same way the `\DescribeMacro` macro describes macros. Supported *<arguments>* are shown in Table 1. Potential *<body content>* can be placed between the begin and end of the environment description to explain the user what kind of material should be placed inside it. The environment also exists in macro form as `\DescribeEnv`, which allows to provide small *<body content>* as an optional argument. Please note that for this optional argument a `\MacroArgs` is automatically inserted, but not for the `\DescribeEnv` environment content.

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

<code>\DescribeLength&lt;\name&gt;{&lt;default value&gt;}</code>
--

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

## 2.3 Format Macros

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

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

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

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

<code>\meta{&lt;meta text&gt;}</code>	<code>\marg{&lt;argument text&gt;}</code>
<code>\oarg{&lt;argument text&gt;}</code>	<code>\parg{&lt;argument text&gt;}</code>
<code>\aarg{&lt;argument text&gt;}</code>	<code>\sarg</code>

This macros allow to typeset meta text and mandatory, optional, picture and beamer overlay arguments as well as a star symbol. They are used internally by `\MacroArgs` and friends. See Table 1 for examples.

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

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

## 2.4 Settings

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

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

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

`\descsep` (Default: 1em in tt font = 10.5pt)

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

## 2.5 Macros and Environments to include LaTeX Code Examples

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

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

(to be written)

## 3 Implementation

### 3.1 Class File

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

### 3.2 Package File

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

### 3.3 Config File



```

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

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

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

54 \urlstyle{sf}

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

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

67 \reversemarginpar

```

### 3.4 Macros and Environments to document Implementations

```

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

```

```

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

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

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

```

### 3.4.1 Color and style definitions

```

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

```

### 3.4.2 General Macros

`\ydocwrite`

```

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

```

`\ydocfname`

```

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

```

`\ydoc@catcodes`

```

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

```

### 3.4.3 Handling Macrocode

**macrocode**

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

**\macro@code**

#1: verbatim macro code

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

`\macro@@code`

#1: verbatim macro code

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

`\linenumberbox`

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

`\PrintMacroCode`

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

`\PrintMacroCode`

```
158 \RequirePackage{listings}

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

```

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

```

`\ydoclistingssettings`

```

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

```

`\macro@impl@args`

#1: number of macro arguments

```

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

```

**\macro@impl@endargs**

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

**\macro@impl@argline**

#1: argument number  
#2: argument description

```
201 \def\macro@impl@argline#1#2{%
202   \par{\texttt{\#\#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   \endgroup
295 }
```

`\Checksum`

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

`\AlsoImplementation`

`\OnlyDescription`

`\StopEventually`

## `\Finale`

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

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

## `\MakePercentComment`

## `\MakePercentIgnore`

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

## `\DocInput`

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

## `\CharacterTable`

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

```

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

```

\ydoc@correct@CharacterTable

```

339 \def\ydoc@correct@CharacterTable
340 {Upper-case \A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R/
    \S\T\U\V\W\X\Y\Z
341 Lower-case \a\b\c\d\e\f\g|h|i|j|k|l|m|n|o|p|q|r/
    \s\t\u\v\w\x\y\z
342 Digits \0\1\2\3\4\5\6\7\8\9
343 Exclamation \! Double quote \" Hash (/
    number) \#
344 Dollar \$ Percent \% /
    Ampersand \&
345 Acute accent \' Left paren \(( Right /
    paren \)
346 Asterisk \* Plus \+ Comma /
    \,
347 Minus \- Point \. Solidus /
    \/
348 Colon \: Semicolon \; Less /
    than \<
349 Equals \= Greater than \> Question/
    mark \?
350 Commercial at \@ Left bracket \[ /
    Backslash \\
351 Right bracket \] Circumflex \^ /
    Underscore \_
352 Grave accent \' Left brace \{ Vertical/
    bar \|
353 Right brace \} Tilde \~}
354 \@onelevel@sanitize\ydoc@correct@CharacterTable

```

355 %

#### **\DoNotIndex**

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

#### **\changes**

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

#### **\RecordChanges**

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

#### **\PrintChanges**

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

#### **\PrintIndex**

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

#### **\CodelineIndex**

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

### `\EnableCrossrefs`

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

### `\GetFileInfo`

Current implementation taken from doc package.

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

### `\ydoc@checkchecksum`

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

```

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

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

418 \RequirePackage{url}
419
420 \def\package{\def\@package}
421 \package{\jobname}
422
423 \def\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}
538 \definecolor{macrodesc}{rgb}{0,0.2,0.6}
539 \definecolor{keydesc}{rgb}{0,0.4,0.9}
540 \definecolor{macroimpl}{rgb}{0,0.1,0.3}
541 \definecolor{meta}{rgb}{0,0.25,0.75}
542 \definecolor{scriptcolor}{rgb}{0.2,0.6,0.2}
543 \definecolor{optioncolor}{rgb}{0.3,0.2,0}
544 \colorlet{optional}{black!65!white}
545 \colorlet{metaoptional}{optional!50!meta}
546 \providecolor{urlcolor}{named}{blue}
547 \providecolor{linkcolor}{named}{blue}
548 \providecolor{filecolor}{named}{blue}
549 \providecolor{citecolor}{named}{blue}
550 \providecolor{anchorcolor}{named}{blue}
551 \providecolor{menucolor}{named}{blue}
552 \providecolor{runcolor}{named}{blue}

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

```

### 3.6.2 Text Formatting Macros

`\meta`

Prints *[meta text](#)*.

```

568 \newrobustcmd*\meta[1]{%
569     {\metastyle{%
570         \ensuremath\langle
571         #1\/%
572         \ensuremath\rangle
573     }}%
574 }

```

### `\marg`

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

```
575 \newrobustcmd*{\marg}[1]{%
576   {\margstyle{%
577     {\ttfamily\braceleft}%
578     \meta{#1}%
579     {\ttfamily\braceright}%
580   }}%
581 }
```

### `\oarg`

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

```
582 \newrobustcmd*{\oarg}[1]{%
583   {\oargstyle{%
584     {\ttfamily[]}%
585     \meta{#1}%
586     {\ttfamily[]}%
587   }}%
588 }
```

### `\parg`

Sets style and adds parentheses.

```
589 \newrobustcmd*{\parg}[1]{%
590   {\pargstyle{%
591     {\ttfamily{}}%
592     \meta{#1}%
593     {\ttfamily{}}%
594   }}%
595 }
```

### `\aarg`

Sets style and adds angles.

```
596 \newrobustcmd*{\aarg}[1]{%
597   {\aargstyle{%
598     {\ttfamily<}%
599     \meta{#1}%
600     {\ttfamily>}%
601   }}%
602 }
```

`\sarg`

Prints star with given style.

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

`\pkg`

`\cls`

`\lib`

`\env`

`\opt`

`\file`

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

```

620 \newcommand*\filestyle[1]{\texttt{\textcolor{file/
    }{#1}}}
621 \colorlet{cls}{black}
622 \colorlet{lib}{black}
623 \colorlet{env}{black}
624 \colorlet{file}{black}
625 \colorlet{pkg}{black}
626 \definecolor{opt}{rgb}{0.5,0.16666,0}

```

`\cs`

`\cmd`

```

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

```

`\Key`

```

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

```

### 3.6.3 Text Formatting Styles

`\macrodescstyle`

Style of described macro names.

```

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

```

`\macrodescstyle`

Style of described macro names.

```

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

```

`\macroargsstyle`

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

```

632 \def\macroargsstyle{\ttfamily}

```

### `\envcodestyle`

Default style for code body content in described environments.

```
633 \def\envcodestyle{\ttfamily}
```

### `\verbstyle`

Style for verbatim text inside macro argument list.

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

### `\metastyle`

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

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

### `\margstyle`

Style for `\marg`.

```
636 \def\margstyle{}
```

### `\Optional`

### `\optional`

### `\optionalstyle`

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

### `\optionalon`

### `\optionaloff`

```
639 \def\optionalon{\protected\def\optional{\/  
    optionalstyle}}  
640 \def\optionaloff{\let\optional\relax}  
641 \optionalon
```

#### `\oargstyle`

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

```
642 \def\oargstyle{\optional}
```

#### `\pargstyle`

Style for `\parg`.

```
643 \def\pargstyle{}
```

#### `\aargstyle`

Style for `\aarg`.

```
644 \def\aargstyle{}
```

#### `\sargstyle`

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

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

### 3.6.4 Dimension Registers

#### `\descindent`

```
646 \newdimen\descindent  
647 \descindent=-\parindent
```

#### `\beforedescskip`

```
648 \newdimen\beforedescskip  
649 \beforedescskip=\bigskipamount
```

#### `\afterdescskip`

```
650 \newdimen\afterdescskip  
651 \afterdescskip=\medskipamount
```

### `\descsep`

Set to 1em in tt font.

```
652 \newdimen\descsep
653 \begingroup
654 \ttfamily
655 \global\descsep=1em\relax
656 \endgroup
```

## 3.6.5 Macro Argument Reading Mechanism

### `\read@Macro@arg`

Reads next token and calls second macro.

```
657 \def\read@Macro@arg{%
658   \futurelet\@let@token\handle@Macro@arg
659 }
```

### `\AlsoMacro`

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

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

### `\ydoc@short@AlsoMacro`

Makes & an alias for \AlsoMacro.

```
672 \begingroup
673 \catcode'\|\active
674 \gdef\ydoc@short@AlsoMacro{%
675   \catcode'\|\active
676   \let|\AlsoMacro
677 }
678 \endgroup
```

### `\ydoc@macrocatcodes`

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

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

### `\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.

```
688 \def\handle@Macro@arg{%
689   \expandafter\let\expandafter\handler\csname /
        handle@Macro@token@\meaning\@let@token\endcsname
690   \ifx\handler\relax
691     \def\handler{\ifhmode\unskip\fi\end@Macro@args}%
692   %<*DEBUG>
693   % \typeout{DEBUG: Stopped at: \expandafter\meaning\
        csname @let@token\endcsname}%
694   % \typeout{}%
695   %\else
696   %\expandafter\ifx\csname @let@token\endcsname\
        AlsoMacro
697   % \typeout{DEBUG: TOKEN: \string\AlsoMacro}%
698   %\else
699   % \typeout{DEBUG: TOKEN: \expandafter\meaning\
        csname @let@token\endcsname}%
700   %\fi
701 %</DEBUG>
702   \fi
703   \handler
704 }
705 \def\define@Macro@handler{%
706   \begingroup
707   \ydoc@macrocatcodes
708   \define@Macro@handler@
709 }
710 \def\define@Macro@handler@#1{%
711   \endgroup
712   \@namedef{handle@Macro@token@\meaning#1}%
713 }
```



`\end@Macro@args`

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

```
714 \def\end@Macro@args{%
715     \y@egroup
716     \after@Macro@args
717 }
```

`\after@Macro@args`

Hook to add additional commands in certain situations.

```
718 \def\after@Macro@args{%
719 }
```

### Macro argument reading macros

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

`\read@Macro@marg`

```
720 \define@Macro@handler{\bgroup}{%
721     \begingroup
722     \afterassignment\read@Macro@marg@
723     \let\@let@token=%
724 }
725 \def\read@Macro@marg@{%
726     \bgroup
727     \margstyle{}%
728     \let\end@Macro@args\empty%
729     {\ttfamily\braceleft}%
730     \aftergroup\read@Macro@marg@@
731     \read@Macro@arg
732 }
733 \def\read@Macro@marg@@{%
734     {\ttfamily\braceright}%
735     \endgroup
736     \read@Macro@arg
737 }
```

`\read@Macro@oarg`

```
738 \define@Macro@handler{[]}{%
739     \begingroup
740     \let\read@Macro@oarg@end\read@Macro@oarg@@end
```

```

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

```

#### \read@Macro@parg

```

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

```

#### \read@Macro@aarg

```

775     \def\read@Macro@aarg<{%
776         \begingroup
777         \let\read@Macro@aarg@end\read@Macro@aarg@@end
778         \let\end@Macro@args\read@Macro@aarg@end
779         \aargstyle{}%
780         {\ttfamily<}%
781         \read@Macro@arg

```

```

782 }
783 \define@Macro@handler{>}{%
784   \read@Macro@aarg@end
785 }
786 \def\read@Macro@aarg@@end#1>>{%
787   #1%
788   {\ttfamily>}%
789   \endgroup
790   \read@Macro@arg
791 }

```

`\read@Macro@angle`

```

792 \define@Macro@handler{<}<{%
793   \futurelet\@let@token\read@Macro@angle@
794 }

```

`\read@Macro@angle@`

```

795 \def\read@Macro@angle@{%
796   \ifx\@let@token<%
797     \expandafter\read@Macro@aarg
798   \else
799     \expandafter\read@Macro@meta
800   \fi
801 }

```

`\read@Macro@meta`

```

802 \def\read@Macro@meta#1>{%
803   \meta{#1}\read@Macro@arg
804 }

```

`\read@Macro@sarg`

```

805 \define@Macro@handler**{%
806   \sarg\read@Macro@arg
807 }

```

### `\read@Macro@verb`

Sets up verbatim mode calls second macro.

```
808 \define@Macro@handler{'}'{%  
809   \begingroup  
810   \let\do\@makeother  
811   \dospecials  
812   \@noligs  
813   \@makeother\'%  
814   \obeyspaces  
815   \read@Macro@verb@  
816 }
```

### `\read@Macro@verb@`

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

```
817 \begingroup  
818 \@makeother\'%  
819 \gdef\read@Macro@verb@#1' {%  
820   \endgroup  
821   \ifx\relax#1\relax  
822     {\verbstyle{\string'}}%  
823   \else  
824     {%  
825       \frenchspacing  
826       \@noligs\verbstyle{#1}}%  
827   \fi  
828   \read@Macro@arg  
829 }  
830 \endgroup
```

### `\read@Macro@cmds`

Simply executes given code.

```
831 \define@Macro@handler!!#1! {%  
832   #1\relax  
833   \read@Macro@arg  
834 }
```

### `\read@Macro@rmSPACE`

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

```

835 \define@Macro@handler{\@sptoken} {%
836   \read@Macro@arg
837 }

```

#### `\read@Macro@addtoken`

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

```

838 \define@Macro@handler{~}#1{%
839   #1\read@Macro@arg
840 }
841 \AtBeginDocument{%
842   \define@Macro@handler{~}#1{%
843     #1\read@Macro@arg
844   }
845 }
846 \define@Macro@handler{\space}#1{%
847   #1\read@Macro@arg
848 }

```

### 3.6.6 Description Macros

#### For Macros

#### `\DescribeMacro`

```

849 \@ifundefined{DescribeMacro}{}{%
850   \PackageInfo{ydoc-desc}{Redefining \string\
851     DescribeMacro}{}%
852 }

```

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.

```

852 \def\DescribeMacro{%
853   \DescribeMacros
854   \let\DescribeMacros\y@egroup
855   \optionalon
856   \def\after@Macro@args{\endDescribeMacros}%
857   \begingroup\makeatletter
858   \DescribeMacro
859 }

```

### `\DescribeScript`

```
860 \def\DescribeScript#1{%
861   \DescribeMacros
862   \let\DescribeMacros\y@egroup
863   \optionalon
864   \def\after@Macro@args{\endDescribeMacros}%
865   \hbox\y@bgroup
866   \texttt{#1}%
867   \ydoc@macrocatcodes
868   \macroargsstyle
869   \read@Macro@arg~%
870 }
```

### `\DescribeKey`

```
871 \def\DescribeKey{%
872   \DescribeKeys
873   \let\DescribeKeys\y@egroup
874   \optionalon
875   \def\after@Macro@args{\endDescribeKeys}%
876   \begingroup\makeatletter
877   \Describe@Macro
878 }
```

### `\Describe@Macro`

```
879 \def\Describe@Macro#1{%
880   \endgroup
881   \edef\name{\expandafter\@gobble\string#1}%
882   \global\@namedef{href@desc@\name}{}%
883   \immediate\write\@mainaux{%
884     \global\noexpand\@namedef{href@desc@\name}{}%
885   }%
886   \hbox\y@bgroup
887   \@ifundefined{href@impl@\name}{}{\hyperlink{impl:\n/
888     name}}%
889   {%
890     \hbox{\vbox to 0pt{\vss\hbox{\raisebox{4ex}{\n/
891       hypertarget{desc:\name}}}}}%
892     \PrintMacroName{#1}%
893   }%
894   \ydoc@macrocatcodes
895   \macroargsstyle
896   \read@Macro@arg
897 }
```

### `\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.

```
896 \newcommand*\MakeShortMacroArgs{%
897   \@ifstar
898     {\@MakeShortMacroArgs\Macro}%
899     {\@MakeShortMacroArgs\MacroArgs}%
900 }
901 \def\@MakeShortMacroArgs#1#2{%
902   \MakeShortVerb{#2}
903   \catcode'#2\active
904   \begingroup
905   \catcode'\~\active
906   \lccode'\~'#2\relax
907   \lowercase{\endgroup\gdef~{\bgroup\let~\egroup#1}}%
908 }
```

### `\DeleteShortMacroArgs`

```
909 \newcommand*\DeleteShortMacroArgs[1]{%
910   \DeleteShortVerb{#1}%
911 }
```

### `\Macro`

Simply uses the two macros below.

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

### `\@Macro`

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

```
913 \def\@Macro{%
914   \begingroup\makeatletter
915   \Describe@Macro
916 }
917 \define@Macro@handler\AlsoMacro{}
918 \define@Macro@handler\DescribeMacro{}
919 \define@Macro@handler\DescribeKey{}
920 \define@Macro@handler\DescribeScript{}
```

### **\MacroArgs**

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

```
921 \newcommand*\MacroArgs{%
922   \begingroup
923   \def\end@Macro@args{\endgroup\xspace}%
924   \ydoc@macrocatcodes
925   \macroargsstyle
926   %<*DEBUG>
927   %\typeout{}%
928   %\typeout{DEBUG: Start MacroArgs}%
929   %</DEBUG>
930   \read@Macro@arg
931 }
932 \RequirePackage{xspace}
```

### **\DescribeMacros**

```
933 \def\DescribeMacros{%
934   \begingroup
935   \let\Macro\@Macro
936   \parindent=0pt\relax
937   \setbox\descbox\vbox\y@bgroup
938 }
```

### **\endDescribeMacros**

```
939 \def\endDescribeMacros{%
940   \y@egroup
941   \PrintMacros
942   \endgroup
943 }
```

### **\DescribeKeys**

```
944 \def\DescribeKeys{%
945   \begingroup
946   \let\PrintMacroName\PrintKeyName
947   \let\Key\@Macro
948   \parindent=0pt\relax
949   \setbox\descbox\vbox\y@bgroup
950 }
```



`\endDescribeKeys`

```
951 \def\endDescribeKeys{%
952   \y@egroup
953   \PrintKeys
954   \endgroup
955 }
956 \def\PrintKeys{\PrintMacros}
```

`\DescribeMacrosTabcolsep`

```
957 \def\DescribeMacrosTabcolsep{\tabcolsep}
```

`\DescribeMacrosTab`

```
958 \def\DescribeMacrosTab{%
959   \DescribeMacros
960   \hbox\y@bgroup
961   \tabcolsep=\DescribeMacrosTabcolsep\relax
962   \DescribeMacrosTab@
963 }
964 \def\DescribeMacrosTab@#1{\tabular{@{}#1@{}}}
```

`\endDescribeMacrosTab`

```
965 \def\endDescribeMacrosTab{%
966   \endtabular\y@egroup
967   \endDescribeMacros
968 }
```

## For Lengths

`\DescribeLength`

```
969 \newcommand*\DescribeLength{%
970   \begingroup
971   \let\DescribeLength\Describe@Length
972   \setbox\descbox\hbox\y@bgroup
973   \tabular{@{}l@{\hspace{2em}}l@{}}%
974   \Describe@Length
975 }
```

`\Describe@Length`

```
976 \newcommand*\Describe@Length[2]{%
977   \PrintLengthName{#1}&
978   (Default: {\macroargsstyle#2\unskip})%
979   \@ifnextchar\DescribeLength
980   {\}%
981   {%
982     \endtabular
983     \y@egroup
984     \PrintLength
985     \endgroup
986   }%
987 }
```

### For Environments

`\DescribeEnv`

```
988 \@ifundefined{DescribeEnv}{}{%
989   \PackageInfo{ydoc-desc}{Redefining \string\
990     DescribeEnv}{}%
991 }
992 \let\DescribeEnv\relax
993
994 \newcommand*\DescribeEnv[2][]{%
995   \begingroup
996   \def\DescribeEnv@name{#2}%
997   \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.

```
996   \ifx\@currenvir\DescribeEnv@string
997     \def\after@Macro@args{%
998       \let\after@Macro@args\empty
999       \setbox\@tempboxa\hbox\y@bgroup
1000       \@ifnextchar\end{}%
1001       {\DescribeEnv@newline}%
1002       #1%
1003     }%
```

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

```
1004   \else
1005     \ifx\relax#1\relax
1006       \def\after@Macro@args{%
1007         \y@bgroup
```

```

1008         \endDescribeEnv
1009     }%
1010 \else
1011     \def\after@Macro@args{%
1012         \setbox\@tempboxa\hbox\y@bgroup
1013         \DescribeEnv@newline\MacroArgs#1%
1014         \endDescribeEnv
1015     }%
1016 \fi
1017 \fi

    Start \vbox and adds first line.

1018 \setbox\descbox\vbox\y@bgroup
1019 \envcodestyle
1020 \let\PrintEnv\PrintSubEnv
1021 \hbox\y@bgroup
1022 \PrintEnvName{\begin}{\DescribeEnv@name}%
1023 \ydoc@macrocatcodes
1024 \macroargsstyle
1025 \read@Macro@arg
1026 }

```

#### **\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.

```

1027 \newcommand*\DescribeEnv@newline [1] [0pt] {%
1028     \strut\y@egroup
1029     {\vskip#1}%
1030     \hbox\y@bgroup\strut
1031     \hspace*{\descsep}%
1032     \ignorespaces
1033 }%

```

#### **\DescribeEnv@string**

Holds the environment name for comparison.

```

1034 \def\DescribeEnv@string{DescribeEnv}

```

#### **\descbox**

Save box to store description content.

```

1035 \newbox\descbox

```

**\endDescribeEnv**

```
1036 \def\endDescribeEnv{%
1037   \y@egroup
1038   \begingroup
1039   \setbox\@tempboxa\lastbox
1040   \ifcase0%
1041     \ifdim\wd\@tempboxa>\descsep1\fi
1042     \ifdim\ht\@tempboxa>\ht\strutbox1\fi
1043     \ifdim\dp\@tempboxa>\dp\strutbox1\fi
1044   \else
1045     \box\@tempboxa
1046   \fi
1047   \endgroup
1048   \hbox\y@bgroup
1049   \PrintEnvName{\end}{\DescribeEnv@name}
1050   \y@egroup
1051   \y@egroup
1052   \PrintEnv
1053   \endgroup
1054 }
```

### 3.6.7 Print Macros

**\PrintMacroName**

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

```
1055 \def\PrintMacroName#1{%
1056   {\macrodescstyle{\strut
1057     \texttt{\char92}%
1058     \escapechar\m@ne
1059     \string#1\strut}}%
1060 }
```

**\PrintKeyName**

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

```
1061 \def\PrintKeyName#1{%
1062   {\keydescstyle{\strut
1063     #1\strut}}%
1064 }
```

### `\PrintLengthName`

Formats length register name.

```
1065 \let\PrintLengthName\PrintMacroName
```

### `\PrintEnvName`

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

```
1066 \def\PrintEnvName#1#2{%
1067   \strut
1068   \string#1\braceleft
1069   {\macrodescstyle#2\strut}%
1070   \braceright
1071 }
```

### `\PrintMacros`

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

```
1072 \def\PrintMacros{%
1073   \par\vspace\beforedescskip
1074   \begingroup
1075   \sbox\@tempboxa{\descframe{\usebox\@descbox}}}%
1076   \Needspace*{\dimexpr\ht\@tempboxa+3\baselineskip\relax}%
1077   \par\noindent
1078   \ifdim\wd\@tempboxa>\dimexpr\linewidth-2\descindent\relax
1079     \makebox[\linewidth][c]{\usebox\@tempboxa}%
1080   \else
1081     \hspace*{\descindent}%
1082     \usebox\@tempboxa
1083   \fi
1084   \endgroup
1085   \par
1086   \vspace\afterdescskip
1087   \par\noindent
1088 }
1089 \def\descframe#1{%
1090   \fbox{\hspace*{\descsep}#1\hspace*{\descsep}}}%
1091 }
```

### `\PrintLength`

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

```
1092 \let\PrintLength\PrintMacros
```

#### `\PrintEnv`

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

```
1093 \let\PrintEnv\PrintMacros
```

#### `\PrintSubEnv`

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

```
1094 \def\PrintSubEnv{%
1095   \hbox{\hbox{\usebox{\descbox}}}%
1096 }
```

### 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.

```
1097 {%
1098 \@firstofone{%
1099   \catcode'\=12
1100   \gdef\bslash
1101 }{\}
1102 }%}
```

#### `\percent`

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

```
1103 \begingroup
1104 \catcode'\%=12
1105 \gdef\percent{%}
1106 \endgroup
```

#### `\braceleft`

`\braceright`

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

```
1107 \begingroup
1108 \catcode'\<=1
1109 \catcode'\>=2
1110 \catcode'\{=12
1111 \catcode'\}=12
1112 \gdef\braceleft <{>
1113 \gdef\braceright <}>
1114 \endgroup
```

### 3.6.9 Other Macros

`\y@bgroup`

`\y@egroup`

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

```
1115 \def\y@bgroup{\bgroup\color@setgroup}
1116 \def\y@egroup{\color@endgroup\egroup}
```

`\codeline`

```
1117 \newcommand*{\codeline}[1][c]{%
1118   \codelinebefore
1119   \hbox to \hsize\bgroup
1120   \ifx i#1\hspace*{\leftmargin}\else
1121     \ifx l#1\else\hss\fi
1122   \fi
1123   \let\xspace\relax
1124   \hbox\bgroup
1125   \aftergroup\codeline@end
1126   \aftergroup#1%
1127   \afterassignment\MacroArgs
1128   \let\@let@token=%
1129 }
1130 \def\codeline@end#1{%
1131   \ifx r#1\else\hss\fi
1132   \egroup
1133   \codelineafter
1134 }
1135 \newcommand*\codelinebefore{\par\smallskip\noindent}
1136 \newcommand*\codelineafter {\par\smallskip\noindent}
```

#### codequote

```
1137 \newenvironment{codequote}{%
1138     \def\{\{\newline\relax\MacroArgs}%
1139     \par\smallskip\bgroup\leftskip=\leftmargin\relax
1140         \rightskip=\rightmargin\noindent\MacroArgs}
1141     {\par\egroup\smallskip\noindent\relax
1142         ignorespacesafterend}
```

#### macroquote

```
1141 \newenvironment{macroquote}{%
1142     \def\{\{\newline\relax\Macro}%
1143     \par\smallskip\bgroup\leftskip=\leftmargin\relax
1144         \rightskip=\rightmargin\noindent\Macro}
1145     {\par\egroup\smallskip\noindent\relax
1146         ignorespacesafterend}
```

### 3.7 Include Code Examples

```
1145 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
1146 \ProvidesPackage{ydoc-expl}[%
1147     %<!DATE>
1148     %<!VERSION>
1149     %<*DRIVER>
1150     2011/08/11 develop
1151     %</DRIVER>
1152     ydoc package to insert live examples of LaTeX /
1153     code]
1154 \RequirePackage{listings}
1155 \lst@RequireAspects{writefile}
1156 \def\ydoc@exafile{\jobname.exa}
```

#### examplecode

```
1156 \lstdefinestyle{examplecode}{numbers=left,firstnumber=
1157     =1,numberstyle=\tiny\color{gray}\sffamily,
1158     numbersep=5pt}%
1159
```

#### exampleextract

```
1157 \lstdefinestyle{exampleextract}{gobble=4}%
1158 \newbox\examplecodebox
1159 \newbox\exampleresultbox
```



`\BoxExample`

```
1160 \def\BoxExample{%
1161   \setbox\examplecodebox\hbox{\color@setgroup
1162     \lstinputlisting[style=examplecode,style=
1163       thisexampleprint]%
1164     {\ydoc@exafile}%
1165   \unskip\color@endgroup}%
1166   \setbox\exampleresultbox\hbox{\color@setgroup
1167     \@@input\ydoc@exafile\relax
1168   \unskip\color@endgroup}%
1169 }
```

`\PrintExample`

```
1169 %<DISABLED>
1170 \RequirePackage{showexpl}
1171 \def\PrintExample{%
1172   \begingroup
1173   \lstset{style=examplecode}%
1174   \MakePercentComment
1175   \LTXinputExample[varwidth]{\ydoc@exafile}%
1176   \endgroup
1177 }
1178 %</DISABLED>
```

`\PrintExample`

```
1179 \def\PrintExample{%
1180   \begingroup
1181   \BoxExample
1182   \@tempdima=\textwidth
1183   \advance\@tempdima by -\wd\examplecodebox\relax
1184   \advance\@tempdima by -\wd\exampleresultbox\relax
1185   \advance\@tempdima by -15pt\relax
1186   \ifdim\@tempdima>\bigskipamount
1187     \hbox to \textwidth{%
1188       \null\hss
1189       \minipage[c]{\wd\exampleresultbox}\fbox{\usebox\
1190         exampleresultbox}\endminipage
1191       \hfill\hfill\hskip\bigskipamount\hskip15pt\hfill\
1192       \hfill
1193       \minipage[c]{\wd\examplecodebox}\usebox\
1194         examplecodebox\endminipage
1195       \hss\null
1196     }%
1197 }
```

```

1194 \else
1195 \vbox{%
1196 \centerline{\fbox{\usebox\exampleresultbox}}}%
1197 \vspace{\bigskipamount}%
1198 \centerline{\usebox\examplecodebox}%
1199 }%
1200 \fi
1201 \endgroup
1202 }

```

#### examplecode

```

1203 \lstnewenvironment{examplecode}[1][\{%
1204 \lstdefinestyle{thisexampleprint}{#1}%
1205 \lstset{style=exampleextract,#1}%
1206 \setbox\@tempboxa\hbox\bgroup
1207 \lst@BeginWriteFile{\ydoc@exafile}%
1208 }
1209 {%
1210 \lst@EndWriteFile
1211 \egroup
1212 \begingroup
1213 \MakePercentComment
1214 \catcode'\^^M=5\relax
1215 \PrintExample
1216 \endgroup
1217 }
1218 \RequirePackage{float}

```

#### example

```

1219 \floatstyle{plain}
1220 \newfloat{example}{tbhp}{loe}
1221 \floatname{example}{\examplename}
1222 \def\examplename{Example}

```

#### exampletable

```

1223 \newenvironment{exampletable}{%
1224 \floatstyle{plaintop}%
1225 \restylefloat{example}%
1226 \example
1227 }{\endexample}

```

```

1228 \expandafter\ifx\csname ydocinclversion\endcsname\
      relax\else
1229 \endinput
1230 \fi
1231
1232 \chardef\ydocinclversion=1
1233
1234 \newread\inFile
1235 \newread\subFile
1236 \newwrite\outFile
1237 \newif\ifContinue
1238 \newlinechar='^^J
1239
1240 \def\makeOther#1{\catcode'#1=12\relax}
1241
1242 \let\inLine\relax
1243 \let\lastLine\relax
1244
1245 \def\includefiles#1#2{%
1246   \begingroup
1247   \immediate\openin\inFile#1\relax
1248   \immediate\openout\outFile#2\relax
1249   \makeOther\@%
1250   \makeOther\ \makeOther\\\makeOther\$%
1251   \makeOther#\makeOther^\makeOther\^^K%
1252   \makeOther_\makeOther\^^A\makeOther\%%
1253   \makeOther~\makeOther{\makeOther\}\makeOther\&%
1254   \endlinechar-1\relax
1255   \Continuetrue
1256   \loop
1257     \let\lastLine\inLine
1258     \read\inFile to\inLine
1259     \ifeof\inFile
1260       \Continuefalse
1261     \else
1262       \expandafter\checkLine\inLine\empty\empty\
        empty\endLine
1263     \fi
1264     \ifContinue
1265   \repeat
1266   \immediate\closein\inFile
1267   \immediate\closeout\outFile
1268   \endgroup
1269   \end
1270 }
1271
1272 \def\copyline{%
1273   \immediate\write\outFile{\inLine}%
1274 }
1275

```

```

1276 \chardef\percentcharnum='\%
1277
1278 \begingroup
1279 \makeOther\%\makeOther\@\relax
1280 \gdef\SubFileOptionString{%\<@\relax
1281 \gdef\CommentChar{%\}\relax
1282 \catcode'\|=0
1283 \makeOther\ \makeOther\|\relax
1284 |gdef|IfFalseString{%\ \iffalse}|relax
1285 |gdef|FiString{%\ \fi}|relax
1286 |endgroup
1287
1288 \def\checkLine#1#2#3#4\endLine{%
1289     \def\firstthree{#1#2#3}%
1290     \ifx\firstthree\SubFileOptionString
1291         \readSubFile#4\endLine
1292     \else
1293         \copyline
1294     \fi
1295 }
1296
1297 \def\readSubFile#1>#2\endLine{%
1298     \immediate\openin\subFile=#1\relax
1299     \ifeof\subFile
1300         % File not found
1301     \else
1302         \message{^^JIncluding subfile '#1'^^J}%
1303         \immediate\write\outFile{\CommentChar<*#1>}%
1304         \ifx\lastLine\IfFalseString
1305             \immediate\write\outFile{\FiString}%
1306         \fi
1307         \copySubFile
1308         \ifx\lastLine\IfFalseString
1309             \immediate\write\outFile{\IfFalseString}%
1310         \fi
1311         \immediate\write\outFile{\CommentChar</#1>}%
1312     \fi
1313     \immediate\closein\subFile
1314 }
1315
1316 \def\copySubFile{%
1317     \read\subFile to\subLine
1318     \ifeof\subFile\else
1319         \immediate\write\outFile{\subLine}%
1320         \expandafter\copySubFile
1321     \fi
1322 }

```