

# The ydoc Class and Packages

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

Version 0.7alpha  
2022/10/26

License: [LPPL v1.3c or later](#)

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

Texdoc: <https://texdoc.org/pkg/ydoc>

Homepage: <https://github.com/MartinScharrer/ydoc>

Repository: <https://github.com/MartinScharrer/ydoc.git>

Issue tracker: <https://github.com/MartinScharrer/ydoc/issues>

## 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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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 T<sub>E</sub>X code and types it verbatim. The environment end is read verbatim as well and must be written on a separate line beginning with a percent followed by exactly four spaces: '% `\end{macrocode}`'.

```

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

```

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

## 2.2 Description Macros and Environments

```
\DescribeMacro<macro><macro arguments>
```

The `\DescribeMacro` is used to describe macros included their arguments. It takes the to be described `<macro>` as first argument (can also be enclosed in `{ }`). The macro name can include `'@'`. Any number of `<macro arguments>` (in a broad sense, see Table 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 stacked inside one framed box. If this is not wanted simply separate them with `\relax` or any other macro or token. See also the `DescribeMacros` environment below.

### Examples:

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

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

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

```
\Macro<macro><macro arguments>
```

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

`\MacroArgs`*(macro arguments)*

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

`\AlsoMacro`*(\macro)(further macro arguments)*

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

**Examples:**

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

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

`\MakeShortMacroArgs`\**{(char)}*

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

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

`\DeleteShortMacroArgs`*{(char)}*

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

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

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

This environment can be used to place multiple macro description into the same framed box. The macros are described using `\Macro`, which has a slightly different

definition than outside of this environment, to place the description into a `\hbox`. The environment stacks these `\hboxes` in a `\vbox`. The macros can also be placed freely using anything which produces a `\hbox`, e.g. `\hbox{\Macro\A ~~~ \Macro\B}` or using a `tabular` (see also `DescribeMacroTab`).

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

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

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

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

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

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

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

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

## 2.3 Format Macros

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

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

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;text&gt;&gt;</code>	<code>\aarg{&lt;text&gt;}</code>
Star	<code>*</code>	<code>*</code>	
Verbatim content	<code>'\$&amp;^%_#&amp;\$\'</code>	<code>\$&amp;^%_#&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.

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

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

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

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

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

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

## 2.4 Settings

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

<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}</code> <code>\end{examplecode}</code>
--

(to be written)

## 3 Implementation

### 3.1 Class File

```
1 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
2 \ProvidesClass{ydoc}[%
3 %<!DATE>
4 %<!VERSION>
5 %<*DRIVER>
6     2011/08/11 develop
7 %</DRIVER>
8     ydoc class: document LaTeX class and packages]
```

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

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

### 3.2 Package File

```
14 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
15 \ProvidesPackage{ydoc}[%
16 %<!DATE>
17 %<!VERSION>
18 %<*DRIVER>
19     2011/08/11 develop
20 %</DRIVER>
21     ydoc package: document LaTeX class and packages]

22 \RequirePackage{ydoc-code}
23 \RequirePackage{ydoc-expl}
24 \RequirePackage{ydoc-desc}
25 \RequirePackage{ydoc-doc}

26
27 \RequirePackage{newverbs}
28 \MakeSpecialShortVerb{\qverb}{\"}
29 \AtBeginDocument{\catcode'\^A=14\relax}
30
31 \input{ydoc.cfg}
```

### 3.3 Config File



```

32 %% Please delete the following line on manual changes/
   :
33 \ProvidesFile{ydoc.cfg}[%
34 %<!DATE>
35 %<!VERSION>
36 %<*DRIVER>
37     2011/08/11 develop
38 %</DRIVER>
39     Default config file for ydoc]

40 \usepackage[T1]{fontenc}
41 \IfFileExists{fourier.sty}{%
42     \usepackage{fourier}
43 }{}

    Use 'lmodern' only for the 'tt' font if fourier is installed.

44 \IfFileExists{lmodern.sty}{
45     \IfFileExists{fourier.sty}{
46         \renewcommand{\ttdefault}{lmtt}
47     }{
48         \usepackage{lmodern}
49     }
50 }{}

51 \urlstyle{sf}

    Use micro-typesetting if pdftex is used:

52 \usepackage{ifpdf}
53 \ifpdf
54 \usepackage{microtype}
55 \fi

56 \usepackage{array}
57 \usepackage{booktabs}
58 \usepackage{multicol}
59 \usepackage{xcolor}
60 \usepackage{listings}
61 \usepackage{booktabs}
62 \usepackage{hyperref}

63 \reversemarginpar

```

### 3.4 Macros and Environments to document Implementations

```

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

```

```

69     2011/08/11 develop
70 %</DRIVER>
71     ydoc package to document macro code]

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

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

```

### 3.4.1 Color and style definitions

```

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

```

### 3.4.2 General Macros

`\ydocwrite`

```

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

```

`\ydocfname`

```

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

```

`\ydoc@catcodes`

```

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

```

### 3.4.3 Handling Macrocode

`macrocode`

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

`\macro@code`

#1: verbatim macro code

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

`\macro@@code`

#1: verbatim macro code

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

`\linenumberbox`

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

`\PrintMacroCode`

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

`\PrintMacroCode`

```
157 \RequirePackage{listings}  
  
158 \def\PrintMacroCode{%  
159   \begingroup  
160   \let\firstlinemacro\empty  
161   \let\lastlinemacro\empty
```

```

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

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

```

`\ydoclistingssettings`

```

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

```

`\macro@impl@args`

#1: number of macro arguments

```

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

```

`\macro@impl@endargs`

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

`\macro@impl@argline`

#1: argument number  
#2: argument description

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

`\macro@impl@arg`

#1: argument description

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

`macro`

#1: implemented macro

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

`key`

#1: key family  
#2: key name

```

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

```

### environment

#1: environment name

```

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

```

### style

#1: style name

```

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

```

### \PrintMacroImpl

#1: macro (token)

```

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

```

```

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

```

### \PrintMacroImplName

#1: macro (token)

```

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

```

### \PrintEnvImplName

#1: environment name

test

```

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

```

### \implstyle

```

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

```

### \bslash

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

```

269 {%
270 \@firstofone{%
271   \catcode'\=12
272   \gdef\bslash
273   }{\}
274 }%}

```



### 3.5 Provide doc macros

```
275 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
276 \ProvidesPackage{ydoc-doc}[%
277 %<!DATE>
278 %<!VERSION>
279 %<*DRIVER>
280     2099/01/01 develop
281 %</DRIVER>
282     ydoc package to provide 'doc' macros]
```

#### `\ydoc@countbslashes`

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

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

#### `\CheckSum`

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

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

### `\MakePercentComment`

### `\MakePercentIgnore`

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

### `\DocInput`

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

### `\CharacterTable`

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

```

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

```

\ydoc@correct@CharacterTable
------------------------------

```

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

```

354 %

### `\DoNotIndex`

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

### `\changes`

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

### `\RecordChanges`

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

### `\PrintChanges`

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

### `\PrintIndex`

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

### `\CodelineIndex`

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

### \EnableCrossrefs

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

### \GetFileInfo

Current implementation taken from doc package.

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

### \ydoc@checkchecksum

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

```

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

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

417 \RequirePackage{url}
418
419 \def\package{\def\@package}
420 \package{\jobname}
421
422 \def\bundle{\def\@bundle}
423 \let\@bundle\@empty
424
425
426 \def\ctanlocation{\def\@ctanlocation##1}
427 \ctanlocation{https://www.ctan.org/pkg/#1}
428
429 \date{Version \fileversion\space -- \filedate}
430
431 \def\@homepage{%
432     CTAN: \@ctan
433 }
434
435 \def\@ctan{%
436     \ydoc@eurl{\@ctanlocation{\ifx\@bundle\@empty\
         @package\else\@bundle\fi}}%
437 }
438
439 \def\@texdoc{%
440     \ydoc@eurl{https://texdoc.org/pkg/\ifx\@bundle\
         @empty\@package\else\@bundle\fi}%
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 \let\@email\empty
449 \let\@plainemail\empty

```

```

450
451 \let\@github\empty
452 \protected\def\github{\def\@github}
453
454 \title{The \texorpdfstring{\pkgtitle{\@package}}{\@
  @package} Package}
455 \def\@bundlesubtitle{Part of the \texorpdfstring{\@
  pkgtitle{\@bundle}}{\@bundle} bundle}
456
457 \protected\def\pkgtitle#1{%
458   \texorpdfstring{\textsf{#1}}{#1}%
459 }
460
461 \def\@begintitlelinks{%
462   \vskip .5em
463   \begingroup
464   \large \lineskip .5em%
465   \begin{tabular}[t]{rl}%
466   }
467
468 \def\@endtitlelinks{%
469   \end{tabular}%
470   \par
471   \endgroup
472 }
473
474 \def\@maketitlelink#1#2{%
475   #1: & #2\[\medskipamount]
476 }
477
478 \def\@license{%
479   \@maketitlelink{License}{\href{https://www.latex-
  project.org/lppl/lppl-1-3c/}{LPPL v1.3c or later}}\
  %
480 }
481
482 \def\ydoc@eurl#1{{\edef\URL{#1}}\expandafter\url\URL\
  }}
483
484 \def\@maketitle{%
485   \newpage
486   \null\vskip 2em
487   \begin{center}%
488     \let\footnote\thanks
489     {\LARGE \@title \par }\vskip 1.5em%
490     \ifx\@bundle\@empty\else
491     {\large \@bundlesubtitle \par }\vskip 1.5em%
492     \fi
493     {\large \lineskip .5em%
494     \begin{tabular}[t]{c}%

```

```

495         \@author
496     \end{tabular}%
497 \par}%
498 \ifx\@plainemail\empty\else
499     {\large \lineskip .5em%
500     \begin{tabular}[t]{c}%
501         \@email
502     \end{tabular}%
503     \par}%
504 \fi
505 \vskip 1em
506 {\large \@date }%
507 \vskip 1em
508 \ifx\@github\@empty
509     {\large \lineskip .5em%
510     \begin{tabular}[t]{c}%
511         \@homepage
512     \end{tabular}%
513     \par}%
514 \vskip 1em
515 \ifx\@repository\@empty\else
516     {\large \lineskip .5em%
517     \begin{tabular}[t]{c}%
518     VC: \@repository
519     \end{tabular}%
520     \par}%
521 \fi
522 \else
523 \@begintitlelinks
524 \@license
525 \@maketitlelink{CTAN}{\@ctan}%
526 \@maketitlelink{Texdoc}{\@texdoc}%
527 \@maketitlelink{Homepage}{\ydoc@eurl{https://github./
528     com/\@github}}%
529 \@maketitlelink{Repository}{\ydoc@eurl{https://github/
530     .com/\@github.git}}%
531 \@maketitlelink{Issue tracker}{\ydoc@eurl{https://
532     github.com/\@github/issues}}%
533 \@endtitlelinks
534 \fi
535 \end{center}%
536 \par\vskip 1em
537 \aftergroup\ydocpdfsettings
538 }
539 \ifpdf
540 \def\ydocpdfsettings{%
541     \hypersetup{%
542         pdfauthor = {\@author\space<\@plainemail>},
543         pdftitle = {\@title},

```



```

542         pdfsubject = {Documentation of LaTeX package/
                    \@package},
543         pdfkeywords = {\@package, LaTeX, TeX}
544     }%
545 }
546 \else
547 \let\ydocpdfsettings\empty
548 \fi
549
550 \let\orig@maketitle\maketitle
551 \def\maketitle{%
552     \ydocpdfsettings
553     \orig@maketitle
554     \let\orig@maketitle\relax
555 }

```

### 3.6 Description Macros and Environments

```

556 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
557 \ProvidesPackage{ydoc-desc}[%
558 %<!DATE>
559 %<!VERSION>
560 %<*DRIVER>
561     2099/01/01 develop
562 %</DRIVER>
563     ydoc package to describe macros, environments, /
        options etc.]
564 \IfFileExists{needspace.sty}{%
565     \RequirePackage{needspace}
566 }{%
567     \def\Needspace{\@ifstar\@gobble\@gobble}
568 }

```

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

```

569 \RequirePackage{shortvrb}

```

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

```

570 \RequirePackage{etoolbox}

```

#### 3.6.1 Color and style definitions

```

571 \RequirePackage{xcolor}

```

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

```

572 \expandafter\def\csmame\string\color@none\endcsmame{%
573     \xcolor@ {}{}{}{}
574 }

```

```

575 \definecolor{macrodesc}{rgb}{0,0.2,0.6}
576 \definecolor{keydesc}{rgb}{0,0.4,0.9}
577 \definecolor{macroimpl}{rgb}{0,0.1,0.3}
578 \definecolor{meta}{rgb}{0,0.25,0.75}
579 \definecolor{scriptcolor}{rgb}{0.2,0.6,0.2}
580 \definecolor{optioncolor}{rgb}{0.3,0.2,0}
581 \colorlet{optional}{black!65!white}
582 \colorlet{metaoptional}{optional!50!meta}
583 \providecolor{urlcolor}{named}{blue}
584 \providecolor{linkcolor}{named}{blue}
585 \providecolor{filecolor}{named}{blue}
586 \providecolor{citecolor}{named}{blue}
587 \providecolor{anchorcolor}{named}{blue}
588 \providecolor{menucolor}{named}{blue}
589 \providecolor{runcolor}{named}{blue}

591 \RequirePackage{hyperref}
592 \hypersetup{%
593     colorlinks=true,
594     pdfborder=0 0 0,
595     pdfborderstyle={},
596     urlcolor=urlcolor,
597     linkcolor=linkcolor,
598     filecolor=filecolor,
599     citecolor=citecolor,
600     anchorcolor=anchorcolor,
601     menucolor=menucolor,
602     runcolor=runcolor,
603 }

```

### 3.6.2 Text Formatting Macros

`\meta`

Prints *meta text*.

```

605 \newrobustcmd*\meta [1]{%
606     {\metastyle{%
607         \ensuremath\langle
608         #1\/%
609         \ensuremath\rangle
610     }}%
611 }

```

`\marg`

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

```

612 \newrobustcmd*{\marg}[1]{%
613   {\margstyle{%
614     {\ttfamily\braceleft}%
615     \meta{#1}%
616     {\ttfamily\braceright}%
617   }}%
618 }

```

### \oarg

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

```

619 \newrobustcmd*{\oarg}[1]{%
620   {\oargstyle{%
621     {\ttfamily[]}%
622     \meta{#1}%
623     {\ttfamily]}}%
624   }}%
625 }

```

### \parg

Sets style and adds parentheses.

```

626 \newrobustcmd*{\parg}[1]{%
627   {\pargstyle{%
628     {\ttfamily{}}%
629     \meta{#1}%
630     {\ttfamily)}}%
631   }}%
632 }

```

### \aarg

Sets style and adds angles.

```

633 \newrobustcmd*{\aarg}[1]{%
634   {\aargstyle{%
635     {\ttfamily<}}%
636     \meta{#1}%
637     {\ttfamily>}}%
638   }}%
639 }

```

### \sarg

Prints star with given style.

```

640 \newrobustcmd*{\sarg}{\sargstyle{*}}

```

`\pkg`

`\cls`

`\lib`

`\env`

`\opt`

`\file`

```
641 \newrobustcmd*\pkg [1]{{\pkgstyle{#1}}}
642 \newrobustcmd*\cls [1]{{\clsstyle{#1}}}
643 \newrobustcmd*\lib [1]{{\libstyle{#1}}}
644 \newrobustcmd*\env [1]{{\envstyle{#1}}}
645
646 \newrobustcmd*\opt{\@ifstar\ys@opt\y@opt}
647 \def\y@opt#1{{\optstyle{#1}}}
648 \def\ys@opt#1{{\optstyle{#1}}\optpar{#1}}
649 \newrobustcmd*\optpar [1]{\marginpar{\hbox to \
marginparwidth{\hss\y@opt{#1}}}}
650
651 \newrobustcmd*\file [1]{{\filestyle{#1}}}
652 \newcommand*\pkgstyle [1]{\texttt{\textcolor{pkg/
}{#1}}}
653 \newcommand*\clsstyle [1]{\texttt{\textcolor{cls/
}{#1}}}
654 \newcommand*\libstyle [1]{\texttt{\textcolor{lib/
}{#1}}}
655 \newcommand*\envstyle [1]{\texttt{\textcolor{env/
}{#1}}}
656 \newcommand*\optstyle [1]{\textsf{\textcolor{opt/
}{#1}}}
657 \newcommand*\filestyle [1]{\texttt{\textcolor{file/
}{#1}}}
658 \colorlet{cls}{none}
659 \colorlet{lib}{none}
660 \colorlet{env}{none}
661 \colorlet{file}{none}
662 \colorlet{pkg}{none}
663 \definecolor{opt}{rgb}{0.5,0.16666,0}
```

`\cs`

`\cmd`

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

`\Key`

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

### 3.6.3 Text Formatting Styles

`\macrodescstyle`

Style of described macro names.

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

`\macrodescstyle`

Style of described macro names.

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

`\macroargsstyle`

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

```
669 \def\macroargsstyle{\ttfamily}
```

`\envcodestyle`

Default style for code body content in described environments.

```
670 \def\envcodestyle{\ttfamily}
```

`\verbstyle`

Style for verbatim text inside macro argument list.

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

`\metastyle`

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

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

`\margstyle`

Style for `\marg`.

```
673 \def\margstyle{}
```

`\Optional`

`\optional`

`\optionalstyle`

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

`\optionalon`

`\optionaloff`

```
676 \def\optionalon{\protected\def\optional{\/  
optionalstyle}}  
677 \def\optionaloff{\let\optional\relax}  
678 \optionalon
```

`\oargstyle`

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

```
679 \def\oargstyle{\optional}
```

`\pargstyle`

Style for `\parg`.

```
680 \def\pargstyle{}
```

`\aargstyle`

Style for `\aarg`.

```
681 \def\aargstyle{}
```

`\sargstyle`

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

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

### 3.6.4 Dimension Registers

`\descindent`

```
683 \newdimen\descindent
684 \descindent=-\parindent
```

`\beforedescskip`

```
685 \newdimen\beforedescskip
686 \beforedescskip=\bigskipamount
```

`\afterdescskip`

```
687 \newdimen\afterdescskip
688 \afterdescskip=\medskipamount
```

`\descsep`

Set to 1em in tt font.

```
689 \newdimen\descsep
690 \begingroup
691 \ttfamily
692 \global\descsep=1em\relax
693 \endgroup
```

### 3.6.5 Macro Argument Reading Mechanism

`\read@Macro@arg`

Reads next token and calls second macro.

```
694 \def\read@Macro@arg{%
695   \futurelet\@let@token\handle@Macro@arg
696 }
```

`\AlsoMacro`

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

```
697 \newcommand*\AlsoMacro{%
698   \begingroup\makeatletter
699   \AlsoMacro@
700 }
701 \def\AlsoMacro@#1{%
702   \endgroup
703   %<*DEBUG>
704   %\typeout{DEBUG: Macro: \string#1}%
705   %</DEBUG>
706   \PrintMacroName{#1}%
707   \read@Macro@arg
708 }
```

`\ydoc@short@AlsoMacro`

Makes & an alias for \AlsoMacro.

```
709 \begingroup
710 \catcode'\|\active
711 \gdef\ydoc@short@AlsoMacro{%
712   \catcode'\|\active
713   \let|\AlsoMacro
714 }
715 \endgroup
```



### `\ydoc@macrocatcodes`

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

```
716 \def\ydoc@macrocatcodes{%
717   \ydoc@short@AlsoMacro
718   \@makeother\'%
719   \@makeother\!%
720   \@makeother\[%
721   \@makeother\]%
722   \@makeother\(%
723   \@makeother\)%
724 }
```

### `\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.

```
725 \def\handle@Macro@arg{%
726   \expandafter\let\expandafter\handler\csname /
       handle@Macro@token@\meaning\@let@token\endcsname
727   \ifx\handler\relax
728     \def\handler{\ifhmode\unskip\fi\end@Macro@args}%
729   %<*DEBUG>
730   % \typeout{DEBUG: Stopped at: \expandafter\meaning\
       csname @let@token\endcsname}%
731   % \typeout{}%
732   %\else
733   %\expandafter\ifx\csname @let@token\endcsname\
       AlsoMacro
734   % \typeout{DEBUG: TOKEN: \string\AlsoMacro}%
735   %\else
736   % \typeout{DEBUG: TOKEN: \expandafter\meaning\
       csname @let@token\endcsname}%
737   %\fi
738   %</DEBUG>
739   \fi
740   \handler
741 }
742 \def\define@Macro@handler{%
743   \begingroup
744   \ydoc@macrocatcodes
745   \define@Macro@handler@
746 }
747 \def\define@Macro@handler@#1{%
748   \endgroup
749   \@namedef{handle@Macro@token@\meaning#1}%
750 }
```

`\end@Macro@args`

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

```
751 \def\end@Macro@args{%
752     \y@egroup
753     \after@Macro@args
754 }
```

`\after@Macro@args`

Hook to add additional commands in certain situations.

```
755 \def\after@Macro@args{%
756 }
```

### Macro argument reading macros

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

`\read@Macro@marg`

```
757 \define@Macro@handler{\bgroup}{%
758     \begingroup
759     \afterassignment\read@Macro@marg@
760     \let\@let@token=%
761 }
762 \def\read@Macro@marg@{%
763     \bgroup
764     \margstyle{}%
765     \let\end@Macro@args\empty%
766     {\ttfamily\braceleft}%
767     \aftergroup\read@Macro@marg@@
768     \read@Macro@marg
769 }
770 \def\read@Macro@marg@@{%
771     {\ttfamily\braceright}%
772     \endgroup
773     \read@Macro@marg
774 }
```

`\read@Macro@oarg`

```
775 \define@Macro@handler{[]}{%
776     \begingroup
777     \let\read@Macro@oarg@end\read@Macro@oarg@@end
```

```

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

```

`\read@Macro@parg`

```

795     \define@Macro@handler{()}{%
796         \begingroup
797         \let\read@Macro@parg@end\read@Macro@parg@@end
798         \let\end@Macro@args\read@Macro@parg@end
799         \pargstyle{}%
800         {\ttfamily ()}%]
801         \read@Macro@arg
802     }
803     \define@Macro@handler{()}{%
804         \read@Macro@parg@end
805     }
806     \def\read@Macro@parg@@end#1){%
807         #1%
808         {\ttfamily )}%]
809         \endgroup
810         \read@Macro@arg
811     }

```

`\read@Macro@aarg`

```

812     \def\read@Macro@aarg<{%
813         \begingroup
814         \let\read@Macro@aarg@end\read@Macro@aarg@@end
815         \let\end@Macro@args\read@Macro@aarg@end
816         \aargstyle{}%
817         {\ttfamily <}%]
818         \read@Macro@arg

```

```

819 }
820 \define@Macro@handler{>}{%
821   \read@Macro@aarg@end
822 }
823 \def\read@Macro@aarg@@end#1>>{%
824   #1%
825   {\ttfamily>}%
826   \endgroup
827   \read@Macro@arg
828 }

```

`\read@Macro@angle`

```

829 \define@Macro@handler{<}<{%
830   \futurelet\@let@token\read@Macro@angle@
831 }

```

`\read@Macro@angle@`

```

832 \def\read@Macro@angle@{%
833   \ifx\@let@token<%
834     \expandafter\read@Macro@aarg
835   \else
836     \expandafter\read@Macro@meta
837   \fi
838 }

```

`\read@Macro@meta`

```

839 \def\read@Macro@meta#1>{%
840   \meta{#1}\read@Macro@arg
841 }

```

`\read@Macro@sarg`

```

842 \define@Macro@handler**{%
843   \sarg\read@Macro@arg
844 }

```

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

```

845 \define@Macro@handler{=}{%
846   =\read@Macro@arg
847 }

```

### `\read@Macro@verb`

Sets up verbatim mode calls second macro.

```
848 \define@Macro@handler{'}'{%
849   \begingroup
850   \let\do\@makeother
851   \dospecials
852   \@noligs
853   \@makeother\'%
854   \obeyspaces
855   \read@Macro@verb@
856 }
```

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

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

```
857 \begingroup
858 \@makeother\'%
859 \gdef\read@Macro@verb@#1' {%
860   \endgroup
861   \ifx\relax#1\relax
862     {\verbstyle{\string'}}%
863   \else
864     {%
865       \frenchspacing
866       \@noligs\verbstyle{#1}}%
867   \fi
868   \read@Macro@arg
869 }
870 \endgroup
```

### `\read@Macro@cmds`

Simply executes given code.

```
871 \define@Macro@handler!!#1!{%
872   #1\relax
873   \read@Macro@arg
874 }
```

### `\read@Macro@rmspace`

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

```

875 \define@Macro@handler{\@sptoken} {%
876   \read@Macro@arg
877 }

```

### \read@Macro@addtoken

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

```

878 \define@Macro@handler{~}#1{%
879   #1\read@Macro@arg
880 }
881 \AtBeginDocument{%
882 \define@Macro@handler{~}#1{%
883   #1\read@Macro@arg
884 }
885 }
886 \define@Macro@handler{\space}#1{%
887   #1\read@Macro@arg
888 }

```

## 3.6.6 Description Macros

### For Macros

### \DescribeMacro

```

889 \@ifundefined{DescribeMacro}{}{%
890   \PackageInfo{ydoc-desc}{Redefining \string\
891     DescribeMacro}{}%
892 }

```

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.

```

892 \def\DescribeMacro{%
893   \DescribeMacros
894   \let\DescribeMacros\y@egroup
895   \optionalon
896   \def\after@Macro@args{\endDescribeMacros}%
897   \begingroup\makeatletter
898   \Describe@Macro
899 }

```

### \DescribeScript

```
900 \def\DescribeScript#1{%
901   \DescribeMacros
902   \let\DescribeMacros\y@egroup
903   \optionalon
904   \def\after@Macro@args{\endDescribeMacros}%
905   \hbox\y@bgroup
906   \texttt{#1}%
907   \ydoc@macrocatcodes
908   \macroargsstyle
909   \read@Macro@arg~%
910 }
```

### \DescribeKey

```
911 \def\DescribeKey{%
912   \DescribeKeys
913   \let\DescribeKeys\y@egroup
914   \optionalon
915   \def\after@Macro@args{\endDescribeKeys}%
916   \begingroup\makeatletter
917   \Describe@Macro
918 }
```

### \Describe@Macro

```
919 \def\Describe@Macro#1{%
920   \endgroup
921   \edef\name{\expandafter\@gobble\string#1}%
922   \global\@namedef{href@desc@\name}{}%
923   \immediate\write\@mainaux{%
924     \global\noexpand\@namedef{href@desc@\name}{}%
925   }%
926   \hbox\y@bgroup
927   \@ifundefined{href@impl@\name}{{\hyperlink{impl:\name}}}%
928   {%
929     \hbox{\vbox to 0pt{\vss\hbox{\raisebox{4ex}{\hyperlink{desc:\name}}}}}%
930   \PrintMacroName{#1}}%
931   }%
932   \ydoc@macrocatcodes
933   \macroargsstyle
934   \read@Macro@arg
935 }
```

### `\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.

```
936 \newcommand*\MakeShortMacroArgs{%
937   \@ifstar
938     {\@MakeShortMacroArgs\Macro}%
939     {\@MakeShortMacroArgs\MacroArgs}%
940 }
941 \def\@MakeShortMacroArgs#1#2{%
942   \MakeShortVerb{#2}
943   \catcode'#2\active
944   \begingroup
945   \catcode'\~\active
946   \lccode'\~'#2\relax
947   \lowercase{\endgroup\gdef~{\bgroup\let~\egroup#1}}%
948 }
```

### `\DeleteShortMacroArgs`

```
949 \newcommand*\DeleteShortMacroArgs[1]{%
950   \DeleteShortVerb{#1}%
951 }
```

### `\Macro`

Simply uses the two macros below.

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

### `\@Macro`

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

```
953 \def\@Macro{%
954   \begingroup\makeatletter
955   \Describe@Macro
956 }
957 \define@Macro@handler\AlsoMacro{}
958 \define@Macro@handler\DescribeMacro{}
959 \define@Macro@handler\DescribeKey{}
960 \define@Macro@handler\DescribeScript{}
```



### `\MacroArgs`

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

```
961 \newcommand*\MacroArgs{%
962   \begingroup
963   \def\end@Macro@args{\endgroup\xspace}%
964   \ydoc@macrocatcodes
965   \macroargsstyle
966   %<*DEBUG>
967   %\typeout{}%
968   %\typeout{DEBUG: Start MacroArgs}%
969   %</DEBUG>
970   \read@Macro@arg
971 }
972 \RequirePackage{xspace}
```

### `\DescribeMacros`

```
973 \def\DescribeMacros{%
974   \begingroup
975   \let\Macro\@Macro
976   \parindent=0pt\relax
977   \setbox\descbox\vbox\y@bgroup
978 }
```

### `\endDescribeMacros`

```
979 \def\endDescribeMacros{%
980   \y@egroup
981   \PrintMacros
982   \endgroup
983 }
```

### `\DescribeKeys`

```
984 \def\DescribeKeys{%
985   \begingroup
986   \let\PrintMacroName\PrintKeyName
987   \let\Key\@Macro
988   \parindent=0pt\relax
989   \setbox\descbox\vbox\y@bgroup
990 }
```

`\endDescribeKeys`

```
991 \def\endDescribeKeys{%
992   \y@egroup
993   \PrintKeys
994   \endgroup
995 }
996 \def\PrintKeys{\PrintMacros}
```

`\DescribeMacrosTabcolsep`

```
997 \def\DescribeMacrosTabcolsep{\tabcolsep}
```

`\DescribeMacrosTab`

```
998 \def\DescribeMacrosTab{%
999   \DescribeMacros
1000   \hbox\y@bgroup
1001   \tabcolsep=\DescribeMacrosTabcolsep\relax
1002   \DescribeMacrosTab@
1003 }
1004 \def\DescribeMacrosTab@#1{\tabular{@{}#1@{}}}
```

`\endDescribeMacrosTab`

```
1005 \def\endDescribeMacrosTab{%
1006   \endtabular\y@egroup
1007   \endDescribeMacros
1008 }
```

### For Lengths

`\DescribeLength`

```
1009 \newcommand*\DescribeLength{%
1010   \begingroup
1011   \let\DescribeLength\Describe@Length
1012   \setbox\descbox\hbox\y@bgroup
1013   \tabular{@{}l@{\hspace{2em}}l@{}}%
1014   \Describe@Length
1015 }
```

### `\Describe@Length`

```
1016 \newcommand*\Describe@Length [2]{%
1017   \PrintLengthName{#1}&
1018   (Default: {\macroargsstyle#2\unskip})%
1019   \@ifnextchar\DescribeLength
1020     {\}%
1021     {%
1022       \endtabular
1023       \y@egroup
1024       \PrintLength
1025       \endgroup
1026     }%
1027 }
```

### For Environments

### `\DescribeEnv`

```
1028 \@ifundefined{DescribeEnv}{%
1029   \PackageInfo{ydoc-desc}{Redefining \string\
1030   DescribeEnv}{%
1031 }
1031 \let\DescribeEnv\relax
```

```
1032 \newcommand*\DescribeEnv [2] []{%
1033   \begingroup
1034   \def\DescribeEnv@name{#2}%
1035   \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.

```
1036   \ifx\@currenvir\DescribeEnv@string
1037     \def\after@Macro@args{%
1038       \let\after@Macro@args\empty
1039       \setbox\@tempboxa\hbox\y@bgroup
1040       \@ifnextchar\end{%
1041         {\DescribeEnv@newline}%
1042         #1%
1043       }%
1043     }
```

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

```
1044   \else
1045     \ifx\relax#1\relax
1046       \def\after@Macro@args{%
1047         \y@bgroup
```

```

1048         \endDescribeEnv
1049     }%
1050 \else
1051     \def\after@Macro@args{%
1052         \setbox\@tempboxa\hbox\y@bgroup
1053         \DescribeEnv@newline\MacroArgs#1%
1054         \endDescribeEnv
1055     }%
1056 \fi
1057 \fi

    Start \vbox and adds first line.

1058 \setbox\descbox\vbox\y@bgroup
1059 \envcodestyle
1060 \let\PrintEnv\PrintSubEnv
1061 \hbox\y@bgroup
1062 \PrintEnvName{\begin}{\DescribeEnv@name}%
1063 \ydoc@macrocatcodes
1064 \macroargsstyle
1065 \read@Macro@arg
1066 }

```

#### `\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.

```

1067 \newcommand*\DescribeEnv@newline [1] [0pt] {%
1068     \strut\y@egroup
1069     {\vskip#1}%
1070     \hbox\y@bgroup\strut
1071     \hspace*{\descsep}%
1072     \ignorespaces
1073 }%

```

#### `\DescribeEnv@string`

Holds the environment name for comparison.

```

1074 \def\DescribeEnv@string{DescribeEnv}

```

#### `\descbox`

Save box to store description content.

```

1075 \newbox\descbox

```

`\endDescribeEnv`

```
1076 \def\endDescribeEnv{%
1077   \y@egroup
1078   \begingroup
1079   \setbox\@tempboxa\lastbox
1080   \ifcase0%
1081     \ifdim\wd\@tempboxa>\descsep1\fi
1082     \ifdim\ht\@tempboxa>\ht\strutbox1\fi
1083     \ifdim\dp\@tempboxa>\dp\strutbox1\fi
1084   \else
1085     \box\@tempboxa
1086   \fi
1087   \endgroup
1088   \hbox\y@bgroup
1089     \PrintEnvName{\end}{\DescribeEnv@name}
1090   \y@egroup
1091   \y@egroup
1092   \PrintEnv
1093   \endgroup
1094 }
```

### 3.6.7 Print Macros

`\PrintMacroName`

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

```
1095 \def\PrintMacroName#1{%
1096   {\macrodescstyle{\strut
1097     \texttt{\char92}}%
1098     \escapechar\m@ne
1099     \string#1\strut}}%
1100 }
```

`\PrintKeyName`

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

```
1101 \def\PrintKeyName#1{%
1102   {\keydescstyle{\strut
1103     #1\strut}}%
1104 }
```

### `\PrintLengthName`

Formats length register name.

```
1105 \let\PrintLengthName\PrintMacroName
```

### `\PrintEnvName`

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

```
1106 \def\PrintEnvName#1#2{%
1107   \strut
1108   \string#1\braceleft
1109   {\macrodescstyle#2\strut}%
1110   \braceright
1111 }
```

### `\PrintMacros`

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

```
1112 \def\PrintMacros{%
1113   \par\vspace\beforedescskip
1114   \begingroup
1115   \sbox\@tempboxa{\descframe{\usebox{\descbox}}}%
1116   \Needspace*{\dimexpr\ht\@tempboxa+3\baselineskip\relax}%
1117   \par\noindent
1118   \ifdim\wd\@tempboxa>\dimexpr\linewidth-2\descindent\relax
1119     \makebox[\linewidth][c]{\usebox\@tempboxa}%
1120   \else
1121     \hspace*{\descindent}%
1122     \usebox\@tempboxa
1123   \fi
1124   \endgroup
1125   \par
1126   \vspace\afterdescskip
1127   \par\noindent
1128 }
1129 \def\descframe#1{%
1130   \fbox{\hspace*{\descsep}#1\hspace*{\descsep}}%
1131 }
```

### `\PrintLength`

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

```
1132 \let\PrintLength\PrintMacros
```

`\PrintEnv`

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

```
1133 \let\PrintEnv\PrintMacros
```

`\PrintSubEnv`

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

```
1134 \def\PrintSubEnv{%
1135   \hbox{\hbox{\usebox{\descbox}}}%
1136 }
```

### 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.

```
1137 {%
1138 \@firstofone{%
1139   \catcode'\=12
1140   \gdef\bslash
1141   }{\}
1142 }%}
```

`\percent`

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

```
1143 \begingroup
1144 \catcode'\%=12
1145 \gdef\percent{%}
1146 \endgroup
```

`\braceleft`

`\braceright`

Defines expandable left and right braces with catcode 12: ‘{<sub>12</sub>’ ‘}<sub>12</sub>’.

```
1147 \begingroup
1148 \catcode '\<=1
1149 \catcode '\>=2
1150 \catcode '\{=12
1151 \catcode '\}=12
1152 \gdef\braceleft <{>
1153 \gdef\braceright <}>
1154 \endgroup
```

### 3.6.9 Other Macros

`\y@bgroup`

`\y@egroup`

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

```
1155 \def\y@bgroup{\bgroup\color@setgroup}
1156 \def\y@egroup{\color@endgroup\egroup}
```

`\codeline`

```
1157 \newcommand*{\codeline}[1][c]{%
1158   \codelinebefore
1159   \hbox to \hsize\bgroup
1160   \ifx i#1\hspace*{\leftmargin}\else
1161     \ifx l#1\else\hss\fi
1162   \fi
1163   \let\xspace\relax
1164   \hbox\bgroup
1165   \aftergroup\codeline@end
1166   \aftergroup#1%
1167   \afterassignment\MacroArgs
1168   \let\@let@token=%
1169 }
1170 \def\codeline@end#1{%
1171   \ifx r#1\else\hss\fi
1172   \egroup
1173   \codelineafter
1174 }
1175 \newcommand*\codelinebefore{\par\smallskip\noindent}
1176 \newcommand*\codelineafter {\par\smallskip\noindent}
```



### codequote

```
1177 \newenvironment{codequote}{%
1178     \def\{\{\newline\relax\MacroArgs}%
1179     \par\smallskip\bgroup\leftskip=\leftmargin\
1180     \rightskip=\rightmargin\noindent\MacroArgs}
1181     {\par\egroup\smallskip\noindent\
1182     ignorespacesafterend}
```

### macroquote

```
1181 \newenvironment{macroquote}{%
1182     \def\{\{\newline\relax\Macro}%
1183     \par\smallskip\bgroup\leftskip=\leftmargin\
1184     \rightskip=\rightmargin\noindent\Macro}
1185     {\par\egroup\smallskip\noindent\
1186     ignorespacesafterend}
```

## 3.7 Include Code Examples

```
1185 \NeedsTeXFormat{LaTeX2e}[1999/12/01]
1186 \ProvidesPackage{ydoc-expl}[%
1187 %<!DATE>
1188 %<!VERSION>
1189 %<*DRIVER>
1190     2011/08/11 develop
1191 %</DRIVER>
1192     ydoc package to insert live examples of LaTeX /
1193     code]
1194
1193 \RequirePackage{listings}
1194 \lst@RequireAspects{writefile}
1195 \def\ydoc@exafile{\jobname.exe}
```

### examplecode

```
1196 \lstdefinestyle{examplecode}{%
1197     language=[latex]tex,
1198     basicstyle=\ttfamily,
1199     columns=fullflexible,
1200     numbers=left,
1201     firstnumber=1,
1202     numberstyle=\tiny\color{gray}\sffamily,
1203     numbersep=5pt,
1204     breaklines,prebreak={\mbox{\tiny$\swarrow$}},
1205     commentstyle=\color{black!60},
1206 }%
```

### exampleresult

```
1207 \lstdefinestyle{exampleresult}{%
1208     firstnumber=1,
1209     gobble=0,
1210     basicstyle=\ttfamily,
1211     columns=fullflexible,
1212     commentstyle=\color{black!60},
1213 }
```

### exampleextract

```
1214 \lstdefinestyle{exampleextract}{gobble=4}%

1215 \newbox\examplecodebox
1216 \newbox\exampleresultbox
```

### \BoxExample

```
1217 \def\BoxExample{%
1218     \setbox\examplecodebox\hbox{\color@setgroup
1219         \lstinputlisting[style=examplecode,style=/  
             thisexampleprint]%
1220         {\ydoc@exafile}%
1221     \unskip\color@endgroup}%
1222     \setbox\exampleresultbox\hbox{\color@setgroup
1223         \lstset{style=exampleresult}%
1224         \@input\ydoc@exafile\relax
1225     \unskip\color@endgroup}%
1226 }
```

### \PrintExample

```
1227 %< *DISABLED >
1228 \RequirePackage{showexpl}
1229 \def\PrintExample{%
1230     \begingroup
1231     \lstset{style=examplecode}%
1232     \MakePercentComment
1233     \LTXinputExample[varwidth]{\ydoc@exafile}%
1234     \endgroup
1235 }
1236 %< /DISABLED >
```

`\PrintExample`

```
1237 \def\PrintExample{%
1238   \begingroup
1239   \BoxExample
1240   \@tempdima=\textwidth
1241   \advance\@tempdima by -\wd\examplecodebox\relax
1242   \advance\@tempdima by -\wd\exampleresultbox\relax
1243   \advance\@tempdima by -15pt\relax
1244   \ifdim\@tempdima>\bigskipamount
1245     \hbox to \textwidth{%
1246       \null\hss
1247       \minipage[c]{\wd\exampleresultbox}\fbox{\usebox\
1248         exampleresultbox}\endminipage
1249       \hfill\hfill\hskip\bigskipamount\hskip15pt\hfill\
1250         \hfill
1251       \minipage[c]{\wd\examplecodebox}\usebox\
1252         examplecodebox\endminipage
1253       \hss\null
1254     }%
1255   \else
1256     \vbox{%
1257       \centerline{\fbox{\usebox\exampleresultbox}}%
1258       \vspace{\bigskipamount}%
1259       \centerline{\usebox\examplecodebox}%
1260     }%
1261   \fi
1262 \endgroup
1263 }
```

`examplecode`

```
1261 \lstnewenvironment{examplecode}[1][\]{%
1262   \lstdefinestyle{thisexampleprint}{#1}%
1263   \setbox\@tempboxa\hbox\bgroup
1264   \lstset{style=exampleextract,#1}%
1265   \lst@BeginWriteFile{\ydoc@exafile}%
1266 }
1267 {%
1268   \lst@EndWriteFile
1269   \egroup
1270   \begingroup
1271   \MakePercentComment
1272   \catcode'\^^M=5\relax
1273   \PrintExample
1274   \endgroup
1275 }
```

```
1276 \RequirePackage{float}
```

**example**

```
1277 \floatstyle{plain}
1278 \newfloat{example}{tbhp}{loe}
1279 \floatname{example}{\examplename}
1280 \def\examplename{Example}
```

**exampletable**

```
1281 \newenvironment{exampletable}{%
1282   \floatstyle{plaintop}%
1283   \restylefloat{example}%
1284   \example
1285 }{\endexample}

1286 \expandafter\ifx\csname ydocinclversion\endcsname\
1287   relax\else
1288   \endinput
1289 \fi

1290 \chardef\ydocinclversion=1
1291
1292 \newread\inFile
1293 \newread\subFile
1294 \newwrite\outFile
1295 \newif\ifContinue
1296 \newlinechar='^^J
1297
1298 \def\makeOther#1{\catcode '#1=12\relax}
1299
1300 \let\inLine\relax
1301 \let\lastLine\relax
1302
1303 \def\includefiles#1#2{%
1304   \begingroup
1305   \immediate\openin\inFile#1\relax
1306   \immediate\openout\outFile#2\relax
1307   \makeOther\@%
1308   \makeOther\ \makeOther\\\makeOther\$%
1309   \makeOther#\makeOther^\makeOther^^K%
1310   \makeOther\_ \makeOther^^A\makeOther%%
1311   \makeOther~\makeOther\{\makeOther\}\makeOther\&%
1312   \endlinechar-1\relax
1313   \Continuetrue
1314   \loop
1315     \let\lastLine\inLine
```

```

1316     \read\inFile to\inLine
1317     \ifeof\inFile
1318         \Continuefalse
1319     \else
1320         \expandafter\checkLine\inLine\empty\empty\
            empty\endLine
1321     \fi
1322     \ifContinue
1323 \repeat
1324 \immediate\closein\inFile
1325 \immediate\closeout\outFile
1326 \endgroup
1327 \end
1328 }
1329
1330 \def\copyline{%
1331     \immediate\write\outFile{\inLine}%
1332 }
1333
1334 \chardef\percentcharnum='\%
1335
1336 \begingroup
1337 \makeOther%\makeOther@\relax
1338 \gdef\SubFileOptionString{%<@}\relax
1339 \gdef\CommentChar{%}\relax
1340 \catcode'\|=0
1341 \makeOther\ \makeOther\|\relax
1342 |gdef|IfFalseString{% \iffalse}|relax
1343 |gdef|FiString{% \fi}|relax
1344 |endgroup
1345
1346 \def\checkLine#1#2#3#4\endLine{%
1347     \def\firstthree{#1#2#3}%
1348     \ifx\firstthree\SubFileOptionString
1349         \readSubFile#4\endLine
1350     \else
1351         \copyline
1352     \fi
1353 }
1354
1355 \def\readSubFile#1>#2\endLine{%
1356     \immediate\openin\subFile=#1\relax
1357     \ifeof\subFile
1358         % File not found
1359     \else
1360         \message{^^JIncluding subfile '#1'^^J}%
1361         \immediate\write\outFile{\CommentChar< *#1>}%
1362         \ifx\lastLine\IfFalseString
1363             \immediate\write\outFile{\FiString}%
1364         \fi

```

```

1365     \copySubFile
1366     \ifx\lastLine\IfFalseString
1367         \immediate\write\outFile{\IfFalseString}%
1368     \fi
1369     \immediate\write\outFile{\CommentChar</#1>}%
1370 \fi
1371 \immediate\closein\subFile
1372 }
1373
1374 \def\copySubFile{%
1375     \read\subFile to\subLine
1376     \ifeof\subFile\else
1377         \immediate\write\outFile{\subLine}%
1378         \expandafter\copySubFile
1379     \fi
1380 }

```