The LaTeX dtxdescribe Package

v1.02 — 2019/07/16

© 2016–2018 Brian Dunn
bd@BDTechConcepts.com

Describe additional object types in `dtx` source files.

Abstract

The doc package includes tools for describing macros and environments in LaTeX source `.dtx` format. The dtxdescribe package adds additional tools for describing booleans, lengths, counters, keys, packages, classes, options, files, commands, arguments, and other objects. dtxdescribe also works with the regular document classes, for those who do not wish to use the ltxdoc class and `.dtx` files.

Each described item is given a margin tag similar to `\DescribeEnv`, and is listed in the index by itself and also by category. Each item may be sorted further by an optional class. All index entries except code lines are hyperlinked.

The `dtxexample` environment is provided for typesetting example code and its results. Contents are displayed verbatim along with a caption and cross-referencing. They are then `\input` and executed, and the result is shown.

Environments are also provided for displaying verbatim or formatted source code, user-interface displays, and sidebars with titles.

Macros are provided for formatting the names of inline LaTeX objects such as packages and booleans, as well as program and file names, file types, internet objects, the names of certain programs, a number of logos, and inline dashes and slashes.

dtxdescribe works with the ltxdoc class, but also works with the standard classes as well, except that the `macro` and `environment` environments are not supported. Either makeidx or splitidx may be loaded by the user. makeidx will be used by default.

Contents

List of Examples

List of Figures

1 Introduction

The doc package provides `\DescribeMacro` and `\DescribeEnv` to help document new macros and environments. Each generates a heading in the documentation, to which `\marg`, `\oarg`, and `\parg` may be added to identify arguments to be passed to the new object. Their names are added to the margin, and index entries are added, as well as group of entries for environments.

dtxdescribe extends this concept to include a number of additional objects, such as booleans and keys. To help identify what is being described in the margin, small tags are added to the name, such as “Env”, “Bool”, or “Key”. These new objects are also listed in the index with the same tag shown after their names, and also by group. Optional classes may be used to further categories index entries.

Modifications have been made to interact with hyperref to provide hyper links for regular index entries as well as the new `\Describe` entries.

Additional macros are provided to generate colored margin tags and warnings, and a new `dtxexample` environment demonstrates code examples.

This documentation and its index show examples of these macros in use.

While the index may appear to be overkill for a small package, keep in mind that it includes a number of fictional entries from the examples. Extensive cross-referencing can be useful for larger works. And, of course, you need not cross-reference everything!

2 Using dtxdescribe

Place `\usepackage{dtxdescribe}` in the `.dtx` file's driver section:

```
%<*driver>
\documentclass{ltxdoc}
    ...
\usepackage{lmodern}
    ...
\usepackage{dtxdescribe}
    ...
\usepackage{packagename} % the name of your new package
    ...
\usepackage[...]{hyperref}
\usepackage[...]{cleveref}
    ...
%</driver>
```

Various objects inside the `dtx` file may be described with `\DescribeBoolean`, `\DescribeLength`, `\DescribeCounter`, and related macros, similar to the already-familiar `\DescribeMacro` and `\DescribeEnv`.

Optional "classes" may be assigned to the objects being described, including the new versions of `\DescribeMacro` and `\DescribeEnv`. These classes are printed in the margin tag and index entry for each item, and also generate additional index entries sorted by class. This is especially useful for key/value sets, where several sets may appear in the same document.

inside a float

The margin tag is not printed if the `\Describe` macros are used inside a float such as a table, but the index entries are still made.

`\margintag{text}`

`\margintag{text}` may be used to place a colored tag in the margin to summarize paragraph contents or draw attention to an index destination.

⚠ `\watchout[text]`

`\watchout[optional text]` may be used to place a red warning sign in the margin, along with optional text.

The `dtxexample` environment may be used to typeset and execute small pieces of LaTeX code as examples of its use. Optional cross-referencing notes may be used to refer to any example float being generated.

3 The macros, and the `dtxexample` environment

3.1 Macros and environments

Env macro
Env environment
⚠ .dtx only

These are only provided by the ltxdoc class and doc package to document a `.dtx` file, where comments are used by *docstrip* to disable these environments in the resulting `.sty` file. When using the regular document classes, the `macro` and `environment` environments would localize any definitions, and `\DescribeMacro` and `\DescribeEnv` should be used instead.

`\DescribeMacro` `[`⟨*class*⟩`]` `{`⟨`\name`⟩`}`

The preexisting macro from the doc package is redefined to create hyperlinked index entries, and include an optional class. A margin tag is created and an index entry is made. When the optional class is used, it is displayed in front of the margin tag, and is used to group an index entry by macro name and another index entry by class. An example would be to describe the float creation and caption setup for a new class of float, such as the `dtxexample` float and the example "photograph" float both found in the index for this document. See example 1 on page 18 for examples.

`\DescribeEnv` `[`⟨*class*⟩`]` `{`⟨*environment name*⟩`}`

The preexisting macro from the doc package is redefined to create hyperlinked index entries, include an optional class, and also to place an 'Env' tag in front of the name in the margin. See example 2 on page 19.

3.2 Arguments

The `\Describe...` macros may be followed by `\marg`, `\oarg`, and `\parg` to describe arguments passed to the macros.

`\marg` `{`⟨*text*⟩`}`

Shows a mandatory argument for a macro or environment.

The results looks like `{`⟨*mandatory*⟩`}`.

`\oarg` `{`⟨*text*⟩`}`

Shows an optional argument for a macro or environment.

The results looks like `[`⟨*optional*⟩`]`.

`\parg` `{`⟨*text*⟩`}`

Used for "picture" arguments, such as coordinates.

The result looks like (⟨*coordinate*⟩).

`\DescribeArgument` `[`⟨*class*⟩`]` `{`⟨*argument*⟩`}`

May be used to describe actions taken when given certain macro arguments. These will be given an 'Arg' margin tag and will appear in the index. The `class` may be used to categorize arguments by their macro or environment name. See example 9 on page 23.

3.3 Booleans, lengths, counters, keys

See example 4 on page 20.

`\DescribeBoolean` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a boolean. Given a 'Bool' tag in the margin and index.

`\DescribeLength` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a length. Given a 'Len' tag in the margin and index.

`\DescribeCounter` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a counter. Given a 'Ctr' tag in the margin and index.

`\DescribeKey` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a key. Given a 'Key' tag in the margin and index. The `class` may be used to categorize keys by their kev/value group. See example 8 on page 22.

3.4 Packages, classes, options

`\DescribePackage` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a package. Given a 'Pkg' tag in the margin and index.

`\DescribeClass` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a LaTeX class. Given a 'Cls' tag in the margin and index.

`\DescribeOption` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes a LaTeX package or class option. Given an 'Opt' tag in the margin and index.

3.5 Files, programs, commands

`\DescribeFile` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes an operating-system file. Given a 'File' tag in the margin and index. The filename may have underscores.

`\DescribeProgram` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes an operating-system program. Given a 'Prog' tag in the margin and index. The program name may have underscores.

`\DescribeCommand` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes an operating-system command. Given a 'Cmd' tag in the margin and index. The command name may have underscores.

3.6 Other source objects

`\DescribeObject` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes an arbitrary programming object, such as a color definition or caption setup. A margin tag and index entry are created with `\ttfamily` type. When a class is used, it is pre-pended to the margin tag, appended to the index entry, and a second index entry is created grouped by class. If a macro name is to be described, use `\DescribeMacro` instead. See example 10 on page 24.

`\DescribeOther` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

Describes an arbitrary non-programming object, such as a license agreement or credits. A margin tag and index entry are created in roman type. When a class is used, it is pre-pended to the margin tag, appended to the index entry, and a second index entry is created grouped by class. See example 11 on page 24.

3.7 In a `description` environment

To describe an object using a `description` environment, use the following. See example 12 on page 25.

`\ItemDescribeMacro` `[`⟨*class*⟩`]` `{`⟨`\name`⟩`}` A description.

`\ItemDescribeEnv` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}` A description.

`\ItemDescribeArgument` `[`⟨*class*⟩`]` `{`⟨*argument*⟩`}` A description.

`\ItemDescribeBoolean` `[⟨class⟩]` `{⟨name⟩}` A description.

`\ItemDescribeLength` `[⟨class⟩]` `{⟨\name⟩}` A description.

`\ItemDescribeCounter` `[⟨class⟩]` `{⟨name⟩}` A description.

`\ItemDescribeKey` `[⟨class⟩]` `{⟨name⟩}` A description.

`\ItemDescribePackage` `[⟨class⟩]` `{⟨package_name⟩}` With underscores.

`\ItemDescribeClass` `[⟨class⟩]` `{⟨class_name⟩}` With underscores.

`\ItemDescribeOption` `[⟨class⟩]` `{⟨name⟩}` A description.

`\ItemDescribeFile` `[⟨class⟩]` `{⟨file_name⟩}` With underscores.

`\ItemDescribeProgram` `[⟨class⟩]` `{⟨program_name⟩}` With underscores.

`\ItemDescribeCommand` `[⟨class⟩]` `{⟨command_name⟩}` With underscores.

`\ItemDescribeObject` `[⟨class⟩]` `{⟨name⟩}` A description.

`\ItemDescribeOther` `[⟨class⟩]` `{⟨name⟩}` A description.

3.8 Defaults

`\DescribeDefault` `{⟨value⟩}`
Default: `value`

Shows the default value of a `\Describe...` item, such as displayed here. Place this macro immediately after the `\Describe...` macro and any arguments, but before the text description.

`\DescribeDefaultcolor`
Default: `green!50!black`

The color of the margin tag used to show the default value. This is used by `\textcolor` to create the margin tag.

3.9 `\margintag`, `\watchout`

`\margintag` `{⟨text⟩}`

`\margintag{example}`

Creates a colored margin tag. May be used to identify the topic of a paragraph or the destination of an arbitrary index entry.

`\margintagcolor`
Default: `blue!70!black`

The color of the `\margintag`.

`\watchout` `[⟨text⟩]`

⚠ `\watchout[example]` Creates a red margin tag with a warning sign and optional text. May be used to warn the reader of special instructions, etc. Without the optional text the warning sign is displayed by itself.

`\watchoutcolor` The color of the `\watchout`.
Default: `red!50!black`

3.10 `dtxexample` environment

Env `dtxexample` * `[`⟨*Notes/cross-references*⟩`]` `{`⟨*caption & label*⟩`}`

The `dtxexample` environment is useful for demonstrating a piece of LaTeX code. The example is a simulated float with its own caption and optional label, along with optional notes and/or cross-referencing commands. The contents of the `dtxexample` environment are printed verbatim, then loaded and executed as LaTeX code, showing the results just below the printed code. In the case of float commands, the floats are generated as expected somewhere nearby, and should be given their own labels. References to the float's labels may be placed in the optional argument to the `dtxexample` environment, and will be printed below the code.

The unstarred version places the code inside a minipage, forbidding a page break in the middle of the code listing. The starred version does not use a minipage. This is required when the code is too large to fit on a single page.

See example 13 for a demonstration of how `dtxexample` works.

`\dtxexamplecodename` The text name of the code section.
Default: `Code:`

`\dtxexampleresultname` The text name of the result section.
Default: `Result:`

3.11 `noindmacro` and `noindenvironment` environments

⚠ `.dtx` only These only make sense if using the ltxdoc class and doc package to document a `.dtx` file, where comments are used by *docstrip* to disable these environments in the resulting `.sty` file. When using the regular document classes, the `noindmacro` and `noindenvironment` environments would localize any definitions, and `\DescribeMacro` and `\DescribeEnv` should be used instead.

Env `noindmacro` `{`⟨`\name`⟩`}` To document macros which should not be included in the index.

Env `noindenvironment` `{`⟨*name*⟩`}` To document environments which should not be included in the index.

Replace

```
\begin{macro}{\macroname} \oarg{optional} \marg{mandatory}
...
```

```
\end{macro}
```

with

```
\begin{noindmacro}{\macroname} \oarg{optional} \marg{mandatory}
...
\end{noindmacro}
```

and similarly for `noindenvironment`.

3.12 sourceverb, sourcedisplay, UIdisplay, docsidebar

Env sourceverb — Default: gobble=2, tabsize=4, xleftmargin=2em

`[`⟨*key/values*⟩`]` Display source code verbatim. Uses optional fancyvrb keys. Includes `gobble=2` to absorb the leading `%` and space character of a `dtx` file source format. Because this is a verbatim environment, it *cannot* be used inside a macro.

Env fsourceverb — Default: gobble=2, tabsize=4, xleftmargin=2em,frame=lines

`[`⟨*key/values*⟩`]` Display source code verbatim inside a frame. A label may be included using the `label` key. Because this is a verbatim environment, it *cannot* be used inside a macro. See example 14 on page page 27.

Env sourcedisplay — Display source code with manual formatting. This is not a verbatim environment. `\textcolor`, `\textbf`, and `\emph` may be used to highlight text. Macros must be escaped with `\cs`, characters such as { must be produced with `\{`, etc. `\\` must be used to force a new line. `\fquad`, `\fqquad`, and `\fqqquad` may be used to force indenting. Because this is *not* a verbatim environment, it *can* be used inside a macro. See example 15 on page 27.

`\fquad` Single-level indent inside a `sourcedisplay`.

`\fqquad` Double-level indent inside a `sourcedisplay`.

`\fqqquad` Triple-level indent inside a `sourcedisplay`.

Env UIdisplay — Displays a user interface, such as a dialog box entry or a menu selection. See example 16 on page 28. Also see the `\UI` macro..

`\userentry` `{`⟨*text to enter*⟩`}` Typeset something for the user to enter. Also see the `\cmds` macro.

`\userentryname` — Default: Enter ⇒ Text to tell the user to enter the following item. Change with `\renewcommand`.

Env docsidebar — `[`⟨*title*⟩`]` Creates a sidebar within the document. See example 17 on page 29.

3.13 Formatted objects

Macros to format references to various kinds of objects.

This dtxdescribe package documentation uses erewhon, roboto, and inconsolata, along with metalogo, to demonstrate the following font effects.

3.13.1 LaTeX objects

\pkg packagename, also for a classname

\env environment

\ctr counter

\bool boolean

\optn option: to a macro, package, class

\TOC TOC: Table of contents.

\LOF LOF: List of figures.

\LOT LOT: List of tables.

3.13.2 Programs and commands

\progcode inline program code: Escape underscores and other special characters such as {, %, $.

\prog *grep*, *make*: A program name. Underscores allowed.

\filenm file_name: Underscores allowed.

\UI **General user-interface text.** What the user sees on the display. Also see the UIdisplay environment.

\cmds **commands to be entered**: What the user enters. Escape underscores and other special characters such as {, %, $. Also see the \userentry macro.

3.13.3 File types

\ODT ODT OpenDocument Format word processing document

\SVG SVG image format

\PNG PNG image format

\GIF GIF image format

\JPG JPG image format

\EPS EPS image format

\PDF PDF image format

\DVI DVI image format

3.13.4 Internet

\UTF UTF: Unicode

\URL URL: Uniform Resource Locator

\element <element>: HTML / CSS element

\attribute attribute: HTML / CSS attribute

\HTML HTML: Hypertext Markup Language

\HTMLfive HTML5: Old-style figure if font supports

\CSS CSS: Cascading Style Sheet

\CSSthree CSS3: Old-style figure if font supports

\EPUB EPUB: E-book file format

3.13.5 Specific programs

\tikz Ti*k*z: Package logo

\MathML MathML: Mathematical Markup Language

\CTAN CTAN: Comprehensive TEX Archive Network

\TDS TDS: TEX Directory Structure

3.13.6 Acronyms, brand names, trademarks

\brand Brandname, Company Name

`\acro` ACRO: Acronym

`\supregistered` Superscript trademark symbol®

3.14 Logos

Several additional logos are provided:

`\LuaTeX` LuaTEX

`\LuaLaTeX` LuaLATEX

`\XeTeX` XƎTEX, with reversed E if graphics is loaded.

`\XeLaTeX` XƎLATEX, with reversed E if graphics is loaded.

`\AmS` $\mathcal{AMS}$

`\LyX` LyX

`\BibTeX` BIBTEX

`\MakeIndex` *MakeIndex*

`\ConTeXt` ConTEXt

`\MiKTeX` MiKTEX

3.15 Dashes and slashes

`\thinskip` A breakable thin skip.

`\endash` An endash: –

`\emdash` An emdash: —

`\thinbrspace` A thin space which allows a line break.

`\thinthinbrspace` A very thin space which allows a line break.

`\Dash` An unbreakeable thin space, emdash, and breakable thin space: A — B

`\dash` An unbreakeable thin space, endash, and breakable thin space: A – B

`\Slash` An unbreakable very thin space, a slash, and a breakable very thin space:

Command	Result	
`A--B`	A–B	(not breakable)
`A \dash B`	A – B	(only breakable before the B)
`A -- B`	A – B	(breakable before or after the dash)
`A---B`	A—B	(not breakable)
`A \Dash B`	A — B	(only breakable before the B)
`A --- B`	A — B	(breakable before or after the dash)
`A/B`	A/B	(not breakable)
`A \Slash B`	A/B	(only breakable before the B)
`A / B`	A / B	(breakable before or after the slash)
`A~/~B`	A / B	(not breakable)

4 Examples

Example 1: Macros

Code:

```
\DescribeMacro{\mymacro} \oarg{optional} \marg{mandatory}
    A typical macro definition.

\DescribeMacro[photograph]{\DeclareFloatingEnvironment}
Create a photograph float. \bigskip

\DescribeMacro[photograph]{\captionsetup}
Caption settings for a photograph float.

\DescribeMacro[photograph]{\cnameref}
\pkg{cleveref} name for the photograph float.
```

Result:

\mymacro `[`⟨*optional*⟩`]` `{`⟨*mandatory*⟩`}` A typical macro definition.

[photograph] \DeclareFloatingEnvironment Create a photograph float.

[photograph] \captionsetup Caption settings for a photograph float.

[photograph] \cnameref cleveref name for the photograph float.

The optional class is used to label and group tags and index entries. See this document's index entries for examples of this "photograph" class and the `dtxexample` class of macros.

hyperlinks The re-defined `\DescribeMacro`, `\DescribeEnv`, and all the following macros create hyperlinked index entries, along with regular uses of `\index`.

Example 2: Environment

Code:

```
\DescribeEnv{myenvironment} \marg{argument} Short description.
```

Result:

Env myenvironment {⟨*argument*⟩} Short description.

add'l tags

index groups

The re-defined `\DescribeEnv` adds an 'Env' tag to the margin, and adds "(environment)" to its own index entry. Note that environments and all the other new objects defined by this package each receives two index entries, one by name, and one grouped with others of its kind.

⚠ too much text

Example 2 shows descriptive text on the same line as the `\DescribeEnvironment`. For macros and environments with many arguments after the name, it may be better to place any additional text in a following paragraph.

Example 3: Second Environment

Code:

```
\DescribeEnv[kindofenvironment]{otherenvironment}
    \oarg{opt args} \parg{coordinates} A description.
```

Result:

Env [kindofenvironment] otherenvironment [⟨*opt args*⟩] (⟨*coordinates*⟩) A description.

The `otherenvironment` will be indexed by itself and also with `myenvironment` under the index entry "environments", and also under the class `kindofenvironment`.

Example 4: Booleans and Counters

Code:

```
\DescribeBoolean[examples]{sampleboolean} Some description.

\DescribeCounter[examples]{samplecounter} Some description.
```

Result:

Bool [examples] `sampleboolean` Some description.

Ctr [examples] `samplecounter` Some description.

Most of the new `\Describe____` macros behave like the new `\DescribeEnv`, placing a tag in the margin, an index entry by name, and another index entry by group.

Example 5: Lengths

Code:

```
\DescribeLength[photograph]{\photowidth} Some description.
```

Result:

Len [photograph] `\photowidth` Some description.

Lengths have a leading backslash, but are otherwise described the same as the rest of the objects.

Example 6: Packages, Classes, and Options

Code:

```
\DescribePackage[examples]{samplepackage}
    About a \LaTeX\ package.

\DescribeClass[examples]{sample_class}
    About a \LaTeX\ class.

\DescribeOption[examples]{sampleoption}
    About an option for a package or class.
```

Result:

Pkg [examples] samplepackage — About a LaTeX package.

Cls [examples] sample_class — About a LaTeX class.

Opt [examples] sampleoption — About an option for a package or class.

Example 7: Files, Commands, and Programs

Code:

```
\DescribeFile[bigfiles]{really_big_file.txt} Some description.

\DescribeFile[bigfiles]{another_big_file.txt} Some description.

\DescribeFile{lone_file.txt} Some description.

\DescribeCommand{OS_command} An operating-system command.

\DescribeProgram{program_name} An operating-system program.
```

Result:

File [bigfiles] really_big_file.txt — Some description.

File [bigfiles] another_big_file.txt — Some description.

File lone_file.txt — Some description.

Cmd OS_command — An operating-system command.

Prog program_name — An operating-system program.

Filenames, program names, and command names may have underscores, such as tested here. A class is used to group “bigfiles” together in the index.

Example 8: Keys

Code:

```
\DescribeKey[groupofkeys]{firstkey} About the first key
    of the |groupofkeys| set.

\DescribeKey[groupofkeys]{secondkey} About the second key
    of |groupofkeys|.

\DescribeKey[examples]{samplekey} About some key of |otherkeys|.

\DescribeKey[examples]{sampletwokey} About another key of |otherkeys|.

\DescribeKey{lonekey} A key without a class.
```

Result:

Key [groupofkeys] `firstkey` About the first key of the `groupofkeys` set.

Key [groupofkeys] `secondkey` About the second key of `groupofkeys`.

Key [examples] `samplekey` About some key of `otherkeys`.

Key [examples] `sampletwokey` About another key of `otherkeys`.

Key `lonekey` A key without a class.

See the index key groups.

Example 9: Arguments

Code:

```
\DescribeArgument[figure]{[H]}
What happens when a figure is [H]ere.

\DescribeArgument[figure]{[M]}
What happens when a figure is in the [M]argin.

\DescribeArgument[\cs{mymacro}]{bold}
What happens when \cs{mymacro} is given the |bold| argument.
```

Result:

Arg [figure] `[H]` What happens when a figure is [H]ere.

Arg [figure] `[M]` What happens when a figure is in the [M]argin.

Arg [`\mymacro`] `bold` What happens when `\mymacro` is given the `bold` argument.

Arguments behave like keys, and may have an optional class to identify their macro or environment, and group their entries in the index.

⚠ macro names Note the need to use `\cs{mymacro}` for the macro's name.

Example 10: Object

Code:

```
\DescribeObject[color]{somecolor}
    The color of something.

\DescribeObject[color]{othercolor}
    The other color.

\DescribeObject{randomobject} About some random object.
```

Result:

[color] somecolor — The color of something.

[color] othercolor — The other color.

randomobject — About some random object.

Describes an arbitrary programming object, using `\ttfamily` text.

Example 11: Other

Code:

```
\DescribeOther{license agreement}
The following is the fictional license agreement:

\DescribeOther{Before \env{myenvironment}}
    Actions to be done \cs{BeforeBeginEnvironment}.

\DescribeOther[otherclass]{Other Item} About the other item.

\DescribeOther[otherclass]{Additional Item} About the add'l item.
```

Result:

license agreement — The following is the fictional license agreement:

Before myenvironment — Actions to be done `\BeforeBeginEnvironment`.

[otherclass] Other Item — About the other item.

[otherclass] Additional Item — About the add'l item.

Describes an arbitrary non-programming object, using roman text.

Example 12: Description environments

Code:

```
\begin{description}
\ItemDescribeMacro[descexamples]{\macroname} Describe the macro.
\ItemDescribeBoolean[descexamples]{booleanname} Describe the boolean.
\ItemDescribeLength[descexamples]{\lengthname} Describe the length.
\ItemDescribeKey[descexamples]{keyname} Describe the key.
\ItemDescribePackage[descexamples]{package_name} Describe the package.
\ItemDescribeClass[descexamples]{class_name} Describe the class.
\ItemDescribeFile[descexamples]{file_name} Describe the file.
\ItemDescribeProgram[descexamples]{program_name} Describe the program.
\ItemDescribeCommand[descexamples]{command_name} Describe the class.
\end{description}
```

Result:

[descexamples] `\macroname` **`\macroname:`** Describe the macro.

Bool [descexamples] `booleanname` **`booleanname:`** Describe the boolean.

Len [descexamples] `\lengthname` **`\lengthname:`** Describe the length.

Key [descexamples] `keyname` **`keyname:`** Describe the key.

Pkg [descexamples] `package_name` **`package_name:`** Describe the package.

Cls [descexamples] `class_name` **`class_name:`** Describe the class.

File [descexamples] `file_name` **`file_name:`** Describe the file.

Prog [descexamples] `program_name` **`program_name:`** Describe the program.

Cmd [descexamples] `command_name` **`command_name:`** Describe the class.

Uses a `description` environment to describe objects.

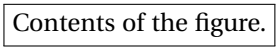

Figure 1: A Figure

Example 13: `dtxexample`

Code:

```
\begin{figure}
    \centering\fbox{Contents of the figure.}
    \caption{A Figure}\label{fig:afigure}
\end{figure}
```

Result:

See fig. 1

Example 13, typeset above, was created with the following code:

```
\begin{dtxexample}[See \cref{fig:afigure}]
    {\env{dtxexample}\label{ex:dtxexample}}
\begin{figure}
    \centering\fbox{Contents of the figure.}
    \caption{A Figure}\label{fig:afigure}
\end{figure}
\end{dtxexample}
```

When the example was created:

1. The “float” of type `example` was created, with the caption “`dtxexample`” and the label `ex:dtxexample`, which points to example 13.
2. The code was displayed verbatim.
3. The code was written to the file `dtxexample_cut.tex`.
4. The code was `\input` from `dtxexample_cut.tex`.
5. Executing the code created the figure with caption “A Figure” and label `fig:afigure`, which points to fig. 1.
6. The cross-reference to the figure was shown on the optional display line by the optional argument to `dtxexample`.
7. The starred form of `dtxexample` was used to create the closing rule below the code, since a float was being generated and nothing followed the code inline. An unstarred version would have created an extra rule.

Example 14: `fsourceverb`

Code:

```
% \begin{fsourceverb}[label=An fsourceverb example]
% \newcommand{fdosomething}[1][whattodo]{
%    doing #1
% }
% \end{fsourceverb}
```

Result:

An fsourceverb example
```
\newcommand{fdosomething}[1][whattodo]{
   doing #1
}
```

(The leading `%` characters would be present in the `dtx` source.)

Example 15: `sourcedisplay`

Code:

```
\begin{sourcedisplay}
\cs{newcommand}\{dosomething\}[1][\textcolor{red}{whattodo}]\{\\
\fquad \textcolor{blue}{doing \textcolor{red}{\#1}}\\
\}
\end{sourcedisplay}
```

Result:

```
\newcommand{dosomething}[1][whattodo]{
  doing #1
}
```

Example 16: `UIdisplay`

Code:

```
Select:
\begin{UIdisplay}
  \textsf{Preferences $\to$ Plugins $\to$ Files $\to$ HTML}
\end{UIdisplay}
For the field
\begin{UIdisplay}
Title heading:
\end{UIdisplay}
\userentry{H1}
```

Result:

Select:

Preferences → **Plugins** → **Files** → **HTML**

For the field

Title heading:

Enter ⇒ **H1**

Example 17: `docsidebar`

Code:

```
Main text.

More main text.

\begin{docsidebar}[A title]
An aside, which may help explain something
incidental to the main text.
\end{docsidebar}

Additional main text.
```

Result:

Main text.

More main text.

> *A title*
>
> An aside, which may help explain something incidental to the main text.

Additional main text.

5 Usage notes

Placement of `\Describe` macros: Typically LaTeX macro and environment definitions are enclosed in `macro` and `environment` environments at their place in the source code. `\DescribeMacro` and `\DescribeEnv` would be used elsewhere in the manual to describe how to use the code. `\DescribeBoolean` and such might be at their place in the source code, unless they are worthy of discussion for the end-user, in which case they should be in the "User's Manual" section of the document.[1] It may be useful to use `\DeclareBoolean` and friends both at the code location and also in the User's Manual section.

Extra spaces: When placing multiple `\Describe`, `\index`, `\margintag`, and `\watchout` macros together, care must be taken to avoid extra space in the printed text where these macros occur. A trailing percent character may be used to avoid the extra space:

```
text text text% <-- avoids extra space
\margintag{A comment.}
\index{An entry}
\index{Another entry}
more inline text
```

Unwanted vertical space: Other environments nested inside a `docsidebar` may produce excessive vertical space. It may be required to insert

```
\vspace*{-\baselineskip}
```

`\margintag` placement: To have the margin tag appear next to the first line of a paragraph, place the `\margintag` or `\watchout` somewhere after the first few words in the paragraph. The `\margintag` may be on its own line, and the rest of the paragraph may follow on the next line. If too many words are printed before the `\margintag`, the words may wrap to the next line before the tag occurs.

Margin tag overlap: To keep margin tags in proper alignment, use a new paragraph or multiple lines between `\margintag`, `\watchout`, or `\Declare` macros

missing tags

`\Describe` inside floats: When these macros are used inside a float, the margin tag is supressed (there is no margin in a float), but the index entries are still created.

[1] Future versions may include `\DeclareBoolean` for use at the point where the boolean is defined, creating an index entry with a code line number, and `\DescribeBoolean` with a page number index entry for the related discussion in the User's Manual portion of the document.

6 Code

6.1 Required packages

Pkg makeidx
Pkg splitidx

One of several index programs must be provided. One of several index programs must be provided.

```
\AtBeginDocument{
    \@ifpackageloaded{makeidx}{}{
    \@ifpackageloaded{splitidx}{}{
        \RequirePackage{makeidx}
        \makeindex
    }}
}
```

Pkg etoolbox

v2.6 or later for `\BeforeBeginEnvironment`, `\AfterEndEnvironment`

```
\RequirePackage{etoolbox}[2011/01/03]%
```

Pkg xparse

Used for the examples.

```
\RequirePackage{xparse}
```

Pkg xifthen

Used for the examples.

```
\RequirePackage{xifthen}
```

Pkg xcolor

Used for the examples.

```
\RequirePackage{xcolor}
\definecolor{myurlcolor}{rgb}{0,0,.7}
\definecolor{mylinkcolor}{rgb}{.7,0,0}
```

Pkg caption

Used for the examples.

```
\RequirePackage{caption}
```

Pkg newfloat

Used for the examples.

```
\RequirePackage{newfloat}
```

Pkg fancyvrb

Used for the examples.

```
\RequirePackage{fancyvrb}
```

Pkg xstring

Used for `\StrSubstitute` for `\DescribeFile`.

```
\RequirePackage{xstring}
```

Pkg hyperref PDF bookmarks

If hyperref is loaded, disable some macros in PDF bookmarks:

```
\AtBeginDocument{
    \@ifpackageloaded{hyperref}{
        \pdfstringdefDisableCommands{%
            \def\quad{ }%
            \def\\{ }%
            \def\pkg#1{#1}%
            \def\ctr#1{#1}%
            \def\bool#1{#1}%
            \def\optn#1{#1}%
            \def\env#1{#1}%
            \def\cs#1{\textbackslash#1}%
            \def\,{ }%
            \def\LuaLaTeX{LuaLaTeX}%
            \def\XeLaTeX{XeLaTeX}%
            \def\TeX{TeX}%
            \def\LaTeX{LaTeX}%
            \def\LaTeXe{LaTeX2e}%
            \def\LuaTeX{LuaTeX}%
            \def\LuaLaTeX{LuaLaTeX}%
            \def\XeTeX{XeTeX}%
            \def\AmS{AMS}%
            \def\Dash{ --- }%
            \def\dash{ -- }%
            \def\Slash{/}%
            \def\prog#1{\detokenize{#1}}%
            \def\progcode#1{#1}%
            \def\filenm#1{\detokenize{#1}}%
            \def\brand#1{#1}%
            \def\acro#1{#1}%
            \def\HTML{HTML}%
            \def\ODT{ODT}%
            \def\SVG{SVG}%
            \def\PNG{PNG}%
            \def\GIF{GIF}%
            \def\JPG{JPG}%
            \def\EPS{EPS}%
            \def\PDF{PDF}%
            \def\DVI{DVI}%
            \def\UTF{UTF}%
            \def\URL{URL}%
            \def\element#1{#1}%
            \def\attribute#1{#1}%
            \def\HTML{HTML}%
            \def\HTMLfive{HTML5}%
            \def\CSS{CSS}%
            \def\CSSthree{CSS3}%
```

```
            \def\EPUB{EPUB}%
            \def\TOC{TOC}%
            \def\LOF{LOF}%
            \def\LOT{LOT}%
        }
    }
```

If hyperref is not loaded, emulate \hyperpage here.

```
    {
        \newcommand*{\hyperpage}[1]{#1}
    }
}
```

Pkg pict2e

```
\RequirePackage{pict2e}
\setlength{\unitlength}{1pt}
```

\warningsign Prints an exclamation point inside a triangle.

Creates a warning sign without relying on the presence of the fourier font. During copy/paste, this shows up as a simple exclamation point.

```
\newcommand*{\warningsign}{%
\begin{picture}(10,9)
\put(4,1){\scriptsize!}
\put(0,0){\line(500,866){5}}
\put(10,0){\line(-500,866){5}}
\put(0,0){\line(1,0){10}}
\end{picture}
}
```

6.2 Gobbling comment characters

DTXD@gobble The .dtx format uses leading percent characters for code to be in the documentation only. Other classes do not.

```
\@ifpackageloaded{doc}{
    \newcommand*{\DTXD@gobble}{2}
}{
    \newcommand*{\DTXD@gobble}{0}
}
```

6.3 Vertical spacing

```
\setlength{\marginparsep}{1em}
\setlength{\marginparpush}{.7ex}

\setlength{\parindent}{0em}
\setlength{\parskip}{2ex}
```

Len `\IndexMin` From ltxdoc.

```
\ifdef{\IndexMin}
    {\setlength{\IndexMin}{40ex}}
    {\newlength{\IndexMin}}
```

6.4 ltxdoc emulation

If the ltxdoc class is not used, some of its macros are replicated here.

```
\@ifclassloaded{ltxdoc}{}{
    \def\cmd#1{\cs{\expandafter\cmd@to@cs\string#1}}
    \def\cmd@to@cs#1#2{\char\number`#2\relax}
    \DeclareRobustCommand\cs[1]{\texttt{\char`\\#1}}
    \providecommand\marg[1]{%
        {\ttfamily\char`\{}\meta{#1}{\ttfamily\char`\}}}
    \providecommand\oarg[1]{%
        {\ttfamily[}\meta{#1}{\ttfamily]}}
    \providecommand\parg[1]{%
        {\ttfamily(}\meta{#1}{\ttfamily)}}
    \providecommand\url{\texttt}
}
```

6.5 doc emulation

If the doc class is not used, some of its macros are replicated here.

```
\AtBeginDocument{
    \@ifpackageloaded{doc}{}{
        \newenvironment*{macro}[1]{%
            \PackageError{dtxdescribe}
                {The 'macro' environment is only\MessageBreak
                    available when using the doc package\MessageBreak
                    with a .dtx source file}
                {This environment only makes sense for .dtx source.}
        }{}
        \newenvironment*{environment}[1]{%
            \PackageError{dtxdescribe}
                {The 'environment' environment is only\MessageBreak
```

```
                    available when using the doc package\MessageBreak
                    with a .dtx source file}
                {This environment only makes sense for .dtx source.}
        }{}
        \def\MacroFont{\fontencoding\encodingdefault
                    \fontfamily\ttdefault
                    \fontseries\mddefault
                    \fontshape\updefault
                    \small}%
        \@ifundefined{actualchar}{\def\actualchar{@}}{}
        \@ifundefined{quotechar}{\def\quotechar{"}}{}
        \@ifundefined{levelchar}{\def\levelchar{!}}{}
        \@ifundefined{encapchar}{\def\encapchar{|}}{}
        \@ifundefined{verbatimchar}{\def\verbatimchar{+}}{}
        \setlength\marginparpush{0pt}  \setlength\marginparwidth{8pc}
        \reversemarginpar
        \DeclareRobustCommand\meta[1]{%
            \ensuremath\langle
            \ifmmode \expandafter \nfss@text \fi
            {%
            \meta@font@select
            \edef\meta@hyphen@restore
                {\hyphenchar\the\font\the\hyphenchar\font}%
            \hyphenchar\font\m@ne
            \language\l@nohyphenation
            #1\/%
            \meta@hyphen@restore
            }\ensuremath\rangle
        }
        \def\meta@font@select{\itshape}
    }
}
```

6.6 Support macros

`\PrintEnvName` {⟨*name*⟩} Prints an environment name.

```
\providecommand*{\PrintEnvName}{}
\renewcommand*{\PrintEnvName}[1]
{\strut{\scriptsize{}Env}\quad\MacroFont#1\ }
```

`\DTXD@printtype` {⟨*text*⟩}

Used to print the object class in the margin:

```
\newcommand*{\DTXD@printtype}[1]
{\raggedleft\strut{\scriptsize\sffamily#1}\quad\MacroFont}
```

`\usage` {⟨*text*⟩}

Allow hyperlinks in the “usage” index entries:

```
\providecommand{\usage}{}
\renewcommand{\usage}[1]{\textit{\hyperpage{#1}}}
```

`\DTXD@origwrindex` Used to bypass hyperref index modifications.

```
\let\DTXD@origwrindex\@wrindex
```

`\DTXD@margintag` {⟨*class*⟩} {⟨*name*⟩} {⟨*margin tag*⟩}

Creates the margin tag for the object being described.

The `class` is used to sub-categories keys into their key/value groups.

```
\newcommand*{\DTXD@margintag}[3]{%
\@ifundefined{@captype}{% not float?
\leavevmode%
\marginpar{%
\DTXD@printtype{%
#3% margintag
\ifblank{#1}{}{ [#1]}% class
}% Desc@Type
\texttt{#2}% name
}% marginpar
}{}% not float?
}
```

`\DTXD@index` {⟨*class*⟩} {⟨*name*⟩} {⟨*margin tag*⟩} {⟨*index tag*⟩} {⟨*main/usage*⟩}

Creates the index entries for the object being described, where name has no backslash or underscore.

The `class` is used to sub-categories keys into their key/value groups. `main` prints code lines in the index, and `usage` prints page numbers.

```
\newcommand*{\DTXD@index}[5]{%
```

The `makeindex` program allows each index entry to call a macro by appending a vertical bar and a macro name to each entry. hyperref adds a call by `\hyperpage` to each index entry, by appending the phrase `|hyperpage` to the entry in the `.idx` file. The doc package uses the same mechanism to distinguish between code line entries (`|main`) and references to the use of a macro (`|usage`). The problem is that `makeindex` can only handle one macro call, but hyperref tries to append its `|hyperpage` to the already-existing `|usage` or `|main`.

The solution used for dtxdescribe is to allow hyperref to modify all regular index entries, but use the original definition of `\@wrindex` for the `\Describe____` macros, before hyperref modified it. Then, the `\usage` macro, defined above, manually adds the hyperlink.

Below, `\@bsphack` and `\@esphack` seem to be required for `\@wrindex` to work. `\ignorespaces` is used in addition because `\Declare` and `\index` entries often come in groups.

```
\@bsphack%
\begingroup%
\DTXD@origwrindex{%
```

Index by name:

Write the name, the formatted name, the index tag, and the class:

```
#2\actualchar{\protect\ttfamily#2} % name
(#4)% index tag
\ifblank{#1}{}{ [#1]}% class
\encapchar #5}%
```

Index by tag and class:

Write the tag and class as a group, under which is the name and the formatted name.

```
\begingroup%
\DTXD@origwrindex{%
#4:\levelchar% index tag
\ifblank{#1}{}{[#1]:\levelchar}% class
#2\actualchar{\protect\ttfamily#2}% name
\encapchar #5}%
```

Possibly index by class and name:

```
\ifblank{#1}{}{% class given
\begingroup%
\DTXD@origwrindex{%
#1\actualchar[#1]:\levelchar% class
#2\actualchar{\protect\ttfamily#2} % name
(#4)% index tag
\encapchar #5}%
}% class given
% \@esphack%
\@esphack%
\ignorespaces%
}
```

`\DTXD@margintagindex` {⟨*class*⟩} {⟨*name*⟩} {⟨*margin tag*⟩} {⟨*index tag*⟩} {⟨*main/usage*⟩}

Creates the margin tag and the index entries. The `class` is used to sub-categories keys into their key/value groups.

```
\newcommand*{\DTXD@margintagindex}[5]{%
% \@bsphack%
```

The margin tag and the name:

```
\DTXD@margintag{#1}{#2}{#3}%
```

The index entries:

```
\DTXD@index{#1}{#2}{#3}{#4}{#5}%
}
```

`\DTXD@macroname` {⟨*control sequence*⟩}

Given a control sequence such as `\name`, prints its name without the backslash.

From: http://tex.stackexchange.com/questions/42318/removing-a-backslash-from-a-character-sequence

```
\begingroup\lccode`\|=`\\
\lowercase{\endgroup\def\removebs#1{\if#1|\else#1\fi}}
\newcommand*{\DTXD@macroname}[1]{\expandafter\removebs\string#1}
```

`\DTXD@verbatimcmd` {⟨`\name`⟩}

While printing to the index file, prints the `\name` verbatim. From `\SpecialIndex` in the doc package.

```
\newcommand*{\DTXD@verbatimcmd}[1]{%
\string\verb\quotechar*\verbatimchar\string#1\verbatimchar%
}
```

`\DTXD@cmdmargintagindex` {⟨*class*⟩} {⟨*name*⟩} {⟨*margin tag*⟩} {⟨*index tag*⟩} {⟨*main/usage*⟩}

Creates the margin tag and index entries where name is a `\macro`.

```
\newcommand*{\DTXD@cmdmargintagindex}[5]{%
\@bsphack%
```

Create a margin tag with the name of the macro:

```
\@ifundefined{@captype}{% not float?
\leavevmode%
\marginpar{%
\DTXD@printtype{%
#3% margin tag
\ifblank{#1}{}{ [#1]}% class
}% Desc@Type
\cmd{#2}% name
}% marginpar
}{}% not float?
```

Create an index entry sorted by the name without its leading backslash, followed by the macro name with the backslash, and the tag. Prepend with the class if given.

Write (class):>name=csname (indextag)|usage

```
\begingroup%
\DTXD@origwrindex{%
\ifblank{#1}{}{#1\actualchar[#1]:\levelchar}% class
\DTXD@macroname{#2}\actualchar\DTXD@verbatimcmd{#2} % name
(#4)% index tag
\encapchar #5}%
```

Create an index entry grouped by the tag, then printed and sorted by the macro name with the backslash, and the tag.

Write indextag:>(class):>csname|usage

```
\begingroup%
\DTXD@origwrindex{%
#4:\levelchar% index tag
\ifblank{#1}{}{[#1]:\levelchar}% class
\DTXD@verbatimcmd{#2}% name
\encapchar #5}%
\@esphack%
\ignorespaces%
}
```

6.7 `\DescribeMacro` and `\DescribeEnvironment`

`\DescribeMacro` `[`⟨*class*⟩`]` `{`⟨`\name`⟩`}`

Redefined to allow hyperlinked index entries and an optional class:

```
\providecommand*{\DescribeMacro}{}
\renewcommand*{\DescribeMacro}[2][]{%
\@bsphack%
```

Create the margin tag with the macro's name:

```
\@ifundefined{@captype}{% not float?
\leavevmode%
\marginpar{%
\raggedleft%
\ifblank{#1}{}{{\scriptsize\textsf{[#1]}} }% class
\cmd{#2}% name
}% marginpar
}{}% not float?
```

Write the index sorted by the name without the backslash, followed by the actual name with the backslash. Append the class if given.

Write name=csname>(class)|usage

```
\begingroup%
\DTXD@origwrindex{%
\DTXD@macroname{#2}\actualchar\DTXD@verbatimcmd{#2}% name
\ifblank{#1}{}{\levelchar[#1]}% class
\encapchar usage}%
```

Only if a class was given:

```
\ifblank{#1}%
{}% no class
{% class given
% Again, and prepend the class:
%
% Write class=(class):>name=csname\verb+|usage+
%    \begin{macrocode}
\begingroup%
\DTXD@origwrindex{%
#1\actualchar[#1]:\levelchar%
\DTXD@macroname{#2}\actualchar\DTXD@verbatimcmd{#2}%
\encapchar usage}%
}% class given
\@esphack%
\ignorespaces%
}
```

`\DescribeEnv` `[`⟨*class*⟩`]` `{`⟨*environment name*⟩`}`

Redefined to allow hyperlinked index entries:

```
\providecommand*{\DescribeEnv}{}
\renewcommand*{\DescribeEnv}[2][]
{\DTXD@margintagindex{#1}{#2}{Env}{environment}{usage}}
```

6.8 New \Describe... macros

\DTX@filename Stores the filename with a sanitized underscore.

```
\newcommand*{\DTXD@filename}{}
```

\DTXD@filemarginparindex {⟨*class*⟩} {⟨*name*⟩} {⟨*margin tag*⟩} {⟨*index tag*⟩} {⟨*main/usage*⟩}

The name may have underscores.

```
\newcommand*{\DTXD@filemarginparindex}[5]{%
```

Create a detokenized version of the filename…

```
\renewcommand{\DTXD@filename}{\detokenize{#2}}%
```

… then replace any underscores with a detokenized _, which will print as an underscore when read back from the index file:

```
\StrSubstitute{\DTXD@filename}%
{\detokenize{_}}{\detokenize{\_}}[\DTXD@filename]%
```

The original filename is printed in the margin. Any underscore characters have already been disabled by the \catcode change.

```
\DTXD@margintag{#1}{#2}{#3}%
```

The detokenized and sanitized version is sent to the index file:

```
\DTXD@index{#1}{\DTXD@filename}{#3}{#4}{#5}%
```

End the group with the disabled underscore, and clean up the extra space from the \catcode command:

```
\endgroup%
\ignorespaces%
}
```

\DTXD@DescribeFile [⟨*class*⟩] {⟨*name*⟩}

The name may have underscores.

```
\newcommand*{\DTXD@DescribeFile}[2][]{%
\DTXD@filemarginparindex{#1}{#2}{File}{file}{usage}%
}
```

\DescribeFile {⟨*name*⟩}

The underscore character is temporarily disabled, then the name is passed directly to \DTXD@DescribeFile.

```
\newcommand*{\DescribeFile}{%
\begingroup\catcode`\_=12 \DTXD@DescribeFile%
}
```

\DTXD@DescribeProgram [⟨*class*⟩] {⟨*name*⟩}

The name may have underscores.

```
\newcommand*{\DTXD@DescribeProgram}[2][]{%
\DTXD@filemarginparindex{#1}{#2}{Prog}{program}{usage}%
}
```

\DescribeProgram {⟨*name*⟩}

The underscore character is temporarily disabled, then the name is passed directly to \DTXD@DescribeProgram.

```
\newcommand*{\DescribeProgram}{%
\begingroup\catcode`\_=12 \DTXD@DescribeProgram%
}
```

\DTXD@DescribeCommand [⟨*class*⟩] {⟨*name*⟩}

The name may have underscores.

```
\newcommand*{\DTXD@DescribeCommand}[2][]{%
\DTXD@filemarginparindex{#1}{#2}{Cmd}{command}{usage}%
}
```

\DescribeCommand {⟨*name*⟩}

The underscore character is temporarily disabled, then the name is passed directly to \DTXD@DescribeCommand.

```
\newcommand*{\DescribeCommand}{%
\begingroup\catcode`\_=12 \DTXD@DescribeCommand%
}
```

\DTXD@DescribePackage [⟨*class*⟩] {⟨*name*⟩} The name may have underscores.

```
\newcommand*{\DTXD@DescribePackage}[2][]{%
```

```
\DTXD@filemarginparindex{#1}{#2}{Pkg}{package}{usage}%
}
```

`\DescribePackage` `{`⟨*name*⟩`}`

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribePackage`.

```
\newcommand*{\DescribePackage}{%
\begingroup\catcode`\_=12 \DTXD@DescribePackage%
}
```

`\DTXD@DescribeClass` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

The name may have underscores.

```
\newcommand*{\DTXD@DescribeClass}[2][]{%
\DTXD@filemarginparindex{#1}{#2}{Cls}{class}{usage}%
}
```

`\DescribeClass` `{`⟨*name*⟩`}`

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribeClass`.

```
\newcommand*{\DescribeClass}{%
\begingroup\catcode`\_=12 \DTXD@DescribeClass%
}
```

`\DescribeOption` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand*{\DescribeOption}[2][]
{\DTXD@margintagindex{#1}{#2}{Opt}{option}{usage}}
```

`\DescribeArgument` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

The `class` may be used to categorize arguments by their macro or environment name.

```
\newcommand*{\DescribeArgument}[2][]
{\DTXD@margintagindex{#1}{#2}{Arg}{argument}{usage}}
```

`\DescribeBoolean` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand*{\DescribeBoolean}[2][]
{\DTXD@margintagindex{#1}{#2}{Bool}{boolean}{usage}}
```

\DescribeLength [⟨*class*⟩] {⟨*name*⟩}

```
\newcommand*{\DescribeLength}[2][]
{\DTXD@cmdmargintagindex{#1}{#2}{Len}{length}{usage}}
```

\DescribeCounter [⟨*class*⟩] {⟨*name*⟩}

```
\newcommand*{\DescribeCounter}[2][]
{\DTXD@margintagindex{#1}{#2}{Ctr}{counter}{usage}}
```

\DescribeKey [⟨*class*⟩] {⟨*name*⟩}

The class may be used to categorize keys by their kev/value group.

```
\newcommand*{\DescribeKey}[2][]
{\DTXD@margintagindex{#1}{#2}{Key}{key}{usage}}
```

\DescribeObject [⟨*class*⟩] {⟨*name*⟩}

May be used to describe an arbitrary piece of code. Creates a margin tag and index entries with \ttfamily.

```
\newcommand*{\DescribeObject}[2][]{%
\@ifundefined{@captype}{% not float?
\@bsphack%
\leavevmode%
\marginpar{%
    \raggedleft%
    \ifblank{#1}{}{\raggedleft{\scriptsize[#1]} }
    \texttt{#2}%
}%
}{}% not float?
\ifblank{#1}%
{\begingroup%
\DTXD@origwrindex{%
#2\actualchar{\protect\ttfamily#2}%
\encapchar usage%
}%
}%
{%
\begingroup%
\DTXD@origwrindex{%
#2\actualchar{\protect\ttfamily#2} [#1]%
\encapchar usage%
}%
\begingroup%
\DTXD@origwrindex{%
```

```
#1\actualchar[#1]:\levelchar#2\actualchar{\protect\ttfamily#2}%
\encapchar usage%
}%
}%
\@esphack%
\ignorespaces%
}
```

`\DescribeOther` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

May be used to describe an arbitrary non-programming object. Creates a margin tag and index entries with roman type.

```
\newcommand*{\DescribeOther}[2][]{%
\@ifundefined{@captype}{% not float?
\@bsphack%
\leavevmode%
\marginpar{%
    \raggedleft%
    \ifblank{#1}{}{\raggedleft{\scriptsize[#1]} }%
    #2%
}%
}{}% not float?
\ifblank{#1}%
{%
\begingroup%
\DTXD@origwrindex{#2\encapchar usage}%
}%
{%
\begingroup%
\DTXD@origwrindex{#2 [#1]\encapchar usage}%
\begingroup%
\DTXD@origwrindex{#1\actualchar[#1]:\levelchar#2\encapchar usage}%
}%
\@esphack%
\ignorespaces%
}
```

6.9 `\DescribeDefault`

`\DescribeDefaultcolor` The color of the margin tag used to show the default value.

```
\newcommand*{\DescribeDefaultcolor}{green!50!black}
```

`\DescribeDefault` `{`⟨*value*⟩`}`

Creates a colored margin tag showing the booleandefault value.

```
\newcommand{\DescribeDefault}[1]{%
    \margintag{%
        \footnotesize%
        \textcolor{\DescribeDefaultcolor}{%
            Default: \texttt{#1}%
        }%
    }%
}
```

6.10 \ItemDescribeMacro, etc.

The following are for use inside a description.

\ItemDescribeMacro [⟨*class*⟩] {⟨\name⟩}

```
\newcommand{\ItemDescribeMacro}[2][]{%
\item[\cmd{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeMacro[#1]{#2}%
}
```

\ItemDescribeEnv [⟨*class*⟩] {⟨*name*⟩}

```
\newcommand{\ItemDescribeEnv}[2][]{%
\item[\env{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeEnv[#1]{#2}%
}
```

\ItemDescribeArgument [⟨*class*⟩] {⟨*argument*⟩}

```
\newcommand{\ItemDescribeArgument}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeArgument[#1]{#2}%
}
```

\ItemDescribeBoolean [⟨*class*⟩] {⟨*name*⟩}

```
\newcommand{\ItemDescribeBoolean}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
```

```
\DescribeBoolean[#1]{#2}%
}
```

`\ItemDescribeLength` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\ItemDescribeLength}[2][]{%
\item[\cmd{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeLength[#1]{#2}%
}
```

`\ItemDescribeCounter` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\ItemDescribeCounter}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeCounter[#1]{#2}%
}
```

`\ItemDescribeKey` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\ItemDescribeKey}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeKey[#1]{#2}%
}
```

`\ItemDescribePackage` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\DTXD@ItemDescribePackage}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribePackage[#1]{#2}%
\endgroup
}

\newcommand{\ItemDescribePackage}{%
\begingroup\catcode`\_=12 \DTXD@ItemDescribePackage%
}
```

`\ItemDescribeClass` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\DTXD@ItemDescribeClass}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeClass[#1]{#2}%
```

```
\endgroup
}

\newcommand{\ItemDescribeClass}{%
\begingroup\catcode`\_=12 \DTXD@ItemDescribeClass%
}
```

`\ItemDescribeOption` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\ItemDescribeOption}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeOption[#1]{#2}%
}
```

`\ItemDescribeFile` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\DTXD@ItemDescribeFile}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeFile[#1]{#2}%
\endgroup
}

\newcommand{\ItemDescribeFile}{%
\begingroup\catcode`\_=12 \DTXD@ItemDescribeFile%
}
```

`\ItemDescribeProgram` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\DTXD@ItemDescribeProgram}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeProgram[#1]{#2}%
\endgroup
}

\newcommand{\ItemDescribeProgram}{%
\begingroup\catcode`\_=12 \DTXD@ItemDescribeProgram%
}
```

`\ItemDescribeCommand` `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\DTXD@ItemDescribeCommand}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeCommand[#1]{#2}%
```

```
\endgroup
}

\newcommand{\ItemDescribeCommand}{%
\begingroup\catcode`\_=12 \DTXD@ItemDescribeCommand%
}
```

\ItemDescribeObject `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\ItemDescribeObject}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeObject[#1]{#2}%
}
```

\ItemDescribeOther `[`⟨*class*⟩`]` `{`⟨*name*⟩`}`

```
\newcommand{\ItemDescribeOther}[2][]{%
\item[\texttt{#2}:]%
\setlength{\parskip}{1.5ex}%
\DescribeOther[#1]{#2}%
}
```

6.11 \margintag, \watchout

\margintagcolor The color of the \margintag.

```
\newcommand*{\margintagcolor}{blue!70!black}
```

\margintag `{`⟨*text*⟩`}`

Prints a colored margin tag.

```
\newcommand{\margintag}[1]{%
\@ifundefined{@captype}{% not float?
\marginpar{\raggedleft\textcolor{\margintagcolor}{#1}}%
\ignorespaces%
}{}% not float?
}
```

\watchoutcolor The color of the \watchout.

```
\newcommand*{\watchoutcolor}{red!50!black}
```

\watchout [⟨*text*⟩]

Prints a warning sign and optional text.

```
\newcommand{\watchout}[1][]{%
\@ifundefined{@captype}{% not float?
    \marginpar{%
        \raggedleft%
        \textcolor{\watchoutcolor}{\warningsign\normalsize\quad#1}%
    }%
    \ignorespaces%
}{}% not float?
}
```

6.12 The dtxexample environment

Also see example 13 on page page 26.

File dtxexample_cut.tex Used to store then \input example code.

[color] DTXD@examplerulecolor The color of the middle rule in the dtxexample.

```
\definecolor{DTXD@examplerulecolor}{rgb}{.9,.9,.9}
```

\dtxexamplecodename The text name of the code section.

```
\newcommand*{\dtxexamplecodename}{Code:}
```

\dtxexampleresultname The text name of the result section.

```
\newcommand*{\dtxexampleresultname}{Result:}
```

Env dtxexample * [⟨*notes/cross-references*⟩] {⟨*caption & label*⟩}

Reads the code listing as a verbatim input using the fancybox package, then displays the code listing as a verbatim output, and also executes the code and displays the result. A title caption is specified, along with optional cross-referencing commands or notes to refer to the results. The unstarred version places the code inside a minipage, forbidding a page break in the middle of the code listing. The starred version does not use a minipage. This is required when the code is too large to fit on a single page.

```
\NewDocumentEnvironment{dtxexample}{s +O{} m}
{% start dtxexample
```

Copy the environment's contents to the file dtxexample_cut.tex:

```
\VerbatimOut[gobble=\DTXD@gobble,tabsize=4]{dtxexample_cut.tex}%
}% start dtxexample
```

When the environment closes:

```
{% end dtxexample
```

Finish the verbatim output:

```
\endVerbatimOut
\par
\addvspace{\bigskipamount}
```

If unstarred, typeset the example in a minipage:

```
\IfBooleanTF{#1}{\vspace{\bigskipamount}}{\minipage{\linewidth}}%
```

Emulated a float of type "example":

```
\captionsetup{type=dtxdexample}%
\hrule\medskip
\caption{#3}
```

Typeset the contents as verbatim:

```
\textcolor{DTXD@examplerulecolor}{\smallskip\hrule}
\smallskip
{\scriptsize\itshape\dtxexamplecodename}
\VerbatimInput[tabsize=4]{dtxexample_cut.tex}
\unskip
\textcolor{DTXD@examplerulecolor}{\hrule}
\smallskip
{\scriptsize\itshape\dtxexampleresultname}

```

Possible add the optional cross-references or notes:

```
\ifstrempty{#2}
{}
{{\itshape\small #2}}
```

If unstarred, close the \minipage.

```
\IfBooleanTF{#1}{}{\endminipage}%
} % end dtxexample
```

Outside of the environment's scope, input the example to generate its output and labels:

```
\AfterEndEnvironment{dtxexample}
{%
```

Execute the code:

```
\par\unskip\input{dtxexample_cut.tex}%
```

Closing rule::

```
\medskip\hrule%
}
```

[dtxexample] A new float type for the examples.

`\DeclareFloatingEnvironment`

```
\DeclareFloatingEnvironment[
fileext=lox,
listname={List of Examples},
name=Example,
placement=hbp
]{dtxdexample}
```

[dtxexample] `\captionsetup` Caption setup for the examples.

```
\captionsetup*[dtxdexample]{
format=hang,
font=bf,
justification=raggedright,
singlelinecheck=false,
skip=0pt,
position=top,
}
```

[dtxexample] `\crefname` Name for cleveref.

```
\AtBeginDocument{
\@ifpackageloaded{cleveref}{\crefname{dtxdexample}{example}{examples}}{}
}
```

6.13 `noindmacro` and `noindenvironment`

Similar to `macro` and `environment`, but not indexed.

Env noindmacro {⟨*name*⟩}

```
\newenvironment{noindmacro}[1]
{
    \setlength{\parskip}{\marginparpush}
    \leavevmode\par\DTXD@margintag{}{\cmd{#1}}{}
}
{\unskip}
```

Env noindenvironment {⟨*name*⟩}

```
\newenvironment{noindenvironment}[1]
{
    \setlength{\parskip}{\marginparpush}
    \leavevmode\par\DTXD@margintag{}{#1}{Env}
}
{\unskip}
```

6.14 sourcedisplay, UIdisplay, docsidebar

For use in a sourcedisplay:

\fquad Forces a quad indent.

```
\newcommand*{\fquad}{\hspace*{1em}}
```

\fqquad Forces a double-quad indent.

```
\newcommand*{\fqquad}{\hspace*{2em}}
```

\fqqquad Forces a triple-quad indent.

```
\newcommand*{\fqqquad}{\hspace*{3em}}
```

Env sourceverb To typeset a block of source code, verbatim.

```
\DefineVerbatimEnvironment{sourceverb}{Verbatim}
    {gobble=\DTXD@gobble,tabsize=4,xleftmargin=2em}
\BeforeBeginEnvironment{sourceverb}{\vspace*{-.5\parskip}}
```

Env fsourceverb To typeset a framed block of source code, verbatim.

```
    \DefineVerbatimEnvironment{fsourceverb}{Verbatim}
        {gobble=\DTXD@gobble,tabsize=4,xleftmargin=2em,frame=lines}
\BeforeBeginEnvironment{fsourceverb}{\vspace*{-.5\parskip}}
```

Env sourcedisplay To typeset a block of source code, allowing direct formatting.

```
\newenvironment{sourcedisplay}
{
    \leavevmode
    \par
    \fqquad\minipage{\linewidth-4em}
    \ttfamily
}
{%
    \endminipage
    \par
}
```

Env UIdisplay To typeset a user interface display.

```
\newenvironment{UIdisplay}
{
    \leavevmode
    \par
    \fqquad\minipage{\linewidth-4em}
    \sffamily\bfseries
}
{
    \endminipage
    \par
}
```

\userentryname Text to tell the user to enter the following item.

```
\newcommand*{\userentryname}{Enter~$\Rightarrow$}
```

\userentry {⟨*text to enter*⟩}

Typesets text to be entered by the users.

```
\newcommand{\userentry}[1]{%
\par
\fqquad%
\begin{minipage}{\linewidth-2em}
    {\footnotesize \userentryname}\quad\cmds{#1}
\end{minipage}
\par
```

```
608 }
```

Env docsidebar To typeset a sidebar in the documentation.

```
609 \newenvironment{docsidebar}[1][]
610 {%
611     \quote\unskip\medskip
612     \setlength{\parskip}{1.5ex}%
613     \ifblank{#1}{}{\textit{#1}\newline}%
614     \rule[.5\bigskipamount]{\linewidth}{.4pt}%
615     \newline%
616 }
617 {%
618     \leavevmode\par
619     \rule[\bigskipamount]{\linewidth}{.4pt}
620     \endquote\unskip
621 }
```

6.15 Formatted objects

Macros to format references to various kinds of objects.

6.15.1 LaTeX objects

\pkg or class

```
622 \providerobustcmd*{\pkg}[1]{\mbox{\textsf{#1}}}
```

\env

```
623 \providerobustcmd*{\env}[1]{\mbox{\texttt{#1}}}
```

\ctr

```
624 \providerobustcmd*{\ctr}[1]{\mbox{\texttt{#1}}}
```

\bool

```
625 \providerobustcmd*{\bool}[1]{\mbox{\texttt{#1}}}
```

\optn

```
626 \providerobustcmd*{\optn}[1]{\mbox{\texttt{#1}}}
```

\TOC

```
\providerobustcmd*{\TOC}{\acro{TOC}}
```

\LOF

```
\providerobustcmd*{\LOF}{\acro{LOF}}
```

\LOT

```
\providerobustcmd*{\LOT}{\acro{LOT}}
```

6.15.2 Programs and commands

\cmds

```
\providerobustcmd*{\cmds}[1]{\mbox{\textbf{\texttt{#1}}}}
```

\progcode

```
\providerobustcmd*{\progcode}[1]{\mbox{\texttt{#1}}}
```

\prog

```
\newcommand*{\DTXD@prog}[1]{%
    \mbox{\textsf{\textsl{\detokenize{#1}}}}%
    \endgroup%
}

\providerobustcmd*{\prog}{%
    \begingroup%
    \catcode`\_=12%
    \DTXD@prog%
}
```

\filenm

```
\newcommand*{\DTXD@filenm}[1]{%
    \mbox{\texttt{\detokenize{#1}}}%
    \endgroup%
}

\providerobustcmd*{\filenm}{%
    \begingroup%
```

```
    \catcode`\_=12%
    \DTXD@filenm%
}
```

\UI General user-interface text.

```
\providerobustcmd*{\UI}[1]{\textbf{\textsf{#1}}}
```

6.15.3 File types

\ODT

```
\providerobustcmd*{\ODT}{\acro{ODT}}
```

\SVG

```
\providerobustcmd*{\SVG}{\acro{SVG}}
```

\PNG

```
\providerobustcmd*{\PNG}{\acro{PNG}}
```

\GIF

```
\providerobustcmd*{\GIF}{\acro{GIF}}
```

\JPG

```
\providerobustcmd*{\JPG}{\acro{JPG}}
```

\EPS

```
\providerobustcmd*{\EPS}{\acro{EPS}}
```

\PDF

```
\providerobustcmd*{\PDF}{\acro{PDF}}
```

\DVI

```
\providerobustcmd*{\DVI}{\acro{DVI}}
```

6.15.4 Internet

\UTF

```
\providerobustcmd*{\UTF}{\acro{UTF}}
```

\URL

```
\providerobustcmd*{\URL}{\acro{URL}}
```

\element

```
\providerobustcmd*{\element}[1]{\texttt{<#1>}}
```

\attribute

```
\providerobustcmd*{\attribute}[1]{\mbox{\texttt{#1}}}
```

\HTML

```
\providerobustcmd*{\HTML}{\acro{HTML}}
```

\HTMLfive

```
\providerobustcmd*{\HTMLfive}{\HTML\textsc{5}}
```

\CSS

```
\providerobustcmd*{\CSS}{\acro{CSS}}
```

\CSSthree

```
\providerobustcmd*{\CSSthree}{\CSS\textsc{3}}
```

\EPUB

```
\providerobustcmd*{\EPUB}{\acro{EPUB}}
```

6.15.5 Specific programs

\tikz

```
\providerobustcmd*{\tikz}{Ti\textit{k}z}
```

\MathML

```
671 \providerobustcmd*{\MathML}{Math\acro{ML}}
```

\CTAN

```
672 \providerobustcmd*{\CTAN}{\acro{CTAN}}
```

\TDS

```
673 \providerobustcmd*{\TDS}{\acro{TDS}}
```

6.15.6 Acronyms, brand names, trademarks

\brand

```
674 \providerobustcmd*{\brand}[1]{\textsc{#1}}
```

\acro

```
675 \providerobustcmd*{\acro}[1]{\textsc{\lowercase{#1}}}
```

\supregistered Superscript trademark symbol.

```
676 \providerobustcmd*{\supregistered}{\textsuperscript{\textregistered}}
```

6.16 Logos

\LuaTeX LuaTEX

```
677 \providerobustcmd*{\LuaTeX}{\mbox{Lua\TeX}}
```

\LuaLaTeX LuaLATEX

```
678 \providerobustcmd*{\LuaLaTeX}{\mbox{Lua\LaTeX}}
```

\XeTeX XETEX, XELATEX
\XeLaTeX

```
679 \providerobustcmd*{\XeTeXrevE}
680     {\hspace{-.1667em}\raisebox{-.5ex}{E}\hspace{-.125em}}
```

```
681
682 \AtBeginDocument{
683 \@ifpackageloaded{graphics}{
684     \renewrobustcmd*{\XeTeXrevE}
685         {\hspace{-.1667em}\raisebox{-.5ex}{\reflectbox{E}}\hspace{-.125em}}
686 }{}
687 }
688
689 \providerobustcmd*{\XeTeX}{\mbox{X\XeTeXrevE\TeX}}
690 \providerobustcmd*{\XeLaTeX}{\mbox{X\XeTeXrevE\LaTeX}}
```

`\AmS` $\mathcal{A}\mathcal{M}\mathcal{S}$

```
691 \providerobustcmd*{\AmS}{%
692     \leavevmode\hbox{$\mathcal A\kern-.2em\lower.376ex%
693     \hbox{$\mathcal M$}\kern-.2em\mathcal S$}%
694 }
```

`\LyX` LyX

```
695 \providerobustcmd*{\LyX}{\textsf{LyX}}
```

`\BibTeX` BibTeX

```
696 \providerobustcmd*{\BibTeX}{\mbox{B\textsc{ib}\TeX}}
```

`\MakeIndex` *MakeIndex*

```
697 \providerobustcmd*{\MakeIndex}{\prog{MakeIndex}}
```

`\ConTeXt` ConTeXt

```
698 \providerobustcmd*{\ConTeXt}{\mbox{Con\TeX{}t}}
```

`\MiKTeX` MiKTeX

```
699 \providerobustcmd*{\MiKTeX}{\mbox{MiK\TeX}}
```

6.17 Dashes and slashes

`\thinskip` A breakable thin skip.

```
700 \DeclareRobustCommand{\thinskip}{\hskip 0.16667em\relax}
```

`\endash` An endash: –

```
\def\endash{-}
```

`\emdash` An emdash: —

```
\def\emdash{--}
```

`\thinbrspace` A thin space which allows a line break.

```
\newcommand{\thinbrspace}{\hspace{.16667em}\penalty\exhyphenpenalty\hspace{0pt}}
```

`\thinthinbrspace` A thin space which allows a line break.

```
\newcommand{\thinthinbrspace}{\hspace{.08333em}\penalty\exhyphenpenalty\hspace{0pt}}
```

`\Dash` An unbreakeable thin space, emdash, and breakable thin space.

```
\newrobustcmd{\Dash}{\unskip\thinspace\emdash\thinbrspace}
```

`\dash` An unbreakeable thin space, endash, and breakable thin space.

```
\newrobustcmd{\dash}{\unskip\thinspace\endash\thinbrspace}
```

`\Slash` An unbreakable very thin space, a slash, and a breakable thin space.

```
\newrobustcmd{\Slash}{\unskip\hspace{.08333em}/\thinthinbrspace}
```

7 Compiling dtxdescribe

To compile the dtxdescribe package:

Enter ⇒ **`pdflatex dtxdescribe.ins`**

To compile the dtxdescribe documentation

Enter ⇒ **`pdflatex dtxdescribe.dtx`**

(Several times)

Enter ⇒ **`makeindex -s gglo.ist -o dtxdescribe.gls dtxdescribe.glo`**

Enter ⇒ **`makeindex -s gind.ist dtxdescribe`**

Enter ⇒ **`pdflatex dtxdescribe.dtx`**

(Several times)

Change History and Index

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

D

E

N

O

P

R

S

T