The LaTeX **dtxdescribe** Package

v1.09 — 2024/01/20

https://github.com/bdtc/dtxdescribe

Describe additional object types in `dtx` or `tex` 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, hooks, sockets, plugs, 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 type of object displayed in parentheses (length, filename, etc). Each item may be sorted further by an optional category displayed in brackets, such as `[category_name]`.

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.

dtxdescribe works with pdfTeX, XeTeX, and LuaTeX, and perhaps other engines as well.

See change in setup of **hyperref** and **cleveref**: Section 2 on page 7.

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 categories may be used to further sort 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

To use **dtxdescribe** with the **ltxdoc** class and `.dtx` files:

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

% hyperref now automatically loaded by \pkg{ltxdoc}.

\AtBeginDocument{
   \hypersetup{...}%                     If needed.
   \pdfstringdefDisableCommands{ ... }%   If needed.
}

\AddToHook{begindocument/before}{% Before .aux file is loaded.
  \usepackage{cleveref}%                 If needed.
  \crefname{somename}{name}{names}%      If needed.
}

    ...
%</driver>
```

To use **dtxdescribe** with the regular classes, such as **article**, use:

```
\documentclass{article}

\usepackage{dtxdescribe}

\usepackage{hyperref}%                   If needed.
   \hypersetup{...}%                     If needed.
   \pdfstringdefDisableCommands{ ... }%   If needed.

\usepackage{cleveref}%                   If needed.
  \crefname{somename}{name}{names}%      If needed.

\begin{document}
...
\end{document}
```

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

Optional “categories” may be assigned to the objects being described, including the new versions of `\DescribeMacro` and `\DescribeEnv`. These categories are printed in the margin tag and index entry for each item, and also generate additional index entries sorted by category. 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. `\listofdtxexamplefloats` prints the list of examples.

3 The macros, and the `dtxexample` environment

3.1 Macros and environments

`macro` (*env.*)
`environment` (*env.*)
⚠ `.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` `[`⟨*category*⟩`]` `{`⟨`\name`⟩`}`

The preexisting macro from the doc package is redefined to create hyperlinked index entries, and include an optional category. A margin tag is created and an index entry is made. When the optional category is used, it is displayed along with the margin tag, and is used to group an index entry by macro name and another index entry by category. An example would be to describe the float creation and caption setup for a new category 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 19 for examples.

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

The preexisting macro from the doc package is redefined to create hyperlinked index entries, include an optional category, and also to place an 'Env' tag in front of the name in the margin. See example 2 on page 20.

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` [⟨*category*⟩] {⟨*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 `category` may be used to categorize arguments by their macro or environment name. See example 10 on page 24.

3.3 Booleans, lengths, counters, hooks, sockets, plugs, keys

See example 4 on page 21.

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

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

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

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

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

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

`\DescribeHook` [⟨*category*⟩] {⟨*name*⟩}

Describes a hook. Given a 'Hook' tag in the margin and index. The `category` may be used to categorize hooks by package. The hook name may or may not have a backslash. Example:

```
\DescribeHook{\hookname}
\DescribeHook[LaTeX]{para/begin}
```

`\DescribeSocket` [⟨*category*⟩] {⟨*name*⟩}

`\DescribePlug` [⟨*category*⟩] {⟨*name*⟩}

Describes a socket or plug. Given a 'Socket' or 'Plug' tag in the margin and index. The `category` may be used to categorize sockets and plugs by package. Example:

```
\DescribeSocket{socketname}
\DescribePlug{plugname}
```

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

Describes a key. Given a 'Key' tag in the margin and index. The `category` may be used to categorize keys by their kev/value group. See example 9 on page 23.

3.4 Packages, classes, options

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

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

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

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

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

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

3.5 Files, programs, commands

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

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

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

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

`\DescribeCommand` [⟨*category*⟩] {⟨*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` [⟨*category*⟩] {⟨*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 category is used, it is added to the margin tag, appended to the index entry, and a second index entry is created grouped by category. If a macro name is to be described, use `\DescribeMacro` instead. See example 11 on page 25.

`\DescribeOther` [⟨*category*⟩] {⟨*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 category is used, it is added to the margin tag, appended to the index entry, and a second index entry is created grouped by category. See example 12 on page 25.

3.7 In a `description` environment

To describe an object using a `description` environment, use the following. See example 13 on page 26.

`\ItemDescribeMacro`	[⟨*category*⟩] {⟨\name⟩} A description.
`\ItemDescribeEnv`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeArgument`	[⟨*category*⟩] {⟨*argument*⟩} A description.
`\ItemDescribeBoolean`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeLength`	[⟨*category*⟩] {⟨\name⟩} A description.
`\ItemDescribeCounter`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeHook`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeSocket`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribePlug`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeKey`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribePackage`	[⟨*category*⟩] {⟨*package_name*⟩} With underscores.
`\ItemDescribeClass`	[⟨*category*⟩] {⟨*class_name*⟩} With underscores.
`\ItemDescribeOption`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeFile`	[⟨*category*⟩] {⟨*file_name*⟩} With underscores.
`\ItemDescribeProgram`	[⟨*category*⟩] {⟨*program_name*⟩} With underscores.
`\ItemDescribeCommand`	[⟨*category*⟩] {⟨*command_name*⟩} With underscores.
`\ItemDescribeObject`	[⟨*category*⟩] {⟨*name*⟩} A description.
`\ItemDescribeOther`	[⟨*category*⟩] {⟨*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 Nesting

`\shownesting` `*` `[`⟨*fraction of* `\linewidth`⟩`]` `{`⟨*container name*⟩`}` `{`⟨*contents*⟩`}`

It may be useful to show which objects contain which other objects. `\shownesting` shows a box enclosing a name for the container, and the container's contents. `\shownesting` be nested, showing boxes inside other boxes, which displays how each environment and macro is fit together inside each other.

The optional argument is the fraction of `\linewidth` to use for the box, from `[0]` to `[1]`. The default is `[1]`. Each `\shownesting` starts its own paragraph, unless the star `*` is used, in which case the `\shownesting*` appears inline with previous text. To place two `\shownesting` boxes side-by-side, use optional arguments to specify less than full `\linewidth` for each box, and use `\shownesting*` for the second box to place it inline.

See example 14 on page 27 for an example.

3.10 `\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` The color of the `\margintag`.
Default: `blue!70!black`

`\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`

3.11 `dtxexample` environment

`dtxexample` (*env.*) `*` `[`⟨*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.

`listofdtxexamplefloats` Prints the list of examples.

See example 15 for a demonstration of how `dtxexample` works.

The examples may be customized by redefining the following, perhaps for another language:

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

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

3.12 `noindmacro` and `noindenvironment` environments

⚠ `.dtx` only These are like `macro` and `environment`, but not indexed. 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, `noindmacro` and `noindenvironment` environments should not be used, as they would localize any definitions. `\DescribeMacro` and `\DescribeEnv` should be used instead.

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

`noindenvironment` (*env.*) {⟨*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`.

3.13 `sourceverb`, `sourcedisplay`, `UIdisplay`, `docsidebar`

`sourceverb` (*env.*) [⟨*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.
Default: `gobble=2,` `tabsize=4,` `xleftmargin=2em`

`fsourceverb` (*env.*) 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 16 on page 29.

`sourcedisplay` (*env.*) 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 17 on page 29.

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

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

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

`UIdisplay` (*env.*) Displays a user interface, such as a dialog box entry or a menu selection. See example 18 on page 30. 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`.

`docsidebar` (*env.*) [⟨*title*⟩] Creates a sidebar within the document. See example 19 on page 31.

3.14 Formatted objects

Macros to format references to various kinds of objects.

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

3.14.1 LaTeX objects

`\pkg` {⟨*packagename*⟩} Prints as packagename. Also for a classname.

`\cs` {⟨*csname*⟩} Prints as `\csname`.

`\env` {⟨*environment*⟩} Prints as `environment`.

`\marg` {⟨*argument*⟩} Prints {⟨*arg*⟩}. Mandatory argument.

`\oarg` {⟨*argument*⟩} Prints [⟨*arg*⟩]. Optional argument.

`\parg` {⟨*argument*⟩} Prints (⟨*arg*⟩). Picture-mode argument.

`\ctr` {⟨*counter*⟩} Prints as `counter`.

`\bool`	{⟨*boolean*⟩} Prints as `boolean`.
`\optn`	{⟨*option*⟩} Prints as `option`, for example to a macro, package, class.
`\TOC`	TOC: Table of contents.
`\LOF`	LOF: List of figures.
`\LOT`	LOT: List of tables.

3.14.2 Programs and commands

`\progcode`	Prints as `inline program code`: Escape underscores and other special characters such as {, %, $.
`\prog`	Prints as ***grep***, ***make***: A program name. Underscores allowed.
`\filenm`	Prints as `file_name`: Underscores allowed.
`\UI`	Prints as **General user-interface text.** What the user sees on the display. Also see the `UIdisplay` environment.
`\cmds`	Prints as **`Commands to be entered`**: What the user enters. Escape underscores and other special characters such as {, %, $. Also see the `\userentry` macro.

3.14.3 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

3.14.4 Internet

`\UTF`	UTF: Unicode
`\URL`	URL: Uniform Resource Locator
`\element`	{⟨*element name*⟩} Prints as `<element>`, an HTML / CSS element

`\attribute`	{⟨*attribute name*⟩} Prints as `attribute`, an HTML / CSS attribute. pdfTeX and XeTeX only. Not for LuaTeX.
`\attrib`	{⟨*attribute name*⟩} Prints as `attribute`, an HTML / CSS attribute. pdfTeX, XeTeX, or LuaTeX.
`\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

3.14.5 Specific programs

`\TikZ`	Ti*k*Z: Package logo
`\CTAN`	CTAN: Comprehensive TeX Archive Network
`\TDS`	TDS: TeX Directory Structure
`\MathML`	MathML: Mathematical Markup Language
`\MathJax`	MathJax: Math on the web.

3.14.6 Acronyms, brand names, trademarks

`\brand`	{⟨*name*⟩} BRANDNAME, COMPANY NAME
`\acro`	{⟨*acronym*⟩} ACRO: Acronym
`\supregistered`	Superscript trademark symbol®

3.15 Logos

Several additional logos are provided.

Also see the metalogo and metalogox packages.

`\dviTeX`	DVI TeX
`\dviLaTeX`	DVI LaTeX
`\pdfTeX`	PDF TeX
`\pdfLaTeX`	PDF LaTeX

`\LuaTeX`	LuaTEX
`\LuaLaTeX`	LuaLATEX
`\XeTeX`	XƎTEX, with reversed E if graphics is loaded.
`\XeLaTeX`	XƎLATEX, with reversed E if graphics is loaded.
`\AmS`	AMS
`\LyX`	LyX
`\BibTeX`	BIBTEX
`\MakeIndex`	*MakeIndex*
`\ConTeXt`	ConTEXt
`\MiKTeX`	MiKTEX

3.16 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]{\DeclareFloatingPhoto}
Create a photograph float environment.

\DescribeMacro[c=photograph]{\photocaptionsetup}
Caption settings for a photograph float.

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

Result:

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

`\DeclareFloatingPhoto` [photograph] Create a photograph float environment.

`\photocaptionsetup` [photograph] Caption settings for a photograph float.

`\cphotonameref` [photograph] cleveref name for the photograph float.

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

`myenvironment` (*env.*) `{`⟨*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:

`otherenvironment` (*env.*)
[kindofenvironment]

`[`⟨*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 category `kindofenvironment`.

Example 4: Booleans and Counters

Code:

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

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

Result:

`sampleboolean` (*bool*) [examples] Some description.

`samplecounter` (*Ctr*) [examples] 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:

`\photowidth` (*Len*) [photograph] Some description.

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

Example 6: Hooks and Sockets

Code:

```
\DescribeHook{\hookname} A hook with a backslash.

\DescribeHook{para/begin} A hook without a backslash.

\DescribeSocket{socketname} A socket.

\DescribePlug{plugname} A socket plug.
```

Result:

`\hookname` (*Hook*) A hook with a backslash.

`para/begin` (*Hook*) A hook without a backslash.

`socketname` (*Socket*) A socket.

`plugname` (*Plug*) A socket plug.

Hooks may or may not have a leading backslash.

Example 7: 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:

`samplepackage` (*Pkg*) [examples] About a LaTeX package.

`sample_class` (*Cls*) [examples] About a LaTeX class.

`sampleoption` (*Opt*) [examples] About an option for a package or class.

Example 8: 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:

`really_big_file.txt` (*file*) [bigfiles] Some description.

`another_big_file.txt` (*file*) [bigfiles] Some description.

`lone_file.txt` (*file*) Some description.

`OS_command` (*Cmd*) An operating-system command.

`program_name` (*Prog*) An operating-system program.

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

Example 9: 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 category.
```

Result:

`firstkey` (*Key*) [groupofkeys]	About the first key of the `groupofkeys` set.
`secondkey` (*Key*) [groupofkeys]	About the second key of `groupofkeys`.
`samplekey` (*Key*) [examples]	About some key of `otherkeys`.
`sampletwokey` (*Key*) [examples]	About another key of `otherkeys`.
`lonekey` (*Key*)	A key without a category.

See the index key groups.

Example 10: 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[\mymacro]{bold}
What happens when \cs{mymacro} is given the |bold| argument.
```

Result:

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

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

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

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

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

Example 11: Object

Code:

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

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

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

Result:

`somecolor` [color]	The color of something.
`othercolor` [color]	The other color.
`randomobject`	About some random object.

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

Example 12: Other

Code:

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

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

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

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

Result:

`license agreement`	The following is the fictional license agreement:
`Before myenvironment`	Actions to be done `\BeforeBeginEnvironment`.
`Other Item` [othercategory]	About the other item.
`Additional Item` [othercategory]	About the add'l item.

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

Example 13: 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 command.
\end{description}
```

Result:

`\macroname` [descexamples]

`\macroname`: Describe the macro.

`booleanname` (*bool*) [descexamples]

`booleanname`: Describe the boolean.

`\lengthname` (*Len*) [descexamples]

`\lengthname`: Describe the length.

`keyname` (*Key*) [descexamples]

`keyname`: Describe the key.

`package_name` (*Pkg*) [descexamples]

`package_name`: Describe the package.

`class_name` (*Cls*) [descexamples]

`class_name`: Describe the class.

`file_name` (*file*) [descexamples]

`file_name`: Describe the file.

`program_name` (*Prog*) [descexamples]

`program_name`: Describe the program.

`command_name` (*Cmd*) [descexamples]

`command_name`: Describe the command.

Uses a `description` environment to describe objects.

Example 14: Nesting

Code:

```
\shownesting{\env{environmentA}}{
      \shownesting{\cs{macroB}}{
          \shownesting{\env{environmentC}}{
              The contents.
          }
      }
}

\shownesting{\env{sidebyside}}{
      \shownesting[.35]{minipage}{
          Left contents.
      }
      \hfill \cs{hfill} \hfill
      \shownesting*[.35]{minipage}{
          Right contents.
      }
}
```

Result:

`environmentA`

`\macroB`

`environmentC`

The contents.

`sidebyside`

minipage

Left contents.

`\hfill`

minipage

Right contents.

Note the use of the optional arguments to select less than full `\linewidth`, and the starred form for the second box to place it inline with the `\hfill` text.

Contents of the figure.

Figure 1: A Figure

Example 15: `dtxexample`

Code:

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

Result:

See fig. 1

Example 15, 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 `dtxexamplefloat` was created, with the caption "`dtxexample`" and the label `ex:dtxexample`, which points to example 15.
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 16: `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 17: `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 18: `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 19: `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 uses of `\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

`makeidx` (*Pkg*)
`splitidx` (*Pkg*)

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

```
\AtBeginDocument{
    \IfPackageLoadedTF{makeidx}{}{
    \IfPackageLoadedTF{splitidx}{}{
        \RequirePackage{makeidx}
        \makeindex
    }}
}
```

`etoolbox` (*Pkg*) v2.6 or later for `\BeforeBeginEnvironment`, `\AfterEndEnvironment`

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

`xparse` (*Pkg*) Used for the examples.

```
\RequirePackage{xparse}
```

`calc` (*Pkg*) Used for `\shownesting`.

```
\RequirePackage{calc}
```

`xcolor` (*Pkg*) Used for the examples.

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

`caption` (*Pkg*) Used for the examples.

```
\RequirePackage{caption}
```

`newfloat` (*Pkg*) Used for the examples.

```
\RequirePackage{newfloat}
```

`fancyvrb` (*Pkg*) Used for the examples.

```
\RequirePackage{fancyvrb}
```

`xstring` (*Pkg*) Used for `\StrSubstitute` for `\DescribeFile`.

```
\RequirePackage{xstring}
```

`hyperref` (*Pkg*)
PDF bookmarks

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

```
\AtBeginDocument{
    \IfPackageLoadedTF{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\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\attrib#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}%
        }
    }% yes hyperref
```

If hyperref is not loaded, emulate `\hyperpage` here.

```
    {% no hyperref
        \newcommand*{\hyperpage}[1]{#1}
    }
}
```

`pict2e` (*Pkg*)

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

6.2 Warning sign

`\warningsign` Prints an exclamation point inside a triangle. Displays as: ⚠

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}
}
```

6.3 Special character handling

The literal backslash character:

```
\begingroup
\catcode`\|=0
\catcode`\\=12
|gdef|DTXD@backslash{\}
|endgroup
```

6.4 Patching hypdoc, splitidx, doc

If hyperref is disabled (by lwarp) then define the missing `\hdclindex`.

```
\IfPackageLoadedTF{doc}{
\IfPackageLoadedTF{hypdoc}{
\AddToHook{begindocument/before}[doc/hyperref]{%
 \ifdoc@hyperref
 \else
  \def\hdclindex#1#2{%
```

```
    \ifx\@nil#2\@nil\else\csname #2\expandafter\endcsname\fi%
  }%
 \fi
}
}{}% hypdoc loaded
}{}% doc loaded
```

Several changes for when **hypdoc** and **splitidx** are used together:

```
\AtBeginDocument{
\IfPackageLoadedTF{doc}{
\IfPackageLoadedTF{hypdoc}{
\IfPackageLoadedTF{splitidx}{
```

splitidx is modified to add `|hdpindex{}` to work with **hypdoc**.

```
\renewcommand*{\@wrsindex}[2][]{%
  \ifx\relax#1\relax%
    \if@splitidx%
      \@wrsindex[idx]{#2}%
    \else%
      \def\@tempa{#2}%
      \if@verbindex\@onelevel@sanitize\@tempa\fi%
      \@wrindex{\@tempa}%
    \fi%
  \else%
%
%    \def\@tempa{#2}%
    \def\@tempa{#2\encapchar hdpindex{}}%       dtxdescribe
%
    \csname index@#1@hook\endcsname%
    \expandafter\ifx\csname @@wrsindex\endcsname\relax%
      \@@@wrsindex{#1}{{\@tempa}{\thepage}}%
    \else%
      \def\@tempb{\@@wrsindex{#1}}%
      \expandafter\@tempb\@tempa||\\%
    \fi%
    \endgroup%
    \@esphack%
  \fi%
}
```

hypdoc guesses the TOC level for the PDF bookmarks, but its algorithm seems to fail at `\StopEventually` for split indices.

“`Paragraph ended before \HD@@guesstoclevel was complete.`”

Its guess is fixed to level 1 until `\PrintChanges` or `\printindex`.

```
\def\HD@guesstoclevel#1{1}
```

Patch **doc**’s `\PrintChanges` to reset the TOC guess to top level.

```
\preto\PrintChanges{\def\HD@guesstoclevel#1{0}}
```

hypdoc adds PDF bookmarks to letter groups in the index. hypdoc must take into account multiple indexes, otherwise, followup indices will have duplicate bookmarks.

Increment the index number at the start of each index, and append the index number to the PDF bookmark.

Also, change the TOC guess to top level from now on, presuming that the indices are at the end.

```
\newcounter{DTXD@indexnumber}

\extendtheindex
    {%
        \addtocounter{DTXD@indexnumber}{1}%
        \def\HD@guesstoclevel##1{0}%
    }
    {}
    {}
    {}

\def\HD@@@bfseries\hfil#1\hfil{%
  \ifx\\#1\\%
  \else%
    \raisebox{\baselineskip}[0pt]{%
      \kern-\HD@margin\relax%
%      \pdfbookmark[\HD@toclevel@subindex]{#1}{HD.#1}}%
      \pdfbookmark[\HD@toclevel@subindex]{#1}%      dtxdescribe
        {HD.#1.\arabic{DTXD@indexnumber}}%          dtxdescribe
      \kern\HD@margin\relax%
    }%
  \fi%
  \hfil#1\hfil%
}%
```

```
}{}% splitidx loaded
}{}% hypdoc loaded
}{}% doc loaded
}% AtBeginDocument
```

6.5 Gobbling comment characters

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

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

6.6 Vertical spacing

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

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

`\IndexMin` (*Len*) From ltxdoc.

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

6.7 Not ltxdoc: 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}
}
```

6.8 Not doc: doc emulation

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

```
\AtBeginDocument{
    \IfPackageLoadedTF{doc}{}{% not 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}
    }% not doc
}% AtBeginDocument
```

6.9 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 category in the margin:

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

`\usage` {⟨*text*⟩}

Allow hyperlinks in the “usage” index entries:

```
\IfPackageLoadedTF{doc}{}{% not doc package

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

}% not doc package
```

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

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

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

Creates the margin tag for the object being described.

The `category` is used to sub-categorize keys into their key/value groups.

```
\newcommand*{\DTXD@margintag}[3]{%
    \@ifundefined{@captype}{% not float?
        \leavevmode%
        \marginpar{%
            {%
                \hbadness=10000%
                \hfuzz=5em%
                \DTXD@printtype{%
                    #3% margintag
                    \ifblank{#1}{}{ [#1]}% category
                }%
                \texttt{#2}% name
            }%
        }% marginpar
    }{}% not float?
}
```

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

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

The `category` is used to sub-categorize 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 category:

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

Index by tag and category:

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

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

Possibly index by category and name:

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

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

Creates the margin tag and the index entries. The `category` is used to sub-categorize 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` {⟨*category*⟩} {⟨*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{%
    {%
        \hbadness=10000%
        \hfuzz=5em%
        \DTXD@printtype{%
            #3% margin tag
            \ifblank{#1}{}{ [#1]}% category
        }%
        \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 category if given.

Write [category]:>name=csname (indextag)|`usage`

```
\begingroup%
\DTXD@origwrindex{%
\ifblank{#1}{}{#1\actualchar[#1]:\levelchar}% category
\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:>[category]:>csname|`usage`

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

6.10 doc: Key handling for object categories

If using doc, the optional key/value argument may also include an object category. This is supported by assigning any unknown key to be the category.

The category may be given using the key `c`, or as the default action when an unknown key is given.

```
\ExplSyntaxOn
\IfPackageLoadedTF{doc}{

\newcommand*{\DTXD@category}{}

\keys_define:nn {doc}
{
    c .cs_set:Np = \DTXD@category,
    unknown .code:n = {%
        \renewcommand*{\DTXD@category}{\l_keys_key_str}%
    },
}

}{}% doc loaded
\ExplSyntaxOff
```

6.11 doc: Handling `\marginpar` inside a float

To avoid a floats lost error, do not print margin tags if inside a float.

```
\IfPackageLoadedTF{doc}{

\def\@doc@describe#1#2{%                    dtxdescribe
    \ifdoc@noprint\else%
      \@ifundefined{@captype}{% not float?  dtxdescribe
          \marginpar{\raggedleft%
                     \strut%
                     \doc@providetarget%
                     \@nameuse{PrintDescribe#1}{#2}%
                \ifdefvoid{\DTXD@category}{}{%  dtxdescribe
                    \space{\footnotesize[\mbox{\DTXD@category}]}%
                }%
          }%
      }{}%
    \fi%
    \ifdoc@noindex\else%
      \@nameuse{Special#1Index}{#2}%
    \fi%
  \@esphack%
  \endgroup%
  \ignorespaces%
}

}{}% doc loaded
```

6.12 doc: Handling categories and detokenizing names

`\@NewDocElement` File names and such may include underscores or other characters, so patch doc to neutralized while processing.

```
\IfPackageLoadedTF{doc}{%

\def\@NewDocElement#1#2#3{%
  \doc@macrolikefalse%
  \doc@topleveltrue%
  \def\doc@idxtype{#3}%
  \def\doc@idxgroup{#3s}%
  \let\doc@printtype\@empty%
  \csname keys_set:nn\endcsname{doc}{#1}%
  \ifx\doc@printtype\@empty%
    \@temptokena{}%
  \else%
    \@temptokena\expandafter{\expandafter%
         \textnormal\expandafter{\expandafter%
         \space\expandafter%
         (\doc@printtype)}}%
  \fi%
  \@nameedef{Print#2Name}##1{%
     {\noexpand\MacroFont%
      \ifdoc@macrolike%
         \noexpand\string##1%                           dtxdescribe
      \else%
         \noexpand\detokenize\expandafter{##1}%         dtxdescribe
      \fi%
      \the\@temptokena%
     }}%
  \expandafter\let\csname PrintDescribe#2\expandafter\endcsname%
                  \csname Print#2Name\endcsname%
  \edef\doc@expr{%
     \ifdoc@macrolike%
       \noexpand\doc@createspecialmacrolikeindexes%
     \else%
       \noexpand\doc@createspecialindexes%
     \fi%
     {#2}%
    }%
   \expandafter\expandafter\expandafter%
   \doc@expr%
   \expandafter\expandafter\expandafter%
     {\expandafter\doc@idxtype\expandafter}\expandafter%
     {\doc@idxgroup}%
  \doc@createdescribe{#2}%
    \ifdoc@macrolike%
      \doc@createenv{TT}{#2}{#3}%
    \else%
      \doc@createenv{TF}{#2}{#3}%
    \fi
}
```

`\DTXDbreak` Inserts a possible line break point. Used in the index to allow line breaks before verbatim category names.

```
\newcommand*{\DTXDbreak}{\space\penalty200}
```

`\DTXD@printobjectname` {⟨*name w/o backslash*⟩} {⟨*name*⟩} Adds the object name to the index in verbatim. These are passed as arguments instead of directly used here because they must have their value when the index is written instead of when `\DTXD@printobjectname` is used when the index is read back.

```
\newcommand*{\DTXD@printobjectname}[2]{%
    #1%
    \actualchar%
    \string\verb% %
    \quotechar%
    *%
    \verbatimchar%
    #2%
    \verbatimchar%
}
```

`\DTXD@maybecategory` If there is a category, it is added verbatim.

```
\newcommand*{\DTXD@maybecategory}{%
    \ifdefvoid{\DTXD@category}%
        {}%
        {%
```

`\DTXDbreak` inserts a possible line break here, allowing a break if the following verbatim is too long.

```
            \string\DTXDbreak%
```

```
            [%
            \string\verb%
            \noexpand\quotechar%
```

```
            *%
            \verbatimchar\DTXD@category\verbatimchar%
        ]}%
}
```

`\DTXD@categorylevelname` The simplified name, without backslash.

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

`\DTXD@maybecategorylevel` {⟨*object_type*⟩} If a category is assigned, index by category.

arg 1: Type of object, shown between parens (), such as `macro`, `boolean`, etc.

`\DTXD@category`: The name of the category, printed in brackets [], such as `\macroname`.

`\DTXD@categorylevelname`: The simplified name of the category, such as `macroname`.

`\@gtempa`: The name of this particular object.

```
\newcommand*{\DTXD@maybecategorylevel}[1]{%
    \ifdefvoid{\DTXD@categorylevelname}%
        {}%
        {%
```

Index:
`categorylevelname=\verb!*+[category]:+>name=\verb!*+<prefix>name+<break>(type)`

```
            \index{%
                \DTXD@categorylevelname%
                \noexpand\actualchar%
                \string\verb% % to fool emacs highlighting
                \noexpand\quotechar%
                *%
                \noexpand\verbatimchar%
                [\DTXD@category]:%
                \verbatimchar%
                \noexpand\levelchar%
           \noexpand\DTXD@printobjectname{\DTXD@gtempa@nobackslash}{\@gtempa}%
                \ifblank{#1}{}{\string\DTXDbreak (#1)}%
```

`\the\@temptokena` is not used here.

```
           \noexpand\doc@handleencap{usage}%
            }% index
        }%
}
```

`\DTXD@findcategorylevelname` Given `\DTXD@category`, create `\DTXD@categorylevelname`, a detokenized name without backslash.

```
\newcommand*{\DTXD@findcategorylevelname}{%
    \edef\DTXD@categorylevelname{\DTXD@category}%
   \edef\DTXD@categorylevelname{\detokenize\expandafter{\DTXD@categorylevelname}}%
    \IfBeginWith{\DTXD@categorylevelname}{\DTXD@backslash}%
     {\StrGobbleLeft{\DTXD@categorylevelname}{1}[\DTXD@categorylevelname]}%
        {}%
}
```

`\DTXD@gtempa@nobackslash` A version without the leading backslash.

```
\newcommand*{\DTXD@gtempa@nobackslash}{}
```

`\DTXD@findgtempa` Detokenize `\@gtempa` and also find another version without any leading backslash.

```
\newcommand*{\DTXD@findgtempa}[1]{%
    \edef\@gtempa{\detokenize{#1}}%
    \IfEndWith{\@gtempa}{ }{\StrGobbleRight{\@gtempa}{1}[\@gtempa]}{}%
    \IfBeginWith{\@gtempa}{\DTXD@backslash}%
        {\StrGobbleLeft{\@gtempa}{1}[\DTXD@gtempa@nobackslash]}%
        {\edef\DTXD@gtempa@nobackslash{\@gtempa}}%
}
```

`\doc@createspecialindexes` {⟨*1: name*⟩} {⟨*2: index type*⟩} {⟨*3: index group*⟩}

```
\def\doc@createspecialindexes#1#2#3{%
  \@temptokena{\space (#2)}%
  \@temptokenb{#3:}%
```

Define `\SpecialMain<name>Index` {⟨*1: name*⟩}

```
  \@nameedef{SpecialMain#1Index}##1{%
```

Find `\@gtempa`, the name of the object.

```
    \noexpand\DTXD@findgtempa{##1}%
```

`\@temptokena`: Index (type).

`\@temptokenb`: Index group.

`\@gtempa`: Detokenized name of the particular object.

`\DTXD@gtempa@nobackslash`: Detokenized name without backslash.

```
    \noexpand\@bsphack%
    \noexpand\DTXD@findcategorylevelname%                 dtxdescribe
```

```
  \ifdoc@toplevel%
%    \noexpand\special@index{##1\noexpand\actualchar
%
%
```

Index:
`name=\verb!*+name+ (type) [category]|main`

```
    \noexpand\special@index{%
        \noexpand\DTXD@printobjectname%                   dtxdescribe
            {\noexpand\DTXD@gtempa@nobackslash}%
            {\noexpand\@gtempa}%
        \ifx\@nil#2\@nil\else \the\@temptokena \fi%
        \noexpand\DTXD@maybecategory%                     dtxdescribe
        \noexpand\encapchar main%
    }%
```

Maybe add the category index entry:

```
    \noexpand\DTXD@maybecategorylevel{#2}%               dtxdescribe
  \fi%
%
%
```

If group is not empty, index:
`group:>name=\verb!*+name+ [category]|main`

```
  \ifx\@nil#3\@nil\else%
  \noexpand\special@index{%
       \the\@temptokenb\noexpand\levelchar%
%     ##1\noexpand\actualchar{\string\ttfamily\space##1}%
      \noexpand\DTXD@printobjectname%                    dtxdescribe
          {\noexpand\DTXD@gtempa@nobackslash}%
          {\noexpand\@gtempa}%
      \noexpand\DTXD@maybecategory%                      dtxdescribe
      \noexpand\encapchar main%
  }%
```

Maybe add the category index entry:

```
  \noexpand\DTXD@maybecategorylevel{#2}%                 dtxdescribe
\fi%
\noexpand\@esphack}%
```

Define `\Special<name>Index` {⟨*1: name*⟩}

```
\@nameedef{Special#1Index}##1{%
  \noexpand\DTXD@findgtempa{##1}%
```

`\@temptokena`: Index (type).

`\@temptokenb`: Index group.

`\@gtempa`: Detokenized name of the particular object.

`\DTXD@gtempa@nobackslash`: Detokenized name without backslash.

```
  \noexpand\@bsphack%
  \noexpand\DTXD@findcategorylevelname%                  dtxdescribe
```

If is a top level object, index:
`name=\verb!*+name+ (type) [category]|usage`

```
\ifdoc@toplevel%
      \noexpand\doc@providetarget%

%   \noexpand\index{##1\noexpand\actualchar{\string\ttfamily\space##1}%
      \noexpand\index{%
          \noexpand\DTXD@printobjectname%                dtxdescribe
              {\noexpand\DTXD@gtempa@nobackslash}%
              {\noexpand\@gtempa}%
          \ifx\@nil#2\@nil\else \the\@temptokena \fi%
          \noexpand\DTXD@maybecategory%                  dtxdescribe
          \noexpand\doc@handleencap{usage}%
      }%
```

Maybe add the category index entry:

```
      \noexpand\DTXD@maybecategorylevel{#2}%             dtxdescribe
```

```
  \fi%
%
%
```

If group is not empty, index:
`group:>name=\verb!*+name+ [category]|usage`

```
  \ifx\@nil#3\@nil\else%
    \noexpand\index{%
        \the\@temptokenb\noexpand\levelchar%
%       ##1\noexpand\actualchar{\string\ttfamily\space##1}
        \noexpand\DTXD@printobjectname%                    dtxdescribe
            {\noexpand\DTXD@gtempa@nobackslash}%
            {\noexpand\@gtempa}%
        \noexpand\DTXD@maybecategory%                      dtxdescribe
        \noexpand\doc@handleencap{usage}%
    }%
```

Maybe add the category index entry:

```
    \noexpand\DTXD@maybecategorylevel{#2}%                 dtxdescribe
  \fi%
%
%
    \noexpand\@esphack}}
```

`c@createspecialmacrolikeindexes` {⟨*1: name*⟩} {⟨*2: index type*⟩} {⟨*3: index group*⟩}

```
\def\doc@createspecialmacrolikeindexes#1#2#3{%
  \@temptokena{\space (#2)}%
  \@temptokenb{#3:}%
```

Define `\Code<name>Index` {⟨*1:* `main` *or* `usage`⟩} {⟨*2: name*⟩}

```
  \@nameedef{Code#1Index}##1##2{%
    \noexpand\DTXD@findgtempa{##2}%
```

`\@temptokena`: Index type.

`\@temptokenb`: Index group.

`\@gtempa`: Detokenized name of the particular object.

`\DTXD@gtempa@nobackslash`: Detokenized name without backslash.

```
    \noexpand\@bsphack%
    \noexpand\DTXD@findcategorylevelname%                  dtxdescribe
```

```
  \noexpand\ifdoc@noindex\noexpand\else%
```

If a top level object, index:
`name=\verb!*+\name+ (type) [category]|<main or usage>`

```
  \ifdoc@toplevel%
    \noexpand\special@index{%
          \noexpand\DTXD@printobjectname%                 dtxdescribe
              {\noexpand\DTXD@gtempa@nobackslash}%
              {\noexpand\@gtempa}%
          \ifx\@nil#2\@nil\else \the\@temptokena \fi%
          \noexpand\DTXD@maybecategory%                   dtxdescribe
          \noexpand\encapchar ##1%
    }%
```

Maybe add the category index entry:

```
    \noexpand\DTXD@maybecategorylevel{#2}%                dtxdescribe
  \fi%
```

If group is not empty, index:
`group:>name=\verb!*+\name+ [category]|<main or usage>`

```
  \ifx\@nil#3\@nil\else%
    \noexpand\special@index{%
          \the\@temptokenb\noexpand\levelchar%
          \noexpand\DTXD@printobjectname%                 dtxdescribe
              {\noexpand\DTXD@gtempa@nobackslash}%
              {\noexpand\@gtempa}%
          \noexpand\DTXD@maybecategory%                   dtxdescribe
          \noexpand\encapchar ##1%
    }%
```

Maybe add the category index entry:

```
    \noexpand\DTXD@maybecategorylevel{#2}%                dtxdescribe
  \fi%
\noexpand\fi%
  \noexpand\@esphack}%
```

Define `\SpecialMain<name>Index` {⟨*1: name*⟩}

```
\@nameedef{SpecialMain#1Index}##1{%
  \expandafter\noexpand\csname Code#1Index\endcsname%
      {main}{##1}}%
```

Define `\Special<name>Index` {⟨*1: name*⟩}

```
\@nameedef{Special#1Index}##1{%
  \noexpand\DTXD@findgtempa{##1}%
```

`\@temptokena`: Index type.

`\@temptokenb`: Index group.

`\@gtempa`: Detokenized name of the particular object.

`\DTXD@gtempa@nobackslash`: Detokenized name without backslash.

```
  \noexpand\@bsphack%
  \noexpand\DTXD@findcategorylevelname%                 dtxdescribe
```

```
\noexpand\ifdoc@noindex\noexpand\else%
```

If a top level object, index:
`name=\verb!*+\name+ (type) [category]|usage`

```
  \ifdoc@toplevel%
    \noexpand\doc@providetarget%
    \noexpand\index{%
          \noexpand\DTXD@printobjectname%               dtxdescribe
              {\noexpand\DTXD@gtempa@nobackslash}%
              {\noexpand\@gtempa}%
          \ifx\@nil#2\@nil\else \the\@temptokena \fi%
          \noexpand\DTXD@maybecategory%                 dtxdescribe
          \noexpand\doc@handleencap{usage}%
    }%
```

Maybe add the category index entry:

```
  \noexpand\DTXD@maybecategorylevel{#2}%                dtxdescribe
  \fi%
```

If group is not empty, index:
`group:>name=\verb!*+\name+ [category]|usage`

```
  \ifx\@nil#3\@nil\else%
    \noexpand\index{%
          \the\@temptokenb\noexpand\levelchar%
          \noexpand\DTXD@printobjectname%               dtxdescribe
              {\noexpand\DTXD@gtempa@nobackslash}%
              {\noexpand\@gtempa}%
          \noexpand\DTXD@maybecategory%                 dtxdescribe
          \noexpand\doc@handleencap{usage}%
    }%
```

Maybe add the category index entry:

```
    \noexpand\DTXD@maybecategorylevel{#2}%              dtxdescribe
  \fi%
\noexpand\fi%
  \noexpand\@esphack}}

}{}% doc loaded
```

6.13 Not doc: `\DescribeMacro` and `\DescribeEnvironment`

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

Redefined to allow hyperlinked index entries and an optional category:

```
\IfPackageLoadedTF{doc}{}{% not doc

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

Create the margin tag with the macro's name:

```
\@ifundefined{@captype}{% not float?
\leavevmode%
\marginpar{%
    {%
        \hbadness=10000%
        \hfuzz=5em%
        \raggedleft%
        \ifblank{#1}{}{{\scriptsize\textsf{[#1]}} }% category
        \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 category if given.

Write name=csname>[category]`|usage`

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

Only if a category was given:

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

}% not doc
```

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

Redefined to allow hyperlinked index entries:

```
\IfPackageLoadedTF{doc}{}{% not doc

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

}% not doc
```

6.14 New `\Describe...` macros

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

The name may have underscores.

```
\IfPackageLoadedTF{doc}{}{% not doc
\newcommand*{\DTXD@filename}{}

\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%
}
}% not doc
```

`\DescribeMacro` Redefine with new definitions.

`\DescribeEnvironment` Redefine with new definitions.

```
\IfPackageLoadedTF{doc}{% doc

\RenewDocElement[macrolike = true ,
                 idxtype   = ,
                 idxgroup  = ,
                 printtype =
                ]{Macro}{macro}

\RenewDocElement[macrolike = false ,
                 idxtype   = env.  ,
                 idxgroup  = environments ,
                 printtype = \textit{env.}
                ]{Env}{environment}

}{}% doc
```

`\DescribeFile` {⟨*name*⟩}

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=file,
    idxgroup=Files,
    printtype=\textit{file}
]{File}{file}

}{% not doc

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

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

}% not doc
```

`\DescribeProgram` {⟨*name*⟩}

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=program,
    idxgroup=Programs,
    printtype=\textit{Prog}
]{Program}{program}

}{% not doc

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

\newcommand*{\DescribeProgram}{%
    \begingroup\catcode`\_=12 \DTXD@DescribeProgram%
}
}% not doc

```

`\DescribeCommand` {⟨*name*⟩}

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=command,
    idxgroup=Commands,
    printtype=\textit{Cmd}
]{Command}{command}

}{% not doc

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

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

}% not doc
```

`\DescribePackage` {⟨*name*⟩}

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=package,
    idxgroup=Packages,
    printtype=\textit{Pkg}
]{Package}{package}

}{% not doc

\newcommand*{\DTXD@DescribePackage}[2][]{%
    \DTXD@filemarginparindex{#1}{#2}{Pkg}{package}{usage}%
}

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

}% not doc

```

`\DescribeClass` {⟨*name*⟩}

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=class,
    idxgroup=Classes,
    printtype=\textit{Cls}
]{Class}{class}

}{% not doc

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

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

}% not doc
```

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=option,
    idxgroup=Options,
    printtype=\textit{Opt}
]{Option}{option}

}{% not doc

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

}% not doc
```

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

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=argument,
    idxgroup=Arguments,
    printtype=\textit{Arg}
]{Argument}{argument}

}{% not doc

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

}% not doc
```

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=boolean,
    idxgroup=Booleans,
    printtype=\textit{bool}
]{Boolean}{boolenv}

}{% not doc

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

}% not doc
```

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=true,
    toplevel=true,
    idxtype=length,
    idxgroup=Lengths,
    printtype=\textit{Len}
]{Length}{length}

}{% not doc

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

}% not doc
```

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=counter,
    idxgroup=Counters,
    printtype=\textit{Ctr}
]{Counter}{counter}

}{% not doc

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

}% not doc
```

`\DescribeHook` [⟨*category*⟩] {⟨*name*⟩}

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=true,
    toplevel=true,
    idxtype=hook,
    idxgroup=Hooks,
    printtype=\textit{Hook}
]{Hook}{hook}

}{% not doc

\newcommand*{\DescribeHook}[2][]
    {\DTXD@margintagindex{#1}{#2}{Hook}{hook}{usage}}

}% not doc
```

`\DescribeSocket` [⟨*category*⟩] {⟨*name*⟩}

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=socket,
    idxgroup=Sockets,
    printtype=\textit{Socket}
]{Socket}{socket}

}{% not doc

\newcommand*{\DescribeSocket}[2][]
    {\DTXD@margintagindex{#1}{#2}{Socket}{socket}{usage}}

}% not doc
```

`\DescribePlug` [⟨*category*⟩] {⟨*name*⟩}

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=plug,
    idxgroup=Plugs,
    printtype=\textit{Plug}
]{Plug}{plug}

}{% not doc

\newcommand*{\DescribePlug}[2][]
```

```
    {\DTXD@margintagindex{#1}{#2}{Plug}{plug}{usage}}

}% not doc
```

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

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=key,
    idxgroup=Keys,
    printtype=\textit{Key}
]{Key}{key}

}{% not doc

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

}% not doc
```

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

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=object,
    idxgroup=Objects,
    printtype=,
]{Object}{object}

}{% not doc

\newcommand*{\DescribeObject}[2][]{%
\@ifundefined{@captype}{% not float?
    \@bsphack%
    \leavevmode%
    \marginpar{%
        \hbadness=10000%
        \hfuzz=5em%
        \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%
}

}% not doc
```

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

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

```
\IfPackageLoadedTF{doc}{% doc

\NewDocElement[
    macrolike=false,
    toplevel=true,
    idxtype=other,
    idxgroup=Other,
    printtype=,
]{Other}{other}

}{% not doc

\newcommand*{\DescribeOther}[2][]{%
\@ifundefined{@captype}{% not float?
    \@bsphack%
    \leavevmode%
    \marginpar{%
        \hbadness=10000%
        \hfuzz=5em%
        \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%
}

}% not doc
```

6.15 `\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}%
        }%
    }%
}
```

6.16 `\ItemDescribeMacro`, etc.

The following are for use inside a `description`.

`\ItemDescribeMacro` [⟨*category*⟩] {⟨\name⟩}

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

`\ItemDescribeEnv` [⟨*category*⟩] {⟨*name*⟩}

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

`\ItemDescribeArgument` [⟨*category*⟩] {⟨*argument*⟩}

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

`\ItemDescribeBoolean` [⟨*category*⟩] {⟨*name*⟩}

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

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

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

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

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

`\ItemDescribeHook` [⟨*category*⟩] {⟨*name*⟩}

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

`\ItemDescribeSocket` [⟨*category*⟩] {⟨*name*⟩}

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

`\ItemDescribePlug` [⟨*category*⟩] {⟨*name*⟩}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.17 `\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?
}
```

6.18 Nesting

Shows a box enclosing a label for the container, and the container's contents. May be nested.

`\shownesting` `[`⟨*fraction of* `\linewidth`⟩`]` `{`⟨*container*⟩`}` `{`⟨*contents*⟩`}`

```
\NewDocumentCommand{\shownesting}{s O{1} m m}{
    \IfBooleanF{#1}{
        \par\smallskip
    }
    \fbox{
        \begin{minipage}{#2\linewidth-2em}
            \hbadness=10000\relax%
            #3\par\smallskip
            \hspace{1em}
            \begin{minipage}{\linewidth-1.5em}
                #4
            \end{minipage}
        \end{minipage}
    }
}
```

6.19 The `dtxexample` environment

Also see example 15 on page 28.

`dtxexample_cut.tex` (*file*) Used to store the `\input` example code.

`DTXD@examplerulecolor` [color] 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:}
```

`dtxexample` (*env.*) `*` `[`⟨*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, else use a float:

```
\IfBooleanTF{#1}%
    {% minipage
        \minipage{\linewidth}%
        \captionsetup{type=dtxexamplefloat}%
    }%
    {% float
        \begin{dtxexamplefloat}%
    }%
```

```
\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 float or `\minipage`.

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

`\DeclareFloatingEnvironment` [dtxexamplefloat]

A new float type for the examples.

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

`\captionsetup` [dtxexamplefloat]

Caption setup for the examples.

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

`\crefname` [dtxexamplefloat]

Name for cleveref. `\crefname` here is required for documents not using the doc class:

```
%\AddToHook{begindocument/after}{% Before .aux file is loaded.
\AtBeginDocument{
    \ifdef{\crefname}{
        \crefname{dtxexamplefloat}{example}{examples}
    }{}
}
```

6.20 `noindmacro` and `noindenvironment`

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

`noindmacro` (*env.*) {⟨*name*⟩}

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

`noindenvironment` (*env.*) {⟨*name*⟩}

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

6.21 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}}
```

`sourceverb` (*env.*) To typeset a block of source code, verbatim.

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

`fsourceverb` (*env.*) 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}}
```

`sourcedisplay` (*env.*) To typeset a block of source code, allowing direct formatting.

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

`UIdisplay` (*env.*) 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
}
```

`docsidebar` (*env.*) To typeset a sidebar in the documentation.

```
\newenvironment{docsidebar}[1][]
{%
    \quote\unskip\medskip
    \setlength{\parskip}{1.5ex}%
    \ifblank{#1}{}{\textit{#1}\newline}%
    \rule[.5\bigskipamount]{\linewidth}{.4pt}%
    \newline%
}
{%
    \leavevmode\par
    \rule[\bigskipamount]{\linewidth}{.4pt}
    \endquote\unskip
}
```

6.22 Formatted objects

Macros to format references to various kinds of objects.

6.22.1 LaTeX objects

`\pkg` {⟨*name*⟩} Also useable for class names.

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

`\cs` {⟨*csname*⟩} From ltxdoc.

```
\providerobustcmd*{\cs}[1]{\texttt{\char‘\\#1}}
```

`\env` {⟨*name*⟩}

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

`\marg` {⟨*argument*⟩} From ltxdoc.

```
\providecommand\marg[1]{%
  {\ttfamily\char‘\{}\meta{#1}{\ttfamily\char‘\}}}
```

`\oarg` [⟨*argument*⟩] From ltxdoc.

```
\providecommand\oarg[1]{%
  {\ttfamily[}\meta{#1}{\ttfamily]}}
```

`\parg` (⟨*argument*⟩) From ltxdoc.

```
\providecommand\parg[1]{%
  {\ttfamily(}\meta{#1}{\ttfamily)}}
```

`\ctr` {⟨*name*⟩}

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

`\bool` {⟨*name*⟩}

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

`\optn` {⟨*name*⟩}

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

`\TOC`

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

`\LOF`

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

`\LOT`

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

6.22.2 Programs and commands

`\cmds` {⟨*commands to print*⟩} No processing is provided for special characters.

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

`\progcode` {⟨*code to print*⟩} No processing is provided for special characters.

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

`\prog` {⟨*program name*⟩} Underscores are allowed.

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

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

`\filenm` {⟨*file name*⟩} Underscores are allowed.

```
\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}}}
```

6.22.3 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}}
```

6.22.4 Internet

`\UTF`

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

`\URL`

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

`\element` {⟨*name*⟩}

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

`\attribute` {⟨*name*⟩}

`\attrib` {⟨*name*⟩}

Each of these is "provided", and any prior meaning will be unchanged. In particular, LuaTeX uses `\attribute`, so its meaning is unchanged if using LuaTeX.

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

\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}}
```

6.22.5 Specific programs

`\TikZ`

```
\providerobustcmd*{\TikZ}{Ti\emph{k}Z}
```

`\CTAN`

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

`\TDS`

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

`\MathML`

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

`\MathJax`

```
\providerobustcmd*{\MathJax}{\brand{MathJax}}
```

6.22.6 Acronyms, brand names, trademarks

`\brand` {⟨*name*⟩}

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

`\acro` {⟨*acronym*⟩}

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

`\supregistered` Superscript trademark symbol.

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

6.23 Logos

`\dviTeX` DVI TeX

```
\providerobustcmd*{\dviTeX}{\mbox{\DVI\,\TeX}}
```

`\dviLaTeX` DVI LaTeX

```
\providerobustcmd*{\dviLaTeX}{\mbox{\DVI\,\LaTeX}}
```

`\pdfTeX` PDF TeX

```
\providerobustcmd*{\pdfTeX}{\mbox{\PDF\,\TeX}}
```

`\pdfLaTeX` PDF LaTeX

```
\providerobustcmd*{\pdfLaTeX}{\mbox{\PDF\,\LaTeX}}
```

`\LuaTeX` LuaTeX

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

`\LuaLaTeX` LuaLaTeX

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

`\XeTeX` `\XeLaTeX` XƎTeX, XƎLaTeX

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

\AtBeginDocument{
\IfPackageLoadedTF{graphics}{
    \renewrobustcmd*{\XeTeXrevE}
        {\hspace{-.1667em}\raisebox{-.5ex}{\reflectbox{E}}\hspace{-.125em}}
}{}
}

\providerobustcmd*{\XeTeX}{\mbox{X\XeTeXrevE\TeX}}
\providerobustcmd*{\XeLaTeX}{\mbox{X\XeTeXrevE\LaTeX}}
```

`\AmS` $\mathcal{AMS}$

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

`\LyX` LyX

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

`\BibTeX` BibTeX

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

`\MakeIndex` *MakeIndex*

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

`\ConTeXt` ConTEXt

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

`\MiKTeX` MiKTEX

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

6.24 Dashes and slashes

`\thinskip` A breakable thin skip.

```
\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)

A Change History

- v0.10
  - General: 2016/12/08 Initial ver 1
- v0.11
  - General: 2018/03/30 1
  - `\DTXD@cmdmargintagindex`: Index tag no longer plural. 43
  - `\DTXD@index`: Index tag no longer plural. 41
  - `\watchout`: Changed to `\raggedleft`. 67
- v1.00
  - General: 2019/01/11 1
    - Added formatted objects. 73
    - Added logos. 77
    - Cut file name changed to `dtxexample_cut.tex` 68
  - `\DescribeClass`: Fix: Allow underscore. 57
  - `\DescribeDefault`: Added. 63
  - `\DescribeDefaultcolor`: Added. 63
  - `\DescribeMacro`: Sans tag font. 53
  - `\DescribePackage`: Fix: Allow underscore. 57
  - `docsidebar`: Added. 72
  - `\DTXD@cmdmargintagindex`: Sans tag font. 42
  - `\DTXD@filemarginparindex`: Fix: File category. 54
  - `\DTXD@printtype`: Sans tag font. 39
  - `\dtxexamplecodename`: Added. 68
  - `\dtxexampleresultname`: Added. 68
  - `\fqqquad`: Added. 71
  - `\fqquad`: Added. 71
  - `\fquad`: Added. 71
  - `fsourceverb`: Added. 71
  - `\ItemDescribeArgument`: Added. 64
  - `\ItemDescribeBoolean`: Added. 64
  - `\ItemDescribeClass`: Added. 65
  - `\ItemDescribeCommand`: Added. 66
  - `\ItemDescribeCounter`: Added. 64
  - `\ItemDescribeEnv`: Added. 64
  - `\ItemDescribeFile`: Added. 66
  - `\ItemDescribeKey`: Added. 65
  - `\ItemDescribeLength`: Added. 64
  - `\ItemDescribeMacro`: Added. 63
  - `\ItemDescribeObject`: Added. 66
  - `\ItemDescribeOption`: Added. 65
  - `\ItemDescribeOther`: Added. 67
  - `\ItemDescribePackage`: Added. 65
  - `\ItemDescribeProgram`: Added. 66
  - `\margintag`: Uses `\margintacolor`. 67
  - `\margintagcolor`: Added. 67
  - `noindenvironment`: Added. 71
  - `noindmacro`: Added. 70
  - `sourcedisplay`: Added. 71
  - `sourceverb`: Added. 71
  - `UIdisplay`: Added. 72
  - `\userentry`: Added. 72
  - `\userentryname`: Added. 72
  - `\watchoutcolor`: Added. 67
- v1.01
  - General: 2019/03/22 1
    - Sanitize PDF bookmarks. 33
  - `\DescribeMacro`: Put margin tag category in brackets. 53
  - `\DescribeObject`: Put margin tag category in brackets. 61
  - `\DescribeOther`: Put margin tag category in brackets. 62
  - `\DTXD@cmdmargintagindex`: Put margin tag category in brackets. 42
  - `\DTXD@margintag`: Put margin tag category in brackets. 40
  - `sourcedisplay`: Reduced width. 71
  - `UIdisplay`: Reduced width. 72
- v1.02
  - General: 2019/07/16 1
    - Fix if not doc package. 38
    - Fix if not hyperref package. 33
    - Fix if not ltxdoc class. 33, 38
  - `\DescribeEnv`: Fix if not ltxdoc class. 54
  - `\DescribeMacro`: Fix if not ltxdoc class. 53
  - `\DescribeObject`: `\raggedleft` margin par. 61
    - Added `\ignorespaces`. 61
  - `\DescribeOther`: `\raggedleft` margin par. 62
    - Added `\ignorespaces`. 62
  - `dtxexample`: Fix if not doc package. 68
  - `\PrintEnvName`: Fix if not ltxdoc class. 39
  - `DTXD@gobble`: Fix if not doc package. 37
  - `\usage`: Fix if not ltxdoc class. 39
- v1.03
  - General: 2022/02/01 1
    - No longer requires xifthen. 33
  - `\attrib`: For LuaTEX. 76
  - `\cs`: Provided. 73
  - `\DescribeHook`: Added. 60
  - `\DescribeMacro`: Reduce hbox warnings. 53
  - `\DescribeObject`: Reduce hbox warnings. 61

- \DescribeOther: Reduce hbox warnings. 62
- \DTXD@cmdmargintagindex: Reduce hbox warnings. 42
- \DTXD@margintag: Reduce hbox warnings. 40
- \dviLaTeX: Added. 77
- \dviTeX: Added. 77
- \ItemDescribeHook: Added. 64
- \marg: Provided. 73
- \MathJax: Added. 77
- \oarg: Provided. 73
- \parg: Provided. 73
- \pdfLaTeX: Added. 78
- \pdfTeX: Added. 78
- \shownesting: Added. 68

v1.04
- General: 2022/02/01 1
  - Corrected copyright date. 1

v1.05
- General: 2022/09/08 1
- \TikZ: Renamed from \tikz, cap Z. 77

v1.06
- General: 2022/12/07 1
  - Fixed for updated doc. 35, 36, 43, 44
- \@NewDocElement: Fixed for updated doc. 44

v1.07
- General: 2023/01/03 1
  - Improved PDF bookmarks. 36, 37
- \doc@createspecialmacrolikeindexes: Added category names for macros. 50
- \DTXD@findgtempa: Detokenized names. 47
- \DTXD@maybecategory: \verb*. 46
  - Improved line breaks. 46
- \DTXD@maybecategorylevel: \verb*. 46
  - Fix occasional crash. 47
- \DTXDbreak: Improved line breaks. 45

v1.08
- General: 2024/01/08 1
  - Updated contact info. 1
- \DescribePlug: Added. 60
- \DescribeSocket: Added. 60
- \ItemDescribePlug: Added. 65
- \ItemDescribeSocket: Added. 64

v1.09
- General: 2024/01/20 1
  - Improved TOC change log and index entries. 1
  - Removed extra spaces. 44
- \doc@createspecialindexes: Removed extra spaces. 49
- \doc@createspecialmacrolikeindexes: Removed extra spaces. 51

B Index