texdoc

Finding & viewing TEX documentation

Manuel Pégourié-Gonnard*

v0.4—2008/08/01

1 Basic Usage, Modes

1.1 `texdoc ⟨name⟩`

The simplest way to use texdoc is just to type[1] `texdoc` followed by the name of the package whose documentation you want to read. It usually finds the documentation for you and opens it in the appropriate reader. That's it: easy and usually fast. The rest of this manual describes what to do if this doesn't work exactly as you like and you want to customise things, and how to do more extensive searchs.

Before the description of texdoc's different modes, just a word words about the typographic conventions in this manual. Things like ⟨*name*⟩ in the above title mean that they should be replaced by what you actually want. For example, if you want to read hyperref's manual, type `texdoc hyperref`. Sometimes there will be complete examples like this:

```
➥ texdoc -s babelbib
1 /usr/local/texlive/2008/texmf-dist/doc/latex/babelbib/babelbib.pdf
2 /usr/local/texlive/2008/texmf-dist/doc/latex/babelbib/tugboat-babelbib.pdf
3 /usr/local/texlive/2008/texmf-dist/doc/latex/babelbib/ChangeLog
4 /usr/local/texlive/2008/texmf-dist/doc/latex/babelbib/README
Please enter the number of the file to view, anything else to skip: 2
```

In this case, what you actually type is in red, and the funny symbol ➥ represents your shell's prompt, which can actually be something like `C:\>` or `name@host:~%` or funnier.

Now, let's talk about this `-s` option you've just seen in the previous example.

---

* for the documentation. The script itself was last updated and revised by Manuel Pégourié-Gonnard with contributions from Reinhard Kotucha, based on the previous texlua versions by Frank Küster. Original (bash) shell script by Thomas Esser, David Aspinall, and Simon Wilkinson. Please sent comments, bug reports, and feature requests to texlive@tug.org.

[1] In a command line. If you don't know how to open one, look for Start→Execute and type `cmd` on Windows, or use the "terminal" icon on Mac OS X. If you are using another flavour of Unix, you probably know what to do.

1.2

```
texdoc -s ⟨name⟩
texdoc --search ⟨name⟩
```

With the two (equivalent) commands above, texdoc also looks for documentation for ⟨*name*⟩, but using the *search mode*, which differs from the normal mode (called *view mode*) on two points:

- It doesn't start a viewer and offers you a *menu* instead.
- It always do a *full search*.

The first point is rather straightforward on the example. The second deserves more explanation.

Usually, texdoc looks for files named ⟨*name*⟩`.pdf` or ⟨*name*⟩`.html` etc. (see 4.4.4), where ⟨*name*⟩ means what you asked for, in TEX Live's documentation directories, and if cannot find such a file, it tries a full search: it finds all files which have ⟨*name*⟩ in their name, or in the directory's name. In search mode, texdoc always performs a full search.

Now look carefully at the previous example. The purpose of search mode is to allow you to find related documentation, such as the TUGboat article on babelbib, which you might want to read, whereas in normal mode texdoc offers you no choice and just displays the user manual `babelbib.pdf`. On the other hand, the view mode is much faster when you know exactly what you want to read.

To try and make you happy, texdoc offers three other modes, introduced below.

1.3

```
texdoc -l ⟨name⟩
texdoc --list ⟨name⟩
```

The *list mode* uses a normal search, but forces texdoc to give you a menu instead of choosing itself the documentation to display. It is usefull when there are many files with the same name but different contents, or many versions of the same file on your system.

```
➥ texdoc -l tex
1 /usr/local/texlive/2008/texmf/doc/man/man1/tex.pdf
2 /usr/local/texlive/2008/texmf-doc/doc/english/knuth/tex/tex.pdf
Please enter the number of the file to view, anything else to skip:
```

Here the first file is the manual page[2] of the `tex` command, while the second is TEX's documented source code...

1.4

```
texdoc -m ⟨name⟩
texdoc --mixed ⟨name⟩
```

As the name says, *mixed mode* is an attempt to provide you the best of the normal (view) and list modes, by mixing them in the following way: If only one file is found, the texdoc opens it, and if many are found, it displays a menu to let you choose. You may want to make this mode the default, see 4.4.2.

[2] converted in pdf. To allow texdoc to find and display real man pages in man format, see 4.4.4.

1.5

```
texdoc ⟨name1⟩ ⟨name2⟩ ⟨...⟩
texdoc ⟨name.ext⟩
```

To conclude this section on basics, let us just mention two points concerning the ⟨*name*⟩ in all previous sections. Is is usually a single name without extension, but you can also use many names at once: then, depending on the mode, texdoc will either open all the corresponding documentation or show you menus for each of the names you mentioned. For each name, you can also specifiy the file exention if you want, eg `texdoc texlive-en.html` lets you read the TeX Live manual in html rather than in pdf format.

You can now stop reading this manual unless you have special needs. If you want to understand the curious `aliased too` messages that you will sometimes see, and control them, read section 2. If you have problems viewing certain type of files or want to choose you preferred reader, look at section 3. Finally, section 4 is the full reference concerning texdoc configuration: while you probably don't want to read it all at once, you can consult 4.4.2 if you want to select your preferred mode and make it the default.

Finally, be aware of the `-h` or `--help` option which provides you a quick reminder of all available command-line options.

2 Aliases, or name substitution

2.1 Basic concept

The usual search modes of texdoc assume that the name of the documentation file is the name of the package, or contains it (at least in the directory name). However, this is not always true, due either to the author choosing a fancy name, or packaging peculiarities. To try helping the user to find the doc even in these cases, texdoc provides an alias mechanism and comes with a list of circa 200 pre-defined aliases.

```
➥ texdoc -l geometry
texdoc info: geometry aliased to geometry/manual.pdf
1 /usr/local/texlive/2008/texmf-dist/doc/latex/geometry/manual.pdf
Please enter the number of the file to view, anything else to skip: 0
```

The concept of alias is very[3] simple: as you can see of the above example, when you type and `geometry` is aliased to `geometry/manual.pdf`, then everything happens as if you actually typed `texdoc geometry/manual.pdf` (without any further alias substitution), and texdoc informs you that something happened so you can understand the results (see 4.4.7 to get rid of this message):

[3] See 2.4 for why it is actually *too* simple.

2.2

```
texdoc -a ⟨options⟩ ⟨name⟩
texdoc --alias ⟨options⟩ ⟨name⟩
texdoc -A ⟨options⟩ ⟨name⟩
texdoc --noalias ⟨options⟩ ⟨name⟩
```

By default, aliased are used in view, list and mixed modes, and disabled in search mode. But you may want to disable it, because the default alias doesn't do what you want[4] or for another reason. In this case, you just have to add `-A` or `--noalias` to the options, like:

```
➥ texdoc -A -l geometry
1 /usr/local/texlive/2008/texmf-doc/doc/polish/tex-virtual-academy-pl/
latex2e/macro/geometry.html
Please enter the number of the file to view, anything else to skip: 0
```

On the contrary, you can force aliasing in search mode by using the `-a` or `--alias` option, though it may not prove very useful.

2.3 Your own aliases

You can define your own aliases, or override the default ones, in texdoc's configuration files. You can get a list of those files by typing `texdoc -f`. For personal aliases, it is recommended that you use the second file, marked by a star (see 4.1 for details). You'll probably need to create in and one or two of the directories containing it.

Creating an alias is easy: you just insert a line like

```
alias geometry = geometry/manual.pdf
```

in your configuration file, and it's all. You can have a look at the configuration file provided (the last one showed by `texdoc -f`) for examples. If you want to permanently unalias something, just insert a line ⟨*name*⟩=⟨*name*⟩: it will overwrite the previous alias.

2.4 Remarks on aliases

Please be aware that this alias feature, or at least its intensive use to try to find the "right" documentation for a given package, should be temporary. Indeed, one problem is that currently aliases do *hide* other files, while it is desirable that they just *add* results in some case. However, defining a coherent behaviour (and how to maintain the needed database) requires work and time, and is therefore reported to future versions.

In this vein, it would be desirable to have a notion of "category", like user documentation of a package, or man page of a program, or reference manual of a program, or documented source code of a package or program, or... If you have ideas about desirable categories and ways they should be handled, feel free to share them at the usual address.

[4] In this case, please report it to texlive@tug.org.

3 Viewer selection

A list of default viewers is defined in texdoc, depending on your platform (Windows, MacOS X, other Unix). On Windows and MacOS, it uses your file associations like when you double-click files in the Explorer or the Finder. On Unix, it tries to find a viewer in the path from a list of "known' viewers.

If you want to use another viewer, you have two ways of telling this to texdoc: in your configuration file or using environment variables. If you hesitate, the configuration file is the recommended way.

To find you configuration file, type `texdoc -f` and pick the file mark with a star (unless you are a system administrator or your home is shred between many machines, see 4.1); you may need to create the file and a few directories. Then you can add lines like:

```
viewer_pdf = (xpdf %s) &
viewer_txt = less
```

Here the `%s` stands for the name of the file to view. The first line sets `xpdf` as the pdf viewer, and use a bit of shell syntax to force it to run in the background (the `()` are here for compatibility with zip support, see 5). The second line sets `less` as the text viewer: it doesn't use `%s`, which means the filename will be placed at the end of the command.

The default extensions allowed are `pdf`, `html`, `txt`, `dvi`, `ps`, and no extension. The `txt` viewer is used for files without extension. See 4.4.4 for how to allow for more extensions.

The corresponding environment variables are `PDFVIEWER`, `BROWSER`, `PAGER`, `DVIVIEWER`, `PSVIEWER`. They follow the same convention as values from the configuration files, and override them if they are set. Since some of those variable are shared by other programs, you can override them by adding `_texdoc` at the end, like in `BROWSER_texdoc`.

4 Full reference

The most useful command-line options, configuration values and all environment variables have been presented. Here we complete our presentation and review all in a systematic way.

4.1 Precedence

Values for a particular setting can come from several sources. They are treated in the following order, where first value found is always used:

1. Command-line options.
2. Environment variables ending with `_texdoc`.
3. Other environment variables.
4. Values from configuration files, read in the following order:
   a) `$TEXMFHOME/texdoc/texdoc-⟨platform⟩.cnf`
   b) `$TEXMFHOME/texdoc/texdoc.cnf`
   c) `$TEXMFLOCAL/texdoc/texdoc-⟨platform⟩.cnf`

d) `$TEXMFLOCAL/texdoc/texdoc.cnf`
e) `$TEXMFMAIN/texdoc/texdoc.cnf`

5. Hard-coded defaults that may depend on the current platform or the current value of another setting.

For the configuration files, ⟨*platform*⟩ stand for the name of the current platform, with names matching those of the directories in `TEXLIVEROOT/bin`, and `TEXMFHOME` and others are the kpse's values, see the kpathsea and web2c manuals. The name with ⟨*platform*⟩ can be used on installation shared between many machines where, for example, not the same viewers are available. However, their use is not recommended in other situations.

4.2 Command-line options

Most of the command-line options correspond to an option that can be set from the config files. For them, we refer the reader to the description of the corresponding configuration option.

4.2.1 `-h, --help` — Shows a quick help message (namely a list of command-line options) and exits successfully.

4.2.2 `-v, --version` — Show the current version of the program and exits successfully.

4.2.3 `-f, --files` — Shows the list of the configuration files for the current installation and platform, with their status (active or not found) and a star marking the recommended file for user settings.

4.2.4 `-w, --view, -l, --list, -m, --mixed, -s, --search, -r, --regex` — See 4.4.2.

4.2.5 `-a, --alias, -A, --noalias` — See 2.

4.2.6 `-i, --interact, -I, --nointeract` — See 4.4.3.

4.2.7 `-e=`⟨*l*⟩`, --extensions=`⟨*l*⟩ — See 4.4.4. *But* be aware that on the command line there should be no space at all, neither in the list (unless quoted according to you shell's convention) not between the `-e` or `--extension` option, the equal sign, and the list. Also take care to quote the special value `*` if necessary. The equal sign is optional.

4.2.8 `-n=`⟨*n*⟩`, --noise-level=`⟨*n*⟩ — See 4.4.7 and be aware that you must avoid spaces on the command line, and the `=` sign is optional.

4.3 Environment variables

They all correspond to some `viewer_`⟨*ext*⟩ setting, and the reader is referred to 3 and 4.4.5 for details. Here is the (obvious) correspondence:

`PAGER_texdoc`	`PAGER`	`viewer_html`
`BROWSER_texdoc`	`BROWSER`	`viewer_txt`
`DVIVIEWER_texdoc`	`DVIVIEWER`	`viewer_dvi`
`PSVIEWER_texdoc`	`PSVIEWER`	`viewer_ps`
`PDFVIEWER_texdoc`	`PDFVIEWER`	`viewer_pdf`

4.4 Configuration files

4.4.1 *General structure.* — Configuration files are line-oriented text files. Comments begin with a `#` and run to the end of line. Lines containing only space are ignored. Space at the beginning or end of a line, as well as around an `=` sign, is ignored. Apart from comments and empty lines, each line must be of one of the following forms:

⟨*config_param*⟩ `=` ⟨*value*⟩
`alias` ⟨*name*⟩ `=` ⟨*target*⟩

where ⟨*config_parameter*⟩ consists of only letters, digits or `-` signs, ⟨*name*⟩ of letters, digits, `-` and `_` signs. ⟨*value*⟩ and ⟨*target*⟩ are free strings (except that not every ⟨*value*⟩ is valid for every ⟨*config_param*⟩, see below) and nothing in it need not be quoted (actually, quotes will be interpreted as part of the value, not as quotation marks).

Lines which do not obey these rules raise a warning. However, unrecognised values of ⟨*config_param*⟩ raise no warning at the moment.

The ⟨*value*⟩ is usually interpreted as a string, except when ⟨*config_param*⟩ ends with:

1. `_list`, then ⟨*value*⟩ is a coma-separated list of strings. Space around commas is ignored. Two consecutive comas or a coma at the beginning or end of the list means the empty string at the corresponding place.
2. `_switch`, then ⟨*value*⟩ must be either `true` or `false` (case-sensitive).
3. `_level`, then ⟨*value*⟩ is a non-negative integer.

4.4.2 `mode =` ⟨*view, list, mixed, search, regex*⟩ — Set the mode to the given value. Default is `view`. The first three values `view`, `list`, `mixed` use the same searching method: first search a file whose name is the ⟨*name*⟩ on the command line and whose extension is in `ext_list` (see 4.4.4), and if nothing is found, then do a full search. This means that a file matches if ⟨*name*⟩ is a substring of its path+name (and its extension is in the list). Here path does not mean the full path, but only the part below `TEXMF/doc`. The `search` mode forces a full search.

The last mode, `regex`, looks for ⟨*name*⟩ in the path+filename as a Lua regex. If you don't know Lua regexes you should be aware that the escape character is `%` and the `-` sign is a special character (which means the same as `*?` in Perl regexes). For more

details, see the Lua reference manual or the book *programming in Lua*. You might want to use[5] `-e='*'` if your regex uses the `$` anchor.

4.4.3 `interact_switch = ⟨true, false⟩` — Turn on or of interaction. Default is on. Turning interaction off prevents texdoc to ask you to choose a file to view when there are multiple choices, and merely just print the list of files found.

4.4.4 `ext_list = ⟨list⟩` — Set the list of recognised extensions to ⟨*list*⟩. Default is

```
pdf, html, txt, dvi, ps,
```

This list is used to filter and sort the results (with the default value: pdf first, etc). Two special values are recognised:

- *The empty element.* This means files without extensions, or more precisely without a dot in their name. This is meant for files like `README`, etc. The file is assumed to be plain text for viewing purpose.
- `*` means any extension. Of course if it is present in the list, it can be the only element!

The value of `ext_list` is overridden on a file per file basis if the given argument already has an extension: the `ext_list` is temporarily set to this extension just for this argument.

For each ⟨*ext*⟩ in `ext_list` there should be a corresponding `viewer_`⟨*ext*⟩ value set. Defaults are defined corresponding to the default `ext_list`, but you can add values if you want. For example, if you want texdoc to be able to find man pages and display them with the `man` command, you can use

```
ext_list = 1, 5, pdf, html, txt, dvi, ps,
viewer_1 = man
viewer_5 = man
```

(This also makes man pages in man format take precedence over their pdf versions.)

4.4.5 `viewer_`⟨*ext*⟩ `=` ⟨*cmd*⟩ — Set the viewer command for files with extension ⟨*ext*⟩ to ⟨*cmd*⟩. For files without extension, `viewer_txt` is used, and there's not `viewer_` variable. In ⟨*cmd*⟩, `%s` can be used as a placeholder for the file name, which is otherwise inserted at the end of the command. The command can be a arbitrary shell construct.

4.4.6 `alias` ⟨*name*⟩ `=` ⟨*othername*⟩ — Everything has already been said in section 2.

4.4.7 `noise_level = ⟨n⟩` — Set the verbosity level to ⟨*n*⟩. This determines whether texdoc will print or not errors or debug information (to stderr). Default level is 3. The numeric codes are as follow:

0. Print nothing (not recommended).
1. Print only error messages.

[5] The quotes in the example are just to make the shell happy.

2. Also print warnings.
3. Also print information messages.
4. Also print debug1 information messages.

Currently, the levels 5 and greater are not used, and the only "debug" information at level 4 is to print the command used to view a file just before executing it.

4.5 Exit codes

The current exit code are as follow:

0. Success.
1. Syntax error.
2. Documentation not found for at least one argument.

5 Bugs, warnings

There is currently no known bug (fingers crossed). But a few things you should be warned about.

First of all, texdoc doesn't always succeed in finding documentation (or finds so many results that it is not useful). Moreover, it cannot handle very correctly packages with many relevant documentation files at the moment (see 2.4). Ideas about how to improve this are most welcome at the usual address.

Second, support for zipped documentation, which have been "available" in previous versions of texdoc, is now disabled by default. The reasons are that this support wasn't portable (didn't work on windows for example), and moreover we won't ship compressed documentation in TeX Live. However, the code has not been totally removed and should be easy to activate again. If you want to use this feature, please:

1. Look in texdoc's code for instructions.
2. Be aware that the corresponding part of the code has not been well tested, and that you are responsible of the unzipping command anyway.
3. Inform us either at the usual address or tldistro@tug.org if you are a Linux, BSD, or whatever, distributor.

Finally, texdoc is also missing a GUI version (texdoctk has never been the GUI version of texdoc, and is unmaintained and probably unmaintainable anyway). This is on the list, but the time line is rather unclear at the moment.

Happy TeXing!