texdoc

Find & view documentation in TEX Live

http://tug.org/texdoc/

Manuel Pégourié-Gonnard

v0.80 2011-05-18

1 Quick guide (2 pages only)

1.1 Basics

Open a command line[1] and type `texdoc` ⟨*name*⟩: the documentation of the ⟨*name*⟩ package will pop up. Of course, replace ⟨*name*⟩ with the actual name of the package you want to learn about. You can also look for the documentation of more than one package at once: just give many names as arguments.

The rest of this section describes the most usual options, like how to see all documentation related to a package or use a different viewer.

1.2 Modes

texdoc has different modes that determine how results will be handled. In the default, "view" mode, it opens the first (supposedly the best) result in a viewer. It is rather handy when you know what you want to read, and want to access it quickly. On the other hand, there may be other relevant documents for the given ⟨*name*⟩, which are ignored in view mode.

The so-called "list mode" makes texdoc list all relevant documentation and ask you which one you want to view. It is useful when there a other interesting sources of information besides the package's main documentation.

There is also a "mixed" mode, intended to combine the best of view mode and list mode: if there is only one relevant result, then texdoc opens it in a viewer, else it offers you a menu.

Usually, texdoc shows only the results it considers relevant. If there are no "good" results, it falls back to less relevant results. You can force it to show you also "bad"

[1] On windows, use "Execute" from the Start menu and type `cmd`. On Mac OS X, use the "terminal" icon. If you are using another flavour of Unix, you probably know what to do.

results even when there are good ones by using the “showall” mode. (This implies using a menu rather than starting a viewer.)

You can select the mode with command-line options: use `texdoc` ⟨*option*⟩ ⟨*name*⟩ with one of the following options: `-w` or `--view` for view mode, `-m` or `--mixed` for mixed mode, `-l` or `--list` for list mode, `-s` or `--showall` for showall mode.

If you always (or mostly) use the same mode, it is probably easier to select it in a configuration file than to always use the command-line option.

1.3 Configuration files

texdoc uses various configuration files, which you can see using the `-f` or `--files` options. The entry marked with a star `(*)` is the file you should use for your personal preferences as a user; you may need to create it (and the parents directories).

In order to select you favorite mode, just insert a line `mode =` ⟨*yourmode*⟩ in this file, where ⟨*yourmode*⟩ is one of `view`, `mixed`, `list` or `showall`.

You can also set your favorite language with `lang =` ⟨*2-letter code*⟩ here, though it is usually detected automatically.

The configuration files can be used to tweak texdoc in many ways, the most useful of which is probably the selection of the viewers for various types of documents.

1.4 Viewers

texdoc’s mechanism for choosing a viewer varies according to your platform. On Windows, OS X or Unix with KDE, Gnome or XFCE, it uses your file associations like when you double-click files in the Explorer, the Finder or your default file manager (except for the text viewer, which is always a pager). Otherwise, it tries to find a viewer in the path from a list of “known” viewers.

You may want to select a different viewer for some kind of file. This is achieved by the various `viewer_`⟨*ext*⟩ configuration options, where ⟨*ext*⟩ is the extension corresponding to the file type. For example, if you want to set xpdf as your default PDF viewer, and run it in the background, insert the line `viewer_pdf = xpdf %s &` in your configuration file. Here, `%s` stands for the name of the file to view.

1.5 Conclusion

We have now covered the most common needs. The next part explains how texdoc proceeds to find and sort results. The default configuration file tries hard to set appropriate values so that you have a good out-of-the-box experience, but you may want to understand the underlying mechanisms and adapt them to your needs. The final part is a full reference for configuration options, including points omitted in the present part.

Your feedback is very welcome of the texdoc mailing list. Feel free to post comments, bug reports, suggestions for improvements (inc. new aliases), even without subscribing.

2 File search, aliases, score

2.1 An overview of how texdoc works

When you type `texdoc` ⟨*keyword*⟩, texdoc browses the trees containing documentation (given by the kpathsea variable `TEXDOCS`), lists all files containing ⟨*keyword*⟩ in their name (including the directory name) and give them a score based on some simple heuristics. For example, a file named ⟨*keyword*⟩`.pdf`, is good, ⟨*keyword*⟩`-`⟨*lang*⟩`.pdf` will score higher if your favorite language ⟨*lang*⟩ is detected or configured, ⟨*keyword*⟩`-doc` will be preferred over ⟨*keyword*⟩`whatever`, files in a directory named exactly ⟨*keyword*⟩ get a bonus, etc.

There is also some filtering based on extensions: only files with known extensions are listed, and some extensions get a lesser score. Also, there is some score adjustments based on keywords; by default, Makefile's get a very bad score since they are almost never documentation.[2]

Then, depending on the mode, the file with the highest score is opened in a viewer, or the list of results is shown. Usually, only results with a positive score are displayed, except in showall mode. Result with very bad score (-100 and below) are never displayed.

However, this model works only if the documentation for ⟨*keyword*⟩ has ⟨*keyword*⟩ in its name, which is not always true. The documentation of the memoir class is in `memman.pdf`, for example, but it will be found anyway since it is in a directory named memoir. But there are more complicated cases: the documentation for mathptmx is in `psnfss2e.pdf`; how can texdoc guess that? It looks for `mathptmx.sty` in the TeX Live package database and finds it in the `psnfss` package, and then adds the documentation files for this package to the results. There are two PDF files: `psfonts.pdf` and `psnfss2e.pdf`, but only the second has a comment from the CTAN catalogue[3] so it is considered better.

However, sometimes even this doesn't work. For example, suppose you're looking for documentation about the standard `article` LaTeX class. Unfortunately, the only file name `article.pdf` is completely unrelated and the name of the real file, `classes.pdf`, is hard to guess (there are plenty of other documentation files in the base `latex` package). Here comes the notion of *alias*: in the default configuration file, article is aliased to classes, so that when you type `texdoc article`, texdoc knows it has to look also for classes. Note that texdoc will also look for the original name, and that a name can be aliased to more than one new name (since version 0.60).

We will soon see how you can configure this, but let's start with a few definitions about how a file can match keyword (all matching is case-insensitive):

1. The keyword is a substring of the file name.

[2] They often end up in the doc tree, since the source of documentation is often in the same directory as the documentation itself in TeX Live. Other source files are discriminated by extension.

[3] `http://texcatalogue.sarovar.org/` — most relevant information from the catalogue is included in the TeX Live package database.

2. The keyword is a “subword” of the file name; words are defined as a sequence of alphanumeric characters delimited by punctuation characters (there is no space in file names in TeX Live) and a subword is a substring both ends of which are a word boundary.
3. The keyword matches “exactly” the file name: that is, the file name is the keyword, possibly plus an extension.

2.2 Alias directives

`alias` ⟨*original keyword*⟩ `=` ⟨*name*⟩
`alias(`⟨*score*⟩`)` ⟨*original keyword*⟩ `=` ⟨*name*⟩

You can define your own aliases in texdoc’s configuration files (see 1.3 or 3.1). For example, insert[4]

```
alias article = classes
```

in order to alias article to classes. Precisely, it means that files matching exactly classes be added to the result list when you look for article, and get a score of 10 (default score for alias results). This is greater than the results of heuristic scoring: it means that results found via aliases will always rank before results associated to the original keyword.

If you want the results associated to a particular alias to have a custom score instead of the default 10, you can use the optional argument to the alias directive. This can be useful if you associate many aliases to a keyword and want one of them to show up first.

You can have a look at the configuration file provided (the last shown by `texdoc -f`) for examples. If you feel one of the aliases you defined locally should be added to the default configuration, please share it on the texdoc mailing list.

Please note than aliasing is case-insensitive, and the aliases don’t cascade: only aliases associated to the original keyword are used. Aliases are additive: if you define your own aliases for a keyword in your configuration file, and there are also aliases for the same keyword in the default configuration, they will add up. You can prevent the default aliases from begin applied for a particular keyword by saying `stopalias` ⟨*keyword*⟩ in your personal configuration file. It will keep the aliases defined before this directive (if any) and prevent all further aliasing on this keyword.

Additionally, starting from version 0.80, aliases for ⟨*keyword*⟩`-`⟨*lang*⟩, where ⟨*lang*⟩ is your preferred language’s 2-letter code (as detected or configured, see the `lang` option) are also used for ⟨*keyword*⟩ and get a `+1` score upgrade.

2.3 Score directives

`adjscore` ⟨*pattern*⟩ `=` ⟨*score adjustment*⟩
`adjscore(`⟨*keyword*⟩`)` ⟨*pattern*⟩ `=` ⟨*score adjustment*⟩

[4] Actually, you don’t need to do this, the default configuration file already includes this directive.

It is possible to adjust the score of results containing some pattern as a subword, either globally (for the result of all searches) or only when searching with a particular keyword. This is done in the configuration file (1.3 or 3.1) using the `adjustscore` directive. Here are a few examples from the default configuration file.

```
adjscore /Makefile = -1000
adjscore /tex-virtual-academy-pl/ = -50
adjscore(tex) texdoc = -10
```

All files named `Makefile` (and also files names `Makefile-foo` if there are any): are "killed" : by adjusting their score with such a large negative value, their final score will most probably be less than -100, so they will never be displayed. Files from the `tex-virtual-academy-pl` directory, on the other hand, are not killed but just get a malus, since they are a common source of "fake" matches which hide better results (even for the lucky ones who can read polish).

The third directive gives a malus for results containing `texdoc` only if the search keyword is `tex`. Otherwise, such results would get a high score because the heuristic scoring would think `texdoc` is the name of TEX's documentation. The value -10 is enough to ensure that those results will have a negative score, so wil not be displayed unless "showall" mode is active.

Warning: Values of scores (like the default score for aliases, the range of heuristic scoring, etc.) may change in a future version of texdoc. Scoring is quite new and may need some adjustments. So, don't be surprised if you need to adapt your scoring directives after a future update of texdoc. This warning should disappear at some point.

2.4 File extensions and base names

The allowed file extensions are defined by the configuration item `ext_list` (default: pdf, html, htm, txt, ps, dvi, no extension). You can configure it with `ext_list =` ⟨*your, list*⟩ in a configuration file. Be aware that it will completely override the default list, not add to it. An empty string in the list means files without extension (no dot in the name), while a star means any extension.

For scoring purposes, there is also a `badext_list` parameter: files whose extension is "bad" according to this list will get a lesser score (currently 0). This only affect heuristic scoring (results found from the original keyword, not from aliases).

Similarly, there are `basename_list` and `badbasename_list` parameters: a file is selected only if its extension is in `ext_list` or its base name in `basename_list`. This is for `readme.`⟨*stuff*⟩ files where ⟨*stuff*⟩ doesn't mean anything about the file type.

3 Full reference

3.1 Precedence of configuration sources

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 same order as they are printed by `texdoc --files`.
5. Hard-coded defaults that may depend on the current platform or the current value of another setting.

The list of configuration files may include entries which depend on the current platform, like `texdoc-x86-64-linux.cnf` for instance. Those files are meant to be used in when a single TEX Live installation is shared across machines with different architectures needing different settings. Their use is not recommended in any other situation.

Finally, the special file `texdoc-dist.cnf` allows you to install a newer version of texdoc (including its default configuration file) in your home: see the web page for instructions on running the development version.

3.2 Command-line options

All command-line options (except the first three below) correspond to configuration item that can be set in the configuration files: we refer the reader to the corresponding section for the meaning of this configuration item.

3.2.1 `-h, --help` — Show a quick help message (namely a list of command-line options) and exit successfully.

3.2.2 `-V, --version` — Show the current version of the program and exit successfully.

3.2.3 `-f, --files` — Show the list of the configuration files for the current installation and platform, with their status (active, not found, or disabled (see 3.4.19)) and exit successfully.

3.2.4 `-w, --view, -l, --list, -m, --mixed, -s, --showall` — Set `mode` to the given value, see 3.4.2.

3.2.5 `-i, --interact, -I, --nointeract` — Set `interact_switch` to true (resp. false), see 3.4.3.

3.2.6 `-M, --machine` — Set `machine_switch` to true, see 3.4.15.

3.2.7 `-q, --quiet` — Set `verbosity_level` to minimum, see 3.4.12.

3.2.8 `-v, --verbose` — Set `verbosity_level` to maximum, see 3.4.12.

3.2.9 `-d, -d=⟨list⟩, --debug, --debug=⟨list⟩` — Set `debug_list`, see 3.4.13. If not list is given, activates all available debug items.

3.2.10 `--just-view` — This is a very special switch: when it is used, texdoc accepts only one argument which must be the a file name with full path (absolute or relative to current directory) and starts a viewer for this file according to its usual viewer rules. Absolutely no search is done. It is unlikely you'll want to use this option directly; it may however be useful when texdoc is used as a back-end by other programs (such as texdoctk).

3.3 Environment variables

They all correspond to some `viewer_⟨ext⟩` setting, and the reader is referred to 3.4.10 for details. Also, environment variables used by older versions of texdoc are accepted. You can append `_texdoc` to every name in the first column: this wins over every other name.

New name	Old name 1	Old name 2	Config. item
PAGER	TEXDOCVIEW_txt	TEXDOC_VIEWER_TXT	viewer_txt
BROWSER	TEXDOCVIEW_html	TEXDOC_VIEWER_HTML	viewer_html
DVIVIEWER	TEXDOCVIEW_dvi	TEXDOC_VIEWER_DVI	viewer_dvi
PSVIEWER	TEXDOCVIEW_ps	TEXDOC_VIEWER_PS	viewer_ps
PDFVIEWER	TEXDOCVIEW_pdf	TEXDOC_VIEWER_PDF	viewer_pdf

Also, on Unix systems, locale-related variables such as `LANG` and `LC_ALL` are used for the default value of `lang`.

3.4 Configuration items

3.4.1 *Structure of configuration files.* — 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.

⟨*configuration item*⟩ = ⟨*value*⟩
alias ⟨*original keyword*⟩ = ⟨*name*⟩
alias(⟨*score*⟩) ⟨*original keyword*⟩ = ⟨*name*⟩
stopalias ⟨*original keyword*⟩
adjscore ⟨*pattern*⟩ = ⟨*score adjustment*⟩
adjscore(⟨*keyword*⟩) ⟨*pattern*⟩ = ⟨*score adjustment*⟩

We will concentrate on the ⟨*configuration item*⟩ part here, since other directives have already been presented (2.2 and 2.3).

In the above, ⟨*value*⟩ never needs to be quoted: quotes would be interpreted as part of the value, not as quotation marks (this also holds for the other directives).

Lines which do not obey these rules raise a warning, as well as unrecognised values of ⟨*configuration item*⟩. The ⟨*value*⟩ can be an arbitrary string, except when the name of the ⟨*configuration item*⟩ 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` (lowercase).
3. `_level`, then ⟨*value*⟩ is an integer.

In these cases, an improper ⟨*value*⟩ will raise a warning too.

3.4.2 `mode =` ⟨*view, list, mixed, showall*⟩ — Set the mode to the given value. Default is `view`. The various modes have been presented in 1.2.

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

3.4.4 `alias_switch =` ⟨*true, false*⟩ — Turn on or off aliasing. Default is on.

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

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

This list is used to filter and sort the results that have the same score(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!

There is a very special case: if the searched ⟨*name*⟩ has `.sty` extension, texdoc enters a special search mode for `.sty` files (not located in the same place as real documentation files) for this ⟨*name*⟩, independently of the current value of `ext_list` and `mode`. In an ideal world, this wouldn't be necessary since every sty file would have a proper documentation in pdf, html or plain text, but...

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

As a special case, if the extension is `sty`, then the `txt` viewer is used; similarly, if it is `htm` the `html` viewer is used. Otherwise, the `txt` viewer is used and a warning is issued.

3.4.6 `badext_list =` ⟨*list*⟩ — Set the list of "bad" extensions to ⟨*list*⟩. Default is "`txt,`". Files with those extensions get a malus of 1 on their heurisitc score if it was previously positive.

3.4.7 `basename_list =` ⟨*list*⟩ — Set the list of allowed base names. Default is `readme`.

3.4.8 `badbasename_list = ⟨list⟩` — Set the list of "bad" base names to ⟨*list*⟩. Default is `readme`. Files with those names get a malus of `1` on their heurisitc score if it was previously positive.

3.4.9 `suffix_list = ⟨list⟩` — Set the list of know documentation suffixes to ⟨*list*⟩. Default is

```
doc, -doc, _doc, .doc, /doc, manual, /manual, -manual, userguide,
/user_guide, -guide, -user, -man, notes
```

When searching for ⟨*keyword*⟩, results (found directly with ⟨*keyword*⟩ as opposed to aliases) are scored in this order (from worse to best):

1. Substring match for ⟨*keyword*⟩.
2. Substring match for ⟨*keyword*⟩⟨*suffix*⟩ for a listed ⟨*suffix*⟩.
3. Exact match for ⟨*keyword*⟩⟨*suffix*⟩ for a listed ⟨*suffix*⟩.
4. Exact match for ⟨*keyword*⟩.

(Adjustements may be done later based on other informations and settings.)

3.4.10 `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 no `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.

3.4.11 `lang =` ⟨*2-letter code*⟩ — Set you preferred language. Defaults to your system's locale.

3.4.12 `verbosity_level =` ⟨*n*⟩ — Set the verbosity level to ⟨*n*⟩ (default: 2). 3 means errors, warnings and informational messages will be printed (on stderr); 2 means only errors and warnings, 1 only errors and 0 nothing except internal errors (obviously not recommended).

3.4.13 `debug_list =` ⟨*list*⟩ — Set the list of activated debug items (default: none; if the option is used without arguments, the list defaults to all known debug items). Implies `--verbose`. Debug information is printed on standard error.

3.4.14 `max_lines =` ⟨*number*⟩ — Set the maximum number of results to be printed without confirmation in list, mixed or showall mode (default: 20). This setting has no effect if interaction is disabled.

3.4.15 `machine_switch =` ⟨*true, false*⟩ — Turn on or off machine-readable output (default: off). With this option active, the value of `interact_switch` is forced to `false`, and each line of output is

⟨*argument*⟩`\t`⟨*score*⟩`\t`⟨*filename*⟩

where ⟨*argument*⟩ is the name of the argument to which the results correspond (mainly useful if there were many arguments), `\t` is the tab (ascii 9) character, and the other entries are pretty self-explanatory. Nothing else is printed on stdout, except if a internal error occurs (in which case exit code will be 1). In the future, more tab-separated fields may be added at the end of the line, but the first 3 fields will remain unchanged.

Currently, there are two additional fields: a two-letter language code, and an unstructured description, both taken from the CTAN catalogue (via the TeX Live database). These fields may be empty and they are not guaranteed to keep the same meaning in future versions of texdoc.

3.4.16 `zipext_list` = ⟨*list*⟩ — List of supported extensions for zipped files (default: empty). Allows compressed files with names like `foobar.`⟨*zip*⟩, with ⟨*zip*⟩ in the given ⟨*list*⟩, to be found and unzipped before the viewer is launched (the temporary file will be destroyed right after).

Warning. Support for zipped documentation is not meant to work on windows, a Unix shell is assumed! If you add anything to this list, please make sure that you also set a corresponding `unzip=`⟨*ext*⟩ value for each ⟨*ext*⟩ in the list. Also make sure you are using blocking (i.e. not returning immediately) viewers.

Remark. TEX Live doesn't ship compressed documentation files, so this option is mainly useful with re-packaged version of TEX Live that do, for example in Linux distributions.

3.4.17 `unzip_`⟨*zipext*⟩ = ⟨*command*⟩ — The unzipping command for zipped files with extension ⟨*zipext*⟩ (default: no default). Define one for each item in `zipext_list`. The command must print the result on stdout, like `gzip -d -c` does.

`rm_file` = ⟨*command*⟩

3.4.18 `rm_dir` = ⟨*command*⟩ — Commands for removing files (resp. directories) on your system (defaults: `rm -f` and `rmdir`). Only useful for zipped documents (see `zipext_list`).

3.4.19 `lastfile_switch` = ⟨*true, false*⟩ — If set to true, prevents texdoc from reading any other configuration file after this one (they will be reported as "disabled" by `texdoc -f`). Mainly useful for installing a newer version of texdoc in your home and preventing the default configuration file from older versions to be used (see the web site for instructions on how to do so).

3.5 Exit codes

The current exit codes are:

0. Success.
1. Internal error.
2. Usage error.

4 Licence

This program is distributed in the hope that it will be useful, but *without any warranty*; without even the implied warranty of *merchantability* or *fitness for a particular purpose*. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

Previous work (texdoc program) in the public domain:

- Contributions from Reinhard Kotucha (2008).
- First texlua versions by Frank Küster (2007).
- Original shell script by Thomas Esser, David Aspinall, and Simon Wilkinson.

Happy TEXing!