NAME

texfot – run TeX, filtering online transcript for interesting messages

SYNOPSIS

texfot [*option*]... *texcmd* [*texarg*...]

DESCRIPTION

`texfot` invokes *texcmd* with the given *texarg* arguments, filtering the online output for "interesting" messages. Its exit value is that of *texcmd*. Examples:

```
# Sample basic invocation:
texfot pdflatex file.tex

# Ordinarily the full output is copied to /tmp/fot.$UID before
# filtering, but that can be omitted, or the filename changed:
texfot --tee=/dev/null lualatex file.tex

# Example of more complex engine invocation:
texfot xelatex --recorder '\nonstopmode\input file'
```

Here is an example of what the output looks like (in its entirety) on an error-free run:

```
/path/to/texfot: invoking: pdflatex hello.ltx
This is pdfTeX, Version 3.141592653-2.6-1.40.24 (TeX Live 2022) (preloaded
Output written on hello.pdf (1 page, 94415 bytes).
```

Aside from its own options, described below, `texfot` just runs the given command with the given arguments (same approach to command line syntax as `env`, `nice`, `timeout`, etc.). Thus, `texfot` works with any engine and any command line options.

`texfot` does not look at the log file or any other possible output file(s); it only looks at the standard output and standard error from the command. stdout is processed first, then stderr. `texfot` writes all accepted lines to its stdout.

The messages shown are intended to be those which likely need action by the author: error messages, overfull and underfull boxes, undefined citations, missing characters from fonts, etc.

FLOW OF OPERATION

Here is the order in which lines of output are checked:

1. If the "next line" needs to be printed (see below), print it.
2. Otherwise, if the line matches any user-supplied list of regexps to accept (given with `--accept`, see below), in that order, print it.
3. Otherwise, if the line matches the built-in list of regexps to ignore, or any user-supplied list of regexps to ignore (given with `--ignore`, see below), in that order, ignore it.
4. Otherwise, if the line matches the list of regexps for which the next line (two lines in all) should be shown, show this line and set the "next line" flag for the next time around the loop. Examples are the common `!` and `filename:lineno:` error messages, which are generally followed by a line with specific detail about the error.
5. Otherwise, if the line matches the list of regexps to show, show it.
6. Otherwise, the default: if the line came from stdout, ignore it; if the line came from stderr, print it (to stdout), with the prefix `[stderr] `. This distinction is made because TeX

engines write relatively few messages to stderr, and it's likely that any such should be considered.

Once a particular check matches, the program moves on to process the next line.

`texfot` matches exclusively line-by-line; however, TeX itself folds output lines, typically at column 79. This means matches might fail because the text being matched was split over two lines. To work around this, you can effectively turn off TeX's folding by setting the `max_print_line` parameter to a large number, either in the environment or on the command:

```
# When errors are missed due to TeX's folding of lines:
texfot pdftex --cnf-line max_print_line=999 file.tex

# Equivalently:
env max_print_line=999 texfot pdftex file.tex
```

Don't hesitate to peruse the source to the script, which is essentially a straightforward loop matching against the different lists as above. You can see the exact regexps being matched in the different categories in the source.

Incidentally, although nothing in this basic operation is specific to TeX engines, all the regular expressions included in the program are specific to TeX. So in practice the program isn't useful except with TeX engines, although it would be easy enough to adapt it (if there was anything else as verbose as TeX to make that useful).

OPTIONS

The following are the options to `texfot` itself (not the TeX engine being invoked; consult the engine documentation or `--help` output for that).

The first non-option terminates `texfot`'s option parsing, and the remainder of the command line is invoked as the TeX command, without further parsing. For example, `texfot --debug tex --debug` will output debugging information from both `texfot` and `tex`. TeX engines, unlike many standard programs, require that options be specified before the input filename or text.

Options may start with either – or ––, and may be unambiguously abbreviated. It is best to use the full option name in scripts, though, to avoid possible collisions with new options in the future.

`--accept` *regexp*
: Accept lines in the TeX output matching (Perl) *regexp*. Can be repeated. This list is checked first, so any and all matches will be shown, regardless of other options. These regexps are not automatically anchored (or otherwise altered), simply used as-is.

`--debug`
`--no-debug`
: Output (or not) what the program is doing to standard error; off by default.

`--ignore` *regexp*
: Ignore lines in the TeX output matching (Perl) *regexp*. Can be repeated. Adds to the default set of ignore regexps rather than replacing. Like the acceptance regexps, these are not automatically anchored (or otherwise altered).

`--interactive`

`--no-interactive`
: By default, standard input to the TeX process is closed so that TeX's interactive mode (waiting for input upon error, the `*` prompt, etc.) is never entered. Giving `--interactive` allows interaction to happen.

`--quiet`
`--no-quiet`
: By default, the TeX command being invoked is reported on standard output; `--quiet` omits that reporting. To get a completely silent run, redirect standard output: `texfot ... >/dev/null`. (The only messages to standard error should be errors from `texfot` itself, so it shouldn't be necessary to redirect that, but of course that could be done as well.)

`--stderr`
`--no-stderr`
: The default is for `texfot` to report everything written to stderr by the TeX command (on stdout). `--no-stderr` omits that reporting. (Some programs, `dvisvgm` is one, can be rather verbose on stderr.)

`--tee` *file*
: By default, the output being filtered is `tee`–ed, before filtering, to make it easy to check the full output in case of problems.

  The default *file* is `$TMPDIR/fot.`*uid*; if `TMPDIR` is not set, `TMP` is used if set; if neither is set, the default directory is `/tmp`. For example: `/tmp/fot.1001`. The *uid* suffix is the effective userid of the process, appended for basic avoidance of collisions between different users on the same system.

  This option allows specifying a different file. Use `--tee /dev/null` to discard the original output.

`--version`
: Output version information and exit successfully.

`--help`
: Display this help and exit successfully.

RATIONALE

I wrote this because, in my work as a TUGboat editor (<https://tug.org/TUGboat>, article submissions always welcome!), I run and rerun many documents, many times each. It was easy to lose warnings I needed to see in the mass of unvarying and uninteresting output from TeX, such as style files being read and fonts being used. I wanted to see all and only those messages which needed some action by me.

I found some other programs of a similar nature, the LaTeX package `silence`, and plenty of other (La)TeX wrappers, but it seemed none of them did what I wanted. Either they read the log file (I wanted to look at only the online output), or they output more, or less, than I wanted, or they required invoking TeX differently (I wanted to keep my build process exactly the same, most critically the TeX invocation, which can get complicated). Hence I wrote this little script.

Here are some keywords if you want to explore other options: texloganalyser, pydflatex, logfilter, latexmk, rubber, arara, and searching for `log` at <https://ctan.org/search>.

`texfot` is written in Perl, and runs on Unix. It may work on Windows if Perl and other software is installed, but I don't use Windows and don't support `texfot` there.

The name comes from the `trip.fot` and `trap.fot` files that are part of Knuth's trip and trap torture tests, which record the online output from the programs. I am not sure what "fot" stands for in trip and trap, but I can pretend that it stands for "filter online transcript" in the present case :).

AUTHORS AND COPYRIGHT

 Email `karl@freefriends.org` with bug reports. It has no home page beyond the package page on CTAN: <https://ctan.org/pkg/texfot>.

```
$Id: texfot,v 1.53 2024/04/17 16:52:12 karl Exp $
```