texlogsieve:*

(yet another program to)
filter and summarize LaTeX log files

Nelson Lago
lago@ime.usp.br

https://gitlab.com/lago/texlogsieve

March 3, 2025

Abstract

texlogsieve reads a LaTeX log file (or the standard input if no file is specified), filters out less relevant messages, and displays a summary report. Highlights:

- Two reports: the most important messages from the log file followed by a summary of repeated messages, undefined references etc.;
- The program goes to great lengths to correctly handle TeX line wrapping and does a much better job at that than existing tools;
- Multiline messages are treated as a single entity;
- Several options to control which messages should be filtered out;
- No messages are accidentally removed.

Introduction

The LaTeX log file is very verbose, which is useful when debugging but a hindrance during document preparation, as warnings such as "missing character", "undefined reference", and others become buried among lots of less relevant messages. This program filters out such less relevant messages and outputs the rest, together with a final summary for the specially important ones. It is a texlua script, similar in spirit to tools such as texfot, texloganalyser, rubber-info, textlog_extract, texlogparser, texlog-filter, pulp, and others.

Note that it does not try to do anything smart about error messages (but it shows a warning in the summary if one is detected; check the "Tips" section regarding this); if there is an error, you probably want to take a look directly at the log file anyway. It also cannot detect if LaTeX stops for user input, so you should **really** run LaTeX in nonstopmode when texlogsieve is reading from a pipe.

texlogsieve **must** be run from the same directory as [pdf|lua|xe]latex, because it searches for the files used during compilation (packages loaded from the current directory, files included with \input etc.).

*This document corresponds to texlogsieve 1.5.0, dated 2025-03-03.

The defaults are reasonable; hopefully, you can just do

```
[pdf|lua|xe]latex -interaction nonstopmode myfile.tex | texlogsieve
```

or

```
texlogsieve myfile.log
```

and be satisfied with the result.

Since it needs to know what messages to expect, `texlogsieve` is currently geared towards LaTeX; I have no idea how it would work with ConTeXt or plain TeX. Still, adding support to them should not be too difficult.

If you want to know more about the TeX log file and the workings of the program, check the initial comments in the code.

1 Unwrapping Long Lines

TeX wraps (breaks) lines longer than `max_print_line` (by default, 79 characters). Most tools detect lines that are exactly 79 characters long and treat the next line as a continuation, but that fails in quite a few cases (check the comments in the `texlogsieve` code for a discussion on that). So, if at all possible, it is a very good idea to set `max_print_line` to a really large value (such as 100,000), effectively disabling line wrapping. It was useful in the 1980s, but not anymore (your terminal or editor wraps automatically)[1].

Still, `texlogsieve` goes to great lengths to correctly handle TeX line wrapping and does a pretty good job at that. It understands the `max_print_line` TeX configuration variable and reads its value from the same places as TeX. Setting `max_print_line` to a value larger than 9999 makes `texlogsieve` ignore line wrapping.

2 Unrecognized Messages

`texlogsieve` automatically handles messages such as "Package blah Info:..." or "LaTeX Warning:...". However, many messages do not follow this pattern. To do its thing, `texlogsieve` should know about these other messages beforehand. This is important for three reasons:

1. Unknown messages are given maximum priority; if you do not want to see them, you have to use `--silence-string`;
2. If the message has more than one line, each line is treated as an independent message. This means you need to use `--silence-string` multiple times;
3. In some rare cases, an unrecognized message may make `texlogsieve` misclassify nearby wrapped lines (if it comes right after a 79 characters long line of a specific type), close file messages (if it includes an unmatched close parens character), or shipout messages (if it includes an unmatched close square bracket character or an open square bracket character followed only by numbers).

While `texlogsieve` recognizes quite a few messages out of the box, you may run into a message generated by some package that it does not know about (you

[1] Likewise, `error_line` and `half_error_line` should be, respectively, 254 and 238 (more about these values here: https://tex.stackexchange.com/a/525972).

can check for this using `-l unknown`). If that is the case, you can use the `--add-[debug|info|warning|critical]-message` options to add it to the list of messages known to the program.

3 Configuration File

`texlogsieve` always searches automatically for the (optional) `texlogsieverc` configuration file in `$TEXINPUTS` (i.e., it searches using `Kpathsea`). In the default configuration, the current directory is in `$TEXINPUTS`, so adding a config file with that name to the project directory is enough to make it work. Options in the config file are exactly the same as the long command-line options described below, but without the preceding "`--`" characters. Lines starting with a "`#`" sign are comments. An example configuration file:

```
no-summary-detail
no-page-delay
# no-page-delay enables shipouts, but we do not want that
no-shipouts
set-to-level-info=Hyperreferences in rotated content will be misplaced
# no need to escape the "\" (or any other) character
silence-string = Using \overbracket and \underbracket from `mathtools'
# silence a string using lua pattern matching
silence-string = ////luaotfload | aux : font no %d+ %(.-%)
silence-files = *.sty
```

If you'd like to also have a generic configuration file for all your projects (a good idea), put it at `$HOME/.texlogsieverc` in unix-like systems; in Windows, put it either at `%LOCALAPPDATA%\texlogsieverc` (`C:\Users\<username>\AppData\Local`) or `%APPDATA%\texlogsieverc` (`C:\Documents and Settings\<username>\Application Data` or `C:\Users\<username>\AppData\Roaming`).

4 Options

`--page-delay`, `--no-page-delay`
: Enable/disable grouping messages by page before display. When enabled, messages are only output after the current page is finished (shipout). The advantage is that the page number is included in the message (default enabled).

`--summary`, `--no-summary`
: Enable/disable final summary (default enabled).

`--only-summary`
: No messages, show only the final summary (default disabled).

`--shipouts`, `--no-shipouts`
: Enable/disable reporting shipouts (default disabled with page-delay, enabled with no-page-delay).

`--file-banner`, `--no-file-banner`
: Show/don't show the "From file…" banner messages (default enabled, except with level `DEBUG` as that would be redundant and confusing).

--repetitions, --no-repetitions
: Allow/prevent repeated messages (default disabled, i.e., repeated messages are supressed).

--be-redundant, --no-be-redundant
: Present/suppress ordinary messages that will also appear in the summary. This affects messages that have special summaries (such as under/overfull boxes or undefined citations). With `--no-be-redundant` (the default), these messages are filtered out and only appear in the final summary.

--box-detail, --no-box-detail
: Include/exclude detailed information on under/overfull boxes in the final summary. With `--no-box-detail`, the summary presents only a list of pages and files that had under/overfull boxes (default enabled).

--ref-detail, --no-ref-detail
: Include/exclude detailed information on undefined references in the final summary. With `--no-ref-detail`, the summary presents only a list of undefined references, without page numbers and filenames (default enabled).

--cite-detail, --no-cite-detail
: Include/exclude detailed information on undefined citations in the final summary. With `--no-cite-detail`, the summary presents only a list of undefined citations, without page numbers and filenames (default enabled).

--summary-detail, --no-summary-detail
: Toggle `--box-detail`, `--ref-detail`, and `--cite-detail` at once.

--heartbeat, --no-heartbeat
: Enable/disable progress gauge in page-delay mode (default enabled).

--color, --no-color
: Enable/disable colored output. On Windows, this will only work with an up-to-date Windows 10 or later (default disabled).

--tips, --no-tips
: Enable/disable suggesting fixes for some known warnings (default enabled).

-l LEVEL, --minlevel=LEVEL
: Filter out messages with severity level lower than `LEVEL`. Valid levels are `DEBUG` (no filtering), `INFO`, `WARNING`, `CRITICAL`, and `UNKNOWN` (default `WARNING`).

-u, --unwrap-only
: Do not filter messages and do not output the summary, only unwrap long, wrapped lines. The output should be very similar (but not equal) to the input file, but with wrapped lines reconstructed. This activates `-l debug`, `--no-summary`, `--no-page-delay`, `--repetitions`, `--be-redundant`, `--shipouts`, and `--no-file-banner`, and also supresses the verbose "open/close file" and "shipout" messages, simulating instead the TEX format, with parens and square brackets. This is useful if you prefer the reports generated by some other tool but want to benefit from texlogsieve's line unwrapping algorithm; the output generated by this option should be parseable by other tools (but you probably need to coerce the other tool not to try to unwrap lines).

--silence-package=PKGNAME
: Filter out messages that can be identified as coming from the given package. Use this option multiple times to suppress messages from several different packages.

--silence-string=EXCERPT OF UNWANTED MESSAGE
: Filter out messages that contain the given string (you only need to provide part of the message text for the whole message to be suppressed). Use this option multiple times to suppress several different messages. The string should be a single line, but that is not a problem for multiline log messages: space characters in the provided string match any sequence of whitespace characters in the message, including newlines. If needed, you may precede the string with "////", in which case you can use lua-style pattern matching (https://www.lua.org/pil/20.2.html). Note that the string is used verbatim: you may need to enclose it in quotes or escape special characters such as "\" for the benefit of the shell, but such quoting and escaping is unnecessary (and harmful) in the configuration file.

--silence-file=FILENAME OR FILE GLOB
: Filter out messages that have been generated while the given file was being processed. Do **not** use absolute or relative paths, only filenames. Simple file globs, such as "`*.cls`", work as expected. If you are only using packages you already know, silencing "`*.sty`" may be a good idea (note that this does not suppress all messages from all packages, only the messages generated while the packages are being loaded). Use this option multiple times to suppress messages from several different files.

--semisilence-file=FILENAME OR FILE GLOB
: Just like the previous option, but non-recursive. This means that messages generated while the given file was being processed are excluded, but messages generated by some other file that was opened by it are not. For example, if "`chapters.tex`" includes (with \input) the files "`chapter1.tex`" and "`chapter2.tex`", using "`--silence-file=chapters.tex`" will prevent messages generated by any of the three files from being displayed. If, however, you use "`--semisilence-file=chapters.tex`", messages generated by `chapters.tex` will be suppressed, but messages generated by `chapter1.tex` or `chapter2.tex` will not.

--add-[debug|info|warning|critical]-message=MESSAGE
: Add `MESSAGE` to the list of messages known to the program with the given severity level; see Section 2 for more information about this. Like `--silence-string`, these should be a single line; unlike `--silence-string`, you need to embed `\n` explicitly to indicate line breaks (this is literally a backslash character followed by the letter "n", **not** a linefeed character). You may precede the string with "////" to use lua-style pattern matching, but embedding `\n` to indicate line breaks is unavoidable. Use these options multiple times to add many different messages.

--set-to-level-[debug|info|warning|critical]=EXCERPT OF MESSAGE
: Redefine the severity level of messages that contain the provided string to the given level. Check the explanation for `--silence-string`, as this works in a similar way. Use these options multiple times to change the severity level of many different messages.

-c CFGFILE, --config-file=CFGFILE
: Read options from the given configuration file in addition to the default config files (see the "Configuration File" section).

-v, --verbose
Print the list of configuration files read and a short summary of the most important active configuration options.

-h, --help Show concise options description.

--version Print program version.

5 Tips

- If the program output is still too verbose to your liking, resist the urge to use `-l critical` or `--only-summary`. Instead, create a configuration file with `no-summary-detail` and appropriate `silence-[something]` and `set-to-level-[something]` options: You will get reasonably quiet output but still be notified of problems. Since you probably always use essentially the same set of packages and classes, the file will be useable in other LaTeX projects too; just put it in one of the places discussed in the "Configuration File" section.
- `texlogsieve` has no smarts to deal with error messages, but you may use options `-l unknown --no-page-delay --no-summary --no-shipouts --no-file-banner` to get almost nothing but errors.
- With `latexmk`, it is enough to put something like this in `latexmkrc`:

  ```
  set_tex_cmds("-halt-on-error -interaction nonstopmode %O %S|texlogsieve");
  ```

  However, this means you get to see the repeated output of all iterations needed to produce the document. To only see the output of the last iteration, you may do something like this:

  ```
  set_tex_cmds("-halt-on-error %O %S");

  $silent = 1; # This adds "-interaction batchmode"
  $silence_logfile_warnings = 1;

  END {
    local $?; # do not override previous exit status
    if (-s "$root_filename.blg"
                and open my $bibfile, '<', "$root_filename.blg") {

        print("**********************\n");
        print("bibtex/biber messages:\n");
        while(my $line = <$bibfile>) {
            if ($line =~ /You.ve used/) {
                last;
            } else {
                print($line);
            };
        };
        close($bibfile);
    };
  ```

```
    if (-s "$root_filename.ilg"
                and open my $indfile, '<', "$root_filename.ilg") {

        print("************************\n");
        print("makeindex/xindy messages:\n");
        while(my $line = <$indfile>) {
            print($line);
        };
        close($indfile);
    };

    if (-s "$root_filename.log") {
        print("***************\n");
        print("LaTeX messages:\n");
        Run_subst("texlogsieve %R.log");
    };
};
```

6 License

Change History