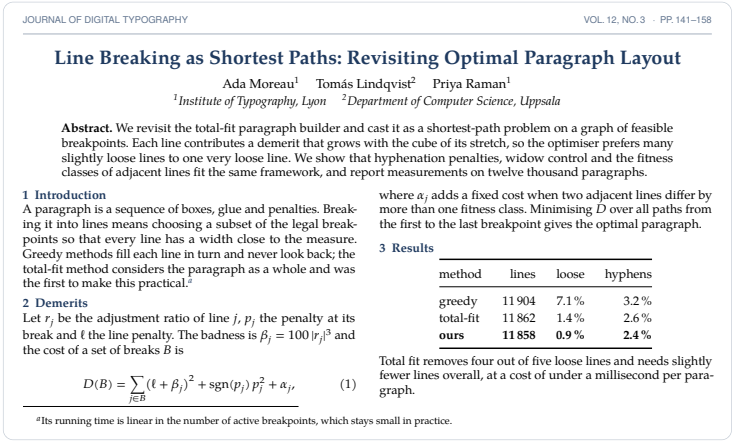


retex

user manual

version 1.8.0

https://gitlab.com/severin-lemaignan/retex



JOURNAL OF DIGITAL TYPOGRAPHY VOL. 12, NO. 3 · PP. 141–158

Line Breaking as Shortest Paths: Revisiting Optimal Paragraph Layout

Ada Moreau[1] Tomás Lindqvist[2] Priya Raman[1]

[1]*Institute of Typography, Lyon* [2]*Department of Computer Science, Uppsala*

Abstract. We revisit the total-fit paragraph builder and cast it as a shortest-path problem on a graph of feasible breakpoints. Each line contributes a demerit that grows with the cube of its stretch, so the optimiser prefers many slightly loose lines to one very loose line. We show that hyphenation penalties, widow control and the fitness classes of adjacent lines fit the same framework, and report measurements on twelve thousand paragraphs.

1 Introduction

A paragraph is a sequence of boxes, glue and penalties. Breaking it into lines means choosing a subset of the legal breakpoints so that every line has a width close to the measure. Greedy methods fill each line in turn and never look back; the total-fit method considers the paragraph as a whole and was the first to make this practical.[2]

2 Demerits

Let r_j be the adjustment ratio of line j, p_j the penalty at its break and ℓ the line penalty. The badness is $\beta_j = 100\,|r_j|^3$ and the cost of a set of breaks B is

$$D(B) = \sum_{j \in B} \left(\ell + \beta_j\right)^2 + \operatorname{sgn}(p_j)\, p_j^2 + \alpha_j, \tag{1}$$

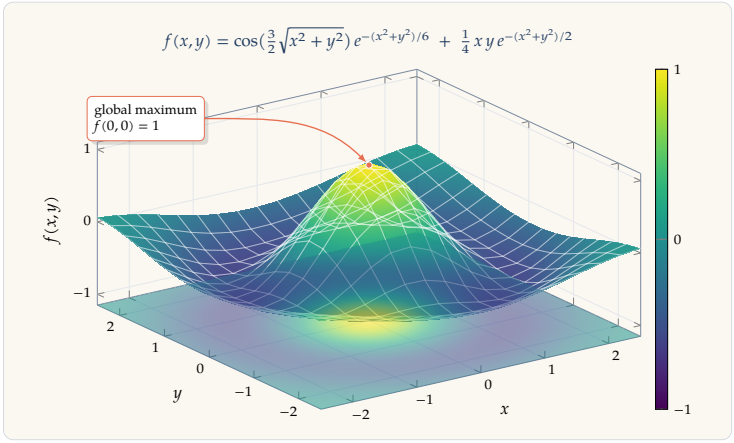
where α_j adds a fixed cost when two adjacent lines differ by more than one fitness class. Minimising D over all paths from the first to the last breakpoint gives the optimal paragraph.

3 Results

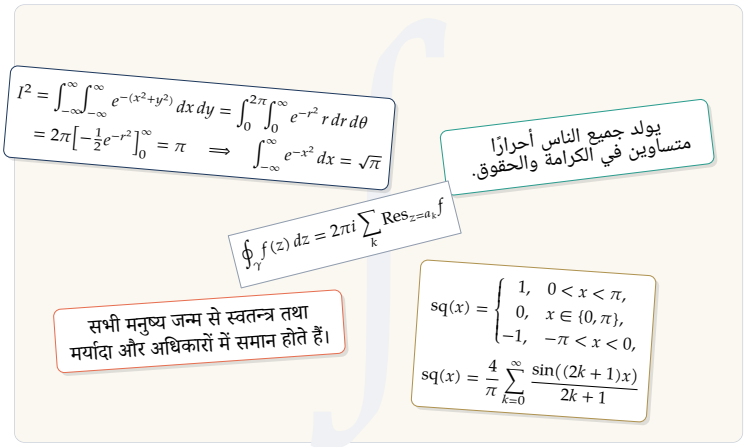
method	lines	loose	hyphens
greedy	11 904	7.1 %	3.2 %
total-fit	11 862	1.4 %	2.6 %
ours	**11 858**	**0.9 %**	**2.4 %**

Total fit removes four out of five loose lines and needs slightly fewer lines overall, at a cost of under a millisecond per paragraph.

[a] Its running time is linear in the number of active breakpoints, which stays small in practice.







The page above was typeset by retex, as part of this manual, from `hero.tex`: an article in two columns, a pgfplots surface, raster images clipped, faded, blended and slanted with TikZ, mathematics in OpenType, and Arabic and Hindi text shaped by HarfBuzz. lualatex makes the same page, pixel for pixel. The colours of the front page are those of one of its lithographs, Ernst Haeckel's siphonophores (*Kunstformen der Natur*, 1904).

If you are reading this in the browser editor, this manual is also a sandbox. Every page you see was typeset in this browser tab, from the files listed on the left, and it is typeset again as you edit them. Nothing leaves your computer. Change a sentence and the PDF changes with it; misspell a command and retex reports the error. The boxes marked *Try it* suggest things to do. The projects page (top left) has a fresh copy of the manual, other examples, and your projects.

Contents

1 Introduction **8**

1.1 Who wrote retex . . . 8

1.2 How this manual is organized . . . 8

1.3 Conventions . . . 10

I Using retex **11**

2 Installing retex **12**

2.1 Release programs . . . 12

2.2 From crates.io or from the sources . . . 12

2.3 The first run . . . 13

2.4 Without TeX Live . . . 13

2.5 Missing TeX Live packages . . . 14

2.6 Platforms . . . 15

3 Compiling documents **17**

3.1 Passes . . . 17

3.2 Where the files go . . . 17

3.3 Errors do not stop the build . . . 18

3.4 Shell escape and the security of documents . . . 18

3.5 Code listings: minted . . . 18

3.6 Images . . . 18

4 Reading the messages **20**

4.1 Errors, warnings, notes and hints . . . 20

4.2 The summary line . . . 21

4.3 Hints about the source . . . 21

4.4 Quieter, louder, and for programs . . . 22

4.5 TeX's own dialog . . . 22

5 Bibliographies and indexes **24**

5.1 BibTeX and biber . . . 24

5.2 makeindex . . . 25

6 Rebuilding as you write **26**

6.1 `--watch` . . . 26

6.2 The figure cache . . . 26

6.3 What else is kept . . . 26

7 Other outputs **27**

7.1 Accessible PDF . . . 27

7.2 PDF for the press: PDF/X . . . 27

7.3 SyncTeX . . . 31

7.4 SVG . . . 31

7.5 OpenDocument presentations and text documents . . . 32

7.6 HTML and Markdown . . . 32

7.7 DVI . . . 32

8 retex in editors **33**

8.1 The language server . . . 33

8.2 Setting up an editor . . . 33

8.3 Without the language server . . . 34

9 The browser editor **35**

9.1 Writing . . . 35

9.2 Files and projects . . . 35

9.3 Projects on GitLab . . . 36

9.4 Other formats . . . 36

9.5 What it cannot do . . . 36

10 retex in programs **37**

10.1 From Python . . . 37

10.2 Examples in Python . . . 38

10.3 From Rust . . . 40

10.4 How a build runs . . . 41

11 Looking inside a document **42**

11.1 Where the time goes . . . 42

11.2 What a command is . . . 43

II Reference **44**

12 The command line **45**
12.1 Building . . . 45
12.2 Files . . . 47
12.3 Messages . . . 47
12.4 Compatibility with LuaTEX . . . 48
12.5 Exit status . . . 49

13 Environment variables and files **50**
13.1 Variables . . . 50
13.2 The settings file . . . 51
13.3 Files . . . 53

14 retex and lualatex **54**
14.1 Point by point . . . 54
14.2 The compatibility mode . . . 54
14.3 Conformance . . . 55
14.4 TEX Live 2026 . . . 56
14.5 Japanese with LuaTEX-ja . . . 56

III Inside retex **57**

15 Performance **58**
15.1 Papers and proposals . . . 58
15.2 Large manuals and presentations . . . 58
15.3 Where the speed comes from . . . 59

16 Clean-room provenance **60**
16.1 Knuth's programs . . . 60
16.2 MetaPost . . . 60
16.3 Bibliographies and indexes . . . 61

17 How retex is tested **62**
17.1 The corpus . . . 62
17.1.1 The manifest . . . 62
17.1.2 The cache . . . 63
17.1.3 A run . . . 64

17.1.4 Adding entries 64
17.2 BibTeX and biber 65
17.3 makeindex 66
17.4 TeX Live 2026 67

A The sandbox **69**
A.1 Text 69
A.2 Mathematics 69
A.3 A picture 69
A.4 A table 69
A.5 Mistakes to fix 69
A.6 Other scripts 70

B History **71**

References **73**

Index **74**

1 Introduction

retex is a TEX engine. Given a LATEX document, it produces the PDF file that lualatex would produce. It reads the same packages, the same fonts and the same hyphenation patterns from an installed TEX Live, and on the documents we test its pages are identical to lualatex's, pixel for pixel, at 300 dpi. It can replace lualatex without other changes: it takes the same command-line options, writes the same log file, and has the same LuaTEX behaviour that packages rely on, from `\directlua` to the node library [12].

It differs from lualatex in what happens around the typesetting.

- It runs the passes itself. Cross-references, tables of contents, bibliographies and indexes settle in one command: retex reruns until the auxiliary files stop changing, and runs BibTeX, biber or makeindex in between when their inputs changed. It has its own implementations of those three programs, with the same output.
- It explains the errors. Each error and warning comes with its file, line and source text, an explanation in words and, where it can, the fix ("did you mean `\textbf`?"); it reports all of them instead of stopping at the first.
- It is faster. One pass takes a third to a half of lualatex's time on most documents, and later passes resume after the preamble, so a full build is two to three times faster (Section 15).
- It reads SVG and WebP images (Figure 1), and writes SVG, OpenDocument, HTML and Markdown as well as PDF. It makes tagged, accessible PDFs. It is also a language server for editors, and it runs entirely in a web browser.

retex is written in Rust, from scratch. It is a clean-room reimplementation. No code of another TEX implementation was read, whatever its license, and its behaviour comes from lualatex's output, LuaTEX's documented Lua interface, published specifications, and the programs that Knuth published as books [6] (Section 16).

1.1 Who wrote retex

retex has been entirely generated by a large language model (Anthropic's Claude). It therefore has no author in the usual sense: it is an automatic translation of the work of Donald Knuth and of the hundreds of volunteers who made TEX, LATEX and LuaTEX what they are, and all the credit goes to them. Séverin Lemaignan maintains the project. retex is released into the public domain (CC0 1.0), and so is this manual.

1.2 How this manual is organized

Section 2 to Section 11 (*Using retex*) explain how to install retex and use it day to day: compiling documents, reading its messages, bibliographies and indexes, rebuilding as you write, the other output formats, editors, and the browser editor. Section 12 to Section 14 (*Reference*) list every option and environment variable, and compare retex with lualatex point by point. Section 15 to Section 17 (*Inside retex*) are for the curious and for contributors: timings, the clean-room rules, and how retex is tested. The appendices hold a sandbox page to experiment with, and the history of the project.

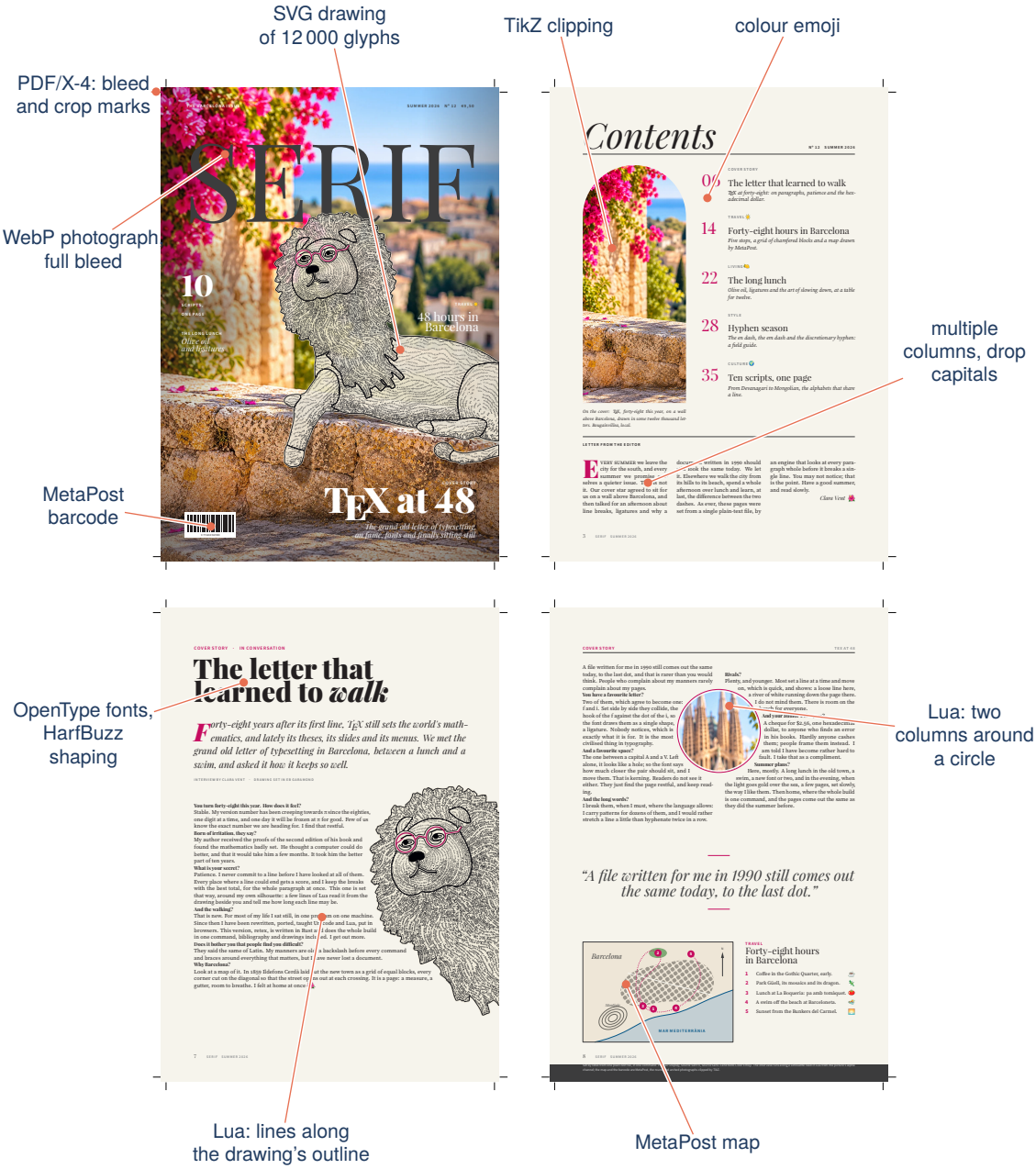


Figure 1: The magazine sample (`samples/magazine/magazine.tex`), four pages that retex sets from one source file in one command, and the features of retex they use. These are the pages of its PDF/X-4 for the press, which the same command writes beside the PDF, with 3 mm of bleed and crop marks.

1.3 Conventions

Commands typed in a terminal, and what retex prints, are shown in grey boxes; the `$` is the shell's prompt:

```
$ retex paper.tex
```

In the text, `--watch` is an option, `RETEX_HINTS` an environment variable, `paper.tex` a file, `\section` a TEX command and ⟨*file*⟩ something to replace. Every option and variable has an entry in Sections 12 and 13, in this form:

`--example` a default, if any

What the option does.

They are all in the index at the end of the manual.

> **Try it** This manual is a project of the browser editor, as any other: the sentence you are reading is in `chapters/introduction.tex`. Hover over `\cref` in the source to see what it is, or over the label in `\cref{sec:performance}` to see the section it refers to; press `F12` on it to go to the label.

PART I

Using retex

2 Installing retex

retex is one program, retex. It reads the LaTeX macro files and the fonts from an installed TeX Live 2025 or 2026, and honours the environment variables and `texmf.cnf` settings that kpathsea and LuaTeX use (`TEXINPUTS`, `TEXFONTS`, `openout_any`, `shell_escape`, and so on). Without an installed TeX Live, it makes a tree of its own from TeX Live's packages, downloaded as documents need them (Section 2.4). It does not need lualatex itself, nor bibtex, biber or makeindex.

2.1 Release programs

Each release on GitLab comes with programs for Linux (x86_64, glibc 2.28 or later), Windows (x86_64) and macOS (Apple silicon and Intel). The latest ones are always at these addresses:

Linux x86_64	`retex-x86_64-linux.tar.gz`
Windows x86_64	`retex-x86_64-windows.zip`
macOS Apple silicon	`retex-aarch64-macos.tar.gz`
macOS Intel	`retex-x86_64-macos.tar.gz`

Unpack the archive and put `retex` somewhere on your `PATH`. The macOS programs are not notarized, so macOS refuses to start a downloaded one until its quarantine flag is removed:

```
$ xattr -d com.apple.quarantine retex
```

2.2 From crates.io or from the sources

With a Rust toolchain (Rust 1.88 or later), retex installs from crates.io:

```
$ cargo install retex
```

or builds from a clone of the repository, as `target/release/retex`:

```
$ git clone https://gitlab.com/severin-lemaignan/retex.git
$ cd retex
$ cargo build --release
```

The repository also has a manual page, `crates/retex-cli/retex.1`.

retex downloads missing TeX Live packages (Section 2.5). That needs an HTTPS client and decoders for TeX Live's archives, in the cargo feature `download`, which is on by default. Without it, retex has no network code at all; it still names the package of a missing file, and uses packages downloaded before:

```
$ cargo install retex --no-default-features
```

2.3 The first run

The first time it compiles a LaTeX document, retex builds its LaTeX format from your TeX Live (a few seconds) and keeps it in its cache directory, with the index of the fonts' names (`RETEX_CACHE`, Section 13). To check what retex found:

```
$ retex --version
retex 1.8.0 (TeX Live 2025, LaTeX 2025-11-01; LuaTeX 1.22 compatible)
TeX Live 2025: /usr/share/texlive (from /usr/share/texlive/release-texlive.txt)
package downloads: ask (default)
```

retex follows the TeX Live it finds: with TeX Live 2026's LaTeX it typesets as TeX Live 2026's lualatex (LuaTeX 1.24) does (Section 14.4).

2.4 Without TeX Live

On a system without TeX Live (a new machine, a container, a `cargo install` without a TeX distribution), retex makes its own tree of TeX Live 2026 in `texlive/2026/` of the cache directory (Section 2.5) and uses it as the only TeX tree. The first run says so, after the question of Section 2.5 when it runs on a terminal (the answer also holds for the packages the document needs later in the run):

```
retex: no TeX Live installed: making retex's own TeX Live 2026 in
  ~/.cache/retex/texlive/2026 (packages are downloaded as documents need
  them; --offline never downloads packages)
```

It downloads the package database, then the package `kpathsea`, whose `texmf.cnf` gives the tree TeX Live's layout and search paths (`TEXMFHOME` is still searched first), and `hyphen-base`. retex then writes the files that TeX Live's installer generates and that no package holds: `language.dat`, `language.def` and `language.dat.lua`, from all the hyphenation patterns the package database lists (the pattern files themselves come when a language is used), and `pdftex.map`, from the font maps of the packages installed. Every other package is downloaded when the format build or a document looks up one of its files. For an `article` with `amsmath`, `graphicx` and `hyperref`, that is 45 packages and 18 MB of archives (77 MB unpacked, with 30 MB more for the package database and its index), most of it the Latin Modern fonts (`lm`, 12 MB). The first run took 27 seconds in a container on a home connection; the next runs need no network. The first download of a run names the repository; the others are one line each. The format of this tree is kept apart from that of an installed TeX Live (Section 13).

The same tree serves a year other than the installed one: `--tl=2026` on a system with TeX Live 2025 uses retex's own TeX Live 2026, and warns that the installed one is not used.

With downloads off (`--offline`, `RETEX_OFFLINE`, the setting `packages.download`, or a retex built without the `download` feature) and no tree made yet, or when the network does not answer, the run stops with one error that says why, and how to install TeX Live with the system's package manager:

```
error: no TeX Live found
  | retex takes the LaTeX kernel, packages and fonts from a TeX Live installation
  | no texmf.cnf with a LaTeX kernel was found beside kpsewhich (on PATH) or in
  | the usual places (/usr/share/texlive, /usr/local/share/texmf, /etc/texmf)
  | retex cannot make its own TeX Live tree: package downloads are off (from --offline)
  | help: install TeX Live: `sudo apt install texlive-luatex texlive-latex-recommended`
```

```
|       or run again without --offline
```

A tree made once works offline. A font asked for by its family name with `\setmainfont`, such as `TeX Gyre Pagella`, is found only once its package is in the tree, since the package database has no font names; `retex packages install tex-gyre` installs that one.

2.5 Missing TEX Live packages

When a document needs a file that your TEX Live does not have, retex can download the TEX Live package that holds it and go on; the file is found in the same run. By default (the setting `packages.download` is `ask`), a build on a terminal asks first, once per run:

```
retex: lmroman10-regular.otf is in the TeX Live package `lm` (11.9 MB), which is not installed
retex: download missing TeX Live packages? [yes/no/always/never] (Enter: yes)
```

`yes` (or Enter) and `no` hold for the rest of the run, its later passes included, and for the rebuilds of `--watch`; `always` and `never` also set `packages.download` to that value in the configuration file (Section 13.2), so that later runs do not ask. A build that cannot ask downloads without asking, as retex did before 1.8: when stdin or stderr is not a terminal (a script, CI, an editor), with `--message-format=json`, with `-interaction=batchmode`, `nonstopmode` or `scrollmode`, and in the compatibility mode. `retex packages install` downloads without asking too.

On a terminal, the progress line counts the packages downloaded, and a line under it names the one being downloaded; at the end of the build, one line sums them up:

```
retex: downloaded 1 TeX Live package (7 KB) from
  https://ftp.math.utah.edu/pub/tex/historic/systems/texlive/2025/tlnet-final;
  --offline never downloads packages
```

When the output is not a terminal (a log file, an editor), each package gets a line of its own as it is downloaded. The packages are those of your TEX Live's year (`--tl`), from TEX Live's own repository: the year's frozen `tlnet-final` for a past year, tlnet (through `mirrors.ctan.org`) for the current one. The first download fetches the year's package database (about 3 MB), checked against TEX Live's signature; each package is checked against the database's checksum. Only the packages' run-time files are installed (no programs, no documentation), in `texlive/<year>/` of the cache directory, beside the version directories (inside `RETEX_CACHE` when it is set). retex searches there after the whole search path of your TEX Live, so a downloaded package never hides an installed file, and once you install a package with your system's tools, that copy is used. A package's dependencies are not downloaded with it: each file is fetched when it is needed. A run downloads at most 200 MB.

Downloads are off with `--offline` for one run, with `RETEX_OFFLINE` in the environment (`export RETEX_OFFLINE=1` for a shell session without a network), or for good with the setting `packages.download` (Section 13.2):

```
$ retex config --global packages.download never
$ retex config packages.download
never
$ retex config --list
packages.download = never
```

The option and the variable win over the setting; `always` turns downloads back on without the question, `ask` with it (`auto`, the value of earlier versions, counts as `always`). The compatibility mode (`--compat=lualatex`) downloads and uses packages as the default mode does, without asking. `--version` and `--verbose` say whether downloads are on, and why. With downloads off, the message for a missing file still names its package and the command that installs it on your system (Section 4), when the year's package database is there. A build that lacked such files ends with a summary: the packages, the files each was needed for, why they were not downloaded (downloads off and by what, downloads paused, the error of a download), and one command of the system's package manager that installs them all:

```
retex: the document needs the TeX Live package `pgf` (for tikz.sty), which is not installed
  not downloaded: package downloads are off (from --offline)
  help: run again without --offline
        or install it with the system's package manager: `sudo apt install texlive-pictures`
        (probably; `apt-file search tikz.sty` tells for sure)
```

With `--message-format=json` it is a `missing-packages` record.

Without a network, retex does not wait long. Most runs do not use the network at all: once the package database is there, a file that no package has is known to be missing without asking the repository, and nothing is fetched for the files your TeX Live has. When a download is needed and the network does not answer, the run gives up after 3 seconds for the name lookup and 3 seconds for the connection, and the runs of the next 10 minutes do not try again (the file `texlive/network-down` in the cache directory records the time). A missing file then gets a note that downloads are paused; `retex packages install` tries at once. A network that answers with an error (no such host, connection refused) costs a few milliseconds and does not pause anything.

`packages.repository`, or `RETEX_REPOSITORY`, names another repository with TeX Live's layout: a mirror's `tlnet`, a local copy, a TeX Live DVD. `retex packages` manages what was downloaded:

`retex packages list` the packages downloaded for the year, with their revision and size;

`retex packages which` ⟨***file***⟩**...** the package of each file, and whether it is installed, downloaded or neither;

`retex packages install` ⟨***package or file***⟩**...** downloads now, to prepare a CI cache;

`retex packages update` fetches the package database again (the current year's changes daily);

`retex packages prune` removes the packages your TeX Live now has; with `--all`, every one.

2.6 Platforms

retex is developed and tested on Linux. **Windows and macOS support is experimental**. The programs are built and checked from Linux only. Both compile, pass the linter and link; the Windows program passes the test suite and typesets the samples identically under Wine (except two that need makeindex, a Linux program there). Neither has been run on a real Windows or macOS machine yet, and reports of what works and what does not are welcome. On Windows every pass runs from the start, as the preamble snapshot and the page checkpoints of `--watch` need the `fork` system call.

> Trying retex needs no installation: the browser editor (Section 9) runs it in a web page, with a TEX tree of its own.

3 Compiling documents

To compile a document, name its main file:

```
$ retex paper.tex
✓ paper.pdf · 12 pages · 284 KB · 1.9 s (3 passes)
```

There is no second run to make, no bibtex to call and no latexmk to run.

3.1 Passes

LaTeX learns where things are as it typesets: the number of a section is known when the section is typeset, its page when the page is shipped out. References to things not typeset yet (a forward `\ref`, a table of contents, a citation) are written to the auxiliary files and read by the next run. With lualatex, that is why one compiles twice, or three times, or runs latexmk.

retex runs the passes itself. After each pass it compares the auxiliary files with those the pass read; while they change, it runs another pass, at most five. Between passes it runs the tools a document needs (Section 5): BibTeX when the citations of the `.aux` file changed, biber when biblatex's control file (`.bcf`) changed, makeindex when the index entries changed. The summary line counts the passes, and `--verbose` says why each one ran:

```
$ retex --verbose paper.tex
retex: running biber paper
retex: pass 2: paper.aux, paper.toc, paper.bbl changed (cross-references,
  table of contents, bibliography)
retex: pass 3: paper.aux changed (cross-references)
retex: wrote paper.pdf (12 pages, 284 KB) in 1.9 s, 3 passes; log in
  .retex/paper-c14cb8c2/paper.log
```

The passes after the first do not start from scratch: retex takes a snapshot of itself at `\begin{document}`, where the preamble ends, and resumes from it while the files the preamble read are unchanged. A preamble that loads many packages is read once per build, not once per pass.

`--passes=1` gives the classic single run of lualatex, and `--compat=lualatex` goes further, for the tools that drive the passes themselves (Section 14.2).

3.2 Where the files go

The PDF is written to the current directory, as lualatex writes it. The other files a LaTeX run makes (the log, `.aux`, `.toc`, `.bbl`, `.idx`, `.out`...) go to a hidden directory next to the document, `.retex/<job>-<hash>/`, where ⟨*job*⟩ is the job's name and ⟨*hash*⟩ tells apart documents of the same name in different directories. The auxiliary files stay out of the directory of sources, and the summary line says where the log is:

```
✗ paper.pdf · 1 page · 7 KB · 0.21 s (2 passes) · 1 error ·
  log .retex/paper-1a2b3c4d/paper.log
```

`--no-aux-dir` keeps the auxiliary files next to the document, as lualatex does; so do `-output-directory` (everything goes to that directory), `-shell-escape` (programs the document runs expect to find them there) and the compatibility mode.

3.3 Errors do not stop the build

lualatex stops at an error and asks what to do, or, in `nonstopmode`, goes on and leaves the user with a log to read. retex reports every error and goes on to the end. The passes that settle the references run too, so the PDF is as complete as the errors allow, and the next build may well show that the error was the only problem. `--interactive` brings back TEX's dialog at each error, and `-halt-on-error` stops at the first one. The exit status says whether there were errors (Section 12.5).

3.4 Shell escape and the security of documents

As with lualatex, a document may run only the programs that `texmf.cnf` allows (`shell_escape_commands`: bibtex, makeindex, kpsewhich, and a few others), and only through `\write18`, Lua's `os.execute` and `io.popen`. `-shell-escape` allows every program, `-no-shell-escape` none. A document that runs makeindex itself, as imakeidx does, gets retex's own makeindex (Section 5.2).

3.5 Code listings: minted

minted colours code with Pygments, a Python program it runs through the shell escape (latexminted). retex does that work itself, in process. Documents that use minted compile faster, without Python and without `-shell-escape`, also in the browser, and their code is coloured as with lualatex, macro for macro, in these languages: Python, C, C++, CUDA, Rust, Java, Go, JavaScript, TypeScript, JSON, Bash (and `sh`, `zsh`), TEX and LATEX, XML, YAML, Markdown, R, Julia and MATLAB. Code in another language Pygments knows is typeset as it is, without colours, with a warning at the line of the listing that tells where to ask for its lexer (an issue on retex's GitLab); a name Pygments does not know is an error, which names the language it looks like a typo of. minted's options work (`escapeinside`, `mathescape`, `texcomments`, `autogobble`, the range options…), with Pygments' styles, except `rangeregex`.

minted 2 (`minted.sty` up to TEX Live 2024, `\usepackage{minted2}` from TEX Live 2025) runs pygmentize through the shell escape, and a few commands that manage its cache (mkdir, rm, mv, a Python one-liner for `autogobble`). In the default mode retex answers these command lines itself, whatever the shell escape mode, and colours the code as pygmentize does. minted 2 stops at the end of its loading unless the shell escape is enabled; retex makes that test pass, for minted 2 alone. Other commands keep the rules of the shell escape (Section 3.4). The package options `cache`, `outputdir`, `kpsewhich`, `finalizecache`, `frozencache` and `draft` work as with lualatex. In the compatibility mode (`--compat=lualatex`), the installed programs run, as with lualatex.

RETEX_MINTED

`external` runs the installed latexminted (minted 3) or pygmentize (minted 2) instead of retex's highlighting (it needs Python, and minted 2 needs `-shell-escape`).

3.6 Images

retex includes the images that lualatex includes (PNG, JPEG, PDF, JBIG2 and JPEG 2000), with the same sizes and the same pages. It also includes these formats:

- SVG, without Inkscape and without `-shell-escape`: a file `figure.svg` is included as other images are, by `\includegraphics` (also without its extension), or by the `svg` package's `\includesvg`. The text of the drawing stays text in the PDF, searchable and selectable. A shape in plain colours that the drawing repeats (through `<use>` elements, typically glyphs or symbols) is written once, as a form XObject that each copy draws with its own transformation.
- GIF (the first frame), WebP and TIFF, which lualatex refuses. They are decoded once and kept in retex's image cache, transparency included. A lossy WebP image is written as a JPEG at quality 90, its transparency as a lossless mask (its decoded pixels, compressed without loss, take about eight times more space). The JPEG differs from the decoded WebP image by about 40 dB PSNR. A lossless WebP image stays lossless. With `\pdfvariable compresslevel` set to 0 no stream is compressed, and WebP images are written as their decoded pixels.

Images are often what makes a TEX run slow. retex reads only their headers to know their size, copies JPEG data and most PNG data into the PDF without decoding them, and keeps what it had to convert (a PDF figure whose page is many streams, as plotting programs write them) in its cache for the next passes and builds. An image that cannot be decoded when the PDF is written is an error, and leaves an empty space.

> **Try it** The front page includes two JPEG images, `images/cc0-litho1.jpg` and `images/cc0-litho2.jpg`, cut, rotated, clipped and blended by TikZ in `hero.tex`; its logo is an SVG file, `images/retex-logo.svg`. Drop an image of your own on the file list and include it in Section A.

4 Reading the messages

A TEX log is written for the program's authors. The error is somewhere in four thousand lines, its line number is that of the file TEX happened to be reading, and the help speaks to someone at a terminal in 1982. retex keeps that log, byte for byte as LuaTEX writes it, and tells you on the terminal what went wrong, where, and what to do about it. Here is a document with two mistakes:

```
\documentclass{article}
\begin{document}
\section{Results}\label{sec:results}
Some \textbff{bold} text.
See Section~\ref{sec:result}.
\typeout{Hello from the document}
\end{document}
```

and what retex says about it:

```
error: undefined command \textbff
  --> paper.tex:4:6
  |
4 | Some \textbff{bold} text.
  |      ^^^^^^^^
  | help: did you mean \textbf or \textbar or \textsf?

note: Hello from the document  paper.tex:6
warning: undefined reference `sec:result`
  --> paper.tex:5:27
  |
5 | See Section~\ref{sec:result}.
  |                           ^
  | help: did you mean `sec:results`? (\label{sec:results} in ./paper.tex, line 3)

✗ paper.pdf · 1 page · 8 KB · 0.20 s (2 passes) · 1 error, 1 warning ·
  log .retex/paper-9988aa08/paper.log
```

4.1 Errors, warnings, notes and hints

Each message has a kind, its place in your files (the file, the line, the column, and the source line with the problem marked), an explanation and help. The kinds are:

error something TEX or a package reports as an error: an undefined command, a missing brace, a math command outside math, a file not found, an error in the bibliography's database. The document is probably not what you meant. Errors are listed first.

warning something LATEX or a package warns about: an undefined reference or citation, a line too wide, a font substituted, a label defined twice. Many layout warnings (lines too wide, spacing stretched) are shown one line each; `--verbose` shows them all in full.

note what the document itself writes to the terminal (`\typeout`, `\message`) or shows (`\show`, `\showthe`, `\showbox`), with the line that did it. The messages of packages go to the log only.

hint something about the source that is legal but probably not what you want (Section 4.3). Hints come after the build.

retex places each message on the line of *your* file that caused it, also when TEX found the

problem deep inside a package's macro; where LaTeX gives a message without a place (an undefined reference at the end of the run), retex finds the place in the source. A long list is cut short; the log has everything.

When it knows how, retex says what fixes the problem: a misspelled command or label gets the names it was probably meant to be, a math command in text the formula it belongs to, a missing brace where it is missing. Editors apply these as quick fixes (Section 8).

For a package or file that is not installed, retex names the TeX Live package that holds it and the command that installs it on your system, when it has TeX Live's package database, `texlive.tlpdb`, of the year in use (`--tl`):

```
error: package `xepersian` not found
  --> paper.tex:2:21
  |
2 | \usepackage{xepersian}
  |                     ^
  | \usepackage{xepersian} loads the file xepersian.sty, which retex cannot
  | find
  | xepersian.sty is in the TeX Live package `xepersian`
  | help: install it: `sudo apt install texlive-lang-arabic` (probably the
  |       Debian package with `xepersian`; `apt-file search xepersian.sty`
  |       tells for sure)
```

The command is apt on Debian and Ubuntu, `dnf install 'tex(file)'` on Fedora, pacman on Arch, and tlmgr for a TeX Live installed with TUG's installer (MacTeX included). retex downloads the database with the first missing package (Section 2.5); this message appears when downloads are off or the download failed. A TeX Live installed with TUG's installer has its own database, of the installed packages only, which names the package of an installed file that has gone missing. Without a database, the message names `tlmgr search --global --file`.

4.2 The summary line

The last line says what came out. On a terminal:

```
✓ paper.pdf · 12 pages · 284 KB · 1.9 s (3 passes)
✗ paper.pdf · 1 page · 8 KB · 0.20 s (2 passes) · 1 error, 1 warning ·
  log .retex/…/paper.log
```

Through a pipe or into a file, a sentence:

```
retex: wrote paper.pdf with 1 error and 1 warning (1 page, 8 KB) in 0.20 s,
  2 passes; log in .retex/paper-9988aa08/paper.log
```

While it runs, a terminal shows the pass and the pages done so far. On terminals that support them, the file names and places are links (`--hyperlinks`): a click opens the file, or your editor at the line (`--hyperlink-format`).

4.3 Hints about the source

After a build retex reads the document's source for common mistakes, which TeX accepts without a word and which show in the output:

- display math as `$$ … $$`, which is plain TEX and gives LATEX the wrong spacing; `eqnarray`, which puts too much space around the relations;
- obsolete font commands (`{\bf …}`, `\it`, `\rm`, `\tt`…);
- a space swallowed after a command (`\LaTeX is`);
- a line break allowed between a word and its reference (`Figure \ref{…}`) where a tie belongs (`Figure~\ref{…}`);
- a hard-coded number where a reference belongs (`Section 2`);
- a `\label` before its `\caption`, which refers to the section instead of the figure;
- straight double quotes in text, three periods instead of an ellipsis, a hyphen in a range of numbers (`10-20`), a sentence space after "e.g." or "i.e.";
- `x^10`, which sets only the 1 as the exponent; a function name in italic letters (`sin x` for `\sin x`);
- hyperref loaded before packages that should come before it, cleveref before hyperref.

Most of them come with their fix. Five kinds are shown at most; the environment variable `RETEX_HINTS` shows all of them (`all`) or none (`0`).

> **Try it** The appendix Section A has a paragraph with a few of these mistakes, on purpose. Open `chapters/sandbox.tex`: the hints are marked in the source as you type, and Ctrl + . on one shows its fix.

4.4 Quieter, louder, and for programs

- `-q` shows only the errors, and the summary line only when the build failed.
- `--verbose` shows every warning in full, the passes run and why, and the tools run.
- `--message-format=short` writes one line per message, in the form compilers use, which editors and terminals recognize:

```
paper.tex:4:6: error: undefined command \textbff (did you mean \textbf or
  \textbar or \textsf?)
paper.tex:6: note: Hello from the document
```

  `-file-line-error` selects this form too.

- `--message-format=json` writes JSON Lines on the standard output, one record per event, for editors and other tools:

```
{"type":"diagnostic","severity":"error","message":"undefined command \\textbff",
 "origin":null,"file":"paper.tex","line":4,"column":6,"end_column":14,
 "source":"Some \\textbff{bold} text.","notes":[],
 "help":["did you mean \\textbf or \\textbar or \\textsf?"],
 "fixes":[{"column":6,"end_column":14,"replacement":"\\textbf"}, …]}
```

  Besides `diagnostic`, the records are `pass`, `progress`, `build-started` (with `--watch`), `finished`, and those of `profile` and `explain`; `--symbols` adds the document's index for editors. Their schema is version 2, and stable: fields are only added.

4.5 TeX's own dialog

`--interactive` stops at each error and asks what to do, as TEX does on a terminal; the log and the messages are those of the default mode otherwise. The compatibility mode

(Section 14.2) gives LuaTEX's terminal output, unchanged, for programs that read it.

5 Bibliographies and indexes

A bibliography or an index needs a program between two LaTeX runs: BibTeX [15] or biber [5] to sort and format the references the document cites, makeindex [9] to sort its index entries. retex runs them itself, between its passes, when their inputs changed, and it runs its own implementations of the three programs, in the same process. Nothing else needs to be installed, and bibliographies and indexes work in the browser too, where no program can be started.

5.1 BibTeX and biber

A document that uses `\bibliographystyle` and `\bibliography` gets BibTeX when the citations, the style or the databases in its `.aux` file changed; a document that uses biblatex gets biber when its control file, `.bcf`, changed (or BibTeX, with biblatex's `backend=bibtex`). There is nothing to configure: the document is written as for lualatex and compiled with retex.

retex's BibTeX writes the same `.bbl` file as BibTeX 0.99d, byte for byte, with the same warnings. It reads the `.aux` file, parses the databases (macros, concatenation, `@preamble`, `@comment`, the recovery from errors), runs the `.bst` style with its 37 built-in functions [16], cross-references and `-min-crossrefs`, sorts, and breaks the lines of the `.bbl` file as BibTeX does. Like BibTeX it works on bytes: UTF-8 passes through, but is measured and cut as bytes. BibTeX's fixed capacities (`max_strings` and the like) are not limits.

retex's biber is the part of biber that documents use: the control file, BibTeX-format databases in UTF-8 and other encodings, the decoding of LaTeX into Unicode, names in the BibTeX format and the extended `family=…` format, dates (ISO 8601 and the EDTF extensions biblatex supports), sorting templates and locales (with the Unicode Collation Algorithm [19] and the tailorings of the locales), labels (`labelalpha`, `extradate`, `extraalpha`), the uniqueness of names and lists (`uniquename`, `uniquelist`, `singletitle`…), inheritance (`crossref`, `xref`, `xdata`, sets), source maps, and the `.bbl` file of biblatex 3.21 as biber 2.21 writes it. It does not implement biber's tool mode, other database formats (biblatexml, RIS, Zotero RDF, EndNote XML), remote databases, or validation of the data model. A document that needs one of those gets the installed biber.

On the test cases of Section 17.2, the `.bbl` files are identical to those of BibTeX on all 1249 cases, and to those of biber on 148 of 149.

Errors in a database are errors of the build, at their line of the `.bib` file, in words:

```
error (biber): syntax error in the .bib file: found ",", expected one of:
  number, name (entry type, key, field, or macro name) or quoted string
  ({...} or "...")
  --> refs.bib:4:12
  |
4 |   year   = ,
  |            ^
  | biber stops at a syntax error: the bibliography is empty until it is fixed
```

After such an error, the warnings that follow from it (every citation of the entry undefined) are not repeated.

RETEX_BIB

`external` runs the installed bibtex and biber instead of retex's own.

5.2 makeindex

A document with an index (makeidx, imakeidx, or the doc package of `.dtx` files) gets makeindex when its index entries changed. For documentation made with doc, retex uses doc's styles, `gind.ist` for the index and `gglo.ist` for the change history (`<job>.glo` into `<job>.gls`). Documents that run makeindex themselves through the shell escape, as imakeidx does for its several indexes, get retex's makeindex too: in the default mode, a `makeindex` command line run by `os.execute` or `io.popen` runs it in process, as the restricted shell escape allows. (In the compatibility mode, the installed program runs, as with lualatex.)

retex's makeindex writes the same `.ind` file as makeindex 2.17, byte for byte, the same transcript (`.ilg`) and the same messages: the style files, the entries with their levels, actual keys, encapsulators and ranges, page numbers of every kind, makeindex's order of keys and of entries that compare equal, the merging of pages into ranges, and the wrapping of lines. It reads the command-line options `-c`, `-g`, `-i`, `-l`, `-q`, `-r`, `-s`, `-o`, `-t` and `-p`. It does not implement `-L` (sorting by the locale) or `-T` (Thai), nor the other index processors (xindy, mendex, upmendex); documents that ask for them get the installed programs.

retex's makeindex makes the index of this manual. The entries of the options and the environment variables are written by the environments that describe them (`retex-manual.sty`), and makeindex sorts them between two passes.

`RETEX_MAKEINDEX`

`external` runs the installed makeindex instead of retex's own, also for the command lines of the shell escape.

Try it Cite another work in this paragraph: `\cite{knuthplass}` (the entries are in `retex-manual.bib`; completion lists them after `\cite{`). The bibliography at the end of the manual follows, sorted by biber. Then add `\index{sandbox}` somewhere: the entry appears in the index.

6 Rebuilding as you write

6.1 `--watch`

```
$ retex --watch --open paper.tex
```

keeps retex running: it rebuilds the document whenever a file the document uses changes (the main file, the files it inputs, its images, its bibliography), and `--open` opens the PDF in your viewer once the first build is done. Viewers that reload a changed file (most do) follow.

A rebuild does not start from the beginning. retex keeps checkpoints of the build at the pages it shipped out; after an edit, the rebuild starts from the last page before the edit. Once the rebuild reaches the same state as the previous build had at the same page, retex writes the PDF at once, with the previous build's pages after that point, and checks them with the rest of the rebuild. On a long document, an edit on page 30 is in the PDF in the time it takes to typeset a page or two. An edit of the preamble restarts from the beginning.

On a terminal, each rebuild replaces the previous one's messages (the scrollback keeps them, and `--no-clear` keeps everything on the screen), and ends with a status line: when it ran, what changed, where the rebuild resumed, and whether the errors are fixed. The window's title shows whether the document is fine (✓ or ✗). `--notify` notifies you when errors appear or are fixed (a desktop notification in iTerm2, WezTerm, Ghostty, kitty and foot, the bell elsewhere). Errors never stop the watching. Enter forces a full rebuild, Ctrl + C ends it.

6.2 The figure cache

TikZ pictures are among the slowest things TEX typesets, and most of them do not change from one build to the next. retex keeps the pictures it typesets in a cache (`figures` in its cache directory, at most 128 MB): a picture whose source and starting state are unchanged is replayed from the cache instead of typeset, with the same output, byte for byte. Rebuilds of documents with many pictures are a quarter to a third faster. For now these are the pictures in a paragraph or a formula; a picture that starts a paragraph of its own (alone in a `figure`, say), runs Lua code, makes PDF objects (shadings, patterns) or writes messages is typeset every time. The compatibility mode does not use the cache.

`RETEX_FIGURES`	on
`off` turns the figure cache off; `stats` shows at the end of each pass what it did; `verify` typesets every picture and compares it with its record.	

6.3 What else is kept

Besides the format and the figures, retex's cache directory holds the index of the fonts' names (the slowest part of a first `\setmainfont`), the lists of the TEX tree's files, and the images it decoded or compressed for the PDF. It is shared by every document, and kept per version of retex: a new version starts with an empty one.

7 Other outputs

7.1 Accessible PDF

A tagged PDF carries, besides the page, the structure of the document: headings, paragraphs, lists, tables, figures with their alternative text, the reading order and the language. Screen readers need it, and the PDF/UA standard [2] describes it. LaTeX can tag documents itself, with `\DocumentMetadata{tagging=on}`.

```
$ retex --accessible paper.tex
```

makes a tagged, accessible PDF (PDF/UA-2, PDF 2.0) with LaTeX's own tagging, for a document that does not ask for it, as if it began with

```
\DocumentMetadata{tagging=on, pdfstandard=ua-2, pdfversion=2.0}
```

A document with its own `\DocumentMetadata` keeps it, and the beamer class is left untagged, as LaTeX does not support it yet. Then retex reports what screen readers would miss:

- a document without a title or a language;
- images without alternative text, given as `\includegraphics[alt={…}]{plot}`;
- TikZ pictures without alternative text, whose drawings LaTeX otherwise hides from screen readers without a word: `\begin{tikzpicture}[alt={…}]`;
- formulas without MathML, and other messages of LaTeX's tagging, in words.

With `--accessible`, a PDF without a title or a language fails the build: retex writes it, but exits with an error. The checks also run for documents that turn tagging on themselves, where they only report (and not in the compatibility mode, Section 14.2). In our tests retex's tagged PDFs have the same structure as lualatex's and pass veraPDF's PDF/UA-2 profile. Tagging changes the layout a little and takes longer, as with lualatex, which is why it is not the default.

7.2 PDF for the press: PDF/X

PDF/X is the subset of PDF that printing presses accept: colour is defined in terms of the inks, the file names the printing condition it was made for, and it contains no interactive elements (links, forms, JavaScript) [3]. A PDF made by LaTeX does not meet these requirements.

```
$ retex -output-format=pdf-x4 book.tex
```

writes `book.pdf` as a PDF/X-4, the version most printers ask for. `-output-format=pdf-x1a` writes a PDF/X-1a, for older print workflows; PDF/X-1a allows neither RGB colour nor transparency. The default printing condition is offset printing on coated paper (FOGRA39, the profile of the colorprofiles package).

```
$ retex -output-format=pdf,pdf-x4 --bleed=3mm --crop-marks book.tex
```

writes `book.pdf` for the screen, with its links, and `book.pdf-x4.pdf` for the press, with 3 mm of bleed and crop marks, from one build. The bleed and the crop marks apply to the

PDF/X only.

The printing condition. `--output-intent` names the printing condition the colours are converted for, by its name in the ICC's registry of characterization data or as a CMYK profile file, such as the one a printer supplies (`--output-intent=printer.icc`). The names known are FOGRA39 (the default), FOGRA51 and FOGRA52 (the PSO v3 conditions of ISO 12647-2:2013), FOGRA27 to FOGRA47, GRACoL, GRACoL2013, SWOP, SWOP5, SNAP and IFRA26. The profile of a name is looked for in TEX Live and in the system's ICC directories (`/usr/share/color/icc` on Linux, where the colord package installs most of these, `/Library/ColorSync/Profiles` on macOS); if it is not found, the build stops with the name of the file needed. FOGRA51 and GRACoL2013 have no free profile in these places; their profiles are distributed by the ECI and by Idealliance.

A document that declares PDF/X itself, with

```
\DocumentMetadata{pdfstandard=X-4}
```

or with the pdfx package, is converted without the option. LATEX records the declaration but does not produce most of what the standard requires.

The conversion. Neither pdfx nor `\DocumentMetadata` is needed. After the last pass, retex rewrites the PDF as follows.

- It adds the parts that identify a PDF/X: the output intent (the ICC profile of the printing condition), the version in the XMP metadata, the `Trapped` key, and a TrimBox on every page (the page itself when the document sets none). Metadata from pdfx, hyperref or `\DocumentMetadata` is kept and completed. An empty title, which hyperref writes when the document gives none, is replaced by the document's name.
- It converts colour to CMYK through the output profile: text, rules, drawings and shadings, and, in a PDF/X-1a, images. Gray and black are converted to black ink only, which keeps small text sharp. A PDF/X-4 keeps its images in RGB, tagged as sRGB, so that the printer converts them with the press's settings, and keeps its transparency, blended in CMYK. In a PDF/X-1a, transparency is flattened: the areas covered by transparent objects become CMYK images at 400 dpi, and the text and drawings above them are left unchanged.
- It removes what a press does not use: links, comments, form fields, actions, JavaScript and transfer functions. Form fields and stamps marked as printable are first drawn into the page, as a viewer prints them. Bookmarks are kept. Layers are kept in a PDF/X-4; PDF/X-1a does not allow them, so layers hidden by default are removed and the others merged into the page.
- It moves the page boxes inside the MediaBox, and the TrimBox inside the BleedBox.
- It sets black text smaller than 12 pt and black lines thinner than 2 pt to overprint, as GWG 2022 asks: the inks under them are kept, so that misregistration of the plates leaves no white edge around them. Black is any tint of the black ink alone (CMYK $0, 0, 0, k$, the conversion of gray); the overprint is set in overprint mode 1, in which the black still replaces the black ink below. A light gray over a colour therefore prints as a mixture of the two. `--no-black-overprint` leaves such text and lines knocking out the inks below.

Each change with a visible effect is reported in a note, for example the number of links removed or the pictures flattened.

The PDF/X-4 output also follows the Ghent Workgroup's specification (GWG 2022 [4]), which printers use for their checks. The standard allows text and drawings in RGB tagged as sRGB; the GWG specification allows RGB only in images.

What the conversion does not change. The output is a valid PDF/X, but some properties of the document determine how it prints, and they are best checked on a proof from the printer.

- *Colour.* RGB colours outside the printable range of the paper, such as saturated blues and greens, print duller. Colours that have to match (a logo, a brand) are best defined in CMYK with values from the printer or designer, for example `\definecolor{brand}{cmyk}{1,.45,0,.2}`. CMYK colours are not converted.
- *Bleed and trim.* Pictures or backgrounds that reach the edge of the page need to extend 3 mm beyond it, because the paper is cut after printing. With `--bleed=3mm`, the page as designed becomes the TrimBox (where the paper is cut), the BleedBox is 3 mm larger on each side, and the page grows to it. The content moves with the boxes, so that what the document draws beyond the edge of its page prints in the bleed: a background placed with eso-pic or TikZ only needs to extend past the paper, for example

  ```
  \AddToShipoutPictureBG*{\AtPageLowerLeft{\put(\LenToUnit{-3mm},\LenToUnit{-3mm})
    {\includegraphics[width=\dimexpr\paperwidth+6mm]{photo}}}}
  ```

  The PDF for the screen shows the page as designed, since viewers cut the drawing at the edge of the page. `--crop-marks` adds crop marks at the corners of the TrimBox, outside the BleedBox, in registration colour (all inks), as printers expect; the page grows to make room for them. A document that sets its own TrimBox, with the crop package or

  ```
  \pdfvariable pageattr{/TrimBox [8.504 8.504 428.03 306.14]
                        /BleedBox [0 0 436.54 314.65]}
  ```

  (a 148 mm by 105 mm card with 3 mm of bleed, in points from the lower-left corner) keeps it, and the bleed is placed around it. Without a TrimBox or `--bleed`, retex uses the whole page. Text is usually kept 5 mm inside the trim.
- *Small text and thin lines.* Inks are not printed in exact register. Text smaller than 12 pt is therefore best printed in black ink alone, not in rich black (black with cyan, magenta and yellow), and small coloured text in as few inks as possible. White text on a coloured background needs 8 pt or more, and lines 0.25 pt or more.
- *Total ink.* Dark mixtures with more than 320% total ink coverage dry poorly on coated paper. Dark colours are usually black with small amounts of the other inks.
- *Images.* Print needs about 300 pixels per inch at the printed size; an image 1200 pixels wide prints at 10 cm. The conversion does not add resolution. PDF/X-1a does not allow JPEG 2000 images; JPEG and PNG are accepted.
- *Fonts.* Fonts are always embedded. Bitmap (Type 3) fonts, which the old TEX fonts are without cm-super or Latin Modern, print poorly; OpenType fonts loaded with fontspec do not have this problem.
- *Interactive content.* Links, forms and comments are not in the PDF/X. The PDF for on-screen reading is a separate file (`-output-format=pdf,pdf-x4`).

Print checks. After writing a PDF/X, retex checks it for what printers' preflight reports beyond the standard, with the limits of the GWG 2022 specification for sheet-fed offset printing. Each finding is a warning with its page and the line of the source that drew it, such as

```
warning (print): page 3: image photo.jpg prints at 120 ppi, below 150 ppi
  --> book.tex:212
```

- an image below 150 pixels per inch at its printed size (a one-bit image below 550), and, as notes, one from 150 to 224 pixels per inch, which may print soft, and one above 1200, which makes the file larger without printing finer;
- total ink above 320%, in a colour or in the pixels of a CMYK image;
- text smaller than 12 pt in rich black, text smaller than 8 pt printed with more than one ink, and white text smaller than 8 pt;
- a white object set to overprint, which does not print;
- black text smaller than 12 pt and black lines thinner than 2 pt drawn over other inks without overprint (in overprint mode 1), which happens in a PDF made with `--no-black-overprint` or by another program;
- a line thinner than 0.25 pt, or of width 0;
- registration colour (the `All` separation) on the page;
- content less than 3 mm inside the trim, and art that reaches the edge of the page and stops before the end of the bleed;
- bitmap (Type 3) fonts.

Sizes and widths are compared in PostScript points (bp), the unit of PDF and of the specification, so that text set at 8 pt in TEX (7.97 bp) is below the limit for white text; `\fontsize{8bp}{10bp}` meets it. A CFF font whose outlines are in 2048 units per em (Playfair Display, for one) is embedded in the PDF as LuaTEX embeds it, with its glyphs drawn 2.048 times larger than the size the PDF sets. Printers' preflight reads the size as set, and so do the checks of small text, with a note that gives the size drawn. The PDF/X conversion writes such fonts with the FontMatrix of their units per em and sets their text at the size it is drawn at, so that in a PDF/X the two sizes agree. The findings of one kind are grouped, one warning per source line, with the pages and their number, so that a page of full-bleed art gives one warning and not one per object. The place in the source comes from the SyncTeX records of the build, kept in memory when no SyncTeX file is asked for, and, for an image, from the `\includegraphics` that read it. Art drawn at shipout (eso-pic, page styles) is placed at the line TEX was reading when the page was shipped out. Crop marks and whatever lies outside the BleedBox are not checked.

The two checks at the trim treat included graphics as whole pictures. For the bleed, the page is rendered at 100 pixels per inch in strips along the four edges of the TrimBox, and each pixel is taken as ink or paper (paper when none of its colour channels is below 247 of 255, about a 3 % tint, so that white objects and the lightest parts of images count as paper). Where there is ink within 0.5 mm inside the trim and none in the band from 0.75 mm to 0.25 mm before the end of the bleed, within 0.5 mm along the edge, the art stops short. The bleed needed is the page's, up to 3 mm, or 3 mm when the page has none; outside the CropBox the page counts as paper. Stretches shorter than 1 mm are ignored, and stretches less than 2 mm apart are reported as one. The warning gives the length along each edge and the places, in millimetres from the top left corner of the trim (or "the whole right edge", "all four edges"):

```
warning (print): page 3: art at the edge of the page stops short of the
  3 mm bleed along 40 mm of the right edge (120 to 160 mm from the top)
```

A drawing made of many small pieces that crosses the trim, or a background under art that stops at the trim, therefore gives no warning. For the safety margin, text and paths in the page's content are judged one by one, and an included PDF or SVG file (a form XObject) as one object, by the box of what it paints with ink, the forms and images inside it included: a graphic that crosses the trim runs off the page and its parts are not checked, and one that lies within 3 mm inside the trim is reported once. An image is one object.

Checking a PDF.

```
$ retex check --pdfx=x-4 flyer.pdf
```

runs the same print checks on a PDF made by any program, with pages as places, and reports what keeps it from being a PDF/X-4 (or, with `--pdfx=x-1a`, a PDF/X-1a): its identification and output intent, its page boxes, fonts that are not embedded, the colour spaces and the transparency the variant does not allow, and interactive content. The exit status is 1 if the file is not a PDF/X of the variant. These checks are not a validation by a printer's preflight tool, which remains the reference.

The list of checks. A build that writes a PDF/X, and retex check, end with a list of the checks that warn or fail, the rules of the standard first and the print checks after them:

```
PDF/X-4 checks of magazine.pdf-x4.pdf: 27 checks, 25 passed, 1 warning
 print
  ⚠ images of 150 ppi or more: page 1
```

With `--verbose`, the list has every check, the ones passed (✓) and the notes (□) included; without warnings or errors, the list is left out. On a terminal without Unicode, the marks are the words `ok`, `note`, `warn` and `fail`.

7.3 SyncTeX

```
$ retex -synctex=1 paper.tex
```

writes `paper.synctex.gz` next to the PDF, for forward search (from a line of the source to its place in the PDF) and inverse search (from the PDF to the source) in editors and viewers [10]. The file has the same records as lualatex's: searches give the same results, and on simple documents the files are identical. `-synctex=-1` writes it uncompressed.

7.4 SVG

`--output-format=svg` writes the whole document as one SVG file, its pages side by side as Inkscape lays out a multi-page document. The text stays text, in the families of the document's fonts, so that it can be edited in Inkscape and searched in a browser; mathematics and scripts that need shaping stay outlines.

7.5 OpenDocument presentations and text documents

These two outputs are experimental. Rather than pictures of the pages, they make documents that office programs edit as their own [14].

`--output-format=odp` writes a beamer presentation as an OpenDocument presentation for LibreOffice Impress (and, through it, PowerPoint): editable text in the document's fonts, at TEX's positions with its line breaks; the frame titles and the title page as title placeholders; `itemize`, `enumerate` and `description` as lists; tables as tables; TikZ pictures and the formulas that cannot be text as pictures; overlays as animations, paragraph by paragraph; `\note` in the notes view; links; slide transitions; and movies, sounds and launched media files, embedded and played in place.

`--output-format=odt` writes an article, a report or a book as an OpenDocument text document that LibreOffice Writer (or Word) treats as its own: the paragraphs flow in reading order (a column after the other, the hyphens of TEX's hyphenation dropped) with their alignment, indents and spacing, in the document's fonts, embedded; headings are headings, with outline levels; the title, authors and date have paragraph styles of their own; lists are lists, footnotes notes, tabulars tables; formulas and TikZ pictures are pictures in the lines, displayed ones paragraphs of their own; floats keep their pictures and captions together, the captions numbered by Writer; labels are bookmarks and references links; two-column text is two columns; the page is the PDF's, its margins LATEX's text block, the page number a field.

7.6 HTML and Markdown

These outputs are also experimental, and come from the same analysis of the document as the text document. `--output-format=html` writes one HTML file: headings, the title page as the header, lists, tables, floats as figures with their captions, footnotes at the end linked both ways, references linked to their labels; the text in the document's fonts (embedded) and sizes, with TEX's alignment, indents and spaces; two-column text in two columns (one on narrow screens); formulas, TikZ pictures and PDF figures as SVG in the page, images as data. `--output-format=md` writes Markdown (CommonMark with GitHub's tables and footnotes): emphasis, strong emphasis and code from the fonts, links, footnotes, pipe tables, and the pictures as PNG files in `<job>_files/`.

7.7 DVI

`-output-format=dvi` writes a DVI file, as LuaTEX does.

> **Try it** In the browser editor, the preview shows this manual as a PDF, as HTML or as Markdown (the buttons above the preview), and the arrow next to *Download PDF* exports it as an accessible PDF, an OpenDocument text document, HTML, Markdown or SVG. The accessible PDF comes with the list of what screen readers would miss: the front page's TikZ pictures have no alternative text.

8 retex in editors

8.1 The language server

`retex lsp`

Run as a language server: LSP [13], JSON-RPC on the standard input and output.

Editors that speak the Language Server Protocol (Neovim, Helix, Emacs, Kate, Zed…) start it for TEX files. It gives them:

- the build's errors and warnings: when a `.tex` file is opened or saved, retex builds the document, as `retex main.tex` would (the PDF is written), and the editor marks its errors and warnings in the source, with retex's explanations, and its fixes as quick fixes;
- the hints about the source as you type (Section 4.3), with "Fix all hints" as a source action;
- hover: a command's arguments, package and definition; a label's number, page and title; a citation's entry;
- go to definition: a command's, also in the TEX tree; a label's; a `.bib` entry's; an input file;
- references: where a label is referred to, where a key is cited;
- completion of commands (with their arguments, as snippets), environments, labels (with their titles) and citation keys;
- the outline of the sections, and links to the input files.

Builds read the files as saved. What the server knows of the document (its commands, labels, citations) comes from the last build: `--message-format=json --symbols` writes that index, for other tools.

The main file is the one a `%!TEX root = main.tex` comment names in the first lines of the file; otherwise the file itself if it has a `\documentclass`; otherwise a `.tex` file with a `\documentclass` in its directory, or in a directory above.

8.2 Setting up an editor

Neovim (0.11 and later)

in `init.lua`:

```
vim.lsp.config('retex', { cmd = { 'retex', 'lsp' }, filetypes = { 'tex' } })
vim.lsp.enable('retex')
```

Helix

in `languages.toml`:

```
[language-server.retex]
command = "retex"
args = ["lsp"]

[[language]]
name = "latex"
language-servers = ["retex"]
```

Emacs (eglot)
in the init file:

```
(add-to-list 'eglot-server-programs
             '((latex-mode LaTeX-mode) "retex" "lsp"))
```

VS Code
needs an extension that starts the server; none is written yet.

8.3 Without the language server

retex fits the editors' usual ways of running TEX as well:

- With `--message-format=short` or `-file-line-error`, each message is one line, `file:line:column: error: message`, which Vim's `:make`, Emacs's compilation mode and the problem matchers of other editors read.
- `-synctex=1` gives forward and inverse search (Section 7.3).
- In a terminal, `--hyperlink-format` makes the places in the messages open your editor at the line: `vscode`, `vscodium`, `cursor`, `textmate`, `macvim`, `kitty`, or a template of your own.
- Tools that run the passes themselves (latexmk, arara, the build systems of editors) get LuaTEX's behaviour from `--compat=lualatex` (Section 14.2); most work with retex's default mode too, which finishes in one run.

9 The browser editor

retex also runs in a web browser. The editor at

`https://severin-lemaignan.gitlab.io/retex`

is a LaTeX editor with a live preview, in the manner of Overleaf, but without a server: the engine, compiled to WebAssembly, typesets the document in the page, from a TeX tree it fetches once and keeps in the browser. Your files stay on your computer. It works in current versions of Chrome, Firefox and Safari, on a phone too.

The editor opens on its *Projects* page. On a first visit, this manual is the project waiting there. It is a project like any other, which you can read and edit.

9.1 Writing

The source is on the left, the PDF on the right. The document is built 0.7 s after you stop typing, or at once with Ctrl + S (Cmd + S on a Mac) or the *Build* button. The first build of a project takes a few seconds, while the files it needs arrive; the next ones resume from `\begin{document}` and take about a second for a short document.

The problems of the build are marked in the source: an icon in the margin, a wavy underline under the problem, and its explanation in a popover when you hover it. The panel under the source lists them all, with the fixes retex knows: *Apply fix* on one, Ctrl + . in the source for the one at the cursor, or *Fix all hints*. The hints about the source (Section 4.3) follow as you type. The panel opens by itself on the first build of a project and on the builds you ask for, not as you type.

The language server (Section 8.1) runs in the page too:

- hover over a command for its arguments, package and definition, over a label in `\ref` or `\cref` for the number, page and title it refers to, over a citation key for its entry;
- completion opens as you type a command after `\`, an environment after `\begin{`, a label after `\ref{` or `\cref{`, a key after `\cite{`; Ctrl + Space opens it anywhere; a command comes with its arguments, the cursor in the first;
- F12, or Ctrl-click (Cmd-click), goes to a definition: a label, a bibliography entry, or the file of the TeX tree where a command is defined, shown read-only at the line.

Between the source and the PDF: Ctrl-click on a line where there is no definition shows its place in the PDF, and a double click in the PDF goes to the source line (SyncTeX, Section 7.3). The PDF's own links work too.

9.2 Files and projects

The file list on the left has the project's files (the button at the top left of the source shows or hides it). Drop files or folders on it (sources, styles, bibliographies, figures) or use its buttons; each file's menu renames, downloads or deletes it, or makes a `.tex` file the one to build. Everything is kept in the browser (IndexedDB): a reload, a closed tab or a restarted computer find the project as it was.

The *Projects* page (the button at the top left of the page) lists the projects kept in this browser, each with its first page, and starts new ones: blank, or from an example (this manual, an article, beamer slides, a report with an index, an ACM paper, an IEEE paper, and Kevin Raymond's flowers drawn in Lua, florilegium).

9.3 Projects on GitLab

Sign in with GitLab on the Projects page to list your gitlab.com projects and open them in the editor, or open a project by its address:

```
https://severin-lemaignan.gitlab.io/retex/?gitlab=group/project
```

with `&ref=`⟨*branch*⟩ and `&main=`⟨*file.tex*⟩ if needed. For another GitLab instance, give the project's whole address:

```
https://severin-lemaignan.gitlab.io/retex/?gitlab=
    https://gitlab.example.org/group/project
```

The project's files are downloaded once (a large repository can take a minute or two), then kept in the browser. The header shows the branch and its state: up to date, how many files you changed, or that the branch has new commits.

Commit lists your changes (files added, modified, deleted, renamed) and commits them to the branch, in one commit, with your message. When the branch cannot take it (protected, no permission, or changed in a way that conflicts), it offers a new branch and a merge request instead. When the branch has new commits, the editor takes them in and merges them with your edits; a conflict shows both versions, for you to choose. Public projects open without signing in, within gitlab.com's limits for requests without an account (these limits are low, so signing in is the normal way).

The editor talks to GitLab directly from the page, with your permission (OAuth); the projects never pass through another server.

9.4 Other formats

The buttons above the preview show the document as a PDF, as HTML or as Markdown. *Download PDF* saves the PDF, and the arrow next to it exports an accessible PDF (with what screen readers would miss among the problems), an OpenDocument text document, HTML, Markdown (with its pictures, in a ZIP file), an OpenDocument presentation or SVG (Section 7).

9.5 What it cannot do

A web page cannot start programs: the shell escape does nothing, and the installed bibtex, biber and makeindex are not there (retex's own are, which is what documents need). The page's TEX tree has the classes and packages that papers, theses and slides use most, a few hundred of them, and their fonts, with Noto Naskh Arabic and Noto Serif Devanagari for the front page of this manual; a package that is not in it can be dropped on the file list like any file.

> **Try it** Change the title on the front page (`retex-manual.tex`), or the colours of the whole manual (the first lines of `retex-manual.sty` define `ink`, `teal`, `sand` and `coral`). Every page changes with them.

10 retex in programs

retex is also a library: a program can compile documents without starting a TEX program, from Python or from Rust. The build is the one `retex doc.tex` runs, every pass included, and the result stays in memory, in all of retex's output formats.

10.1 From Python

```
import retex

tex = retex.Retex()
doc = tex(r"""
\documentclass{article}
\begin{document}
Hello from Python! $e^{i\pi} + 1 = 0$
\end{document}
""")

pdf = doc.to_pdf()        # bytes
svg = doc.to_svg()        # str: every page, stacked
odt = doc.to_odt()        # bytes; also to_odp()
html = doc.to_html()      # str: one file, fonts and pictures in it
md = doc.to_markdown()    # str
doc.save("hello.pdf")     # or .svg, .odt, .odp, .html, .md
```

The package is built with maturin [17], in `crates/retex-python`:

```
$ maturin build --release
$ pip install ../../target/wheels/retex-*.whl
```

Its wheel works with every CPython from 3.9 on. It needs a TEX Live installation, as the program does (Section 2).

`Retex()` takes keyword arguments:

argument	meaning
`passes`	at most this many passes (default 5; 1 runs TEX once, as lualatex)
`shell_escape`	let `\write18` run programs (default `False`)
`accessible`	tag the PDF and check it, as `--accessible` (Section 7.1)
`working_dir`	where the document's own files are (pictures, `.bib` files): relative names are relative to it
`job_name`	the job name, `\jobname` (default `document`)
`files`	a dictionary of names to contents (`str` or `bytes`): files the document reads, put next to it
`keep_files`	a directory to keep the build's files in (the log, the `.aux` file) instead of a temporary one

`tex(source)` (or `tex.compile(source)`) compiles a document given as text, and

```
doc = tex.compile_file("paper.tex")
```

compiles a file, its directory being the working directory. One call does both steps:

```
doc = retex.compile(source, passes=1)    # Retex(passes=1)(source)
```

A `Document` has the outputs above and what the build reported: `pages`, `passes`, `seconds`, `log` (the log of the last pass) and `diagnostics`, retex's errors, warnings, notes and hints (Section 4), with `errors` and `warnings` for two of the kinds. Each diagnostic has `severity`, `message`, `origin`, `file`, `line`, `column`, `source`, `notes` and `help`. A document with errors still has its PDF, as with the program; a build that writes no output raises `retex.RetexError`, whose `diagnostics` and `log` say why:

```
try:
    doc = tex(source)
except retex.RetexError as e:
    for d in e.diagnostics:
        print(d)          # document.tex:3: error: undefined command \oops
```

In Jupyter, a document shows as its pages. `doc.markdown(media)` gives the Markdown text with its pictures, a dictionary of names to bytes, that the text refers to as ⟨*media*⟩/⟨*name*⟩.

Pages are also images: `doc.to_png(page=0, dpi=150)` gives a page as a PNG file, `doc.to_numpy(...)` as a numpy array of bytes, of shape (⟨*height*⟩, ⟨*width*⟩, 3), and `doc.to_pil(...)` as a Pillow image; `page=None` gives a list, one image per page. Pages are numbered from 0. `alpha=True` leaves what the page does not paint transparent (4 channels), and `crop=True` cuts the image down to what the page paints, with a margin of 2 points:

```
doc = retex.compile(r"""\documentclass{standalone}
\begin{document}$\displaystyle\int_0^1 x^2\,dx = \frac13$\end{document}""")
formula = doc.to_numpy(dpi=300, alpha=True, crop=True)
```

The pages are drawn in the process, from their SVG documents (glyphs as outlines) by resvg, a few tens of milliseconds a page at 150 dpi; numpy and Pillow are needed only by the methods that return their objects. What the SVG output leaves out is not drawn either (Section 7).

10.2 Examples in Python

The scripts of this section are in `crates/retex-python/examples`, complete.

A formula in a matplotlib figure. The formula is typeset by LaTeX, with any package, rather than by matplotlib's own mathematics; the image is cut down to the formula, its background transparent:

```
DPI = 300
formula = retex.compile(r"""\documentclass{standalone}
\begin{document}$\displaystyle \varphi(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}$
\end{document}""").to_numpy(dpi=DPI, alpha=True, crop=True)

fig, ax = plt.subplots(figsize=(6, 3.5), dpi=150)
ax.plot(x, np.exp(-x**2 / 2) / np.sqrt(2 * np.pi))
ax.add_artist(AnnotationBbox(OffsetImage(formula, zoom=fig.dpi / DPI),
                             (-2.6, 0.3), frameon=False))
```

One PDF per person. For certificates, letters or badges, a template is filled in and compiled in a loop. A `Retex` keeps its options from one build to the next. Text put in a template this way is TEX source: a name with `&` or `%` needs a backslash before them.

```
tex = retex.Retex(passes=2)  # remember picture needs a second pass
for name in ["Ada Lovelace", "Alan Turing", "Grace Hopper"]:
    source = TEMPLATE.replace("@NAME@", name)
    tex(source).save(f"certificate-{name.split()[-1].lower()}.pdf")
```

A contact sheet. This script puts every page of a document, as a thumbnail, in one image, to show a long document on one screen:

```
doc = retex.Retex().compile_file("thesis.tex")
pages = doc.to_pil(page=None, dpi=24)
w, h = pages[0].size
sheet = Image.new("RGB", (8 * w, -(-len(pages) // 8) * h), "white")
for i, page in enumerate(pages):
    sheet.paste(page, ((i % 8) * w, (i // 8) * h))
sheet.save("contact_sheet.png")
```

A check for continuous integration. This script builds every document of a directory and prints its errors with their file and line, as the program prints them; it fails if one has an error:

```
for path in sorted(pathlib.Path(sys.argv[1]).rglob("*.tex")):
    if r"\documentclass" not in path.read_text(errors="replace"):
        continue  # a chapter, \input by another file
    try:
        problems = tex.compile_file(path).errors
    except retex.RetexError as e:
        problems = e.diagnostics
    for d in problems:
        print(d)
    failed += bool(problems)
```

An animated picture. The curves are computed by numpy and drawn by TikZ, a `tikzpicture` for each frame. With the `tikz` option of the standalone class, each picture is a page, and `page=None` gives every page as an image, a frame of a GIF file (Figure 2, left):

```
a = np.linspace(0, 2 * np.pi, 300)
pictures = []
for degrees in range(0, 360, 5):
    t = np.radians(degrees)
    curves = []
    for k in range(6):   # a ribbon of 6 curves
        x, y = 2 * np.sin(3 * a + t - k * 0.08), 2 * np.sin(2 * a)
        path = " -- ".join(f"({u:.3f},{v:.3f})" for u, v in zip(x, y))
        hue = (degrees + 15 * k) % 360
        curves.append(rf"\definecolor{{c}}{{Hsb}}{{{hue},0.65,1}}"
                      rf"\draw[c, opacity={1 - k / 8:.2f}] {path};")
    pictures.append(r"\begin{tikzpicture}"
                    r"\fill[black!90] (-2.4,-2.4) rectangle (2.4,2.4);"
                    + "\n".join(curves) + r"\end{tikzpicture}")
```

```
doc = retex.compile(r"\documentclass[tikz]{standalone}\begin{document}"
                    + "\n".join(pictures) + r"\end{document}")
frames = doc.to_pil(page=None, dpi=72)
frames[0].save("lissajous.gif", save_all=True,
               append_images=frames[1:], duration=60, loop=0)
```

The 72 frames take about 20 seconds.

From TEX to pixels and back. A glyph is drawn as an array, then drawn again as a TikZ picture of dots, one a cell, each as large as the cell is dark. The result is a halftone that is a vector drawing (Figure 2, right):

```
glyph = retex.compile(r"""\documentclass[border=4pt]{standalone}
\begin{document}\fontsize{60}{60}\selectfont\itshape\&\end{document}""")
ink = 1 - glyph.to_numpy(dpi=40).mean(axis=2) / 255  # 1 is black

dots = []
for r, c in cells:   # the cells of a grid over the image
    v = ink[r*cell:(r+1)*cell, c*cell:(c+1)*cell].mean()
    dots.append(rf"\fill ({c},{-r}) circle ({0.55 * v**0.5:.3f});")
retex.compile(r"""\documentclass[tikz]{standalone}
\begin{document}\begin{tikzpicture}[x=3mm, y=3mm]
""" + "\n".join(dots) + r"""
\end{tikzpicture}\end{document}""").save("halftone.pdf")
```

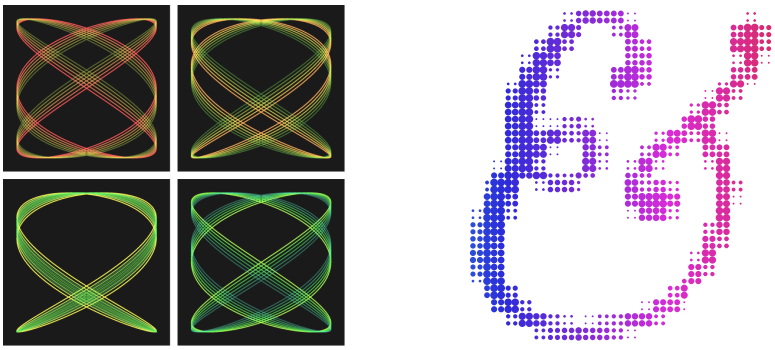


Figure 2: Two pictures from Python: frames of the animated Lissajous curves, and the halftone ampersand, drawn by TikZ.

10.3 From Rust

The `retex` crate is the program and the library [18]:

```
let doc = retex::Retex::new()
    .passes(3)
    .file("logo.png", std::fs::read("logo.png")?)
    .compile(source)?;          // or .compile_file("paper.tex")
std::fs::write("paper.pdf", doc.pdf())?;
std::fs::write("paper.svg", doc.to_svg()?)?;
for e in doc.errors() {
    eprintln!("{e}");
}
```

The options are those of the Python package, as builder methods. A `Document` has the outputs, `pdf`, `to_svg`, `to_odt`, `to_odp`, `to_html` and `to_markdown`; the images of the pages, `render(page, dpi)` (an `Image` of RGBA pixels), `render_with(page, &RenderOptions)`, `render_all` and `to_png(page, dpi)`; and what the build reported, `diagnostics`, `errors`, `log`, `pages`, `passes` and `seconds`. A failed build is `retex::Error::Failed`, with its diagnostics and log.

10.4 How a build runs

- A build runs in a temporary directory. The auxiliary files, the log and the PDF go there, and it is removed after the build (unless `keep_files`); a document given as text is written there as ⟨*job*⟩`.tex`.
- Builds run one at a time. TEX reads and writes files relative to the current directory, so a build sets the process's current directory to the working directory while it runs, and puts it back after. Builds from several threads wait for each other. Python's other threads run during a build (it releases the GIL), but should not rely on the current directory meanwhile.
- A build runs on a thread of its own, with the large stack TEX's recursion needs; an internal error in retex is reported as an error (`RetexError`, `Error::Internal`), not a crash of the program.
- A build does not use snapshots. Later passes run from the start rather than from the snapshot of the preamble the program takes (Section 15), which forks a process; a build of several passes takes longer than with the program.
- The OpenDocument, HTML and Markdown outputs need the marks of the document's structure (Section 7), so the first of these conversions builds the document again with them; SVG converts the PDF itself.

11 Looking inside a document

Two commands answer the questions a log does not: where the time of a build goes, and what a command is.

11.1 Where the time goes

retex profile [⟨*options*⟩] ⟨*file.tex*⟩

Typeset one pass of the document, with the auxiliary files of the last build, and show where the time went.

The report goes to the standard output, after the usual messages; the time is sampled every millisecond. Here it profiles the front page of this manual, alone (`samples/hero/hero.tex` in the repository):

```
$ retex profile hero.tex
profile of hero.tex: one pass in 2.38 s (sampled every 1 ms)

Phases
    14.9 ms   1%  start (format, Lua)
     1.19 s  50%  preamble
     1.17 s  49%  document (1 page)
     8.5 ms   0%  end (finishing the PDF)

Reading package files (with the packages they load)
     0.14 s   6%  fontspec
     0.13 s   6%  tcolorbox
     0.11 s   5%  pgfplots
...
Time by package (where the running code was defined)
     0.66 s  28%  expl3
     0.47 s  20%  tikz
     0.43 s  18%  pgfplots
     0.28 s  12%  LaTeX kernel
     0.21 s   9%  your document
...
Engine work
     1.74 s  73%  TeX (macros and typesetting)
     0.41 s  17%  Lua
     0.22 s   9%  font loading
...
Macros (own time, with the macros they call, calls)
        own      with     calls
     0.14 s    0.14 s         3  \pgfplots@addplotimpl@expression@@
     0.12 s    0.12 s       106  \extract@font
...
```

The sections are: the phases of the pass (and the slowest pages), the time reading each package's files, the time in each package's macros (the package that defined the macro running: the document, a package, the class, the LaTeX kernel), the engine's work by kind, and the macros with the most time of their own. `--stacks=`⟨*file*⟩ also writes the samples as collapsed stacks, for flame graph tools.

11.2 What a command is

> **retex explain** ⟨*name*⟩… [⟨*file.tex*⟩]
>
> Run the preamble of the document and, at `\begin{document}`, say what each command is. Nothing is typeset.

For each name, retex says whether it is a macro (with its parameters and what it expands to), a primitive, or something else; where it was defined, file and line, and where earlier definitions were made; the other names with the same meaning; the command behind a robust command, a command with an optional argument or a document command (`\NewDocumentCommand`, with its argument specification); whether retex runs it natively; and, for an undefined name, the defined names that look like it.

```
$ retex explain '\section' paper.tex
```

Without a document, the names are explained in a minimal article. Names may be given without their backslash. The exit status is 1 if a name is undefined.

PART II

Reference

12 The command line

```
retex [options] file[.tex] [commands]
retex [options] \first-line
retex [options] &format [args]
retex profile [options] file[.tex]
retex explain [options] \name... [file[.tex]]
retex packages list|which|install|update|prune [args]
retex config [--global] key [value]
retex lsp
retex --luaonly file [args]
```

retex typesets the LaTeX (or plain TeX) document ⟨*file*⟩ into `file.pdf`. If the first argument begins with a backslash, all the arguments are a line of input; if it begins with &, the next word names the format to use, as with luatex. When retex runs under its own name and the first argument is `profile`, `explain`, `lsp`, `packages` or `config`, it is a command (Sections 2.5, 8.1 and 11).

Options may start with one dash or two, may be abbreviated to a unique prefix, and take their value after = or as the next argument. Every option of luatex is accepted; those with no meaning for retex (`-kpathsea-debug`, `-mktex`, `-socket`…) are ignored, and unknown options are reported and skipped.

12.1 Building

`--watch`

Stay running and rebuild whenever a file the document uses changes, from the last page before the edit (Section 6). An edit of the preamble restarts the build. Enter starts a full rebuild.

`--no-clear`

With `--watch`, do not clear the screen for a rebuild.

`--notify`

With `--watch`, notify when errors appear or are fixed (a desktop notification in iTerm2, WezTerm, Ghostty, kitty and foot; the bell elsewhere).

`--open`

Open the PDF in the system's viewer after the first build without errors.

`--accessible`

Make a tagged, accessible PDF (PDF/UA-2, PDF 2.0) with LaTeX's own tagging, and check what screen readers need (Section 7.1).

`--passes=N` 5

Run at most ⟨*N*⟩ passes; `--passes=1` gives the classic single run.

`--interactive`

Stop at each error and ask what to do, as TEX does on a terminal (`-interaction=errorstopmode`). By default retex reports every error and goes on.

`-halt-on-error`

Stop at the first error.

`-shell-escape`

Allow `\write18` and the Lua functions that run programs. `-no-shell-escape` disallows them. By default only the programs listed by `shell_escape_commands` in `texmf.cnf` may run.

`-synctex=N`

Write `<job>.synctex.gz` for forward and inverse search (Section 7.3); −1 writes it uncompressed.

`-output-format=FORMAT` pdf

`pdf`, `dvi`, `svg` (one SVG file holding all the pages), `odp` (an OpenDocument presentation, from beamer), `odt` (an OpenDocument text document), `html` (one HTML file) or `md` (Markdown and a directory of pictures). The last four are experimental (Section 7).

`--list-formats`

Print the values of `-output-format`, one a line with what each writes, and exit.

`-draftmode`

Typeset without writing the PDF.

`--offline`

Never download missing TEX Live packages (Section 2.5); also `RETEX_OFFLINE`. Without it, a build on a terminal asks before its first download, unless the setting `packages.download` says `always` or `never`.

`--tl=YEAR`

The TEX Live year to use; also `RETEX_TL`, for programs that run retex without passing it options, and the setting `texlive.year` (Section 13.2), for every run. The option wins over the variable, and the variable over the setting. Without any of them, retex uses the year of the installed TEX Live: the one in `release-texlive.txt` at the root of the tree it finds (`TEXMFROOT`), or, in a tree without that file, the year of its LATEX kernel's release, with a warning. Without a TEX Live, the year is retex's default, 2026. A TEX Live older than 2025 is used with a warning. A year other than the installed TEX Live's, or any year without one, is taken from retex's own tree of that year (Section 2.4); the installed TEX Live is then not used, which a warning says. `--version` and `--verbose` name the year, the tree and why they were chosen, and so do the log (in the default mode) and the JSON `finished` record:

```
TeX Live 2025: /usr/share/texlive (from /usr/share/texlive/release-texlive.txt)
TeX Live 2026: ~/.cache/retex/texlive/2026 (from --tl: retex's own tree)
```

12.2 Files

`-output-directory=DIR`

Write all output files to ⟨*DIR*⟩, the auxiliary files too.

`-jobname=NAME`

Name the output files ⟨*NAME*⟩ instead of after the input file.

`--no-aux-dir`

Write the auxiliary files next to the document instead of under `.retex/<job>-<hash>/` (Section 3.2).

`-recorder`

List the files read and written in `<job>.fls`.

12.3 Messages

`-q, --quiet`

Show only the errors, and the summary line only when the build failed; no progress line.

`--verbose`

Show every warning in full (by default, more than three layout warnings are shown on one line each, and long lists are cut short), the passes run and why, the tools run, and at the start of a build the TEX Live used (`--tl`).

`--message-format=FORMAT` human

`human`: each message with the source line, a marker under the problem, notes and help. `short`: one line per message, `FILE:LINE:COLUMN: error: MESSAGE (HELP)`; `-file-line-error` selects it too. `json`: JSON Lines on the standard output, one record per event (Section 4); the human messages are not written.

`--symbols`

With `--message-format=json`, write a `symbols` record before `finished`: the document's index for editors, from the last pass. It has the commands defined (with their arguments, meaning and place), the environments, the labels and citations, the entries of the bibliography files, and the places of labels, references, citations, sections and inputs in the document's files.

`--color=WHEN` auto

Colour the messages: `auto` when they go to a terminal (see `NO_COLOR` and `CLICOLOR_FORCE`), `always`, even into a pipe or a file (`retex --color=always doc.tex 2>&1 | less -R`), or `never`.

`--hyperlinks=WHEN` auto

Make the places in the messages, the PDF and the log clickable (OSC 8 hyperlinks; the text does not change): `auto` on terminals known to support them (iTerm2, WezTerm, kitty, foot, Ghostty, Alacritty, Konsole, VTE terminals such as GNOME Terminal, Windows Terminal, VS Code), not inside screen or tmux; `always`; `never`.

`--hyperlink-format=FORMAT` file

Where a place opens: `file` (the file, with the system's handler), `vscode`, `vscodium`, `cursor`, `textmate`, `macvim` (the editor, at the line), `kitty` (kitty's open actions, at the line), or a template with `{path}`, `{line}`, `{column}` and `{host}`. Also `RETEX_HYPERLINK_FORMAT`.

`--stacks=FILE`

With `retex profile`: also write the samples to ⟨*FILE*⟩ as collapsed stacks, for flame graph tools.

12.4 Compatibility with LuaTEX

`--compat=lualatex`

Behave as lualatex: LuaTEX's exact terminal output, the auxiliary files next to the document, TEX's dialog at errors; the passes still run until the auxiliary files settle, unless `--passes=1` (Section 14.2). Also when retex runs under the name lualatex or luatex, then with one pass.

`-interaction=MODE` nonstopmode

`batchmode`, `nonstopmode`, `scrollmode` or `errorstopmode` (the default with `--interactive` or in the compatibility mode).

-ini, -fmt=NAME

Build a format (IniTEX), or use the format ⟨*NAME*⟩ instead of lualatex's.

-lua=FILE

Run the Lua initialization script ⟨*FILE*⟩ before the job.

-safer

Disable the Lua functions that run programs or open files.

--luaonly FILE ARGS

Run the Lua script ⟨*FILE*⟩ and exit, as texlua does.

-v, --version

Print the version, with the TEX Live release and the LATEX version retex finds, and on a second line the TEX Live year used, its tree and why (`--tl`), and exit.

-h, --help

Print a summary of the options and exit.

12.5 Exit status

0	the document was typeset without errors
1	there were errors, those of BibTeX, biber and makeindex included, and with `--accessible` a PDF without a title or a language
2	the command line was wrong
101	an internal error: a bug in retex, reported as such
130	interrupted by Ctrl + C (which also ends `--watch`); the terminal is left clean

Warnings and `\show` never change the exit status, in the default mode.

13 Environment variables and files

13.1 Variables

retex reads the variables of kpathsea (`TEXINPUTS`, `TEXMFHOME`, `TEXMFCNF`, `TEXFONTS`…) as LuaTEX does, and these:

`RETEX_CACHE`

The directory of retex's formats and caches: the font names, the lists of the tree's files, the images and figure pages compressed for the PDF, the figure cache. Default: `$XDG_CACHE_HOME/retex/<version>`, or `~/.cache/retex/<version>`; on macOS `~/Library/Caches/retex/<version>`, on Windows `%LOCALAPPDATA%\retex\<version>`.

`RETEX_TL`

The TEX Live year, as `--tl` gives it (which wins), for programs that run retex without passing it options: editors, latexmk, the language server, CI scripts. It wins over the setting `texlive.year`.

`RETEX_OFFLINE`

Never download TEX Live packages, as `--offline` (Section 2.5): any value but empty, `0`, `false`, `no` and `off`, so `RETEX_OFFLINE=1` and `RETEX_OFFLINE=true` both do.

`RETEX_REPOSITORY`

The TEX Live repository packages come from: an `https://` or `file://` URL, or a directory with `tlpkg/` and `archive/`. It wins over the setting `packages.repository`.

`RETEX_CONFIG`

The settings file to read instead of `~/.config/retex/retex.toml` (Section 13.2).

`RETEX_TEXLIVE_KEYRING`

A binary OpenPGP keyring with the keys that sign the repository's package database, instead of TEX Live's key, which is built into retex: for a private repository signed with a key of its own.

`RETEX_BIB`

`external` runs the installed bibtex and biber instead of retex's own (Section 5.1).

`RETEX_MAKEINDEX`

`external` runs the installed makeindex instead of retex's own (Section 5.2).

`RETEX_MINTED`

`external` runs the installed latexminted or pygmentize (Python and Pygments) instead of retex's own highlighting (Section 3.5).

RETEX_HINTS

`all` shows every kind of hint about the source, `0` none (Section 4.3).

RETEX_FIGURES on

`off`, `stats` or `verify`: the figure cache (Section 6.2).

RETEX_HYPERLINK_FORMAT

The default of `--hyperlink-format`.

SOURCE_DATE_EPOCH

The date of the document and of the PDF file, for reproducible output.

NO_COLOR

No colours in the messages (if not empty). Colours are also left out when the messages do not go to a terminal, or when `TERM` is `dumb`, which also turns off the progress line.

CLICOLOR_FORCE

Colours even when the messages do not go to a terminal (if set to anything but `0` or nothing).

LC_ALL, LC_CTYPE, LANG

Unless the locale's character set is UTF-8, the messages use ASCII for their symbols (`|` for |, `...` for …).

TEXEDIT

The editor to run when `E` is answered at an error prompt (`--interactive`).

For debugging retex itself: `RETEX_DEBUG_PASSES` reports what the pass driver does, `RETEX_TIMING` reports timings, `RETEX_NO_SNAPSHOT` turns off the preamble snapshot, and `RETEX_CHECKPOINT_MS` sets the least time, in milliseconds, between the page checkpoints of `--watch` (150 by default).

13.2 The settings file

The user's settings are in `retex.toml`, in the platform's configuration directory: `~/.config/retex/retex.toml` on Linux (`$XDG_CONFIG_HOME/retex/retex.toml` when that variable is set), `~/Library/Application Support/retex/retex.toml` on macOS, `%APPDATA%\retex\retex.toml` on Windows. `RETEX_CONFIG` names another file. There is no file until a setting is made, and every setting has a default. The file is TOML; retex reads its tables and its `key = "value"` lines, and ignores comments (`#`):

```
[texlive]
year = "2025"           # the TeX Live year, as --tl=2025

[packages]
download = "never"      # or "always", or "ask" (the default)
```

```
repository = "https://mirror.example.org/texlive/tlnet"
```

Setting	Value
`texlive.year`	the TEX Live year of every run (`--tl`); `--tl` and `RETEX_TL` win over it. Default: the installed TEX Live's year
`packages.download`	`ask` (the default) asks on a terminal whether to download missing TEX Live packages, and downloads them without a terminal; `always` downloads them, `never` does not (Section 2.5); `--offline` and `RETEX_OFFLINE` win over it
`packages.repository`	the TEX Live repository packages come from: a URL or a directory; `RETEX_REPOSITORY` wins over it. Default: TEX Live's own

`retex config` reads and changes the file, keeping its comments and the lines it does not know, and checks the values it is given:

```
$ retex config texlive.year 2025       # set
$ retex config texlive.year            # show
2025
$ retex config --unset texlive.year    # back to the default
$ retex config --list                  # every setting in the file
$ retex config --help                  # the settings, and the file's path
```

`--global` is accepted and changes nothing: the settings are always the user's. A project has no settings file of its own; the options and variables are there for that.

13.3 Files

`<job>.pdf`	the output, in the current directory or the output directory
`.retex/<job>-<hash>/`	the log file and the auxiliary files of the document, by default (Section 3.2)
`<cache>/lualatex.fmt`	the LaTeX format, built at the first run, and again when TeX Live or its LaTeX kernel changes
`<cache>/texlive-<year>/lualatex.fmt`	the LaTeX format of retex's own tree of a year (Section 2.4)
`<cache>/figures/`	the figure cache (Section 6.2)
`<cache>/../texlive/<year>/`	the TeX Live packages downloaded for a year, and its package database (Section 2.5); inside `RETEX_CACHE` when it is set. Without an installed TeX Live, the whole TeX tree of the year, with its `texmf.cnf` and the generated files in `texmf-var/` (Section 2.4)
`<cache>/../texlive/network-down`	the time the network last did not answer a download; none is tried in the next 10 minutes (Section 2.5)
`~/.config/retex/retex.toml`	the settings (Section 13.2)

14 retex and lualatex

14.1 Point by point

	lualatex	retex
PDF output	the reference	pixel-identical pages; font subsets get other tags
`.log` file	LuaTEX's	the same
reruns for references, contents, bibliographies, indexes	by hand, or with latexmk	automatic, with BibTeX, biber or makeindex when needed, in process
later passes	run the whole document again	resume after the preamble
terminal output	TEX's messages	errors and warnings with the source line, the cause and a fix; hints about common mistakes
auxiliary files	next to the document	in `.retex/<job>-<hash>/`; only the PDF next to the document
rebuild on change	external tools	`--watch`, from the page edited
SVG images	converted by Inkscape, with `-shell-escape` (svg package)	`\includegraphics` directly, the text searchable
other images	PNG, JPEG, JBIG2, PDF; JPEG 2000 through `\saveimageresource` only	the same, and `\includegraphics` of JPEG 2000, GIF (first frame), WebP and TIFF
code listings (minted)	Pygments, through Python and the shell escape	coloured in process, the same in 18 languages, minted 2 included, without `-shell-escape`
SVG output	none (dvisvgm, from DVI)	one multi-page SVG file
presentations	none	OpenDocument, from beamer (experimental)
word processor documents	none	OpenDocument text, HTML, Markdown (experimental)
accessible PDF	`\DocumentMetadata`	the same, or `--accessible`, with checks of what screen readers miss
SyncTeX	`-synctex=1`	the same, with the same records
command-line options	luatex's	the same set, and retex's own
editors	external language servers	`retex lsp`
in a browser	no	yes

14.2 The compatibility mode

Some tools drive TEX themselves: they run the passes (latexmk, arara), or read the terminal output and the log (l3build, the test suites of packages). For them, retex behaves as lualatex with `--compat=lualatex`: the auxiliary files next to the document, TEX's dialog at errors, and LuaTEX's exact terminal output. It still runs the passes the document needs; `--passes=1` makes it one, as lualatex. Each pass prints its output as it runs, as lualatex run several times does; `--verbose` also names the passes and the tools run between them. As in LuaTEX, the lines are cut after 79 characters (`max_print_line` of `texmf.cnf`); `max_print_line=1000` in the environment makes them longer. Run under the name lualatex or luatex (through a

link), retex is in this mode and runs one pass:

```
$ ln -s "$(command -v retex)" ~/bin/lualatex
$ latexmk -lualatex paper.tex
```

	`retex`	`retex --compat=lualatex`
passes	until the auxiliary files settle (at most 5, `--passes`)	the same; one under the name lualatex
later passes	resume after the preamble	run the whole document
BibTeX, biber, makeindex	run between passes, in process	the same; with one pass, the document or the tool driving the passes runs them
auxiliary files	in `.retex/<job>-<hash>/`	next to the document
terminal output	errors and warnings with the source line, the cause and a fix	LuaTeX's
at an error	reported, and the run goes on (`nonstopmode`)	TeX's dialog (`errorstopmode`)
exit status 1	TeX's errors and those of BibTeX, biber and makeindex	TeX's errors
minted	coloured in process, minted 2 included	the installed latexminted or pygmentize (Python)
TikZ figure cache	on (Section 6.2)	off: the log stays LuaTeX's
accessibility checks	for tagged documents (Section 7.1)	none
`--watch`, `retex explain`, `retex profile`	yes	no

With `--accessible`, a PDF without a title or a language also makes the exit status 1 (the PDF is still written).

14.3 Conformance

The LaTeX kernel, expl3, pgf and babel come with test suites, run by l3build, that compare a run's log with the expected one. retex passes nearly every test that LuaTeX itself passes (TeX Live 2025, LuaTeX 1.22):

suite	passes with retex	passes with LuaTeX
LaTeX kernel (latex2e)	1105 of 1114	1108
expl3 (latex3)	255 of 257	255
pgf/TikZ	110 of 122	111
babel	16 of 147	16

The LaTeX kernel tests left differ in one character of the error context LuaTeX prints for invalid UTF-8 input. Most babel tests were written for pdfTeX, so LuaTeX itself passes few of them.

Beyond the test suites, retex is compared with lualatex on a corpus of real documents (Section 17.1): 524 arXiv papers and 93 CTAN package manuals. On 574 of the 617 that

both compile, every page is identical to lualatex's at 150 dpi; the others differ on a few pages, mostly by a fraction of a point (one pixel of antialiasing, a glyph shifted in a figure). retex writes a PDF for every one of them, with the same number of pages.

14.4 TEX Live 2026

With TEX Live 2026's trees and formats, retex typesets as TEX Live 2026's lualatex (LuaHBTEX 1.24): the public samples are as close to it as they are to TEX Live 2025's, real documents are pixel-identical to fresh TEX Live 2026 builds, and the four test suites pass every test that LuaHBTEX 1.24 passes but three accepted for version 1.0 (two latex2e tests of a Windows code page, one pgf test of the manual's images). retex chooses LuaTEX 1.24's behaviour (the italic correction of characters in math, mostly) only for formats of LATEX 2026-06-01 or later, so TEX Live 2025 behaves exactly as before. Section 17.4 has the details.

14.5 Japanese with LuaTEX-ja

retex runs luatexja (LuaTEX-ja, in TEX Live 2025 and 2026, with the Harano Aji fonts) as lualatex does: its JFM glue, kanjiskip and xkanjiskip, line breaking rules, ruby (luatexja-ruby, also under lua-ul's underlines), luatexja-fontspec, vertical writing (`\tate` boxes, with `luatexja_cmapidv` too: `Identity-V` fonts in RTT boxes), ideographic variation sequences and proportional metrics (`palt`). luatexja relies on luaotfload's internals; retex's native luaotfload gives it the font tables it reads, with their features, coverage tables and unrounded metrics. The manuals of LuaTEX-ja (`luatexja-ja`, `luatexja-en`, `luatexja-ruby`, `lltjp-geometry` and the classes' `.dtx` files; the first two with Harano Aji and Latin Modern Mono in place of fonts TEX Live does not have) are pixel-identical to lualatex's, except where a page prints the contents of a Lua table in the order of `pairs`, which LuaTEX itself changes from run to run.

PART III

Inside retex

15 Performance

15.1 Papers and proposals

Measured on one machine (8 cores, TeX Live 2025, warm font caches). One pass: the median of five runs, with the auxiliary files present. Full build: from a clean directory, lualatex with BibTeX or biber and two more lualatex runs, against a single `retex doc.tex`.

document	lualatex	retex
one-page article ("Hello world"), one pass	0.64 s	0.15 s
3-page ACM article (acmart, BibTeX), one pass	6.3 s	1.9 s
the same, full build	18.0 s	2.8 s
9-page ACM article (acmart, biblatex), one pass	9.5 s	4.1 s
the same, full build	28.8 s	8.0 s
30-page project proposal (biblatex, TikZ, many packages), one pass	8.1 s	4.0 s
the same, full build	25.7 s	8.0 s

Measured on 2026-10-31 with retex 1.7.0-beta.1 (full builds: the median of three runs). retex runs BibTeX and biber in process: the biblatex documents gain most, as biber takes 1.7 to 3.9 s of each lualatex build. The bibliographies and the pages are identical to lualatex's.

15.2 Large manuals and presentations

Package manuals and presentations from CTAN, on the same machine, one document at a time. One pass: lualatex's third run against a second `retex --compat=lualatex` run, both with the auxiliary files of a finished build. Full build: three lualatex runs against a single `retex doc.tex` from a clean directory, which reruns until the auxiliary files settle. All of them give pages pixel-identical to lualatex's.

		one pass		full build	
document	pages	lualatex	retex	lualatex ×3	retex
pgf/TikZ manual	1323	423 s	134 s	1103 s	511 s
chemfig manual	93	74 s	56 s	228 s	172 s
tabularray manual	222	76 s	23 s	226 s	64 s
forest (`forest.dtx`)	170	5.3 s	2.6 s	15.8 s	7.8 s
tikz-cd manual	19	4.3 s	2.7 s	13.0 s	6.8 s
metropolis beamer demo	33	3.5 s	1.9 s	10.8 s	3.8 s

(Measured on 2026-10-31 with retex 1.7.0-beta.1, on an idle machine.)

The corpus of Section 17.1 gives newer figures for the largest of them, and for ProfCollege's manual (619 pages of MetaPost and TikZ, through luamplib): full builds measured on 2026-10-30 with retex 1.6.0-beta.7, three documents at a time on the same machine, so slower than the table above for both programs.

document	pages	lualatex	retex	speed-up
ProfCollege manual	619	1197 s	775 s	1.54×
pgf/TikZ manual	1323	834 s	396 s	2.11×
tabularray manual	222	268 s	69 s	3.87×
chemfig manual	93	241 s	183 s	1.32×

On the whole corpus, a full build is a median 2.6 times faster than lualatex's, and the 617 documents together build 2.0 times faster (3757 s against 7506 s). More than half of a ProfCollege pass goes to Lua (mostly luamplib's work around the MetaPost interpreter, which takes little itself), pgf and xint; native code for xint and for luamplib's glue are the next steps.

15.3 Where the speed comes from

- Later passes resume after the preamble, from a snapshot of the engine taken at `\begin{document}`, and `--watch` from a checkpoint at the last page before the edit.
- Some of the macros where documents spend their time (parts of expl3, among them its regular expressions and integer arrays; pgfkeys and pgfmath) have implementations in Rust that retex uses when they are defined as the packages define them, and checks against the macros' own results.
- The tools run in process. No BibTeX, biber or makeindex program is started, which saves in particular biber's Perl start-up.
- Images and figures are cached. retex reads only the headers of images, copies their data without decoding it, compresses figure pages once, and replays TikZ pictures (Section 6.2).
- Fonts load once. Their names are indexed in the cache, and the PDF's fonts and streams are compressed on other threads.

`retex profile` (Section 11) shows where a document spends its own time.

16 Clean-room provenance

retex is written from scratch and released under CC0. No code of another TEX implementation (LuaTEX, pdfTEX, XeTEX, luaotfload, HarfBuzz…) has been read, whatever its license. Its behaviour is established from the output of lualatex on test documents of our own, from the values that LuaTEX's documented Lua interface returns, and from published documentation and specifications. Two kinds of programs are exceptions, used as references: Knuth's, whose terms allow it, and MetaPost, which is in the public domain (both below). The complete policy is in `CONTRIBUTING.md`, in the repository.

16.1 Knuth's programs

Three of Knuth's published programs, one with a change file, are used as references for the algorithms retex shares with them:

file	what it is	used for
`tex.web`	*TEX: The Program* [6]	TEX82's semantics
`etex.ch`	the e-TEX change file to `tex.web`, by P. Breitenlohner and the NTS team	the e-TEX extensions
`mf.web`	*METAFONT: The Program* [7]	the random number generator LuaTEX shares with METAFONT, and the algorithms MetaPost inherits from METAFONT

Reading them is permitted, for three reasons:

- They are published works whose terms allow unrestricted copying. `tex.web` and `mf.web` begin with: "Unlimited copying and redistribution of this file are permitted as long as this file is not modified. Modifications are permitted, but only if the resulting file is not named tex.web" (resp. `mf.web`). `etex.ch` allows copying as long as the copy is not changed. The two programs are also printed as volumes B and D of *Computers & Typesetting*. They are not under the GPL, MIT or Apache licenses: their terms govern copies of the files themselves.
- retex neither copies nor modifies them. They are read as documentation, as a textbook is; retex's code is an independent implementation in Rust, and none of the files is part of the repository.
- They describe the reference behaviour. For TEX82, e-TEX and METAFONT's random number generator, these programs *are* the specification: LuaTEX documents its behaviour as that of TEX and e-TEX, and its random numbers follow METAFONT's generator.

The files, with the license at the top of each, are on CTAN:

- `https://ctan.org/tex-archive/systems/knuth/dist/tex/tex.web`
- `https://ctan.org/tex-archive/systems/knuth/dist/mf/mf.web`

and `etex.ch` is in TEX Live's source tree, as `texk/web2c/etexdir/etex.ch`.

16.2 MetaPost

retex's MetaPost (the `retex-mp` crate, behind LuaTEX's `mplib` library that luamplib uses) is a port to Rust of MetaPost 2.11, whose source is in the public domain:

file	what it is
`mp.w`	the MetaPost interpreter, by John Hobby and Taco Hoekwater
`mpmath.w`, `mpmathdouble.w`, `mpmathdecimal.w`	the scaled, double and decimal number systems
`psout.w`, `tfmin.w`	the exported figures and the TEX font metrics it reads

These CWEB files state that they are in the public domain, so retex may port them; it does so as it does everything else, under CC0. MetaPost is built on METAFONT, and `mf.web` (above) explains much of what `mp.w` takes for granted.

- The libraries MetaPost links, which are not in the public domain (MPFR, GMP, IBM's decNumber), and LuaTEX's Lua binding of the library (`lmplib.c`) were not read. The binary number system (MPFR) is not implemented.
- Instead, the decimal arithmetic is written from the *General Decimal Arithmetic* specification by Mike Cowlishaw, the one decNumber implements, and decNumber's user's guide; its numbers have at most 38 digits (MetaPost allows 1000). The `mplib` Lua interface follows the LuaTEX manual and what lualatex returns on test documents.
- The port is checked against `mpost` (the transcripts of `mpost -ini` runs, in each number system) and lualatex with luamplib; behaviour found this way is marked "Observed" in the code.

16.3 Bibliographies and indexes

For BibTeX, biber and makeindex the same rule holds.

Not read the source code of BibTeX (`bibtex.web` and its change files), bibtex8 and bibtexu, biber (its Perl modules, its table of LATEX to Unicode: retex's is built from the Unicode data and the LATEX documentation), makeindex (any copy), xindy, mendex, upmendex, and any other implementation (citeproc, hayagriva, pybtex, tectonic's port of BibTeX...).

Allowed their documentation (*BibTeXing* [15], *Designing BibTeX Styles* [16], the biblatex [11] and biber [5] manuals, makeindex's manual page and papers [1, 9]), biber's `bcf.rng` schema of the control file (part of its specification), the `.bst`, `.ist` and biblatex files (they are inputs, as packages are for the engine), the Unicode specifications and data (UCA, DUCET, CLDR), and running the installed programs on test inputs to compare their outputs.

Much of what these programs do is not in their documentation and was found by running them; such rules are marked "Observed" in the code. makeindex's sort algorithm, for one, was identified by matching the numbers of comparisons it reports against candidate quicksorts (a median of three from six entries, partitions under four left to an insertion sort): a textbook design, written from that description.

17 How retex is tested

retex's aim is the same output as lualatex's. Its tests compare the two at every scale: single primitives, the test suites of packages, real documents, and the tools that run between passes. Everything here runs from the repository with `cargo xtask`; `CONTRIBUTING.md` has the rules for contributions (each change passes `cargo xtask ci`).

Unit and differential tests (`cargo xtask ci`): the crates' own tests; TEX input run by retex and checked against outputs observed from LuaTEX; the native macros checked against the macros they replace; error messages against expected terminal output.

Samples (`cargo xtask oracle`): the documents of `tests/gate/` (one feature each: text, mathematics, fonts and graphics, TikZ, languages), typeset by lualatex and retex, their pages rendered at 300 dpi and compared pixel by pixel, their logs normalized and compared. The showcase documents of `samples/` (a paper, slides, a multilingual document, a magazine) are retex's own; `cargo xtask cross` checks that other platforms give the same pages.

Test suites (`cargo xtask latex2e`, `l3`, `pgf`, `babel`): the l3build suites of the LATEX kernel, expl3, pgf and babel, with retex as the engine (Section 14.3).

The corpus (`cargo xtask corpus`): real documents (Section 17.1).

Bibliographies and indexes (`cargo xtask bib`, `cargo xtask index`): retex's BibTeX, biber and makeindex against the installed programs (Sections 17.2 and 17.3).

Fuzzing : the tokenizer, the font parser and the PDF writer.

17.1 The corpus

The l3build suites test packages one feature at a time; the corpus tests what people compile: papers with their own macros, publishers' classes, shipped bibliographies and many figures, and the documentation of popular packages. It holds 524 arXiv papers (from 2005 to 2026, 99 subject categories, the journal classes of revtex, amsart, IEEEtran, llncs, acmart, elsarticle…) and 89 CTAN package manuals. Five more manuals, tagged `luatexja` (LuaTEX-ja's, from TEX Live 2025's archive), need luatexja and the Harano Aji fonts (Debian's `texlive-lang-japanese`, or TEX Live 2026).

```
$ cargo build --release -p retex
$ cargo xtask corpus fetch                  # download what is not cached yet
$ cargo xtask corpus run -j 4               # everything; a table and a summary
$ cargo xtask corpus run --tag beamer       # the entries with a tag
$ cargo xtask corpus run '2104.*' siunitx   # entries by name (wildcards)
$ cargo xtask corpus run --engine lualatex  # self-test: all must be identical
$ cargo xtask corpus run --engine "target/release/retex profile"
$ cargo xtask corpus run '24*' --prelude '\DocumentMetadata{tagging=on}'
$ cargo xtask corpus list --tag tikz-heavy
```

The manifest

`corpus/manifest.toml` lists the entries, in a subset of TOML (one-line strings, integers and arrays of strings):

```
[[entry]]
name = "1706.03762v7"
title = "Attention Is All You Need (cs.CL, 2017)"
```

```
source = "arxiv:1706.03762v7"
sha256 = "2e7a7d9ee2520d22eb23dae0cb148e167be02a30d6cb3f11b1c67723194339c6"
main = "ms.tex"
tags = ["arxiv", "cs", "article", "natbib", "microtype", "hyperref", "xcolor"]
```

source
: `arxiv:`⟨*id*⟩ with a version, downloaded from `https://arxiv.org/e-print/`⟨*id*⟩; `texlive:`⟨*pkg*⟩`.doc` or `.source`, an archive of the frozen TEX Live 2025 repository (`tlnet-final` on the TEX Live historic archive; `RETEX_CORPUS_TLNET` selects another mirror), which pins the version of CTAN documentation, as CTAN keeps only the current one (a zip file in its documentation, as ProfCollege ships the sources of its manual, is unpacked where it is); or `url:`⟨*url*⟩. Entries may share a source.

sha256
: of the downloaded archive: `fetch` refuses an archive whose hash differs.

main
: the main file, relative to the archive's root. The engines run in its directory, on a copy of the whole tree.

tags
: the source (`arxiv`, `ctan`), the arXiv field, the document class, notable packages, and `tikz-heavy` (15 or more pictures), `expl3`, `figures` (10 or more `\includegraphics`), `bbl` (a shipped `.bbl` file), `dtx` and `large` (50 pages or more).

exclude
: why the entry is not run (lualatex fails on it, for example); `--all` includes such entries.

timeout
: the limit of each engine run, in seconds (600 by default).

prepare
: jobs each engine runs before the document, each `JOB=CODE`: the beamer user guide's pictures of the themes, for example. They are not timed, and only the document's PDF is compared. The ProfCollege manual's job writes `ProfCollegeT.sty`, the name its author's copy of the package has, and leaves out the random exercises of `\CourseNombre` and the calculator screens, which crash TEX Live 2025's lualatex; it also fixes the random seeds (TEX's, Lua's and so MetaPost's), and a second job builds the game cards the manual includes.

also
: more archives, each `SOURCE@SHA256`, whose files are put next to the main file: the marginalia manual, in the `.doc` archive, reads its `.dtx`, which is in the `.source` one.

args
: options both engines get: `-shell-escape` for the ProfCollege manual, which lists directories through the shell. Only for documents from TEX Live's archive, which are trusted.

The documents are not in the repository: their licenses vary.

The cache

Archives are unpacked into `RETEX_CORPUS_DIR`, by default `~/.cache/retex-corpus` (`~/Library/Caches/retex-corpus` on macOS): outside the checkout, so that `cargo clean` and worktrees do not download everything again (each arXiv download costs the arXiv

servers), and not tied to a version of retex. With every source fetched and the references made, it takes about 700 MB. It holds `src/<source>/`, the unpacked trees (symbolic links removed, the archive's hash recorded), and `reference/<entry>/`, lualatex's PDF, log and terminal output with `reference.json` (exit status, runs, time, pages, first error), reused while the archive, the main file and `lualatex --version` are the same (`--fresh-reference` recompiles it). Downloads use curl; requests to arXiv are at least three seconds apart, also across processes, as arXiv's terms ask.

A run

For each selected entry:

1. **The reference.** `lualatex -interaction=nonstopmode` reruns until no file but the log and the PDF changes (at most five runs), with biber, BibTeX and makeindex between runs when their inputs changed. A `.bbl` file shipped with the source, as arXiv requires, is regenerated only when every database it came from is there. The result is cached.
2. **The candidate** compiles a fresh copy of the tree once, with `--compat=lualatex` so that its log is LuaTeX's (`--no-compat` leaves it out, for a command that does not accept it). A candidate whose program name starts with `lua` is driven like the reference.
3. **The comparison**: exit status, normalized log, page count, and the pages as images (pdftoppm, `--dpi`, 150 by default).

Each entry gets a status: `identical` (the same exit status and pages), `exit` (the same pages, another exit status), `differs`, `pages` (another page count), `no-pdf`, `crash` (a signal, or an exit code of 100 or more, as a panic's 101), `timeout`, or `ref-fail` if lualatex wrote no PDF. The table shows both engines' exit status and time, the pages, the speed-up and whether the logs match; the summary counts the statuses and lists the first error of each failure.

The file `target/corpus/results.json` (or `--results`) holds every entry's results, and the directory `target/corpus/out/<entry>/` keeps, for the entries that are not identical, the candidate's log and output, the log difference and up to three images of the differences (red: ink only in lualatex's pages, blue: only in the candidate's). `bench/corpus-baseline.json` (written by `--save-baseline`) records each entry's status and number of differing pages; every run is compared with it (or with `--compare`), and an entry that gets worse fails the command. Other options: `-j` ⟨N⟩ runs ⟨N⟩ entries at a time (times are noisier: use `-j 1` for speed-ups), `--timeout`, `--keep`.

Both engines compile in a copy under the temporary directory (`TMPDIR`). epstopdf converts EPS figures with Ghostscript, and Ubuntu's AppArmor profile for gs lets it write only in the home directory, `/tmp`, `/mnt` and `/media`: with `TMPDIR` elsewhere the figures are left out and the page counts differ. `run` warns when gs cannot write there.

Adding entries

```
$ cargo xtask corpus search \
    'cat:math.AG AND submittedDate:[201801010000 TO 201812312359]'
$ cargo xtask corpus add 1706.03762 2104.01234v2 -j 4
$ cargo xtask corpus add texlive:siunitx.doc --main doc/latex/siunitx/siunitx.tex
$ cargo xtask corpus add texlive:biblatex.doc --name biblatex-ex-01 \
    --main doc/latex/biblatex/examples/01-introduction.tex
```

`search` queries the arXiv API (`cat:`, `ti:`, `co:`, `submittedDate:[… TO …]`). `add` takes arXiv ids (the latest version when none is given), `texlive:` or `url:` sources. For each it downloads the archive, computes its hash, finds the main file (arXiv's `00README.json` or `00README.XXX`, else a `.tex` or `.dtx` file with `\documentclass` and `\begin{document}`: the shallowest, then the usual names, then the largest), derives the tags, takes the title and field from the arXiv API, and compiles it with lualatex (`--no-check` skips that). Entries that lualatex compiles are appended to the manifest, and so are those whose first error is an undefined pdfTEX primitive (`\pdfoutput`, which heads many arXiv papers), tagged `pdftex-primitives`; the others are appended with `exclude` set to the reason. `cargo xtask corpus recheck` compiles the excluded entries again (after installing missing TEX Live packages, say) and brings back those that now pass. Most arXiv papers are written for pdflatex; those that lualatex cannot compile are excluded rather than patched, as a patched source would no longer be a real-world document.

17.2 BibTeX and biber

```
$ cargo xtask bib sets                          # the sets of cases
$ cargo xtask bib list --set bibtex-unit        # the cases of a set
$ cargo xtask bib prepare -j 3                  # materialize the cases
$ cargo xtask bib ref -j 3                      # outputs of bibtex and biber
$ cargo xtask bib run -j 3                      # the candidate, compared
$ cargo xtask bib run --self -j 3               # the programs against themselves
$ cargo xtask bib run 'bibtex-unit/names-*'     # cases by id (wildcards)
$ cargo xtask bib run --tool biber --biber "path/to/biber --some-option"
```

Each case is a run of one program on one job. `prepare` writes into a directory of the cache every input the run reads: the `.aux` file, the style and the databases for BibTeX; for biber, the document is first compiled with `lualatex -draftmode`, and the case keeps the `.bcf` file and the databases it names. Every run happens in a hard-linked copy of the case's directory, with a fixed environment (`SOURCE_DATE_EPOCH`, the `C.UTF-8` locale, `TZ=UTC`, and for biber an empty home directory), and is killed after the time limit or when it writes more than 64 MB. `ref` runs the installed programs; their outputs are cached, keyed by the program's version and the inputs. `run` also runs the candidate, by default the `bibtool` example of the `retex-bib` crate (`bibtool bibtex JOB`, `bibtool biber JOB`).

A case **passes** when the `.bbl` files are identical byte for byte, the exit statuses equal, and the logs equal once normalized: for BibTeX, the whole `.blg` file but its banner, capacities and statistics; for biber, the `WARN` and `ERROR` messages without their time stamps, with file paths reduced to their names, and sorted, as biber writes its warnings in an order that varies from run to run. The gate fails when a case that passes in the baseline (`bench/bib-baseline.json`) fails. The cache is `RETEX_BIB_DIR`, else `bib/` in the corpus cache.

set	cases	pass	covers
`bibtex-unit`	101	101	names, string functions, the database parser, the stack machine, cross-references, sorting, 8-bit input, line wrapping, errors and warnings
`bibtex-styles`	1083	1083	every installed `.bst` (361) on three databases, citing everything
`bibtex-corpus`	65	65	the databases and styles of 33 corpus documents
`biber-unit`	33	32	dates, names, uniqueness, relations, LaTeX decoding, encodings, sorting and locales, label templates, source maps, refsections, errors
`biber-examples`	50	50	biblatex's example documents
`biber-sweep`	66	66	biblatex's options on `biblatex-examples.bib`

(TeX Live 2025: BibTeX 0.99d, biber 2.21, biblatex 3.21; two biblatex examples are skipped: one needs `xeCJK.sty`, the other makes up random keys.) The one failure, `biber-unit/sorting-template`, is a `uniquename` value biber gives a name written in lower case. Beyond the sets, on the databases of the corpus with every entry cited, 55 of 71 `.bbl` files are identical to biber's; the differences are uniqueness values in rarer configurations and a few name hashes of names with unusual LaTeX.

biber sorts with Perl's `Unicode::Collate`: the Unicode Collation Algorithm with the DUCET of Unicode 13.0, and CLDR's tailorings per locale. retex has its own implementation of the UCA, with that DUCET and the tailorings of the locales the corpus uses, as compact data files generated from the Unicode and CLDR data (`cargo xtask bib collation-data`). A byte-identical `.bbl` file needs biber's exact order, which a newer collation table would not give.

17.3 makeindex

```
$ cargo xtask index sets                       # the sets of cases
$ cargo xtask index prepare --set index-corpus # compile the corpus documents
$ cargo xtask index ref -j 3                   # outputs of makeindex
$ cargo xtask index run -j 3                   # the candidate, compared
$ cargo xtask index run --self                 # makeindex against itself
$ cargo xtask index run --candidate "path/to/prog --opt"
```

Each case is a makeindex command line and its input files, run in a fresh directory with `LC_ALL=C.UTF-8` and no `INDEXSTYLE`. A case **passes** when the files written are identical byte for byte (the transcript without its banner), and so are the terminal output and the exit status. The cache is `RETEX_INDEX_DIR`, else `index/` in the corpus cache; the baseline `bench/index-baseline.json`.

set	cases	pass	covers
`index-unit`	68	66	reading entries, the order of keys, page numbers, ranges and encapsulators, output styles, the command line
`index-fuzz`	400	400	generated entries, half of them malformed
`index-styles`	46	45	every installed `.ist` file (44), and doc's styles
`index-corpus`	35	35	the index files of 29 corpus documents

Some of what makeindex does, found by running it:

- Without a style file, the page types sort as lowercase roman, uppercase roman, arabic, lowercase alphabetic, uppercase alphabetic, but with the sizes of the documented order `rnaRA`, position by position: an arabic page number of 26 or more sorts with the letters.
- A single `c`, `d`, `l` or `m` is an alphabetic page number, a single `i`, `v` or `x` a roman one, unless the previous page number was of the other type and the same case.
- An explicit range is broken by a page of another type, goes on from there, and swallows the pages of other encapsulators while it is open.
- The lines makeindex counts, in its messages, are not all the line ends: those of delimiters, encapsulators, page suffixes and group headings are not counted.

The known differences: keys whose numbers are equal but written differently (`1`, `01`) or of 2^{31} or more, where makeindex's comparison overflows; page numbers that collide on makeindex's scale; and `-l` with keys that end in a space, where makeindex reads past the key.

17.4 TeX Live 2026

retex is tested against TeX Live 2026's lualatex (LuaHBTeX 1.24) as well as 2025's. The 2026 engines are observed only (run, never read): `luatex` and `luahbtex` from TeX Live 2026's archives, run over a 2026 tree (the runtime files of latex, latex-lab, l3kernel, l3packages, pgf, xcolor, hyperref, beamer, amsmath, babel and their dependencies; the rest from the installed 2025 system), with formats built by each engine. `tools/tl-wrappers.sh` writes wrappers that put them first on the `PATH`, and the suites run with `--tl 2026`, at the revisions of `xtask/<suite>-tl2026.lock`:

```
$ tools/tl-wrappers.sh /tmp/tl26 <tl2026>/bin/x86_64-linux/luahbtex \
    <tree> <luahbtex-formats> <retex-formats>
$ PATH=/tmp/tl26/bin:$PATH cargo xtask latex2e --tl 2026 --baseline
$ PATH=/tmp/tl26/bin:$PATH cargo xtask latex2e --tl 2026 --engine /tmp/tl26/retex
```

`RETEX_TEST_TL` runs the native differential tests against a year's tree and format.

suite (`--tl 2026`)	LuaHBTeX 1.24 passes	retex fails of those
latex2e 2026-06-01	497 of 1194	2
expl3 2026-09-09	255 of 260	0 (and passes 2 more)
pgf 3.1.12	113 of 116	1
babel 26.14	15 of 145	0

(Many latex2e tests fail with LuaHBTeX 1.24 too, as the tree mixes 2026 packages into a 2025 system.) What the 2026 kernel needed: any token whose command is a parameter character acts as `#` in a definition (the kernel's hook code relies on it); Type 1 fonts get ToUnicode maps; direction nodes take the current attributes, which tagpdf reads; and natives for pgf 3.1.12 and for the expl3 of September 2026.

retex chooses LuaTeX 1.24's behaviour when it loads a format of LaTeX 2026-06-01 or later (a format records no engine version; that LaTeX came with TeX Live 2026). The changes are in math, in the italic correction of a character nucleus, all observed with width probes and box displays: traditional text fonts also add it for a character followed by another of the same family (`$\mathrm{fo}$` is 8.83337 pt in 1.24, 8.05559 pt in 1.22); OpenType math fonts also

add it at the end of a list and before non-character noads, and an accented character always includes it; an OpenType character's box includes it (delimiter variants, large operators with limits); an OpenType operator with a character nucleus includes it in its box, loses it with an explicit `\nolimits` and before a subscript that is no limit.

A The sandbox

This page is here to be changed. Its source is `chapters/sandbox.tex`; in the browser editor, open it from the file list, and the PDF follows as you type.

> **Try it** The whole manual takes a few seconds to typeset, in a browser more. To try things quickly, build the sandbox alone: remove the `%` before `\includeonly{chapters/sandbox}` in the main file, `retex-manual.tex`. The references to other sections still work, from the last full build. Put the `%` back to see the whole manual again.

A.1 Text

TEX breaks paragraphs into lines all at once, choosing the breaks that make the whole paragraph look best [8]. Make this paragraph longer and the breaks move. Change `\emph{`*a word*`}` into `\textbf{`**a word**`}`, add a footnote,[1] or a reference to the introduction, Section 1.

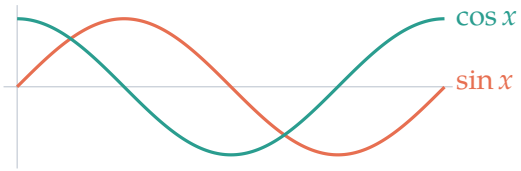
A.2 Mathematics

The Gaussian integral of the front page, once more:

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}, \qquad \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}. \tag{2}$$

Equation (2) is set in TeX Gyre Pagella Math, an OpenType math font.

A.3 A picture



A.4 A table

document	lualatex	retex
one-page article	0.64 s	0.15 s
the pgf/TikZ manual	423 s	134 s

A.5 Mistakes to fix

The paragraph below, between the comments, is written with common mistakes; TEX accepts them, and the PDF shows them. retex's hints mark each one in the source: fix them one by one, or all at once with *Fix all hints*.

> **Try it** Remove the `%` at the start of the two lines above, in `chapters/sandbox.tex`: the hints appear in the source as you type, and the build reports the reference to a figure that does not exist, with what it could have meant.

[1] Like this one.

A.6 Other scripts

The front page has Arabic and Hindi, shaped by HarfBuzz: مرحبا بالعالم is “hello, world” in Arabic, written from right to left, and नमस्ते दुनिया the same in Hindi. The languages are set up in `retex-manual.sty`, with babel.

B History

Work on retex began on 2026-10-16. The first versions were built in milestones (M0 to M7), each with its samples (`samples/m`⟨*N*⟩`/`) and a gate: `cargo xtask ci` green, and every sample's pages identical to lualatex's. Below are the releases, or the milestones they completed, with their dates and what they added; pre-releases are given with their release. `CHANGELOG.md` has every change, and `ROADMAP.md` the plans.

M0: Hello World (0.1.0, 2026-10-16)

An article of one line, raster-identical to lualatex's: TEX's scaled arithmetic; file lookup as kpsewhich; TEX82, e-TEX and enough LuaTEX to build the LATEX format, with `\directlua` through an embedded Lua 5.3; hyphenation; OpenType fonts and a native replacement for luaotfload; a PDF writer compatible with LuaTEX's; automatic passes.

M1 to M5 (0.9.0, 2026-10-17)

The first public release. Text documents (M1), the LuaTEX Lua interface (M2), mathematics with OpenType MATH fonts, amsmath and unicode-math (M3), graphics, hyperref and the PDF back end (M4), pgf and TikZ (M5). BibTeX and biber run between passes; later passes resume from a snapshot taken at `\begin{document}`; `--watch`; readable errors by default, LuaTEX's own output with `--compat=lualatex`.

M6 and 1.0.0 (2026-10-20; release candidates 0.9.9 and 1.0.0-rc.2 on 2026-10-19)

The first stable release. Languages and scripts (M6): babel and polyglossia, right-to-left text, complex scripts shaped with HarfRust. Errors and warnings that say how to fix them, and hints about the source; every luatex option; SyncTeX; TrueType fonts embedded. Every test that LuaTEX passes in the latex2e, l3 and pgf suites passes, five exceptions aside.

1.1.0 (2026-10-20)

Native macros (hot LATEX, expl3, pgfmath, xcolor and etoolbox macros run as engine operations); SVG images without Inkscape and SVG output; TEX Live 2026.

1.2.0 (2026-10-21)

`--watch` rebuilds that resume from the last page before an edit; experimental Windows and macOS support, with programs for each release; a manual page.

1.3.0 (2026-10-24; alphas and rc.1 on 2026-10-23)

retex in the browser (experimental); a reworked command line, with `--message-format=short|json`; native expl3 regular expressions, intarrays and pgfkeys; `retex profile` and `retex explain`; ODP and ODT output (experimental).

1.4.0 (2026-10-26; betas from 2026-10-25)

BibTeX, biber and makeindex in process, so that no external program is needed; HTML and Markdown output (experimental); the corpus of arXiv papers and CTAN manuals compared with lualatex; in the browser editor, the problems marked in the source with their fixes, SyncTeX and links.

1.5.0 (2026-10-27)

`--accessible`, tagged PDF/UA-2 output with checks of what screen readers need; the figure cache for unchanged TikZ pictures; browser rebuilds that resume from `\begin{document}`.

1.6.0 (2026-10-31; betas from 2026-10-28)

`retex lsp`, a language server, also in the browser editor; this manual; GitLab projects in the browser editor; more image formats; TEX Live 2026 checked; MetaPost, ported from MetaPost 2.11, for luamplib figures; native code in every pass; OpenType math

kerns.

1.7.0 (2026-11-03; alphas and betas from 2026-10-30)

retex as a library, from Rust and Python, pages as images (numpy arrays) (Section 10); luatexja; minted without Python, highlighted in process; the documentation of every TEX Live 2026 package compiled, and the problems it showed fixed; the lualibs functions packages call; this front page.

1.8.0 (2026-11-06; alphas and betas from 2026-11-03)

PDF/X-4 and PDF/X-1a for the press (Section 7.2); missing TEX Live packages downloaded when a document needs them (Section 2.5), and without a TEX Live installed, a TEX Live tree of retex's own; the TEX Live year (`--tl`); the settings file (Section 13.2); fuzzing of the engine, the Lua libraries, MetaPost, BibTeX and the image and SVG readers, with the hangs and crashes it found fixed.

References

[1] Pehong Chen and Michael A. Harrison. “Index Preparation and Processing”. In: *Software: Practice and Experience* 18.9 (1988), pp. 897–915.

[2] *Document Management Applications — Electronic Document File Format Enhancement for Accessibility — Part 2: Use of ISO 32000-2 (PDF/UA-2)*. ISO 14289-2:2024. International Organization for Standardization. 2024.

[3] *Graphic Technology — Prepress Digital Data Exchange Using PDF — Part 4: Complete Exchange of CMYK and Spot Colour Printing Data Using PDF 1.6 (PDF/X-4)*. ISO 15930-7:2010. International Organization for Standardization. 2010.

[4] *GWG2022 Specification*. Ghent Workgroup. 2022. URL: https://www.gwg.org/.

[5] Philip Kime and François Charette. *Biber: A Backend Bibliography Processor for `biblatex`*. Version 2.21. 2025.

[6] Donald E. Knuth. *TEX: The Program*. Vol. B. Computers & Typesetting. Reading, Massachusetts: Addison-Wesley, 1986.

[7] Donald E. Knuth. *METAFONT: The Program*. Vol. D. Computers & Typesetting. Reading, Massachusetts: Addison-Wesley, 1986.

[8] Donald E. Knuth and Michael F. Plass. “Breaking Paragraphs into Lines”. In: *Software: Practice and Experience* 11.11 (1981), pp. 1119–1184.

[9] Leslie Lamport. *MakeIndex: An Index Processor for LaTeX*. `makeindex.pdf`, in every TeX distribution. 1987.

[10] Jérôme Laurens. “Direct and Reverse Synchronization with SyncTeX”. In: *TUGboat* 29.3 (2008), pp. 365–371.

[11] Philipp Lehman, Philip Kime, and Moritz Wemheuer. *The `biblatex` Package*. Version 3.21. 2025.

[12] LuaTeX development team. *LuaTeX Reference Manual*. Version 1.22. 2025. URL: https://www.luatex.org/documentation.html.

[13] Microsoft. *Language Server Protocol Specification*. Version 3.17. 2022. URL: https://microsoft.github.io/language-server-protocol/.

[14] *Open Document Format for Office Applications (OpenDocument) Version 1.3*. OASIS. 2021.

[15] Oren Patashnik. *BibTeXing*. `btxdoc`, in every TeX distribution. 1988.

[16] Oren Patashnik. *Designing BibTeX Styles*. `btxhak`, in every TeX distribution. 1988.

[17] PyO3 project. *Maturin: build and publish crates with Rust and Python bindings*. URL: https://www.maturin.rs/.

[18] *retex: API documentation of the library*. URL: https://docs.rs/retex.

[19] Ken Whistler and Markus Scherer. *Unicode Collation Algorithm. Unicode Technical Standard #10*. Unicode Consortium. URL: https://www.unicode.org/reports/tr10/.

Index

`--accessible` (option), 45
`--color=WHEN` (option), 48
`--compat=lualatex` (option), 48
`--example` (option), 10
`--hyperlink-format=FORMAT` (option), 48
`--hyperlinks=WHEN` (option), 48
`--interactive` (option), 46
`--list-formats` (option), 46
`--luaonly FILE ARGS` (option), 49
`--message-format=FORMAT` (option), 48
`--no-aux-dir` (option), 47
`--no-clear` (option), 45
`--notify` (option), 45
`--offline` (option), 46
`--open` (option), 45
`--passes=N` (option), 45
`--stacks=FILE` (option), 48
`--symbols` (option), 48
`--tl=YEAR` (option), 47
`--verbose` (option), 47
`--watch` (option), 45
`-draftmode` (option), 46
`-h, --help` (option), 49
`-halt-on-error` (option), 46
`-ini, -fmt=NAME` (option), 49
`-interaction=MODE` (option), 48
`-jobname=NAME` (option), 47
`-lua=FILE` (option), 49
`-output-directory=DIR` (option), 47
`-output-format=FORMAT` (option), 46
`-q, --quiet` (option), 47
`-recorder` (option), 47
`-safer` (option), 49
`-shell-escape` (option), 46
`-synctex=N` (option), 46
`-v, --version` (option), 49

accessible PDF, 27
auxiliary files, 17

biber, 24
BibTeX, 24
browser editor, 35

clean room, 60
`CLICOLOR_FORCE` (environment variable), 51
compatibility mode, 54
corpus, 62

diagnostics, 20

embedding, 37

figure cache, 26

GitLab, 36

hints, 21
HTML, 32

images, 18
images, of pages, 38

language server, 33
`LC_ALL, LC_CTYPE, LANG` (environment variable), 51
library, 37
luatexja, 56

makeindex, 25
Markdown, 32
minted, 18

`NO_COLOR` (environment variable), 51

OpenDocument, 32

passes, 17
PDF/X, 27
printing, 27
Python, 37
    examples, 38

`retex explain` (command), 43
`retex lsp` (command), 33
`retex profile` (command), 42
`RETEX_BIB` (environment variable), 24, 50
`RETEX_CACHE` (environment variable), 50
`RETEX_CONFIG` (environment variable), 50
`RETEX_FIGURES` (environment variable), 26, 51
`RETEX_HINTS` (environment variable), 51
`RETEX_HYPERLINK_FORMAT` (environment variable), 51
`RETEX_MAKEINDEX` (environment variable), 25, 50
`RETEX_MINTED` (environment variable), 18, 50

RETEX_OFFLINE (environment variable), 50
RETEX_REPOSITORY (environment variable), 50
RETEX_TEXLIVE_KEYRING (environment variable), 50
RETEX_TL (environment variable), 50

sandbox, 69
SOURCE_DATE_EPOCH (environment variable), 51
SVG, 31
SyncTeX, 31

tagging, 27
TEXEDIT (environment variable), 51

watch mode, 26