preview-latex

A LaTeX preview mode for AUCTeX in Emacs.
Version 13.3, 2024-01-14

Jan-Åke Larsson
David Kastrup and others

This manual is for preview-latex, a LaTeX preview mode for AUCTeX (version 13.3 from 2024-01-14).

Table of Contents

preview-latex

preview-latex is a package embedding preview fragments into Emacs source buffers under the AUCTEX editing environment for LATEX. It uses `preview.sty` for the extraction of certain environments (most notably displayed formulas). Other applications of this style file are possible and exist.

The name of the package is really '`preview-latex`', all in lowercase letters, with a hyphen. If you typeset it, you can use a sans-serif font to visually offset it.

Copying

For the conditions for copying parts of preview-latex, see the General Public Licenses referred to in the copyright notices of the files, the General Public Licenses accompanying them and the explanatory section in Section "Copying" in *the AUCTEX manual*.

This manual specifically is covered by the GNU Free Documentation License (see Appendix C [Copying this Manual], page 36).

1 Introduction

Does your neck hurt from turning between previewer windows and the source too often? This AUCTEX component will render your displayed LATEX equations right into the editing window where they belong.

The purpose of preview-latex is to embed LATEX environments such as display math or figures into the source buffers and switch conveniently between source and image representation.

1.1 What use is it?

WYSIWYG (what you see is what you get) sometimes is considered all the rage, sometimes frowned upon. Do we really want it? Wrong question. The right question is *what* we want from it. Except when finetuning the layout, we don't want to use printer fonts for on-screen text editing. The low resolution and contrast of a computer screen render all but the coarsest printer fonts (those for low-quality newsprint) unappealing, and the margins and pagination of the print are not wanted on the screen, either. On the other hand, more complex visual compositions like math formulas and tables can't easily be taken in when seen only in the source. preview-latex strikes a balance: it only uses graphic renditions of the output for certain, configurable constructs, does this only when told, and then right in the source code. Switching back and forth between the source and preview is easy and natural and can be done for each image independently. Behind the scenes of preview-latex, a sophisticated framework of other programs like '`dvipng`', Dvips and Ghostscript are employed together with a special LATEX style file for extracting the material of interest in the background and providing fast interactive response.

1.2 Activating preview-latex

After installation, the package may need to be activated (and remember to activate AUCTEX too). If preview-latex is installed via the Emacs package manager (ELPA), activation should be automatic upon installation.

The usual activation (if it is not done automatically) would be

```
(load "preview-latex.el" nil t t)
```

If you still don't get a "Preview" menu in LATEX mode in spite of AUCTEX showing its "Command", your installation is broken. One possible cause are duplicate Lisp files that might be detectable with *M-x list-load-path-shadows* RET.

1.3 Getting started

Once activated, preview-latex and its documentation will be accessible via its menus (note that preview-latex requires AUCTEX to be loaded). When you have loaded a LATEX document (a sample document `circ.tex` is included in the distribution, but most documents including math and/or figures should do), you can use its menu or *C-c C-p C-d* (for '`Preview/Document`'). Previews will now be generated for various objects in your document. You can use the time to take a short look at the other menu entries and key bindings in the '`Preview`' menu. You'll see the previewed objects change into a roadworks sign when preview-latex has determined just what it is going to preview. Note that you can freely

navigate the buffer while this is going on. When the process is finished you will see the objects typeset in your buffer.

It is a bad idea, however, to edit the buffer before the roadworks signs appear, since that is the moment when the correlation between the original text and the buffer locations gets established. If the buffer changes before that point of time, the previews will not be placed where they belong. If you do want to change some obvious error you just spotted, we recommend you stop the background process by pressing *C-c C-k*.

To see/edit the LaTeX code for a specific object, put the point (the cursor) on it and press *C-c C-p C-p* (for '`Preview/at point`'). It will also do to click with the middle mouse button on the preview. Now you can edit the code, and generate a new preview by again pressing *C-c C-p C-p* (or by clicking with the middle mouse button on the icon before the edited text).

If you are using the `desktop` package, previews will remain from one session to the next as long as you don't kill your buffer.

1.4 Basic modes of operation

preview-latex has a number of methods for generating its graphics. Its default operation is equivalent to using the 'LaTeX' command from AUCTeX. If this happens to be a call of PDFLaTeX generating PDF output (you need at least AUCTeX 11.51 for this), then Ghostscript will be called directly on the resulting PDF file. If a DVI file gets produced, first Dvips and then Ghostscript get called by default.

The image type to be generated by Ghostscript can be configured with

M-x customize-option RET *preview-image-type* RET

The default is '`png`' (the most efficient image type). A special setting is '`dvipng`' in case you have the '`dvipng`'

program installed. In this case, '`dvipng`' will be used for converting DVI files and Ghostscript (with a '`PNG`' device) for converting PDF files. '`dvipng`' is much faster than the combination of Dvips and Ghostscript. You can get downloads, access to its CVS archive and further information from its project site (`https://savannah.nongnu.org/projects/dvipng`).

1.5 More documentation

After the installation, documentation in the form of an info manual will be available. You can access it with the standalone info reader with

```
info preview-latex
```

or by pressing *C-h i d m preview-latex* RET in Emacs. Once preview-latex is activated, you can instead use *C-c C-p* TAB (or the menu entry '`Preview/Read documentation`').

Depending on your installation, this printed manual may also be available in the form of `preview-latex.pdf`.

Detailed documentation for the LaTeX style used for extracting the preview images is placed in `preview.pdf` in a suitable directory during installation; on typical TeX Live-based systems,

```
texdoc preview
```

will display it.

1.6 Availability

The preview-latex project is now part of AUCTEX and accessible as part of the AUCTEX project page (`https://savannah.gnu.org/projects/auctex`). You can get its files from the AUCTEX download area (`https://ftp.gnu.org/pub/gnu/auctex/`). As of AUCTEX 11.81, preview-latex should already be integrated into AUCTEX, so no separate download will be necessary.

Anonymous Git is available at `git://git.savannah.gnu.org/auctex.git` or `https://git.savannah.gnu.org/git/auctex.git`. You can also browse the repository (`https://git.savannah.gnu.org/cgit/auctex.git`) via web interface.

1.7 Contacts

Bug reports should be sent by using *`M-x preview-report-bug`* `RET`, as this will fill in a lot of information interesting to us. If the installation fails (but this should be a rare event), report bugs to `bug-auctex@gnu.org`.

There is a general discussion list for AUCTEX which also covers preview-latex, look at `https://lists.gnu.org/mailman/listinfo/auctex`. For more information on the mailing list, send a message with just the word "help" as subject or body to `auctex-request@gnu.org`. For the developers, there is the `auctex-devel@gnu.org` list; it would probably make sense to direct feature requests and questions about internal details there. There is a low-volume read-only announcement list available to which you can subscribe by sending a mail with "subscribe" in the subject to `info-auctex-request@gnu.org`.

Offers to support further development will be appreciated. If you want to show your appreciation with a donation to the main developer, you can do so via PayPal to `dak@gnu.org`, and of course you can arrange for service contracts or for added functionality. Take a look at the `TODO` list for suggestions in that area.

2 Installation

Installation is now being covered in Section "Installation" in *the AUCTEX manual*.

3 Key bindings and user-level lisp functions

preview-latex adds key bindings starting with *C-c C-p* to the supported modes of AUCTEX (See Section "Key Index" in `auctex`). It will also add its own '`Preview`' menu in the menu bar, as well as an icon in the toolbar.

The following only describes the interactive use: view the documentation strings with *C-h f* if you need the Lisp information.

C-c C-p C-p
`preview-at-point`
Preview/Generate previews (or toggle) at point

If the cursor is positioned on or inside of a preview area, this toggles its visibility, regenerating the preview if necessary. If not, it will run the surroundings through preview. The surroundings include all areas up to the next valid preview, unless invalid previews occur before, in which case the area will include the last such preview in either direction. And overriding any other action, if a region is active (`transient-mark-mode`), it is run through `preview-region`.

`mouse-2` The middle mouse button has a similar action bound to it as `preview-at-point`, only that it knows which preview to apply it to according to the position of the click. You can click either anywhere on a previewed image, or when the preview is opened and showing the source text, you can click on the icon preceding the source text. In other areas, the usual mouse key action (typically: paste) is not affected.

`mouse-3` The right mouse key pops up a context menu with several options: toggling the preview, regenerating it, removing it (leaving the unpreviewed text), copying the text inside of the preview, and copying it in a form suitable for copying as an image into a mail or news article. This is a one-image variant of the following command:

C-c C-p C-w
`preview-copy-region-as-mml`
Copy a region as MML

This command is also available as a variant in the context menu on the right mouse button (where the region is the preview that has been clicked on). It copies the current region into the kill buffer in a form suitable for copying as a text including images into a mail or news article using mml-mode (see Section "Composing" in *Emacs MIME*).

If you regenerate or otherwise kill the preview in its source buffer before the mail or news gets posted, this will fail. Also you should generate images you want to send with `preview-transparent-border`

set to `nil`, or the images will have an ugly border. preview-latex detects this condition and asks whether to regenerate the region with borders switched off. As this is an asynchronous operation running in the background, you'll need to call this command explicitly again to get the newly generated images into the kill ring.

Preview your articles with `mml-preview` (on *C-c C-m P*)

to make sure they look fine.

C-c C-p C-e
`preview-environment`
Preview/Generate previews for environment
: Run preview on LaTeX environment. The environments in `preview-inner-environments` are treated as inner levels so that for instance, the `split` environment in `\begin{equation}\begin{split}...\end{split}\end{equation}` is properly displayed. If called with a numeric argument, the corresponding number of outward nested environments is treated as inner levels.

C-c C-p C-s
`preview-section`
Preview/Generate previews for section
: Run preview on this LaTeX section.

C-c C-p C-r
`preview-region`
Preview/Generate previews for region
: Run preview on current region.

C-c C-p C-b
`preview-buffer`
Preview/Generate previews for buffer
: Run preview on the current buffer.

C-c C-p C-d
`preview-document`
Preview/Generate previews for document
: Run preview on the current document.

C-c C-p C-c C-p
`preview-clearout-at-point`
Preview/Remove previews at point
: Clear out (remove) the previews that are immediately adjacent to point.

C-c C-p C-c C-s
`preview-clearout-section`
Preview/Remove previews from section
: Clear out all previews in current section.

C-c C-p C-c C-r
`preview-clearout`
Preview/Remove previews from region
: Clear out all previews in the current region.

C-c C-p C-c C-b
`preview-clearout-buffer`
Preview/Remove previews from buffer
: Clear out all previews in current buffer. This makes the current buffer lose all previews.

C-c C-p C-c C-d
`preview-clearout-document`
Preview/Remove previews from document

Clear out all previews in current document. The document consists of all buffers that have the same master file as the current buffer. This makes the current document lose all previews.

C-c C-p C-f
`preview-cache-preamble`
Preview/Turn preamble cache on

Dump a pregenerated format file. For the rest of the session, this file is used when running on the same master file. Use this if you know your LaTeX takes a long time to start up, the speedup will be most noticeable when generating single or few previews. If you change your preamble, do this again. preview-latex will try to detect the necessity of that automatically when editing changes to the preamble are done from within Emacs, but it will not notice if the preamble effectively changes because some included file or style file is tampered with.

Note that support for preamble cache is limited for LaTeX variants. c.f. `https://github.com/davidcarlisle/dpctex/issues/15`

- XeLaTeX cannot use preamble cache at all. The reason is intrinsic in XeLaTeX, so preview-latex can't help.
- LuaLaTeX works with preamble cache only when the preamble is simple enough, i.e., when it doesn't load opentype fonts and it doesn't use lua codes in preamble.

C-c C-p C-c C-f
`preview-cache-preamble-off`
Preview/Turn preamble cache off

Clear the pregenerated format file and stop using preambles for the current document. If the caching gives you problems, use this.

C-c C-p C-i
`preview-goto-info-page`
Preview/Read Documentation

Read the info manual.

M-x preview-report-bug RET
`preview-report-bug`
Preview/Report Bug

This is the preferred way of reporting bugs as it will fill in what version of preview-latex you are using as well as versions of relevant other software, and also some of the more important settings. Please use this method of reporting, if at all possible and before reporting a bug, have a look at Chapter 5 [Known problems], page 13.

C-c C-k
LaTeX/TeX Output/Kill Job

Kills the preview-generating process. This is really an AUCTeX keybinding, but it is included here as a hint. If you are generating a preview and then make

a change to the buffer, preview-latex may be confused and place the previews wrong.

4 Simple customization

Customization options can be found by typing *M-x customize-group* RET *preview* RET. Remember to set the option when you have changed it. The list of suggestions can be made very long (and is covered in detail in Chapter 6 [For advanced users], page 16), but some are:

- Change the color of the preview background

  If you use a non-white background in Emacs, you might have color artifacts at the edges of your previews. Playing around with the option `preview-transparent-color` in the '`Preview Appearance`' group might improve things. With some settings, the cursor may cover the whole background of a preview, however.

  This option is specific to the display engine in use.

- Showing `\label`s

  When using preview-latex, the `\label`s are hidden by the previews. It is possible to make them visible in the output by using the LaTeX package `showkeys` alternatively `showlabels`. However, the boxes of these labels will be outside the region preview-latex considers as the preview image. To enable a similar mechanism internal to preview-latex, enable the `showlabels` option in the variable `preview-default-option-list` in the '`Preview Latex`' group.

  It must be noted, however, that a much better idea may be to use the RefTeX package for managing references. See Section "RefTeX in a Nutshell" in *The RefTeX Manual*.

- Open previews automatically

  The current default is to open previews automatically when you enter them with cursor left/right motions. Auto-opened previews will close again once the cursor leaves them again (this is also done when doing incremental search, or query-replace operations), unless you changed anything in it. In that case, you will have to regenerate the preview (via e.g., *C-c C-p C-p*). Other options for `preview-auto-reveal` are available via `customize`.

- Automatically cache preambles

  Currently preview-latex asks you whether you want to cache the document preamble (everything before `\begin{document}`) before it generates previews for a buffer the first time. Caching the preamble will significantly speed up regeneration of previews. The larger your preamble is, the more this will be apparent. Once a preamble is cached, preview-latex will try to keep track of when it is changed, and dump a fresh format in that case. If you experience problems with this, or if you want it to happen without asking you the first time, you can customize the variable `preview-auto-cache-preamble`.

- Attempt to keep counters accurate when editing

  Since preview-latex frequently runs only small regions through LaTeX, values like equation counters are not consistent from run to run. If this bothers you, customize the variable `preview-preserve-counters` to `t` (this is consulted by `preview-required-option-list`). LaTeX will then output a load of counter information during compilation, and this information will be used on subsequent updates to keep counters set to useful values. The additional information takes additional time to analyze, but this is relevant mostly only when you are regenerating all previews at once, and maybe you will be less tempted to do so when counters appear more or less correct.

- Preview your favourite LaTeX constructs

  If you have a certain macro or environment that you want to preview, first check if it can be chosen by cutomizing `preview-default-option-list` in the '`Preview Latex`' group.

  If it is not available there, you can add it to `preview-default-preamble` also in the '`Preview Latex`' group, by adding a `\PreviewMacro` or `\PreviewEnvironment` entry (see Section 6.1.2 [Provided commands], page 20) *after* the `\RequirePackage` line. For example, if you want to preview the `center` environment, press the `Show` button and the last `INS` button, then add

  ```
  \PreviewEnvironment{center}
  ```

  in the space that just opened. Note that since `center` is a generic formatting construct of LaTeX, a general configuration like that is not quite prudent. You better to do this on a per-document base so that it is easy to disable this behavior when you find this particular entry gives you trouble.

  One possibility is to save such settings in the corresponding file-local variable instead of your global configuration (see Section "Local Variables in Files" in *GNU Emacs Manual*). A perhaps more convenient place for such options would be in a configuration file in the same directory with your project (see Section 6.1.1 [Package options], page 16).

  The usual file for preview-latex preconfiguration is `prauctex.cfg`. If you also want to keep the systemwide defaults, you should add a line

  ```
  \InputIfFileExists{preview/prauctex.cfg}{}{}
  ```

  to your own version of `prauctex.cfg` (this is assuming that global files relating to the `preview` package are installed in a subdirectory `preview`, the default behavior).

- Don't preview inline math

  If you have performance problems because your document is full of inline math (`$...$`), or if your usage of `$` conflicts with preview-latex's, you can turn off inline math previews. In the '`Preview Latex`' group, remove `textmath` from `preview-default-option-list` by customizing this variable.

5 Known problems

A number of issues are known concerning the interoperation with various other software. Some of the known problems can be solved by moving to newer versions of the problematic software or by simple patches.

If you find something not mentioned here, please send a bug report using *M-x preview-report-bug* RET, which will fill in a lot of information interesting to us and send it to the `bug-auctex@gnu.org` list. Please use the bug reporting commands if at all possible.

5.1 Font problems with Dvips

Some fonts have been reported to produce wrong characters with preview-latex. preview-latex calls Dvips by default with the option `-Pwww` in order to get scalable fonts for nice results. If you are using antialiasing, however, the results might be sufficiently nice with bitmapped fonts, anyway. You might try `-Ppdf` for another stab at scalable fonts, or other printer definitions. Use

M-x customize-option RET *preview-fast-dvips-command* RET

and

M-x customize-option RET *preview-dvips-command* RET

in order to customize this.

One particular problem is that several printer setup files (typically in a file called `/usr/share/texmf/dvips/config/config.pdf` if you are using the `-Ppdf` switch) contain the `G` option for 'character shifting'. This option will result in '`fi`' being rendered as '`£`' (British Pounds sign) in several fonts, unless your version of Dvips has a long-standing bug in its implementation fixed (only very recent versions of Dvips have).

5.2 Too small bounding boxes

The bounding box of a preview is determined by the LaTeX package using the pure TeX bounding boxes. If there is material extending outside of the TeX box, that material will be missing from the preview image. This happens for the label-showing boxes from the `showkeys` package. This particular problem can be circumvented by using the `showlabels` option of the preview package.

In general, you should try to fix the problem in the TeX code, like avoiding drawing outside of the picture with PSTricks.

One possible remedy is to set `preview-fast-conversion` to 'Off' (see Section 6.2 [The Emacs interface], page 23). The conversion will take more time, but will then use the bounding boxes from EPS files generated by Dvips.

Dvips generally does not miss things, but it does not understand PostScript constructs like `\resizebox` or `\rotate` commands, so will generate rather wrong boxes for those. Dvips can be helped with the `psfixbb` package option to preview (see Section 6.1 [The LaTeX style file], page 16), which will tag the corners of the included TeX box. This will mostly be convenient for *pure* PostScript stuff like that created by PSTricks, which Dvips would otherwise reserve no space for.

5.3 x-symbol interoperation

Thanks to the work of Christoph Wedler, starting with version '`4.0h/beta`' of x-symbol, the line parsing of AUCTEX and preview-latex is fully supported. Earlier versions exhibit problems. However, versions before '`4.2.2`' will cause a drastic slowdown of preview-latex's parsing pass, so we don't recommend to use versions earlier than that.

If you wonder what x-symbol is, it is a package that transforms various tokens and subscripts to a more readable form while editing and offers a few input methods handy especially for dealing with math. Take a look at `http://x-symbol.sourceforge.net/`.

x-symbol versions up to '`4.5.1-beta`' at least require an 8bit-clean TEX implementation (meaning that its terminal output should not use '`^^`'-started escape sequences) for cooperation with preview-latex. Later versions may get along without it, like preview-latex does now.

If you experience problems with `circ.tex` in connection with both x-symbol and Latin-1 characters, you may need to change your language environment or, as a last resort, customize the variable `LaTeX-command-style` by replacing the command `latex` with `latex -translate-file=cp8bit`.

5.4 Middle-clicks paste instead of toggling

This is probably the fault of your favorite package. `isearch.el` is known to be affected while searches are in progress, but the code is such a complicated mess that no patch is in sight. Better just end the search with `RET` before toggling and resume with *C-s C-s* or similar afterwards. Since previews over the current match will auto-open, anyway, this should not be much of a problem in practice.

5.5 No images are displayed with gs 9.27 and earlier

preview-latex tries to adjust the foreground and background colors of generated images to those of Emacs. Unfortunately, incompatible changes introduced in Ghostscript 9.27 breaks the traditional method partially, and preview-latex can display no images under certain circumstances.

A new method implemented alternatively works only with Ghostscript > 9.27. If you are using Ghostscript 9.27 or earlier, customize the option `preview-pdf-adjust-color-method`.

`preview-pdf-adjust-color-method` [User Option]

Method to adjust colors of images generated from PDF. It is not consulted when the LATEX command produces DVI files.

When the option is `t` (default), preview-latex adjusts the FG and BG colors of the generated images by the new method. This method requires that Ghostscript has working `DELAYBIND` feature, thus is invalid with gs 9.27 (and possibly < 9.27).

When it is `compatible`, preview-latex uses traditional method. This option is provided for backward compatibility with older gs. See the below explanation for detail.

When `nil`, no adjustment is done and "black on white" image is generated regardless of Emacs color. This is provided for fallback for gs 9.27 users with customized foreground color. See the below explanation for detail.

When the LaTeX command produces PDF rather than DVI and Emacs has non-trivial foreground color, the traditional method (`compatible`) makes gs >= 9.27 to stop with error. Here, "non-trivial foreground color" includes customized themes.

If you use such non-trivial foreground color and the version of Ghostscript equals to 9.27, you have two options:

1. Choose the value `compatible` and customize `preview-reference-face` to have default (black) foreground color. This makes the generated image almost non-readable on dark background, so the next option would be your only choice in that case.
2. Choose the value `nil`, which forces plain "black on white" appearance for the generated image. You can at least read what are written in the image although they may not match with your Emacs color well.

The default value used to be `compatible` for short period before Ghostscript 9.50 was released but now is `t`.

5.6 Black texts are too hard to read on dark background

Unfortunately, foreground color adjustment discussed in the previous node doesn't work for XeLaTeX for technical reason. The texts are always rendered as black in the preview images, so it's almost impossible to read them on dark background. Hence XeLaTeX users who like dark background in Emacs frame should customize `preview-pdf-adjust-color-method` to `nil`.

6 For advanced users

This package consists of two parts: a LaTeX style that splits the output into appropriate parts with one preview object on each page, and an Emacs-lisp part integrating the thing into Emacs (aided by AUCTeX).

6.1 The LaTeX style file

The main purpose of this package is the extraction of certain environments (most notably displayed formulas) from LaTeX sources as graphics. This works with DVI files postprocessed by either Dvips and Ghostscript or dvipng, but it also works when you are using PDFTeX for generating PDF files (usually also postprocessed by Ghostscript).

Current uses of the package include the preview-latex package for WYSIWYG functionality in the AUCTeX editing environment, generation of previews in LyX, as part of the operation of the pst-pdf package, the tbook XML system and some other tools.

Producing EPS files with Dvips and its derivatives using the `-E` option is not a good alternative: People make do by fiddling around with `\thispagestyle{empty}` and hoping for the best (namely, that the specified contents will indeed fit on single pages), and then trying to guess the baseline of the resulting code and stuff, but this is at best dissatisfactory. The preview package provides an easy way to ensure that exactly one page per request gets shipped, with a well-defined baseline and no page decorations. While you still can use the preview package with the 'classic'

```
dvips -E -i
```

invocation, there are better ways available that don't rely on Dvips not getting confused by PostScript specials.

For most applications, you'll want to make use of the `tightpage` option. This will embed the page dimensions into the PostScript or PDF code, obliterating the need to use the `-E -i` options to Dvips. You can then produce all image files with a single run of Ghostscript from a single PDF or PostScript (as opposed to EPS) file.

Various options exist that will pass TeX dimensions and other information about the respective shipped out material (including descender size) into the log file, where external applications might make use of it.

The possibility for generating a whole set of graphics with a single run of Ghostscript (whether from LaTeX or PDFLaTeX) increases both speed and robustness of applications. It is also feasible to use dvipng on a DVI file with the options

```
-picky -noghostscript
```

to omit generating any image file that requires Ghostscript, then let a script generate all missing files using Dvips/Ghostscript. This will usually speed up the process significantly.

6.1.1 Package options

The package is included with the customary

```
\usepackage[options]{preview}
```

You should usually load this package as the last one, since it redefines several things that other packages may also provide.

The following options are available:

`active` is the most essential option. If this option is not specified, the `preview` package will be inactive and the document will be typeset as if the `preview` package were not loaded, except that all declarations and environments defined by the package are still legal but have no effect. This allows defining previewing characteristics in your document, and only activating them by calling LaTeX as

```
latex '\PassOptionsToPackage{active}{preview} \input{filename}'
```

`noconfig` Usually the file `prdefault.cfg` gets loaded whenever the `preview` package gets activated. `prdefault.cfg` is supposed to contain definitions that can cater for otherwise bad results, for example, if a certain document class would otherwise lead to trouble. It also can be used to override any settings made in this package, since it is loaded at the very end of it. In addition, there may be configuration files specific for certain `preview` options like `auctex` which have more immediate needs. The `noconfig` option suppresses loading of those option files, too.

`psfixbb` Dvips determines the bounding boxes from the material in the DVI file it understands. Lots of PostScript specials are not part of that. Since the TeX boxes do not make it into the DVI file, but merely characters, rules and specials do, Dvips might include far too small areas. The option `psfixbb` will include `/dev/null` as a graphic file in the ultimate upper left and lower right corner of the previewed box. This will make Dvips generate an appropriate bounding box.

`dvips` If this option is specified as a class option or to other packages, several packages pass things like page size information to Dvips, or cause crop marks or draft messages written on pages. This seriously hampers the usability of previews. If this option is specified, the changes will be undone if possible.

`pdftex` If this option is set, PDFTeX is assumed as the output driver. This mainly affects the `tightpage` option.

`xetex` If this option is set, XeTeX is assumed as the output driver. This mainly affects the `tightpage` option.

`displaymath`
will make all displayed math environments subject to preview processing. This will typically be the most desired option.

`floats` will make all float objects subject to preview processing. If you want to be more selective about what floats to pass through to a preview, you should instead use the `\PreviewSnarfEnvironment` command on the floats you want to have previewed.

`textmath` will make all text math subject to previews. Since math mode is used throughly inside of LaTeX even for other purposes, this works by redefining `\(`, `\)` and `$` and the `math` environment (apparently some people use that). Only occurences of these text math delimiters in later loaded packages and in the main document will thus be affected.

`graphics` will subject all `\includegraphics` commands to a preview.

`sections` will subject all section headers to a preview.

`delayed` will delay all activations and redefinitions the `preview` package makes until `\begin{document}`. The purpose of this is to cater for documents which should be subjected to the `preview` package without having been prepared for it. You can process such documents with

```
latex '\RequirePackage[active,delayed,options]{preview}
\input{filename}'
```

This relaxes the requirement to be loading the `preview` package as last package.

driver loads a special driver file `prdriver.def`. The remaining options are implemented through the use of driver files.

`auctex` This driver will produce fake error messages at the start and end of every preview environment that enable the Emacs package preview-latex in connection with AUCTEX to pinpoint the exact source location where the previews have originated. Unfortunately, there is no other reliable means of passing the current TEX input position *in* a line to external programs. In order to make the parsing more robust, this option also switches off quite a few diagnostics that could be misinterpreted.

You should not specify this option manually, since it will only be needed by automated runs that want to parse the pseudo error messages. Those runs will then use `\PassOptionsToPackage` in order to effect the desired behaviour. In addition, `prauctex.cfg` will get loaded unless inhibited by the `noconfig` option. This caters for the most frequently encountered problematic commands.

`showlabels`

During the editing process, some people like to see the label names in their equations, figures and the like. Now if you are using Emacs for editing, and in particular preview-latex, I'd strongly recommend that you check out the RefTEX package which pretty much obliterates the need for this kind of functionality. If you still want it, standard LATEX provides it with the `showkeys` package, and there is also the less encompassing `showlabels` package. Unfortunately, since those go to some pain not to change the page layout and spacing, they also don't change `preview`'s idea of the TEX dimensions of the involved boxes. So if you are using `preview` for determing bounding boxes, those packages are mostly useless. The option `showlabels` offers a substitute for them.

`tightpage`

It is not uncommon to want to use the results of `preview` as graphic images for some other application. One possibility is to generate a flurry of EPS files with

```
dvips -E -i -Pwww -o outputfile.000 inputfile
```

However, in case those are to be processed further into graphic image files by Ghostscript, this process is inefficient since all of those files need to be processed one by one. In addition, it is necessary to extract the bounding box comments from the EPS files and convert them into page dimension parameters for Ghostscript in order to avoid full-page graphics. This is not even possible

if you wanted to use Ghostscript in a *single* run for generating the files from a single PostScript file, since Dvips will in that case leave no bounding box information anywhere.

The solution is to use the `tightpage` option. That way a single command line like

```
gs -sDEVICE=png16m -dTextAlphaBits=4 -r300
-dGraphicsAlphaBits=4 -dSAFER -q -dNOPAUSE
-sOutputFile=outputfile%d.png inputfile.ps
```

will be able to produce tight graphics from a single PostScript file generated with Dvips *without* use of the options `-E -i`, in a single run.

The `tightpage` option actually also works when using the `pdftex` option and generating PDF files with PDFTEX. The resulting PDF file has separate page dimensions for every page and can directly be converted with one run of Ghostscript into image files.

If neither `dvips` or `pdftex` have been specified, the corresponding option will get autodetected and invoked.

If you need this in a batch environment where you don't want to use `preview`'s automatic extraction facilities, no problem: just don't use any of the extraction options, and wrap everything to be previewed into `preview` environments. This is how LyX does its math previews.

If the pages under the `tightpage` option are just too tight, you can adjust by setting the length `\PreviewBorder` to a different value by using `\setlength`. The default value is `0.50001bp`, which is half of a usual PostScript point, rounded up. If you go below this value, the resulting page size may drop below `1bp`, and Ghostscript does not seem to like that. If you need finer control, you can adjust the bounding box dimensions individually by changing the macro `\PreviewBbAdjust` with the help of `\renewcommand`. Its default value is

```
\newcommand \PreviewBbAdjust
{-\PreviewBorder -\PreviewBorder
\PreviewBorder  \PreviewBorder}
```

This adjusts the left, lower, right and upper borders by the given amount. The macro must contain 4 TEX dimensions after another, and you may not omit the units if you specify them explicitly instead of by register. PostScript points have the unit `bp`.

`lyx` This option is for the sake of LyX developers. It will output a few diagnostics relevant for the sake of LyX' preview functionality (at the time of writing, mostly implemented for math insets, in versions of LyX starting with 1.3.0).

`counters` This writes out diagnostics at the start and the end of previews. Only the counters changed since the last output get written, and if no counters changed, nothing gets written at all. The list consists of counter name and value, both enclosed in `{}` braces, followed by a space. The last such pair is followed by a colon (`:`) if it is at the start of the preview snippet, and by a period (`.`) if it is at the end. The order of different diagnostics like this being issued depends on the order of the specification of the options when calling the package.

Systems like preview-latex use this for keeping counters accurate when single previews are regenerated.

`footnotes`
: This makes footnotes render as previews, and only as their footnote symbol. A convenient editing feature inside of Emacs.

The following options are just for debugging purposes of the package and similar to the corresponding TEX commands they allude to:

`tracingall`
: causes lots of diagnostic output to appear in the log file during the preview collecting phases of TEX's operation. In contrast to the similarly named TEX command, it will not switch to `\errorstopmode`, nor will it change the setting of `\tracingonline`.

`showbox` This option will show the contents of the boxes shipped out to the DVI files. It also sets `\showboxbreadth` and `\showboxdepth` to their maximum values at the end of loading this package, but you may reset them if you don't like that.

6.1.2 Provided commands

`\begin{preview}...\end{preview}`
: The `preview` environment causes its contents to be set as a single preview image. Insertions like figures and footnotes (except those included in minipages) will typically lead to error messages or be lost. In case the `preview` package has not been activated, the contents of this environment will be typeset normally.

`\begin{nopreview}...\end{nopreview}`
: The `nopreview` environment will cause its contents not to undergo any special treatment by the `preview` package. When `preview` is active, the contents will be discarded like all main text that does not trigger the `preview` hooks. When `preview` is not active, the contents will be typeset just like the main text.

  Note that both of these environments typeset things as usual when preview is not active. If you need something typeset conditionally, use the `\ifPreview` conditional for it.

`\PreviewMacro`
: If you want to make a macro like

  `\includegraphics` (actually, this is what is done by the `graphics` option to `preview`) produce a preview image, you put a declaration like

  ```
  \PreviewMacro[*[[!]{\includegraphics}
  ```

  or, more readable,

  ```
  \PreviewMacro[{*[][]{}}]{\includegraphics}
  ```

  into your preamble. The optional argument to `\PreviewMacro` specifies the arguments `\includegraphics` accepts, since this is necessary information for properly ending the preview box. Note that if you are using the more readable form, you have to enclose the argument in a `[{` and `}]` pair. The inner braces are necessary to stop any included `[]` pairs from prematurely ending the optional

argument, and to make a single `{}` denoting an optional argument not get stripped away by TEX's argument parsing.

The letters simply mean

`*` indicates an optional `*` modifier, as in `\includegraphics*`.

`[` indicates an optional argument in brackets. This syntax is somewhat baroque, but brief.

`[]` also indicates an optional argument in brackets. Be sure to have encluded the entire optional argument specification in an additional pair of braces as described above.

`!` indicates a mandatory argument.

`{}` indicates the same. Again, be sure to have that additional level of braces around the whole argument specification.

`?`*delimiter*{*true case*}{*false case*}
is a conditional. The next character is checked against being equal to *delimiter*. If it is, the specification *true case* is used for the further parsing, otherwise *false case* will be employed. In neither case is something consumed from the input, so {*true case*} will still have to deal with the upcoming delimiter.

`@`{*literal sequence*}
will insert the given sequence literally into the executed call of the command.

`-` will just drop the next token. It will probably be most often used in the true branch of a `?` specification.

`#`{*argument*}{*replacement*}
is a transformation rule that calls a macro with the given argument and replacement text on the rest of the argument list. The replacement is used in the executed call of the command. This can be used for parsing arbitrary constructs. For example, the `[]` option could manually be implemented with the option string `?[{#{[#1]}{[{#1}]}}{}`. PStricks users might enjoy this sort of flexibility.

`:`{*argument*}{*replacement*}
is again a transformation rule. As opposed to `#`, however, the result of the transformation is parsed again. You'll rarely need this.

There is a second optional argument in brackets that can be used to declare any default action to be taken instead. This is mostly for the sake of macros that influence numbering: you would want to keep their effects in that respect. The default action should use `#1` for referring to the original (not the patched) command with the parsed options appended. Not specifying a second optional argument here is equivalent to specifying `[#1]`.

`\PreviewMacro*`

A similar invocation `\PreviewMacro*` simply throws the macro and all of its arguments declared in the manner above away. This is mostly useful for having things like `\footnote` not do their magic on their arguments. More often than not, you don't want to declare any arguments to scan to `\PreviewMacro*` since you would want the remaining arguments to be treated as usual text and typeset in that manner instead of being thrown away. An exception might be, say, sort keys for `\cite`.

A second optional argument in brackets can be used to declare any default action to be taken instead. This is for the sake of macros that influence numbering: you would want to keep their effects in that respect. The default action might use `#1` for referring to the original (not the patched) command with the parsed options appended. Not specifying a second optional argument here is equivalent to specifying `[]` since the command usually gets thrown away.

As an example for using this argument, you might want to specify

```
\PreviewMacro*[{[]}][#1{}]{\footnote}
```

This will replace a footnote by an empty footnote, but taking any optional parameter into account, since an optional paramter changes the numbering scheme. That way the real argument for the footnote remains for processing by preview-latex.

`\PreviewEnvironment`

The macro

`\PreviewEnvironment` works just as `\PreviewMacro` does, only for environments.

`\PreviewEnvironment*`

And the same goes for `\PreviewEnvironment*` as compared to `\PreviewMacro*`.

`\PreviewSnarfEnvironment`

This macro does not typeset the original environment inside of a preview box, but instead typesets just the contents of the original environment inside of the preview box, leaving nothing for the original environment. This has to be used for figures, for example, since they would

1. produce insertion material that cannot be extracted to the preview properly,
2. complain with an error message about not being in outer par mode.

`\PreviewOpen`
`\PreviewClose`

Those Macros form a matched preview pair. This is for macros that behave similar as `\begin` and `\end` of an environment. It is essential for the operation of `\PreviewOpen` that the macro treated with it will open an additional group even when the preview falls inside of another preview or inside of a `nopreview` environment. Similarly, the macro treated with `\PreviewClose` will close an environment even when inactive.

`\ifPreview`

In case you need to know whether `preview` is active, you can use the conditional `\ifPreview` together with `\else` and `\fi`.

6.2 The Emacs interface

You can use *M-x customize-group* RET *preview-latex* RET in order to customize these variables, or use the menus for it. We explain the various available options together with explaining how they work together in making preview-latex work as intended.

`preview-LaTeX-command`

When you generate previews on a buffer or a region, the command in `preview-LaTeX-command` gets run (that variable should only be changed with Customize since its structure is somewhat peculiar, though expressive). As usual with AUCTEX, you can continue working while this is going on. It is not a good idea to change the file until after preview-latex has established where to place the previews which it can only do after the LATEX run completes. This run produces a host of pseudo-error messages that get parsed by preview-latex at the end of the LATEX run and give it the necessary information about where in the source file the LATEX code for the various previews is located exactly. The parsing takes a moment and will render Emacs busy.

`preview-LaTeX-command-replacements`

This variable specifies transformations to be used before calling the configured command. One possibility is to have '`\pdfoutput=0` ' appended to every command starting with '`pdf`'. This particular setting is available as the shortcut `preview-LaTeX-disable-pdfoutput`. Since preview-latex can work with PDF files by now, there is little incentive for using this option, anymore (for projects not requiring PDF output, the added speed of `dvipng` might make this somewhat attractive).

`preview-required-option-list`

`preview-LaTeX-command` uses `preview-required-option-list` in order to pass options such as `auctex`, `active` and `dvips` to the `preview` package. This means that the user need (and should) not supply these in the document itself in case he wants to be able to still compile his document without it turning into an incoherent mass of little pictures. These options even get passed in when the user loads `preview` explicitly in his document.

The default includes an option `counters` that is controlled by the boolean variable

`preview-preserve-counters`

This option will cause the `preview` package to emit information that will assist in keeping things like equation counters and section numbers reasonably correct even when you are regenerating only single previews.

`preview-default-option-list`
`preview-default-preamble`

If the document does not call in the package `preview` itself (via `\usepackage`) in the preamble, the preview package is loaded using default options

from `preview-default-option-list` and additional commands specified in `preview-default-preamble`.

`preview-fast-conversion`
: This is relevant only for DVI mode. It defaults to 'On' and results in the whole document being processed as one large PostScript file from which the single images are extracted with the help of parsing the PostScript for use of so-called DSC comments. The bounding boxes are extracted with the help of TEX instead of getting them from Dvips. If you are experiencing bounding box problems, try setting this option to 'Off'.

`preview-prefer-TeX-bb`
: If this option is 'On', it tells preview-latex never to try to extract bounding boxes from the bounding box comments of EPS files, but rather rely on the boxes it gets from TEX. If you activated `preview-fast-conversion`, this is done, anyhow, since there are no EPS files from which to read this information. The option defaults to 'Off', simply because about the only conceivable reason to switch off `preview-fast-conversion` would be that you have some bounding box problem and want to get Dvips' angle on that matter.

`preview-scale-function`
`preview-reference-face`
`preview-document-pt-list`
`preview-default-document-pt`
: `preview-scale-function` determines by what factor images should be scaled when appearing on the screen. If you specify a numerical value here, the physical size on the screen will be that of the original paper output scaled by the specified factor, at least if Emacs' information about screen size and resolution are correct. The default is to let `preview-scale-from-face` determine the scale function. This function determines the scale factor by making the size of the default font in the document match that of the on-screen fonts.

  The size of the screen fonts is deduced from the font `preview-reference-face` (usually the default face used for display), the size of the default font for the document is determined by calling `preview-document-pt`.

  This function consults the members of `preview-document-pt-list` in turn until it gets the desired information. The default consults first `preview-parsed-font-size`,

  then calls `preview-auctex-font-size`

  which asks AUCTEX about any size specification like `12pt` to the documentclass that it might have detected when parsing the document, and finally reverts to just assuming `preview-default-document-pt` as the size used in the document (defaulting to 10pt).

  If you find that the size of previews and the other Emacs display clashes, something goes wrong. `preview-parsed-font-size` is determined at `\begin{document}` time; if the default font size changes after that, it will not get reported. If you have an outdated version of `preview.sty` in your path, the size might not be reported at all. If in this case AUCTEX is unable to find a size specification, and if you are using a document class with a different

default value (like 'KomaScript'), the default fallback assumption will probably be wrong and preview-latex will scale up things too large. So better specify those size options even when you know that LaTeX does not need them: preview-latex might benefit from them. Another possibility for error is that you have not enabled AUCTeX's document parsing options. The fallback method of asking AUCTeX about the size might be disabled in future versions of preview-latex since in general it is more reliable to get this information from the LaTeX run itself.

`preview-fast-dvips-command`
`preview-dvips-command`

The regular command for turning a DVI file into a single PostScript file is `preview-fast-dvips-command`, while `preview-dvips-command` is used for cranking out a DVI file where every preview is in a separate EPS file. Which of the two commands gets used depends on the setting of `preview-fast-conversion`. The printer specified here is `-Pwww` by default, which will usually get you scalable fonts where available. If you are experiencing problems, you might want to try playing around with Dvips options (See Section "Command-line options" in `dvips`).

The conversion of the previews into PostScript or EPS files gets started after the LaTeX run completes when Emacs recognizes the first image while parsing the error messages. When Emacs has finished parsing the error messages, it activates all detected previews. This entails throwing away any previous previews covering the same areas, and then replacing the text in its visual appearance by a placeholder looking like a roadworks sign.

`preview-nonready-icon-specs`

This is the roadworks sign displayed while previews are being prepared. You may want to customize the font sizes at which preview-latex switches over between different icon sizes, and the ascent ratio which determines how high above the base line the icon gets placed.

`preview-error-icon-specs`
`preview-icon-specs`

Those are icons placed before the source code of an opened preview and, respectively, the image specs to be used for PostScript errors, and a normal open preview in text representation.

`preview-inner-environments`

This is a list of environments that are regarded as inner levels of an outer environment when doing `preview-environment`. One example when this is needed is in `\begin{equation}\begin{split}...\end{split}\end{equation}`, and accordingly `split` is one entry in `preview-inner-environments`.

6.3 The preview images

`preview-image-type`
`preview-image-creators`
`preview-gs-image-type-alist`

What happens when LaTeX is finished depends on the configuration of `preview-image-type`. What to do for each of the various settings is specified in the variable `preview-image-creators`. The options to pass into Ghostscript and what Emacs image type to use is specified in `preview-gs-image-type-alist`.

`preview-image-type` defaults to `png`. For this to work, your version of Ghostscript needs to support the `png16m` device. If you are experiencing problems here, you might want to reconfigure `preview-gs-image-type-alist` or `preview-image-type`. Reconfiguring `preview-image-creators` is only necessary for adding additional image types.

Most devices make preview-latex start up a single Ghostscript process for the entire preview run (as opposed to one per image) and feed it either sections of a PDF file (if PDFLaTeX was used), or (after running Dvips) sections of a single PostScript file or separate EPS files in sequence for conversion into PNG format which can be displayed much faster by Emacs. Actually, not in sequence but backwards since you are most likely editing at the end of the document. And as an added convenience, any preview that happens to be on-screen is given higher priority so that preview-latex will first cater for the images that are displayed. There are various options customizable concerning aspects of that operation, see the customization group '`Preview Gs`' for this.

Another noteworthy setting of `preview-image-type` is '`dvipng`': in this case, the `dvipng`

program will get run on DVI output (see below for PDF). This is in general much faster than Dvips and Ghostscript. In that case, the option

`preview-dvipng-command`

will get run for doing the conversion, and it is expected that

`preview-dvipng-image-type`

images get produced ('`dvipng`' might be configured for other image types as well). You will notice that `preview-gs-image-type-alist` contains an entry for `dvipng`: this actually has nothing to with '`dvipng`' itself but specifies the image type and Ghostscript device option to use when '`dvipng`' can't be used. This will obviously be the case for PDF output by PDFLaTeX, but it will also happen if the DVI file contains PostScript specials in which case the affected images will get run through Dvips and Ghostscript once '`dvipng`' finishes.

Note for pLaTeX and upLaTeX users: It is known that `dvipng` is not compatible with pLaTeX and upLaTeX. If `preview-image-type` is set to '`dvipng`' and (u)pLaTeX is used, '`dvipng`' just fails and preview-latex falls back on Dvips and Ghostscript.

`preview-gs-options`

Most interesting to the user perhaps is the setting of this variable. It contains the default antialiasing settings `-dTextAlphaBits=4` and

`-dGraphicsAlphaBits=4`. Decreasing those values to 2 or 1 might increase Ghostscript's performance if you find it lacking.

Running and feeding Ghostscript from preview-latex happens asynchronously again: you can resume editing while the images arrive. While those pretty pictures filling in the blanks on screen tend to make one marvel instead of work, rendering the non-displayed images afterwards will not take away your attention and will eventually guarantee that jumping around in the document will encounter only prerendered images.

6.4 Misplaced previews

If you are reading this section, the first thing is to check that your problem is not caused by x-symbol in connection with an installation not supporting 8-bit characters (see Section 5.3 [x-symbol interoperation], page 14). If not, here's the beef:

As explained previously, Emacs uses pseudo-error messages generated by the '`preview`' package in order to pinpoint the exact source location where a preview originated. This works in running text, but fails when preview material happens to lie in macro arguments, like the contents of `\emph`. Those macros first read in their entire argument, munge it through, perhaps transform it somehow, process it and perhaps then typeset something. When they finally typeset something, where is the location where the stuff originated? TEX, having read in the entire argument before, does not know and actually there would be no sane way of defining it.

For previews contained inside such a macro argument, the default behaviour of preview-latex is to use a position immediately after the closing brace of the argument. All the previews get placed there, all at a zero-width position, which means that Emacs displays it in an order that preview-latex cannot influence (currently in Emacs it is even possible that the order changes between runs). And since the placement of those previews is goofed up, you will not be able to regenerate them by clicking on them. The default behaviour is thus somewhat undesirable.

The solution (like with other preview problems) is to tell the LATEX '`preview`' package how to tackle this problem (see Section 6.1 [The LaTeX style file], page 16). Simply, you don't need `\emph` do anything at all during previews! You only want the text math previewed, so the solution is to use `\PreviewMacro*\emph` in the preamble of your document which will make LATEX ignore `\emph` completely as long as it is not part of a larger preview (in which case it gets typeset as usual). Its argument thus becomes ordinary text and gets treated like ordinary text.

Note that it would be a bad idea to declare `\PreviewMacro*[{{}}]\emph` since then both `\emph` as well as its argument would be ignored instead of previewed. For user-level macros, this is almost never wanted, but there may be internal macros where you might want to ignore internal arguments.

The same mechanism can be used for a number of other text-formatting commands like `\textrm`, `\textit` and the like. While they all use the same internal macro `\text@command`, it will not do to redefine just that, since they call it only after having read their argument in, and then it already is too late. So you need to disable every of those commands by hand in your document preamble.

Actually, we wrote all of the above just to scare you. At least all of the above mentioned macros and a few more are already catered for by a configuration file `prauctex.cfg` that

gets loaded by default unless the ‘`preview`’ package gets loaded with the `noconfig` option. You can make your own copy of this file in a local directory and edit it in case of need. You can also add loading of a file of your liking to `preview-default-preamble`,

or alternatively do the manual disabling of your favorite macro in `preview-default-preamble`,

which is customizable in the ‘`Preview Latex`’ group.

Appendix A ToDo

- Support other formats than just LaTeX

  plain TeX users and ConTeXt users should not have to feel left out. While ConTeXt is not supported yet by released versions of AUCTeX, at least supporting plain would help people, and be a start for ConTeXt as well. There are plain-based formats like MusiXTeX that could benefit a lot from preview-latex. The main part of the difficulties here is to adapt `preview.dtx` to produce stuff not requiring LaTeX.

- Support nested snippets

  Currently you can't have both a footnote (which gets displayed as just its footnote number) and math inside of a footnote rendered as an image: such nesting might be achieved by rerunning preview-latex on the footnote contents when one opens the footnote for editing.

- Support other text properties than just images

  Macros like '`\textit`' can be rendered as images, but the resulting humungous blob is not suitable for editing, in particular since the line filling from LaTeX does not coincide with that of Emacs. It would be much more useful if text properties just switched the relevant font to italics rather than replacing the whole text with an image. It would also make editing quite easier. Then there are things like footnotes that are currently just replaced by their footnote number. While editing is not a concern here (the number is not in the original text, anyway), it would save a lot of conversion time if no images were generated, but Emacs just displayed a properly fontified version of the footnote number. Also, this might make preview-latex useful even on text terminals.

- Find a way to facilitate Source Specials

  Probably in connection with adding appropriate support to `dvipng`, it would be nice if clicking on an image from a larger piece of source code would place the cursor at the respective source code location.

- Make `preview.dtx` look reasonable in AUCTeX

  It is a bit embarrassing that `preview.dtx` is written in a manner that will not give either good syntax highlighting or good indentation when employing AUCTeX.

- Web page work

  Currently, preview-latex's web page is not structured at all. Better navigation would be desirable, as well as separate News and Errata eye catchers.

- Manual improvements
  - Pepper the manual with screen shots and graphics

    This will be of interest for the HTML and TeX renditions of the texinfo manual. Since Texinfo now supports images as well, this could well be nice to have.
  - Fix duplicates

    Various stuff appears several times.

- Implement rendering pipelines for Emacs

  The current preview-latex interface is fundamentally flawed, not only because of a broken implementation. A general batchable and daemonizable rendering infrastructure that can work on all kinds of preview images for embedding into buffers is warranted. The

current implementation has a rather adhoc flavor and is not easily extended. It will not work outside of AUCTEX, either.

- Integrate into RefTEX

  When referencing to equations and the like, the preview-images of the source rather than plain text should be displayed. If the preview in question covers labels, those should appear in the bubble help and/or a context menu. Apropos:

- Implement LATEX error indicators

  Previews on erroneous LATEX passages might gain a red border or similar.

- Pop up relevant online documentation for frequent errors

  A lot of errors are of the "badly configured" variety. Perhaps the relevant info pages should be delivered in addition to the error message.

- Implement a table editing mode where every table cell gets output as a separate preview. Alternatively, output the complete table metrics in a way that lets people click on individual cells for editing purposes.

- Benchmark and kill Emacs inefficiencies

  Both the LATEX run under Emacs control as well as actual image insertion in Emacs could be faster. CVS Emacs has improved in that respect, but it still is slower than desirable.

- Improve image support under Emacs

  The general image and color handling in Emacs is inefficient and partly defective. This is still the case in CVS. One option would be to replace the whole color and image handling with GDK routines when this library is available, since it has been optimized for it.

Appendix B Frequently Asked Questions

B.1 Introduction

B.1.1 How can I contribute to the FAQ?

Send an email with the subject:

```
Preview FAQ
```

to `auctex-devel@gnu.org`.

B.2 Requirements

B.2.1 Which version of Emacs is needed?

preview-latex nominally requires GNU Emacs with a version of at least 26.1.

B.2.2 Which versions of Ghostscript and AUCTEX are needed?

We recommend to use GNU or AFPL Ghostscript with a version of at least 7.07.

preview-latex has been distributed as part of AUCTEX since version 11.80. If your version of AUCTEX is older than that, or if it does not contain a working copy of preview-latex, complain to wherever you got it from.

B.2.3 I have trouble with the display format...

We recommend keeping the variable `preview-image-type` set to `dvipng` (if you have it installed) or `png`. This is the default and can be set via the '`Preview/Customize`' menu.

All other formats are known to have inconveniences, either in file size or quality. There are some Emacs versions around not supporting PNG; the proper way to deal with that is to complain to your Emacs provider. Short of that, checking out PNM or JPEG formats might be a good way to find out whether the lack of PNG format support might be the only problem with your Emacs.

B.2.4 For which OS does preview work?

It is known to work under the X Window System for Linux and for several flavors of Unix: we have reports for HP and Solaris.

There are several development versions of Emacs around for native MacOS Carbon, and preview-latex is working with them, too.

With Windows, both native Emacs and Cygwin Emacs should work. However, it is known that MiKTEX (`https://miktex.org/`) sometimes doesn't work with preview-latex. In that case, use TEX Live (`https://tug.org/texlive/`) instead.

B.3 Installation Trouble

B.3.1 I just get 'LaTeX found no preview images'.

The reason for this is that LaTeX found no preview images in the document in question.

One reason might be that there are no previews to be seen. If you have not used preview-latex before, you might not know its manner of operation. One sure-fire way to test if you just have a document where no previews are to be found is to use the provided example document `circ.tex` (you will have to copy it to some directory where you have write permissions). If the symptom persists, you have a problem, and the problem is most likely a LaTeX problem. Here are possible reasons:

Filename database not updated
: Various TeX distributions have their own ways of knowing where the files are without actually searching directories. The normal preview-latex installation should detect common tools for that purpose and use them. If this goes wrong, or if the files get installed into a place where they are not looked for, the LaTeX run will fail.

An incomplete manual installation
: This should not happen if you followed installation instructions. Unfortunately, people know better all the time. If only `preview.sty` gets installed without a set of supplementary files also in the `latex` subdirectory, preview-latex runs will not generate any errors, but they will not produce any previews, either.

An outdated `preview` installation
: The `preview.sty` package is useful for more than just preview-latex. For example, it is part of TeX Live. So you have to make sure that preview-latex does not get to work with outdated style and configuration files: some newer features will not work with older TeX style files, and really old files will make preview-latex fail completely. There usual is a local `texmf` tree, or even a user-specific tree that are searched before the default tree. Make sure that the first version of those files that gets found is the correct one.

B.4 Customization

B.4.1 How to include additional environments like `enumerate`

By default, preview-latex is intended mainly for displaying mathematical formulas, so environments like `enumerate` or `tabular` (except where contained in a float) are not included. You can include them however manually by adding the lines:

```
\usepackage[displaymath,textmath,sections,graphics,floats]{preview}
\PreviewEnvironment{enumerate}
```

in your document header, that is before

```
\begin{document}
```

In general, `preview` should be loaded as the last thing before the start of document.

Be aware that

```
\PreviewEnvironment{...}
```

does not accept a comma separated list! Also note that by putting more and more

```
\PreviewEnvironment{...}
```

in your document, it will look more and more like a DVI file preview when running preview-latex. Since each preview is treated as one large monolithic block by Emacs, one should really restrict previews to those elements where the improvement in visual representation more than makes up for the decreased editability.

B.4.2 What if I don't want to change the document?

The easiest way is to generate a configuration file in the current directory. You can basically either create `prdefault.cfg` which is used for any use of the '`preview`' package, or you can use `prauctex.cfg` which only applies to the use from with Emacs. Let us assume you use the latter. In that case you should write something like

```
\InputIfFileExists{preview/prauctex.cfg}{}{}
\PreviewEnvironment{enumerate}
```

in it. The first line inputs the system-wide default configuration (the file name should match that, but not your own `prauctex.cfg`), then you add your own stuff.

B.4.3 Suddenly I get gazillions of ridiculous pages?!?

When preview-latex works on extracting its stuff, it typesets each single preview on a page of its own. This only happens when actual previews get generated. Now if you want to configure preview-latex in your document, you need to add your own `\usepackage` call to '`preview`' so that it will be able to interpret its various definition commands. It is an error to add the `active` option to this invocation: you don't want the package to be active unless preview-latex itself enables the previewing operation (which it will).

B.4.4 Does preview-latex work with presentation classes?

preview-latex should work with most presentation classes. However, since those classes often have macros or pseudo environments encompassing a complete slide, you will need to use the customization facilities of `preview.sty` to tell it how to resolve this, whether you want no previews, previews of whole slides or previews of inner material.

B.5 Troubleshooting

B.5.1 Preview causes all sort of strange error messages

When running preview-latex and taking a look at either log file or terminal output, lots of messages like

```
! Preview: Snippet 3 started.
<-><->

l.52 \item Sie lassen sich als Funktion $
                                        y = f(x)$ darstellen.
! Preview: Snippet 3 ended.(491520+163840x2494310).
<-><->

l.52 \item Sie lassen sich als Funktion $y = f(x)$
                                                  darstellen.
```

appear (previous versions generated messages looking even more like errors). Those are not real errors (as will be noted in the log file). Or rather, while they **are** really TEX

error messages, they are intentional. This currently is the only reliable way to pass the information from the LaTeX run of preview-latex to its Emacs part about where the previews originated in the source text. Since they are actual errors, you will also get AUCTeX to state

```
Preview-LaTeX exited as expected with code 1 at Wed Sep  4 17:03:30
```

after the LaTeX run in the run buffer. This merely indicates that errors were present, and errors will always be present when preview-latex is operating. There might be also real errors, so in case of doubt, look for them explicitly in either run buffer or the resulting `.log` file.

B.5.2 Why do my DVI and PDF output files vanish?

In order to produce the preview images preview-latex runs LaTeX on the master or region file. The resulting DVI or PDF file can happen to have the same name as the output file of a regular LaTeX run. So the regular output file gets overwritten and is subsequently deleted by preview-latex.

B.5.3 My output file suddenly only contains preview images?!

As mentioned in the previews FAQ entry, preview-latex might use the file name of the original output file for the creation of preview images. If the original output file is being displayed with a viewer when this happens, you might see strange effects depending on the viewer, e.g. a message about the file being corrupted or the display of all the preview images instead of your typeset document. (Also see Section B.4 [Customization], page 32.)

B.6 preview-latex when not using LaTeX

B.6.1 Does preview-latex work with PDFLaTeX?

Yes, as long as you use AUCTeX's own PDFLaTeX mode and have not messed with '`TeX-command-list`'.

B.6.2 Does preview-latex work with 'elatex'?

No problem here. If you configure your AUCTeX to use '`elatex`', or simply have '`latex`' point to '`elatex`', this will work fine. Modern TeX distributions use eTeX for LaTeX, anyway.

B.6.3 Does preview-latex work with ConTeXt?

In short, no. The '`preview`' package is LaTeX-dependent. Adding support for other formats requires volunteers.

B.6.4 Does preview-latex work with plain TeX?

Again, no. Restructuring the '`preview`' package for '`plain`' operation would be required. Volunteers welcome.

In some cases you might get around by making a wrapper pseudo-Master file looking like the following:

```
\documentclass{article}
\usepackage{plain}
```

```
\begin{document}
\begin{plain}
\input myplainfile
\end{plain}
\end{document}
```

Appendix C Copying this Manual

The full license text can be read here:

C.1 GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related

matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties:

any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

   You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

   You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing

distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted

document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have

been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

R

S

U

W