preview-latex

A LaTeX preview mode for AUCTeX in Emacs.
Version 11.82

by David Kastrup, Alan Shutko,
Jan-Åke Larsson and Nick Alcock.

Table of Contents

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). In XEmacs, and in any prepackaged versions worth their salt, activation should be automatic upon installation. If this seems not the case, complain to your installation provider.

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. If you are using XEmacs, you will probably need to upgrade the package to the newest one; things are being fixed just as I am writing this.

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-variable 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 (`http://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.dvi`' or '`preview-latex.ps`'.

Detailed documentation for the LaTeX style used for extracting the preview images is placed in '`preview.dvi`' in a suitable directory during installation; on typical teTeX-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 (`http://savannah.gnu.org/projects/auctex`). You can get its files from the AUCTeX download area (`ftp://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.

You will also find '`.rpm`' files there for Fedora and possibly SuSE. Anonymous CVS is available as well.

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 `http://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 Info file '`auctex`', node '`Key Index`'). 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` or `zmacs-regions`), 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 *M-m P*, or *C-c C-m P* in Emacs 22) 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.

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 10.

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 13), 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. Its default is different in Emacs 21 and Emacs 22, and it is not available in XEmacs.

- Showing `\labels`

  When using preview-latex, the `\labels` 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-options-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 17) *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 13).

  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 Problems with Ghostscript

Most of the problems encountered come from interaction with Ghostscript. It is a good idea to have a fairly recent version of Ghostscript installed. One problem occurs if you have specified the wrong executable under Windows: the command line version of Ghostscript is called '`GSWIN32C.EXE`', not '`GSWIN32.EXE`'.

When Ghostscript fails, the necessary information and messages from Ghostscript go somewhere. If Ghostscript fails before starting to process images, you'll find the information at the end of the process buffer you can see with *C-c C-l*. If Ghostscript fails while processing a particular image, this image will be tagged with clickable buttons for the error description and for the corresponding source file.

The default options configurable with

> *M-x customize-variable* ⟨RET⟩ *preview-gs-options* ⟨RET⟩

include the options '`-dTextAlphaBits=4`' and '`-dGraphicsAlphaBits=4`'. These options have been reported to make Ghostscript 5.50 fail, but should work under Ghostscript 6.51 and later. If you are experiencing problems, it might help to customize them away. Of course, this also takes away the joy of antialiasing, so upgrading Ghostscript might not be the worst idea after all.

The device names have changed over time, so when using an old Ghostscript, you may have problems with the devices demanded by the customizable variable `preview-image-creators`. In that case, make sure they fit your version of Ghostscript, at least the entry corresponding to the current value of `preview-image-type`. While not being best in file size and image quality, setting `preview-image-creators` to `jpeg` should probably be one of the best bets for the purpose of checking basic operation, since that device name has not changed in quite some time. But JPEG is not intended for text, but for photographic images. On a more permanent time scale, the best choice is to use PNG and complain to your suppliers if either Emacs or Ghostscript fail to properly accommodate this format.

5.2 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-variable* ⟨RET⟩ *preview-fast-dvips-command* ⟨RET⟩

and

M-x customize-variable ⟨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).

5.3 Emacs problems

- GNU Emacs versions

  Don't use Emacsen older than 21.3 on X11-based systems. On most other systems, you'll need at least Emacs 22.1 or one of the developer versions leading up to it. Details can be found in section "Prerequisites" in *the AUCTeX manual*.

- Emacsen on Windows operating systems

  For Emacs 21, no image support is available in Emacs under Windows. Without images, preview-latex is useless. The current CVS version of Emacs available from http://savannah.gnu.org/projects/emacs now supports images including the PNG format, so Emacs 22 should work out of the box once it is released. Precompiled versions are available from http://crasseux.com/emacs and http://nqmacs.sf.net.

  For detailed installation instructions for Windows, see section "Installation under MS Windows" in *the AUCTeX manual*.

- XEmacs

  There is are two larger problems known with older XEmacs releases. One leads to seriously mispositioned baselines and previews hanging far above other text on the same line. This should be fixed as of XEmacs-21.4.9.

  The other core bug causes a huge delay when XEmacs's idea of the state of processes (like ghostscript) is wrong, and can lead to nasty spurious error messages. It should be fixed in version 21.4.8.

  Previews will only remain from one session to the next if you have version 1.81 or above of the 'edit-utils' package, first released in the 2002-03-12 sumo tarball.

5.4 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 19). 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 13), 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.

5.5 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`.

5.6 Middle-clicks paste instead of toggling

This is probably the fault of your favorite package. '`flyspell.el`' and '`mouse-drag.el`' are known to be affected in versions before Emacs 21.3. Upgrade to the most recent version. What version of XEmacs might contain the fixes is unknown.

'`isearch.el`' also shows this effect 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.

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 PDFTeXfor 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 ps4pdf 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.

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

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 by default 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 Info file '`dvips`', node '`Command-line options`').

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`.

`preview-use-balloon-help`
: If you turn this XEmacs-only option 'on', then moving the mouse over previews and icons will show appropriate help texts. This works by switching on `balloon-help-mode` in the buffer if it is not already enabled. The default now is 'off' since some users reported problems with their version of XEmacs. GNU Emacs has its corresponding `tooltip-mode` enabled by default and in usable condition.

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 `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`'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.

`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 5.5 [x-symbol interoperation], page 12). 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 13). 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 '`gs.el`' 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 (X)Emacs is needed?

See also the table at the end of the section.

preview-latex nominally requires GNU Emacs with a version of at least 21.1. However, Emacs 22 (currently under development) offers superior performance and wider platform support, and is even now the recommended platform to use.

While recent versions of XEmacs 21.4 are supported, doing this in a satisfactory manner has proven to be difficult due to technical shortcomings and differing API's which are hard to come by. If preview-latex is an important part of your editing workflow, you are likely to get better results and support by switching to Emacs. Of course, you can improve support for your favorite editor by giving feedback in case you encounter bugs.

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.

Proper PDFLaTeX support will require AUCTeX version 11.50 or later. Versions before that are not supported.

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 idea to deal with that is to complain to your Emacs providers. 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.

Under Windows, you should try the most recent versions of preview-latex since a lot of typical Windows problems have been ironed out lately. Under XEmacs, both Cygwin and native ports should work. Image support for GNU Emacs under Windows is expected with

Emacs 22 which has not yet been released at the time of this writing. CVS versions of it already work, however.

The entry "X11/Unix" currently means Linux, Solaris or HP/UX, as well as the X-specific version for Mac/OSX.

OS	Emacs version	XEmacs version
X11/Unix	21.1	21.4.9
Win9x cygwin	21.4?	21.4.8
Win9x native	21.4?	21.4.8

With display errors, XEmacs versions as early as 21.1.14 might also work.

B.3 Installation Trouble

B.3.1 I could not install the precompiled RPM binaries

Note that the binaries require RPM version 3.

B.3.2 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:

Incompatible RPM installation

The RPM packages are intended to run on a Redhat system. So the TeX files, provided by package '`preview-latex-common`' go into '`/usr/share/texmf/tex/latex/preview`' and '`/usr/share/texmf/doc/latex/styles/preview`'. If for your system the TeX files are in different places you have to set appropriate links.

Another possibility is to get the tar archive, edit the '`preview-latex.spec`' accordingly, repack and then use the appropriate '`rpm`' command for building RPMs from a tar archive. If this works, please don't forget to send us the spec file, so that we may be able to help others with your platform.

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 TeXlive. 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.3.3 I have problems with the XEmacs installation

Please note that the XEmacs installation is different, since XEmacs has a package system that gets used here. Please make sure that you read and follow the installation instructions for XEmacs.

The XEmacs Lisp files provided by the '`preview-latex-xemacs`' RPM package are in XEmacs package format and will be installed right into the XEmacs package tree. The location is detected when the RPM file is built. If it is wrong for your system, doing an

```
rpmbuild --rebuild preview-latex-0.9.1-1.fedora.src.rpm
```

should do the trick and should generate RPMs for your system (even if that System happens to be SuSE rather than Fedora).

Alternatively you could install preview-latex manually in your home directory either in '`~/.xemacs`' (for XEmacs version below 21.4.x) or in '`~/.xemacs/xemacs-packages`' (for versions starting with 21.4).

B.3.4 After installation of the XEmacs RPM package, AUCTeX does not work

Most likely you installed the RPM files and you have an XEmacs version which uses a different architecture of directories, (SuSE is known to have a no standard architecture, besides it changes it from SuSE version to SuSE version). Your init file will contain an invocation of preview-latex according to the installation instructions in Chapter 2 [Installation], page 4.

In consequence, the attempt to load preview-latex when entering AUCTeX mode fails because of preview-latex being in a different directory, and this AUCTeX's actication is aborted. Please notice that the installation instructions for Emacs and XEmacs differ.

B.4 Customization

B.4.1 Why don't I get balloon help like in the screen shots?

Some users have reported problems with their XEmacs version, so balloon help is no longer switched on by default. Use the Preview/Customize menu or ⟨M-x⟩ *`customize-variable`* in order to customize `preview-use-balloon-help` to 'On'. This only concerns XEmacs: tooltips under GNU Emacs are enabled by default and unproblematic.

B.4.2 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.4.3 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.4.4 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.4.5 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.6 preview-latex when not using LaTeX

B.6.1 Does preview-latex work with PDFLaTeX?

Yes. PDFLaTeX works fine with AUCTeX version 11.50 or later. For earlier versions, you need to have `LaTeX-command-style` configured manually to call PDFLaTeX.

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.

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 License

(This text is stolen from the Texinfo manual, Edition 4.0).

The programs currently being distributed that relate to preview-latex include lisp files for Emacs and style files for LaTeX. These programs are *free*; this means that everyone is free to use them and free to redistribute them on a free basis. The preview-latex related programs are not in the public domain; they are copyrighted and there are restrictions on their distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of these programs that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of the programs that relate to preview-latex, that you receive source code or else can get it if you want it, that you can change these programs or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of the preview-latex related programs, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the programs that relate to preview-latex. If these programs are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions of the licenses for the programs currently being distributed that relate to preview-latex are found in the General Public Licenses that accompany them.

Index

R

S

U

W