AUC TeX

A much enhanced LaTeX mode for GNU Emacs.
Almost Version 12

by Kresten Krab Thorup
updated for 6.1 to 11 by Per Abrahamsen
updated for almost 12 by David Kastrup

Copying

(This text stolen from the TEXinfo 2.16 distribution).

The programs currently being distributed that relate to AUC TEX include lisp files for GNU Emacs. These programs are *free*; this means that everyone is free to use them and free to redistribute them on a free basis. The AUC TEX 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 AUC TEX, 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 AUC TEX 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 AUC TEX. 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 AUC TEX are found in the General Public Licenses that accompany them.

1 Introduction to AUC TeX

This section of the AUC TEX manual gives a brief overview of what AUC TEX is, and the section is also available as a '`README`' file. It is **not** an attempt to document AUC TEX. Real documentation for AUC TEX is available in the rest of the manual, which you can find in the '`doc`' directory.

Read the '`INSTALLATION`' file for information about how to install AUC TEX. It is identical to the Installation chapter in the AUC TEX manual.

If you are upgrading from the previous version of AUC TEX, the latest changes can be found in the '`CHANGES`' file. If you are upgrading from an older version, read the History chapter in the AUC TEX manual.

AUC TEX is a comprehensive customizable integrated environment for writing input files for LaTEX using GNU Emacs.

AUC TEX lets you run TEX/LaTEX and other LaTEX-related tools, such as a output filters or post processor from inside Emacs. Especially 'running LaTEX' is interesting, as AUC TEX lets you browse through the errors TEX reported, while it moves the cursor directly to the reported error, and displays some documentation for that particular error. This will even work when the document is spread over several files.

AUC TEX automatically indents your 'LaTEX-source', not only as you write it — you can also let it indent and format an entire document. It has a special outline feature, which can greatly help you 'getting an overview' of a document.

Apart from these special features, AUC TEX provides a large range of handy Emacs macros, which in several different ways can help you write your LaTEX documents fast and painlessly.

All features of AUC TEX are documented using the GNU Emacs online documentation system. That is, documentation for any command is just a key click away!

AUC TEX is written entirely in Emacs-Lisp, and hence you can easily add new features for your own needs. It was not made as part of any particular employment or project (apart from the AUC TEX project itself). AUC TEX is distributed under the 'GNU Emacs General Public License' and may therefore almost freely be copied and redistributed.

The next sections are a short introduction to some 'actual' features. For further information, refer to the built-in online documentation of AUC TEX.

1.1 Indentation and formatting

AUC TEX may automatically indent your document as you write it. By pressing ⟨lfd⟩ instead of ⟨ret⟩ at the end of a line, the current line is indented by two spaces according to the current environment level, and the cursor is moved down one line. By pressing ⟨tab⟩, the current line is indented, and the cursor stays where it is. The well-known Emacs feature `format-paragraph` (*M-q*) is reimplemented especially for AUC TEX to follow the indentation. A special command `LaTeX-fill-buffer` lets you indent an entire document like the well-known C utility indent (this time, only according to the LaTEX structure `:-)`).

1.2 Completion

By studying your '`\documentclass`' command (in the top of your document), and consulting a precompiled list of (La)TEX symbols from a large number of TEX and LaTEX files, AUC TEX is aware of the LaTEX commands you should be able to use in this particular document. This 'knowledge' of AUC TEX is used for two purposes.

1. To make you able to 'complete' partly written LaTEX commands. You may e.g. write *`\renew`* and press *M-*⟨tab⟩ (`TeX-complete-symbol`), and then AUC TEX will complete the word '`\renewcommand`' for you. In case of ambiguity it will display a list of possible completions.
2. To aid you inserting environments, that is \begin - \end pairs. This is done by pressing C-c C-e (LaTEX-environment), and you will be prompted for which 'environment' to insert.

1.3 Editing your document

A number of more or less intelligent keyboard macros have been defined to aid you editing your document. The most important are listed below.

`LaTeX-environment`
: (*C-c C-e*) Insert a '`\begin{}`' — '`\end{}`' pair as described above.

`LaTeX-section`
: (*C-c C-s*) Insert one of '`\chapter`', '`\section`', etc.

`TeX-font`
: (*C-c C-f C-r*, *C-c C-f C-i*, *C-c C-f C-b*) Insert one of '`\textrm{ }`'), '`\textit{ \/}`' '`\textbf{ }`' etc.

A number of additional functions are available. But it would be far too much to write about here. Refer to the rest of the AUC TEX documentation for further information.

1.4 Running LaTEX

When invoking one of the commands `TeX-command-master` (*C-c C-c*) or `TeX-command-region` (*C-c C-r*) LaTEX is run on either the entire current document or a given region of it. The Emacs view is split in two, and the output of TEX is printed in the second half of the screen, as you may simultaneously continue editing your document. In case TEX find any errors when processing your input you can call the function `TeX-next-error` (*C-c `*) which will move the cursor to the first given error, and display a short explanatory text along with the message TEX gave. This procedure may be repeated until all errors have been displayed. By pressing *C-c C-w* (`TeX-toggle-debug-boxes`) you can toggle whether the browser also should notify over-full/under-full boxes or not.

Once you've successfully formatted your document, you may preview or print it by invoking `TeX-command-master` again.

1.5 Outlines

Along with AUC TEX comes support for outline mode for Emacs, which lets you browse the sectioning structure of your document, while you will still be able to use the full power of the rest of the AUC TEX functionality.

1.6 Availability

The most recent version is always available at

http://savannah.nongnu.org/download/auctex/stable.pkg/

WWW users may want to check out the AUC TEX page at

http://www.nongnu.org/auctex/

1.7 Contacts

There has been established a mailing list for help, bug reports, feature requests and general discussion about AUC TEX. You're very welcome to join. Traffic average at an article by day, but they come in bursts. If you are only interested in information on updates, you could refer to the newsgroups '`comp.text.tex`' and '`gnu.emacs.sources`'.

If you want to contact the AUC TEX mailing list, send mail to `mailto:auc-tex-subscribe@sunsite.dk` in order to join. Articles should be sent to `mailto:auc-tex@sunsite.dk`.

To contact the current maintainers of AUC TEX directly, email `mailto:auc-tex_mgr@sunsite.dk`.

2 Inserting Frequently Used Commands

The most commonly used commands/macros of AUC TEX are those which simply insert templates for often used TEX and/or LaTEX constructs, like font changes, handling of environments, etc. These features are very simple, and easy to learn, and help you avoid stupid mistakes like mismatched braces, or '`\begin{}`'-'`\end{}`' pairs.

2.1 Insertion of Quotes, Dollars, and Braces

In TEX, literal double quotes '`"like this"`' are seldom used, instead two single quotes are used '`''like this''`'. To help you insert these efficiently, AUC TEX allows you to continue to press `"` to insert two single quotes. To get a literal double quote, press `"` twice.

TeX-insert-quote *count* Command

(`"`) Insert the appropriate quote marks for TeX.

Inserts the value of `TeX-open-quote` (normally '`''`') or `TeX-close-quote` (normally '`''`') depending on the context. With prefix argument, always inserts '`"`' characters.

TeX-open-quote User Option

String inserted by typing `"` to open a quotation.

TeX-close-quote User Option

String inserted by typing `"` to close a quotation.

If you include the style file '`german`' `TeX-open-quote` and `TeX-close-quote` will both be set to '`"`'.

In AUC TEX, dollar signs should match like they do in TEX. This has been partially implemented, we assume dollar signs always match within a paragraph. The first '`$`' you insert in a paragraph will do nothing special. The second '`$`' will match the first. This will be indicated by moving the cursor temporarily over the first dollar sign. If you enter a dollar sign that matches a double dollar sign '`$$`' AUC TEX will automatically insert two dollar signs. If you enter a second dollar sign that matches a single dollar sign, the single dollar sign will automatically be converted to a double dollar sign.

TeX-insert-dollar *arg* Command

(*$*) Insert dollar sign.

Show matching dollar sign if this dollar sign end the TEX math mode. Ensure double dollar signs match up correctly by inserting extra dollar signs when needed.

With optional *arg*, insert that many dollar signs.

To avoid unbalanced braces, it is useful to insert them pairwise. You can do this by typing *`C-c {`*.

TeX-insert-braces Command

(*`C-c {`*) Make a pair of braces and position the cursor to type inside of them.

2.2 Inserting Font Specifiers

Perhaps the most used keyboard commands of AUC TEX are the short-cuts available for easy insertion of font changing macros. They all put the font change inside a TEX group, a practice that help preventing subtle errors. The most significant advantage of using these command instead of typing it in yourself, is that the braces will always match correctly.

If you give an argument (that is, type *C-u*) to the font command, the innermost font will be replaced, i.e. the font in the TEX group around point will be changed. The following table shows the available commands, with ⋆ indicating the position where the text will be inserted.

C-c C-f C-r
: Insert roman \textrm{⋆} text.

C-c C-f C-b
: Insert **bold face** '`\textbf{⋆}`' text.

C-c C-f C-i
: Insert *italics* '`\textit{⋆}`' text.

C-c C-f C-e
: Insert *emphasized* '`\emph{⋆}`' text.

C-c C-f C-s
: Insert *slanted* '`\textsl{⋆}`' text.

C-c C-f C-t
: Insert `typewriter` '`\texttt{⋆}`' text.

C-c C-f C-c
: Insert SMALL CAPS '`\textsc{⋆}`' text.

C-c C-f C-d
: Delete the innermost font specification containing point.

TeX-font *arg* Command

(*C-c C-f*) Insert template for font change command.

If *replace* is not nil, replace current font. *what* determines the font to use, as specified by `TeX-font-list`.

TeX-font-list User Option

List of fonts used by TeX-font.

Each entry is a list with three elements. The first element is the key to activate the font. The second element is the string to insert before point, and the third element is the string to insert after point. An optional fourth element means always replace if not nil.

2.3 Inserting chapters, sections, etc.

Insertion of sectioning macros, that is '`\chapter`', '`\section`', '`\subsection`', etc. and accompanying '`\label`''s may be eased by using *`C-c C-s`*. This command is highly customizable, the following describes the default behavior.

When invoking you will be asked for a section macro to insert. An appropriate default is automatically selected by AUC TeX, that is either: at the top of the document; the top level sectioning for that document style, and any other place: The same as the last occurring sectioning command.

Next, you will be asked for the actual name of that section, and last you will be asked for a label to be associated with that section. The label will be prefixed by the value specified in `LaTeX-section-hook`.

LaTeX-section *arg* Command

(*`C-c C-s`*) Insert a sectioning command.

Determine the type of section to be inserted, by the argument *arg*.

- If *arg* is nil or missing, use the current level.
- If *arg* is a list (selected by C-u), go downward one level.
- If *arg* is negative, go up that many levels.
- If *arg* is positive or zero, use absolute level:
  - 0 : part
  - 1 : chapter
  - 2 : section
  - 3 : subsection
  - 4 : subsubsection
  - 5 : paragraph
  - 6 : subparagraph

The following variables can be set to customize the function.

`LaTeX-section-hook`
: Hooks to be run when inserting a section.

`LaTeX-section-label`
: Prefix to all section references.

The precise behavior of `LaTeX-section` is defined by the contents of `LaTeX-section-hook`.

LaTeX-section-hook User Option

List of hooks to run when a new section is inserted.

The following variables are set before the hooks are run

level Numeric section level, default set by prefix arg to `LaTeX-section`.

name Name of the sectioning command, derived from *level*.

title The title of the section, default to an empty string.

toc Entry for the table of contents list, default nil.

done-mark
Position of point afterwards, default nil meaning after the inserted text.

A number of hooks are already defined. Most likely, you will be able to get the desired functionality by choosing from these hooks.

`LaTeX-section-heading`
Query the user about the name of the sectioning command. Modifies *level* and *name*.

`LaTeX-section-title`
Query the user about the title of the section. Modifies *title*.

`LaTeX-section-toc`
Query the user for the toc entry. Modifies *toc*.

`LaTeX-section-section`
Insert LaTeX section command according to *name*, *title*, and *toc*. If *toc* is nil, no toc entry is inserted. If *toc* or *title* are empty strings, *done-mark* will be placed at the point they should be inserted.

`LaTeX-section-label`
Insert a label after the section command. Controlled by the variable `LaTeX-section-label`.

To get a full featured `LaTeX-section` command, insert

```
(setq LaTeX-section-hook
      '(LaTeX-section-heading
        LaTeX-section-title
        LaTeX-section-toc
        LaTeX-section-section
        LaTeX-section-label))
```

in your '`.emacs`' file.

The behavior of `LaTeX-section-label` is determined by the variable `LaTeX-section-label`.

LaTeX-section-label User Option

Default prefix when asking for a label.

If it is a string, it is used unchanged for all kinds of sections. If it is nil, no label is inserted. If it is a list, the list is searched for a member whose car is equal to the name of the sectioning command being inserted. The cdr is then used as the prefix. If the name is not found, or if the cdr is nil, no label is inserted.

By default, chapters have a prefix of '`cha:`' while sections and subsections have a prefix of '`sec:`'. Labels are not automatically inserted for other types of sections.

2.4 Inserting Environment Templates

A large apparatus is available that supports insertions of environments, that is '`\begin{}`' — '`\end{}`' pairs.

AUC TEX is aware of most of the actual environments available in a specific document. This is achieved by examining your '`\documentclass`' command, and consulting a precompiled list of environments available in a large number of styles.

You insert an environment with *C-c C-e*, and select an environment type. Depending on the environment, AUC TEX may ask more questions about the optional parts of the selected environment type. With *C-u C-c C-e* you will change the current environment.

LaTeX-environment *arg* Command

(*C-c C-e*) AUC TEX will prompt you for an environment to insert. At this prompt, you may press ⟨TAB⟩ or ⟨SPC⟩ to complete a partially written name, and/or to get a list of available environments. After selection of a specific environment AUC TEX may prompt you for further specifications.

If the optional argument *arg* is not-nil (i.e. you have given a prefix argument), the current environment is modified and no new environment is inserted.

As a default selection, AUC TEX will suggest the environment last inserted or, as the first choice the value of the variable `LaTeX-default-environment`.

LaTeX-default-environment User Option

Default environment to insert when invoking '`LaTeX-environment`' first time.

If the document is empty, or the cursor is placed at the top of the document, AUC TEX will default to insert a 'document' environment.

Most of these are described further in the following sections, and you may easily specify more, as described in 'Customizing environments'.

You can close the current environment with *C-c]*, but we suggest that you use *C-c C-e* to insert complete environments instead.

LaTeX-close-environment Command

(*C-c]*) Insert an '`\end`' that matches the current environment.

2.4.1 Equations

When inserting equation-like environments, the '`\label`' will have a default prefix, which is controlled by the following variables:

LaTeX-equation-label User Option

Prefix to use for 'equation' labels.

LaTeX-eqnarray-label User Option

Prefix to use for 'eqnarray' labels.

LaTeX-amsmath-label User Option

Prefix to use for amsmath equation labels. Amsmath equations include '`align`', '`alignat`', '`xalignat`', '`aligned`', '`flalign`' and '`gather`'.

2.4.2 Floats

Figures and tables (i.e., floats) may also be inserted using AUC TEX. After choosing either 'figure' or 'table' in the environment list described above, you will be prompted for a number of additional things.

float-to This field is the option of float environments that controls how they are placed in the final document. In LaTEX this is a sequence of the letters '`htbp`' as described in the LaTEX manual. The value will default to the value of `LaTeX-float`.

caption This is the caption of the float.

label The label of this float. The label will have a default prefix, which is controlled by the variables `LaTeX-figure-label` and `LaTeX-table-label`.

Moreover, in the case of a 'figure' environment, you will be asked if you want to insert a 'center' environment inside the figure.

LaTeX-float User Option

Default placement for floats.

LaTeX-figure-label User Option

Prefix to use for figure labels.

LaTeX-table-label User Option

Prefix to use for table labels.

2.4.3 Itemize-like

In an itemize-like environment, nodes (i.e., '`\item`'s) may be inserted using *C-c* ⟨LFD⟩.

LaTeX-insert-item Command

(*C-c* ⟨LFD⟩) Close the current item, move to the next line and insert an appropriate '`\item`' for the current environment. That is, 'itemize' and 'enumerate' will have '`\item` ' inserted, while 'description' will have '`\item[]`' inserted.

2.4.4 Tabular-like

When inserting Tabular-like environments, that is, 'tabular' 'array' etc., you will be prompted for a template for that environment.

2.4.5 Customizing environments

See Section 9.3 [Adding Environments], page 31, for how to customize the list of known environments.

3 Advanced Editing Features

The previous chapter described how to write the main body of the text easily and with a minimum of errors. In this chapter we will describe some features for entering more specialized sorts of text, and for indenting and navigating through the document.

3.1 Entering Mathematics

TeX is written by a mathematician, and has always contained good support for formatting mathematical text. AUC TeX supports this tradition, by offering a special minor mode for entering text with many mathematical symbols. You can enter this mode by typing `C-c ~`.

LaTeX-math-mode Command

(`C-c ~`) Toggle LaTeX-math-mode. This is a minor mode rebinding the key `LaTeX-math-abbrev-prefix` to allow easy typing of mathematical symbols. ` will read a character from the keyboard, and insert the symbol as specified in `LaTeX-math-list`. If given a prefix argument, the symbol will be surrounded by dollar signs.

You can use another prefix key (instead of `) by setting the variable LaTeX-math-abbrev-prefix.

LaTeX-math-abbrev-prefix User Option

A string containing the prefix of `LaTeX-math-mode` commands; This value defaults to `.

The variable `LaTeX-math-list` holds the actual mapping.

LaTeX-math-list User Option

A list containing key command mappings to use in `LaTeX-math-mode`. The car of each element is the key and the cdr is the macro name.

The AUC TeX distributions includes a reference card for `LaTeX-math-mode` with a list of all math mode commands.

3.2 Completion

Emacs lisp programmers probably know the `lisp-complete-symbol` command, usually bound to `M-⟨TAB⟩`. Users of the wonderful ispell mode know and love the `ispell-complete-word` command from that package. Similarly, AUC TeX has a `TeX-complete-symbol` command, usually bound to `M-⟨TAB⟩`. Using `LaTeX-complete-symbol` makes it easier to type and remember the names of long LaTeX macros.

In order to use `TeX-complete-symbol`, you should write a backslash and the start of the macro. Typing `M-⟨TAB⟩` will now complete as much of the macro, as it unambiguously can. For example, if you type "`\renewc`" and then '`M-⟨TAB⟩`, it will expand to "`\renewcommand`".

TeX-complete-symbol Command

(`M-⟨TAB⟩`) Complete TeX symbol before point.

A more direct way to insert a macro is with `TeX-insert-macro`, bound to *C-c C-m*. It has the advantage over completion that it knows about the argument of most standard LaTeX macros, and will prompt for them. It also knows about the type of the arguments, so it will for example give completion for the argument to '`\include`'. Some examples are listed below.

TeX-insert-macro Command

(*C-c C-m*) Prompt (with completion) for the name of a TeX macro, and if AUC TeX knows the macro, prompt for each argument.

As a default selection, AUC TeX will suggest the macro last inserted or, as the first choice the value of the variable `TeX-default-macro`.

TeX-default-macro User Option

Default macro to insert when invoking `TeX-insert-macro` first time.

A faster alternative is to bind the function `TeX-electric-macro` to '\'. This can be done by setting the variable `TeX-electric-escape`

TeX-electric-escape User Option

If this is non-nil when AUC TeX is loaded, the TeX escape character '\' will be bound to `TeX-electric-macro`

The difference between `TeX-insert-macro` and `TeX-electric-macro` is that space will complete and exit from the minibuffer in `TeX-electric-macro`. Use ⟨TAB⟩ if you merely want to complete.

TeX-electric-macro Command

Prompt (with completion) for the name of a TeX macro, and if AUC TeX knows the macro, prompt for each argument. Space will complete and exit.

By default AUC TeX will put an empty set braces '`{}`' after a macro with no arguments to stop it from eating the next whitespace. This can be stopped by entering `LaTeX-math-mode`, see Section 3.1 [Mathematics], page 11, or by setting `TeX-insert-braces` to nil

TeX-insert-braces User Option

If non-nil, append a empty pair of braces after inserting a macro.

Completions work because AUC TeX can analyze TeX files, and store symbols in emacs lisp files for later retrieval. See Chapter 8 [Automatic], page 25, for more information.

AUC TeX will also make completion for many macro arguments, for example existing labels when you enter a '`\ref`' macro with `TeX-insert-macro` or `TeX-electric-macro`, and BibTeX entries when you enter a '`\cite`' macro. For this kind of completion to work, parsing must be enabled as described in see Chapter 6 [Parsing Files], page 21. For '`\cite`' you must also make sure that the BibTeX files have been saved at least once after you enabled automatic parsing on save, and that the basename of the BibTeX file does not conflict with the basename of one of TeX files.

3.3 Commenting

It is often necessary to comment out temporarily a region of TeX or LaTeX code. This can be done with the commands *C-c ;* and *C-c %*. *C-c ;* will comment out all lines in the current region, while *C-c %* will comment out the current paragraph. To uncomment, simply type *C-u - C-c ;* to uncomment all lines in the region, or *C-u - C-c %* uncomment all comment lines around point.

By default, these commands will insert or remove a single '%'. To insert more than one, give an argument. *C-u 5 C-c %* will add five '%' to each line, while *C-u - 5 C-c %* will remove up to 5 '%' from each line.

TeX-comment-region *count* Command

(*C-c ;*) Add or remove '%' from the beginning of each line in the current region, as specified by *count*.

TeX-comment-paragraph *count* Command

(*C-c %*) Add or remove '%' from the beginning of each line in the current paragraph, as specified by *count*. When removing '%''s the paragraph is considered to consist of all preceding and succeeding lines starting with a '%', until the first non-comment line.

3.4 Marking, Formatting and Indenting

AUC TeX contains very advanced handling of indentation and reformatting of the LaTeX source. If you have already tried AUC TeX with `auto-fill-mode` enabled, you may have noted that the source is automatically indented and formatted as you write it. More over, AUC TeX is able to format sections of text on demand.

It is important to realize, that AUC TeX comes with 'formatting' in two fashions. Either letting TeX format the file, or letting AUC TeX make the ASCII document look better.

Indentation is done by LaTeX environments and by TeX groups, that is the body of an environment is indented by the value of `LaTeX-indent-level` (default 2). Also, items of an 'itemize-like' environment are indented by the value of `LaTeX-item-indent`, default −2. This indentation makes it easier to see the structure of the document, and to catch errors such as a missing close brace. Thus, the indentation is done for precisely the same reasons that you would indent ordinary computer programs.

The following is a short sample of an itemize environment indented by AUC TeX. If more environments are nested, they are indented 'accumulated' just like most programming languages usually are seen indented in nested constructs.

```
\begin{itemize}
\item Insertion of templates for logical-structural compositions such as
  environments and sections.
\item Hot-keys for easy access to certain often used constructs, e.g.,
  font changes, accented letters, and mathematical symbols.
\item Running application programs (such as \TeX), and then parsing
  the output so that errors in the document may be located
  easily.
```

```
\item Support for multi-file documents.
\item Online help for \AllTeX\ error messages.
\item Outlining\Dash i.e., manipulating the document as a composition
  of nested/sequential logical constructs.
\item Instant formatting and indentation of the \ascii-document in
  order to make it easier to read.
\item `Completion' (and thereby spell-checking) of partially written
  control sequences.
\end{itemize}
```

You can format and indent single lines, paragraphs, environments, or sections.

⟨TAB⟩
: `LaTeX-indent-line` will indent the current line.

⟨LFD⟩
: `reindent-then-newline-and-indent` indents the current line, and then inserts a new line (much like ⟨RET⟩) and move the cursor to an appropriate position by the left margin.

M-q
: Alias for *C-c C-q C-p*

C-c C-q C-p
: `LaTeX-fill-paragraph` will reformat or 'fill' the current paragraph.

C-c C-q C-e
: `LaTeX-fill-environment` will reformat or 'fill' the current environment. This may e.g. be the 'document' environment, in which case the entire document will be formatted.

C-c C-q C-s
: `LaTeX-fill-section` will reformat or 'fill' the current logical sectional unit.

M-g
: Alias for *C-c C-q C-r*

C-c C-q C-r
: `LaTeX-fill-region` will format or 'fill' the current region.

Warning: The formatting cannot handle tabular-like environments. Those will be completely messed-up if you try to format them.

LaTeX-indent-level User Option
: Number of spaces to add to the indentation for each '`\begin`' not matched by a '`\end`'.

LaTeX-item-indent User Option
: Number of spaces to add to the indentation for '`\item`''s in list environments.

TeX-brace-indent-level User Option
: Number of spaces to add to the indentation for each '`{`' not matched by a '`}`'.

3.5 Outlining the Document

AUC TeX supports the standard outline minor mode using LaTeX sectioning commands as header lines. See section "Outline Mode" in *GNU Emacs Manual*. By default `outline-minor-mode` will use the prefix key *C-c* which is also used by AUC TeX, so it is suggested that you choose another prefix key by inserting

```
(setq outline-minor-mode-prefix "\C-c\C-o") ; Or whatever...
```

in your '`.emacs`' file.

You can add your own headings by setting the variable `TeX-outline-extra`.

TeX-outline-extra Variable

List of extra TeX outline levels.

Each element is a list with two entries. The first entry is the regular expression matching a header, and the second is the level of the header. A '`^`' is automatically prepended to the regular expressions in the list, so they must match text at the beginning of the line.

See `LaTeX-section-list` for existing header levels.

The following example add '`\item`' and '`\bibliography`' headers, with '`\bibliography`' at the same outline level as '`\section`', and '`\item`' being below '`\subparagraph`'.

```
(setq TeX-outline-extra
      '(("[ \t]*\\\\\\(bib\\)?item\\b" 7)
        ("\\\\bibliography\\b" 2)))
```

You may want to check out the unbundled '`out-xtra`' package for even better outline support. It is available from your favorite emacs lisp archive.

4 Formatting and Printing

The most powerful features of AUC TeX may be those allowing you to run (La)TeX and other external commands like BibTeX and `makeindex` from within Emacs, viewing and printing the results, and moreover allowing you to *debug* your documents.

4.1 Executing Commands

Formatting the document with TeX or LaTeX, viewing with a previewer, printing the document, running BibTeX, making an index, or checking the document with `lacheck` or `chktex` all require running an external command.

There are two ways to run an external command, you can either run it on all of the current documents with `TeX-command-master`, or on the current region with `TeX-command-region`.

TeX-command-master Command

(*C-c C-c*) Query the user for a command, and run it on the master file associated with the current buffer. The name of the master file is controlled by the variable `TeX-master`. The available commands are controlled by the variable `TeX-command-list`.

See Chapter 10 [Installation], page 35, for a discussion about `TeX-command-list` and Chapter 5 [Multifile], page 19 for a discussion about `TeX-master`.

TeX-command-region Command

(*C-c C-r*) Query the user for a command, and run it on the "region file". Some commands (typically those invoking TeX or LaTeX) will write the current region into the region file, after extracting the header and tailer from the master file. If mark is not active, use the old region. The name of the region file is controlled by the variable `TeX-region`. The name of the master file is controlled by the variable `TeX-master`. The header is all text up to the line matching the regular expression `TeX-header-end`. The trailer is all text from the line matching the regular expression `TeX-trailer-start`. The available commands are controlled by the variable `TeX-command-list`.

AUC TeX will allow one process for each document, plus one process for the region file to be active at the same time. Thus, if you are editing *n* different documents, you can have *n* plus one processes running at the same time. If the last process you started was on the region, the commands described in Section 4.2 [Debugging], page 17 and Section 4.4 [Control], page 18 will work on that process, otherwise they will work on the process associated with the current document.

TeX-region User Option

The name of the file for temporarily storing the text when formatting the current region.

TeX-header-end User Option

A regular expression matching the end of the header. By default, this is '`\begin{document}`' in LaTeX mode and '`%**end of header`' in TeX mode.

TeX-trailer-start User Option

A regular expression matching the start of the trailer. By default, this is '`\end{document}`' in LaTeX mode and '`\bye`' in TeX mode.

AUC TeX will try to guess what command you want to invoke, but by default it will assume that you want to run TeX in TeX mode and LaTeX in LaTeX mode. You can overwrite this by setting the variable `TeX-command-default`.

TeX-command-default User Option

The default command to run in this buffer. Must be an entry in `TeX-command-list`.

If you want to overwrite the values of `TeX-header-end`, `TeX-trailer-start`, or `TeX-command-default`, you can do that for all files by setting them in either `TeX-mode-hook`, `plain-TeX-mode-hook`, or `LaTeX-mode-hook`. To overwrite them for a single file, define them as file variables (see section "File Variables" in *The Emacs Editor*). You do this by putting special formatted text near the end of the file.

```
% Local Variables:
% TeX-header-end: "% End-Of-Header"
% TeX-trailer-start: "% Start-Of-Trailer"
% TeX-command-default: "SliTeX"
% End:
```

AUC TeX will try to save any buffers related to the document, and check if the document needs to be reformatted. If the variable `TeX-save-query` is non-nil, AUC TeX will query before saving each file. By default AUC TeX will check emacs buffers associated with files in the current directory, in one of the `TeX-macro-private` directories, and in the `TeX-macro-global` directories. You can change this by setting the variable `TeX-check-path`.

TeX-check-path User Option

Directory path to search for dependencies.

If nil, just check the current file. Used when checking if any files have changed.

4.2 Catching the errors

Once you've formatted your document you may 'debug' it, i.e. browse through the errors (La)TeX reported.

TeX-next-error Command

(`C-c ``) Go to the next error reported by TeX. The view will be split in two, with the cursor placed as close as possible to the error in the top view. In the bottom view, the error message will be displayed along with some explanatory text.

Normally AUC TeX will only report real errors, but you may as well ask it to report 'bad boxes' as well.

TeX-toggle-debug-bad-boxes Command

(`C-c C-w`) Toggle whether AUC TeX should stop at bad boxes (i.e. over/under full boxes) as well as at normal errors.

As default, AUC TEX will display that special '`*help*`' buffer containing the error reported by TEX along with the documentation. There is however an 'expert' option, which allows you to display the real TEX output.

TeX-display-help User Option

When non-nil AUC TEX will automatically display a help text whenever an error is encountered using `TeX-next-error` (*C-c '*).

4.3 Checking for problems

Running TEX or LaTEX will only find regular errors in the document, not examples of bad style. Furthermore, description of the errors may often be confusing. The utility `lacheck` can be used to find style errors, such as forgetting to escape the space after an abbreviation or using '`...`' instead of '`\ldots`' and many other problems like that. You start `lacheck` with *C-c C-c C h e c k* ⟨RET⟩. The result will be a list of errors in the '`*compilation*`' buffer. You can go through the errors with *C-x '* (`next-error`, see section "Compilation" in *The Emacs Editor*), which will move point to the location of the next error.

Another newer program which can be used to find errors is `chktex`. It is much more configurable than `lacheck`, but doesn't find all the problems `lacheck` does, at least in its default configuration. You must install the programs before using them, and for `chktex` you must also modify `TeX-command-list`. You can get `lacheck` from '`<URL:ftp://sunsite.dk/pub/text/lacheck/>`' or alternatively `chktex` from '`<URL:ftp://ftp.dante.de/pub/tex/support/chktex/>`'. Search for 'chktex' in '`tex.el`' to see how to switch between them.

4.4 Controlling the output

A number of commands are available for controlling the output of an application running under AUC TEX

TeX-kill-job Command

(*C-c C-k*) Kill currently running external application. This may be either of TEX, LaTEX, previewer, BibTEX, etc.

TeX-recenter-output-buffer Command

(*C-c C-l*) Recenter the output buffer so that the bottom line is visible.

TeX-home-buffer Command

(*C-c ^*) Go to the 'master' file in the document associated with the current buffer, or if already there, to the file where the current process was started.

5 Multifile Documents

You may wish to spread a document over many files (as you are likely to do if there are multiple authors, or if you have not yet discovered the power of the outline commands (see Section 3.5 [Outline], page 15)). This can be done by having a "master" file in which you include the various files with the TeX macro '\input' or the LaTeX macro '\include'. These files may also include other files themselves. However, to format the document you must run the commands on the top level master file.

When you, for example, ask AUC TeX to run a command on the master file, it has no way of knowing the name of the master file. By default, it will assume that the current file is the master file. If you insert the following in your '.emacs' file AUC TeX will use a more advanced algorithm.

```
(setq-default TeX-master nil) ; Query for master file.
```

If AUC TeX finds the line indicating the end of the header in a master file (`TeX-header-end`), it can figure out for itself that this is a master file. Otherwise, it will ask for the name of the master file associated with the buffer. To avoid asking you again, AUC TeX will automatically insert the name of the master file as a file variable (see section "File Variables" in *The Emacs Editor*). You can also insert the file variable yourself, by putting the following text at the end of your files.

```
% Local Variables:
% TeX-master: "master"
% End:
```

You should always set this variable to the name of the top level document. If you always use the same name for your top level documents, you can set `TeX-master` in your '.emacs' file.

```
(setq-default TeX-master "master") ; All master files called "master".
```

TeX-master User Option

The master file associated with the current buffer. If the file being edited is actually included from another file, then you can tell AUC TeX the name of the master file by setting this variable. If there are multiple levels of nesting, specify the top level file.

If this variable is `nil`, AUC TeX will query you for the name.

If the variable is `t`, then AUC TeX will assume the file is a master file itself.

If the variable is `shared`, then AUC TeX will query for the name, but will not change the file.

It is suggested that you use the File Variables (see section "File Variables" in *The Emacs Editor*) to set this variable permanently for each file.

TeX-one-master User Option

Regular expression matching ordinary TeX files.

You should set this variable to match the name of all files, for which it is a good idea to append a `TeX-master` file variable entry automatically. When AUC TeX adds the name of the master file as a file variable, it does not need to ask next time you edit the file.

If you dislike AUC TeX automatically modifying your files, you can set this variable to '`"<none>"`'. By default, AUC TeX will modify any file with an extension of '.tex'.

AUC TEX keeps track of macros, environments, labels, and style files that are used in a given document. For this to work with multifile documents, AUC TEX has to have a place to put the information about the files in the document. This is done by having an '`auto`' subdirectory placed in the directory where your document is located. Each time you save a file, AUC TEX will write information about the file into the '`auto`' directory. When you load a file, AUC TEX will read the information in the '`auto`' directory about the file you loaded *and the master file specified by* `TeX-master`. Since the master file (perhaps indirectly) includes all other files in the document, AUC TEX will get information from all files in the document. This means that you will get from each file, for example, completion for all labels defined anywhere in the document.

AUC TeX will create the '`auto`' directory automatically if `TeX-auto-save` is non-nil. Without it, the files in the document will not know anything about each other, except for the name of the master file. See Section 8.3 [Automatic Local], page 27.

TeX-save-document Command

(`C-c C-d`) Save all buffers known to belong to the current document.

TeX-save-query User Option

If non-nil, then query the user before saving each file with `TeX-save-document`.

6 Automatic Parsing of TEX files.

AUC TEX depends heavily on being able to extract information from the buffers by parsing them. Since parsing the buffer can be somewhat slow, the parsing is initially disabled. You are encouraged to enable them by adding the following lines to your '`.emacs`' file.

```
(setq TeX-parse-self t) ; Enable parse on load.
(setq TeX-auto-save t) ; Enable parse on save.
```

The later command will make AUC TEX store the parsed information in an '`auto`' subdirectory in the directory each time the TEX files are stored, see Section 8.3 [Automatic Local], page 27. If AUC TEX finds the pre-parsed information when loading a file, it will not need to reparse the buffer. The information in the '`auto`' directory is also useful for multifile documents see Chapter 5 [Multifile], page 19, since it allows each file to access the parsed information from all the other files in the document. This is done by first reading the information from the master file, and then recursively the information from each file stored in the master file.

The variables can also be done on a per file basis, by changing the file local variables.

```
% Local Variables:
% TeX-parse-self: t
% TeX-auto-save: t
% End:
```

Even when you have disabled the automatic parsing, you can force the generation of style information by pressing *C-c C-n*. This is often the best choice, as you will be able to decide when it is necessary to reparse the file.

TeX-parse-self User Option

Parse file after loading it if no style hook is found for it.

TeX-auto-save User Option

Automatically save style information when saving the buffer.

TeX-normal-mode *arg* Command

(*C-c C-n*) Remove all information about this buffer, and apply the style hooks again. Save buffer first including style information. With optional argument, also reload the style hooks.

When AUC TeX saves your buffer, it will by default convert all tabs in your buffer into spaces. To disable this behaviour, insert the following in your '`.emacs`' file.

```
(setq TeX-auto-untabify nil)
```

TeX-auto-untabify User Option

Automatically remove all tabs from a file before saving it.

Instead of disabling the parsing entirely, you can also speed it significantly up by limiting the information it will search for (and store) when parsing the buffer. You can do this by setting the default values for the buffer local variables `TeX-auto-regexp-list` and `TeX-auto-parse-length` in your '`.emacs`' file.

```
;; Only parse \documentstyle information.
(setq-default TeX-auto-regexp-list 'LaTeX-auto-minimal-regexp-list)
;; The documentstyle command is usually near the beginning.
(setq-default TeX-auto-parse-length 2000)
```

This example will speed the parsing up significantly, but AUC TEX will no longer be able to provide completion for labels, macros, environments, or bibitems specified in the document, nor will it know what files belong to the document.

These variables can also be specified on a per file basis, by changing the file local variables.

```
% Local Variables:
% TeX-auto-regexp-list: TeX-auto-full-regexp-list
% TeX-auto-parse-length: 999999
% End:
```

TeX-auto-regexp-list User Option

List of regular expressions used for parsing the current file.

TeX-auto-parse-length User Option

Maximal length of TeX file that will be parsed.

The pre-specified lists of regexps are defined below. You can use these before loading AUC TEX by quoting them, as in the example above.

TeX-auto-empty-regexp-list Constant

Parse nothing

LaTeX-auto-minimal-regexp-list Constant

Only parse documentstyle.

LaTeX-auto-label-regexp-list Constant

Only parse LaTEX labels.

LaTeX-auto-regexp-list Constant

Parse common LaTEX commands.

plain-TeX-auto-regexp-list Constant

Parse common plain TEX commands.

TeX-auto-full-regexp-list Constant

Parse all TEX and LaTEX commands that AUC TEX can use.

7 Internationalization

There are several problems associated with editing non-English TeX with GNU Emacs. Modern versions of GNU Emacs and TeX are usable for European (Latin, Cyrillic, Greek) based languages, but special versions of TeX and Emacs are needed for Korean, Japanese, and Chinese.

7.1 Using AUC TeX for European languages.

First you will need a way to write non-ASCII characters. You can either use macros, or teach TeX about the ISO character sets. I prefer the latter, it has the advantage that the usual standard emacs word movement and case change commands will work.

With LaTeX2e, just add '`\usepackage[latin1]{inputenc}`'. Other languages than Western European ones will probably have other encoding needs.

To be able to display non-ASCII characters you will need an appropriate font and a version of GNU Emacs capable of displaying 8-bit characters. The manner in which this is supported differs between Emacsen, so you need to take a look at your respective documentation.

A compromise is to use use an European character set when editing the file, and convert to TeX macros when reading and writing the files.

'`iso-cvt.el`'
: Much like '`iso-tex.el`' but is bundled with Emacs 19.23 and later.

'`x-compose.el`'
: Similar package bundled with new versions of XEmacs.

'`X-Symbol`'
: a much more complete package for both Emacs and XEmacs that can also handle a lot of mathematical characters and input methods.

AUC TeX supports style files for several languages. Each style file may modify some AUC TeX to better support the language, and will run a language specific hook that will allow you to for example change ispell dictionary, or run code to change the keyboard remapping. The following will for example choose a Danish dictionary for documents including the '`dk.sty`' file. This requires parsing to be enabled, see Chapter 6 [Parsing Files], page 21.

```
(add-hook 'TeX-language-dk-hook
          (function (lambda () (ispell-change-dictionary "danish"))))
```

The following style files are recognized.

'`dk`'
: Runs style hook `TeX-language-dk-hook`.

'`dutch`'
: Runs style hook `TeX-language-nl-hook`.

'`german`'
: Runs style hook `TeX-language-de-hook`. Gives '`"`' word syntax and makes the `"` key insert a literal '`"`'.

'`plfonts`'
'`plhb`'
: Runs style hook `TeX-language-pl-hook`. Gives '`"`' word syntax and makes the `"` key insert a literal '`"`'. Pressing `"` twice will insert '`"<`' or '`">`' depending on context.

7.2 Japanese TeX

To write Japanese text with AUC TeX you need to have versions of TeX and Emacs that support Japanese. There exist at least two variants of TeX for Japanese text (jTeX and pTeX), and AUC TeX can be used with MULE supported Emacsens.

To install Japanese support for AUC TeX, copy '`tex-jp.el`' to AUC TeX installed directory. Next two commands will automatically install contributed files.

```
make contrib
make install-contrib
```

See '`INSTALLATION`' and '`Makefile`' for more information.

To use the Japanese TeX variants, simply enter `japanese-tex-mode`, `japanese-latex-mode`, or `japanese-slitex-mode`, and everything should work. If not, send mail to Shinji Kobayashi '`<koba@flab.fujitsu.co.jp>`', who kindly donated the code for supporting Japanese in AUC TeX. None of the primary AUC TeX maintainers understand Japanese, so they can not help you.

If you usually use AUC TeX in Japanese, setting following variables is useful.

TeX-default-mode User Option

Mode to enter for a new file when it can't be determined whether the file is plain TeX or LaTeX or what.

To use Japanese TeX always, set japanese command for example:

```
(setq TeX-default-mode 'japanese-latex-mode)
```

japanese-TeX-command-default User Option

The default command for TeX-command in japanese TeX mode.

The default value is '`jTeX`'.

japanese-LaTeX-command-default User Option

The default command for TeX-command in japanese LaTeX mode.

The default value is '`jLaTeX`'.

japanese-LaTeX-default-style User Option

The default style/class when creating new japanese LaTeX document.

The default value is '`j-article`'.

See '`tex-jp.el`' for more information.

8 Automatic Customization

Since AUC TeX is so highly customizable, it makes sense that it is able to customize itself. The automatic customization consists of scanning TeX files and extracting symbols, environments, and things like that.

The automatic customization is done on three different levels. The global level is the level shared by all users at your site, and consists of scanning the standard TeX style files, and any extra styles added locally for all users on the site. The private level deals with those style files you have written for your own use, and use in different documents. You may have a '`~/lib/TeX/`' directory where you store useful style files for your own use. The local level is for a specific directory, and deals with writing customization for the files for your normal TeX documents.

If compared with the environment variable `TEXINPUTS`, the global level corresponds to the directories built into TeX. The private level corresponds to the directories you add yourself, except for '`.`', which is the local level.

By default AUC TeX will search for customization files in all the global, private, and local style directories, but you can also set the path directly. This is useful if you for example want to add another person's style hooks to your path. Please note that all matching files found in `TeX-style-path` are loaded, and all hooks defined in the files will be executed.

TeX-style-path User Option

List of directories to search for AUC TeX style files. Each must end with a slash.

By default, when AUC TeX searches a directory for files, it will recursively search through subdirectories.

TeX-file-recurse User Option

Whether to search TeX directories recursively: nil means do not recurse, a positive integer means go that far deep in the directory hierarchy, t means recurse indefinitely.

By default, AUC TeX will ignore files name '`.`', '`..`', '`SCCS`', '`RCS`', and '`CVS`'.

TeX-ignore-file User Option

Regular expression matching file names to ignore.

These files or directories will not be considered when searching for TeX files in a directory.

8.1 Automatic Customization for the Site

Assuming that the automatic customization at the global level was done when AUC TeX was installed, your choice is now: will you use it? If you use it, you will benefit by having access to all the symbols and environments available for completion purposes. The drawback is slower load time when you edit a new file and perhaps too many confusing symbols when you try to do a completion.

You can disable the automatic generated global style hooks by setting the variable `TeX-auto-global` to nil.

TeX-macro-global User Option

Directories containing the site's TeX style files.

TeX-style-global User Option

Directory containing hand generated TeX information. Must end with a slash.

These correspond to TeX macros shared by all users of a site.

TeX-auto-global User Option

Directory containing automatically generated information.

For storing automatic extracted information about the TeX macros shared by all users of a site.

8.2 Automatic Customization for a User

You should specify where you store your private TeX macros, so AUC TeX can extract their information. The extracted information will go to the directories listed in `TeX-auto-private`

Use *`M-x TeX-auto-generate`* to extract the information.

TeX-macro-private User Option

Directories where you store your personal TeX macros. Each must end with a slash.

This defaults to the directories listed in the '`TEXINPUTS`' and '`BIBINPUTS`' environment variables.

TeX-auto-private User Option

List of directories containing automatically generated information. Must end with a slash.

These correspond to the personal TeX macros.

TeX-auto-generate *TEX AUTO* Command

(*`M-x TeX-auto-generate`*) Generate style hook for *TEX* and store it in *AUTO*. If *TEX* is a directory, generate style hooks for all files in the directory.

TeX-style-private User Option

List of directories containing hand generated information. Must end with a slash.

These correspond to the personal TeX macros.

8.3 Automatic Customization for a Directory

AUC TEX can update the style information about a file each time you save it, and it will do this if the directory `TeX-auto-local` exist. `TeX-auto-local` is by default set to '`"auto/"`', so simply creating an '`auto`' directory will enable automatic saving of style information.

The advantage of doing this is that macros, labels, etc. defined in any file in a multifile document will be known in all the files in the document. The disadvantage is that saving will be slower. To disable, set `TeX-auto-local` to nil.

TeX-style-local User Option

Directory containing hand generated TEX information. Must end with a slash.

These correspond to TEX macros found in the current directory.

TeX-auto-local User Option

Directory containing automatically generated TEX information. Must end with a slash.

These correspond to TEX macros found in the current directory.

9 Writing Your own Style Support

See Chapter 8 [Automatic], page 25, for a discussion about automatically generated global, private, and local style files. The hand generated style files are equivalent, except that they by default are found in '`style`' directories instead of '`auto`' directories.

If you write some useful support for a public TEX style file, please send it to us.

9.1 A Simple Style File

Here is a simple example of a style file.

```
;;; book.el - Special code for book style.

(TeX-add-style-hook "book"
 (function (lambda () (setq LaTeX-largest-level
                            (LaTeX-section-level ("chapter"))))))
```

This file specifies that the largest kind of section in a LaTeX document using the book document style is chapter. The interesting thing to notice is that the style file defines an (anonymous) function, and adds it to the list of loaded style hooks by calling `TeX-add-style-hook`.

The first time the user indirectly tries to access some style specific information, such as the largest sectioning command available, the style hooks for all files directly or indirectly read by the current document is executed. The actual files will only be evaluated once, but the hooks will be called for each buffer using the style file.

TeX-add-style-hook *style hook* Function

Add *hook* to the list of functions to run when we use the TEX file *style*.

9.2 Adding Support for Macros

The most common thing to define in a style hook is new symbols (TEX macros). Most likely along with a description of the arguments to the function, since the symbol itself can be defined automatically.

Here are a few examples from '`latex.el`'.

```
(TeX-add-style-hook "latex"
 (function
  (lambda ()
     (TeX-add-symbols
     '("arabic" TeX-arg-counter)
     '("label" TeX-arg-define-label)
     '("ref" TeX-arg-label)
     '("newcommand" TeX-arg-define-macro [ "Number of arguments" ] t)
     '("newtheorem" TeX-arg-define-environment
       [ TeX-arg-environment "Numbered like" ]
       t [ TeX-arg-counter "Within counter" ])))))
```

TeX-add-symbols *symbol ...* Function

Add each *symbol* to the list of known symbols.

Each argument to `TeX-add-symbols` is a list describing one symbol. The head of the list is the name of the symbol, the remaining elements describe each argument.

If there are no additional elements, the symbol will be inserted with point inside braces. Otherwise, each argument of this function should match an argument of the TeX macro. What is done depends on the argument type.

If a macro is defined multiple times, AUC TeX will chose the one with the longest definition (i.e. the one with the most arguments).

Thus, to overwrite

```
'("tref" 1) ; one argument
```

you can specify

```
'("tref" TeX-arg-label ignore) ; two arguments
```

`ignore` is a function that does not do anything, so when you insert a '`tref`' you will be prompted for a label and no more.

`string` Use the string as a prompt to prompt for the argument.

`number` Insert that many braces, leave point inside the first.

`nil` Insert empty braces.

`t` Insert empty braces, leave point between the braces.

`other symbols`
: Call the symbol as a function. You can define your own hook, or use one of the predefined argument hooks.

`list` If the car is a string, insert it as a prompt and the next element as initial input. Otherwise, call the car of the list with the remaining elements as arguments.

`vector` Optional argument. If it has more than one element, parse it as a list, otherwise parse the only element as above. Use square brackets instead of curly braces, and is not inserted on empty user input.

A lot of argument hooks have already been defined. The first argument to all hooks is a flag indicating if it is an optional argument. It is up to the hook to determine what to do with the remaining arguments, if any. Typically the next argument is used to overwrite the default prompt.

`TeX-arg-conditional`
: Implements if EXPR THEN ELSE. If EXPR evaluates to true, parse THEN as an argument list, else parse ELSE as an argument list.

`TeX-arg-literal`
: Insert its arguments into the buffer. Used for specifying extra syntax for a macro.

`TeX-arg-free`
: Parse its arguments but use no braces when they are inserted.

`TeX-arg-eval`
: Evaluate arguments and insert the result in the buffer.

`TeX-arg-file`
: Prompt for a tex or sty filename, and use it without the extension. Run the file hooks defined for it.

`TeX-arg-label`
: Prompt for a label completing with known labels.

`TeX-arg-macro`
: Prompt for a TeX macro with completion.

`TeX-arg-environment`
: Prompt for a LaTeX environment with completion.

`TeX-arg-cite`
: Prompt for a BibTeX citation.

`TeX-arg-counter`
: Prompt for a LaTeX counter.

`TeX-arg-savebox`
: Prompt for a LaTeX savebox.

`TeX-arg-file`
: Prompt for a filename in the current directory, and use it without the extension.

`TeX-arg-input-file`
: Prompt for a filename in the current directory, and use it without the extension. Run the style hooks for the file.

`TeX-arg-define-label`
: Prompt for a label completing with known labels. Add label to list of defined labels.

`TeX-arg-define-macro`
: Prompt for a TeX macro with completion. Add macro to list of defined macros.

`TeX-arg-define-environment`
: Prompt for a LaTeX environment with completion. Add environment to list of defined environments.

`TeX-arg-define-cite`
: Prompt for a BibTeX citation.

`TeX-arg-define-counter`
: Prompt for a LaTeX counter.

`TeX-arg-define-savebox`
: Prompt for a LaTeX savebox.

`TeX-arg-corner`
: Prompt for a LaTeX side or corner position with completion.

`TeX-arg-lr`
: Prompt for a LaTeX side with completion.

`TeX-arg-tb`
: Prompt for a LaTeX side with completion.

`TeX-arg-pagestyle`
: Prompt for a LaTeX pagestyle with completion.

`TeX-arg-verb`
: Prompt for delimiter and text.

`TeX-arg-pair`
: Insert a pair of numbers, use arguments for prompt. The numbers are surrounded by parentheses and separated with a comma.

`TeX-arg-size`
: Insert width and height as a pair. No arguments.

`TeX-arg-coordinate`
: Insert x and y coordinates as a pair. No arguments.

If you add new hooks, you can assume that point is placed directly after the previous argument, or after the macro name if this is the first argument. Please leave point located after the argument you are inserting. If you want point to be located somewhere else after all hooks have been processed, set the value of `exit-mark`. It will point nowhere, until the argument hook sets it.

9.3 Adding Support for Environments

Adding support for environments is very much like adding support for TeX macros, except that each environment normally only takes one argument, an environment hook. The example is again a short version of '`latex.el`'.

```
(TeX-add-style-hook "latex"
 (function
  (lambda ()
    (LaTeX-add-environments
     '("document" LaTeX-env-document)
     '("enumerate" LaTeX-env-item)
     '("itemize" LaTeX-env-item)
     '("list" LaTeX-env-list)))))
```

The only hook that is generally useful is `LaTeX-env-item`, which is used for environments that contain items. It is completely up to the environment hook to insert the environment, but the function `LaTeX-insert-environment` may be of some help. The hook will be called with the name of the environment as its first argument, and extra arguments can be provided by adding them to a list after the hook.

For simple environments with arguments, for example defined with '`\newenvironment`', you can make AUC TeX prompt for the arguments by giving the prompt strings in the call to `LaTeX-add-environments`. For example, if you have defined a `loop` environment with the three arguments *from*, *to*, and *step*, you can add support for them in a style file.

```
%% loop.sty

\newenvironment{loop}[3]{...}{...}
```

```
;; loop.el

(TeX-add-style-hook "loop"
 (function
  (lambda ()
    (LaTeX-add-environments
     '("loop" "From" "To" "Step")))))
```

If an environment is defined multiple times, AUC TeX will chose the one with the longest definition. Thus, if you have an enumerate style file, and want it to replace the standard LaTeX enumerate hook above, you could define an '`enumerate.el`' file as follows, and place it in the appropriate style directory.

```
(TeX-add-style-hook "latex"
 (function
  (lambda ()
    (LaTeX-add-environments
     '("enumerate" LaTeX-env-enumerate foo)))))

(defun LaTeX-env-enumerate (environment &optional ignore) ...)
```

The symbol `foo` will be passed to `LaTeX-env-enumerate` as the second argument, but since we only added it to overwrite the definition in '`latex.el`' it is just ignored.

LaTeX-add-environments *env* . . . Function

Add each *env* to list of loaded environments.

LaTeX-insert-environment *env* [*extra*] Function

Insert environment of type *env*, with optional argument *extra*.

9.4 Adding Other Information

You can also specify bibliographical databases and labels in the style file. This is probably of little use, since this information will usually be automatically generated from the TeX file anyway.

LaTeX-add-bibliographies *bibliography* . . . Function

Add each *bibliography* to list of loaded bibliographies.

LaTeX-add-labels *label* . . . Function

Add each *label* to the list of known labels.

9.5 Automatic Extraction of New Things

The automatic TeX information extractor works by searching for regular expressions in the TeX files, and storing the matched information. You can add support for new constructs to the parser, something that is needed when you add new commands to define symbols.

For example, in the file 'macro.tex' I define the following macro.

```
\newcommand{\newmacro}[5]{%
\def#1{#3\index{#4@#5~cite{#4}}\nocite{#4}}%
\def#2{#5\index{#4@#5~cite{#4}}\nocite{#4}}%
}
```

AUC TeX will automatically figure out that 'newmacro' is a macro that takes five arguments. However, it is not smart enough to automatically see that each time we use the macro, two new macros are defined. We can specify this information in a style hook file.

```
;;; macro.el - Special code for my own macro file.

;;; Code:

(defvar TeX-newmacro-regexp
  '("\\\\newmacro{\\\\\\([a-zA-Z]+\\)}{\\\\\\([a-zA-Z]+\\)}"
    (1 2) TeX-auto-multi)
  "Matches \newmacro definitions.")

(defvar TeX-auto-multi nil
  "Temporary for parsing \\newmacro definitions.")

(defun TeX-macro-cleanup ()
  ;; Move symbols from `TeX-auto-multi' to `TeX-auto-symbol'.
  (mapcar (function (lambda (list)
            (mapcar (function (lambda (symbol)
                      (setq TeX-auto-symbol
                            (cons symbol TeX-auto-symbol))))
                    list)))
          TeX-auto-multi))

(defun TeX-macro-prepare ()
  ;; Clear `Tex-auto-multi' before use.
  (setq TeX-auto-multi nil))

(add-hook 'TeX-auto-prepare-hook 'TeX-macro-prepare)
(add-hook 'TeX-auto-cleanup-hook 'TeX-macro-cleanup)

(TeX-add-style-hook "macro"
 (function
  (lambda ()
    (TeX-auto-add-regexp TeX-newmacro-regexp)
    (TeX-add-symbols '("newmacro"
                       TeX-arg-macro
                       (TeX-arg-macro "Capitalized macro: \\")
```

```
                                t
                                "BibTeX entry: "
                                nil)))))

;;; macro.el ends here
```

When this file is first loaded, it adds a new entry to `TeX-newmacro-regexp`, and defines a function to be called before the parsing starts, and one to be called after the parsing is done. It also declares a variable to contain the data collected during parsing. Finally, it adds a style hook which describes the '`newmacro`' macro, as we have seen it before.

So the general strategy is: Add a new entry to `TeX-newmacro-regexp`. Declare a variable to contain intermediate data during parsing. Add hook to be called before and after parsing. In this case, the hook before parsing just initializes the variable, and the hook after parsing collects the data from the variable, and adds them to the list of symbols found.

TeX-auto-regexp-list Variable

List of regular expressions matching TEX macro definitions.

The list has the following format ((REGEXP MATCH TABLE) ...), that is, each entry is a list with three elements.

REGEXP. Regular expression matching the macro we want to parse.

MATCH. A number or list of numbers, each representing one parenthesized subexpression matched by REGEXP.

TABLE. The symbol table to store the data. This can be a function, in which case the function is called with the argument MATCH. Use `TeX-match-buffer` to get match data. If it is not a function, it is presumed to be the name of a variable containing a list of match data. The matched data (a string if MATCH is a number, a list of strings if MATCH is a list of numbers) is put in front of the table.

TeX-auto-prepare-hook nil Variable

List of functions to be called before parsing a TEX file.

TeX-auto-cleanup-hook nil Variable

List of functions to be called after parsing a TEX file.

10 Installation of AUC TEX

10.1 Compiling

The following describes how to install AUC TEX under Unix. You may also be able to do use these instructions under some other operating systems, if you have already installed the proper GNU tools, such as '`make`'.

To install AUC TEX for an entire site (which may just be your own personal Linux box), issue the following two commands as root:

```
make
make lispdir=/usr/local/share/emacs/site-lisp install
```

except that instead of `/usr/local/...` you should use the location of your sites emacs installation. AUC TEX will then be installed in a subdirectory named '`auctex`' of the '`site-lisp`' directory, and the file '`tex-site.el`' will be stored directly in the '`site-lisp`'. You can now tell your users to enable AUC TEX by adding

```
(require 'tex-site)
```

to their '`.emacs`' file.

If you use xemacs instead, or if your emacs binary is named something else than '`emacs`', specify this by using the commands

```
make EMACS=xemacs
make lispdir=/usr/local/share/emacs/site-lisp install
```

to install.

If you want to install AUC TEX in your personal account, you should chose a directory for all your emacs add-ons, for example an '`elisp`' subdirectory in your home directory. You can then install AUC TEX with the commands

```
make
make lispdir=$HOME/elisp install
```

You will then need to add the following lines to your '`.emacs`' file:

```
(setq load-path (cons "~/elisp" load-path))
(require 'tex-site)
```

10.2 Customizing

Next, you should edit the file '`tex-site.el`' to fit your local site. You do this by looking at the customization section in the beginning of '`tex.el`' and copy the definitions that are wrong for your site to '`tex-site.el`'. Do *not* edit '`tex.el`' directly, or you will have to do all the work over again when you upgrade AUC TEX. AUC TEX will not overwrite your old '`tex-site.el`' file next time you install, so you will be able to keep all your customizations.

There are two variables with a special significance.

TeX-lisp-directory User Option

The directory where you want to install the AUC TEX lisp files.

This variable is set automatically by the `make install` command. If you don't issue a `make install`, for example if you don't want to install AUC TEX in a different place, you will have to set this variable manually to the location of the compiled files.

TeX-macro-global User Option

Directories containing the site's TEX style files.

Normally, AUC TEX will only allow you to complete a short list of built-in macros and environments and on the macros you define yourself. If you issue the *M-x TeX-auto-generate-global* command after loading AUC TEX, you will be able to complete on all macros available in the standard style files used by your document. To do this, you must set this variable to a list of directories where the standard style files are located. The directories will be searched recursively, so there is no reason to list subsirectories explicitly.

You probably also need to change `TeX-command-list` to make sure that the commands used for starting TEX, printing, etc. work on your system. Copy the definition from '`tex.el`' to '`tex-site.el`' and edit the command names appropriately.

Finally, copy and edit `TeX-printer-list` to contain the printers available at your site.

To extract information from your sites TEX macros, type *M-x TeX-auto-generate-global* in your emacs. This will only work if you have set `TeX-macro-global` correctly in '`tex-site.el`'.

10.3 Contributed files

There are several files that are not part of AUC TEX proper, but included in the distribution in case they are useful.

'`hilit-LaTeX.el`'
: Better highlighting for the obsolete '`hilit19`' package.

'`bib-cite.el`'
: Better support for bibliographies and much more.

'`tex-jp.el`'
: Support for Japanese.

Read the comments in the start of each file for more information about how to install, what they do, and who wrote and maintains them.

Appendix A The History of AUC TEX

See the file 'history.texi' for older changes.

A.1 News in 12

- Support for the KOMA-Script classes. Contributed by Mark Trettin <Mark.Trettin@gmx.de>.
- Many more LaTeX and LaTeX2e commands are supported. Done by Masayuki Ataka <ataka@milk.freemail.ne.jp>
- Support for 'prosper.sty', see http://prosper.sourceforge.net/. Contributed by Phillip Lord <p.lord@russet.org.uk>.
- `comment-region` now inserts `%%` by default. Suggested by "Davide G. M. Salvetti" <salve@debian.org>.
- You can now switch between using the 'font-latex' (all emacsen), the 'tex-font' (Emacs 21 only) or no special package for font locking. Customize `TeX-install-font-lock` for this.
- Now use `-t landscape` by default when landscape option appears. Suggested by Erik Frisk <frisk@isy.liu.se>.
- Use 'tex-fptex.el' for fpTeX support. Contributed by Fabrice Popineau <Fabrice.Popineau@supelec.fr>.
- New user option `LaTeX-top-caption-list` specifies environments where the caption should go at top. Contributed by ataka@milk.freemail.ne.jp (Masayuki Ataka).
- Allow explicit dimensions in 'graphicx.sty'. Contributed by ataka@milk.freemail.ne.jp (Masayuki Ataka).
- Limited support for 'verbatim.sty'. Contributed by ataka@milk.freemail.ne.jp (Masayuki Ataka).
- Better support for asmmath items. Patch by ataka@milk.freemail.ne.jp (Masayuki Ataka).
- More accurate error parsing. Added by David Kastrup <David.Kastrup@t-online.de>.
- Bug fixes.

Appendix B Wishlist

This is a list of projects for AUC TEX. Bug reports and requests we can not fix or honor right away will be added to this list. If you have some time for emacs lisp hacking, you are encouraged to try to provide a solution to one of the following problems. It might be a good idea to mail me first, though.

- Page count when compiling should (optionally) go to modeline of the window where the compilation command was invoked, instead of the output window. Suggested by Karsten Tinnefeld <tinnefeld@irb.informatik.uni-dortmund.de>.
- Command to insert a macrodefinition in the preamble, without moving point from the current location. Suggested by "Jeffrey C. Ely" <ely@nwu.edu>.
- Filling messes up comments, but only at the end of the file. Reported by Juergen Reiss <psy3022@rzbox.uni-wuerzburg.de>.
- *C-c C-q C-e* doesn't work properly on nested itemize environments. Reported by "Robert B. Love" <rlove@raptor.rmNUG.ORG>.
- One suggestion for AUC-TeX: I think that the font command C-c C-f C-r, which produces \textrm{} in a LaTeX file, should instead produce either \textrm{} or \mathrm{}, depending on whether one is in math mode or not. — John Palmieri <palmieri@math.mit.edu>
- A way to add and overwrite math mode entries in style files, and to decide where they should be. Suggested by Remo Badii <Remo.Badii@psi.ch>.
- Create template for (first) line of tabular environment.
- I think prompting for the master is the intended behaviour. It corresponds to a 'shared' value for TeX-master.

  There should probably be a 'none' value which wouldn't query for the master, but instead disable all features that relies on TeX-master.

  This default value for TeX-master could then be controled with mapping based on the extension.
- *C-c '* should alway stay in the current window, also when it find a new file.
- `LaTeX-fill-environment` does not indent the closing '`\end`'.
- Rewrite '`ltx-help.el`' and put it in '`latex.el`'. Fix also:

```
From: Denby Wong <DnB@slip224.qlink.QueensU.CA>

 1) change documentation regarding where to get the
    latest version (at CTAN at pip.shsu.edu for me)
    under info/latex2e-help-texinfo/

 2) change or provide choice over which version to
    use.  There are three references to the info
    node "(latex)" in the file which should be
    "(latex2e)"  for the new file.

From: Piet van Oostrum <piet@cs.ruu.nl>
```

```
One of the annoying things of latex-help is that if you ask for \LARGE, you
get \large if you have case-fold-search=t. This is really info's problem as
it doesn't reset it for a search of the node, but it would be easy to stick
a (let (case-fold-search) in latex-help.
```

- It should be possible to insert a default preamble containing e.g. usepackage declarations, perhaps depending on the document class.
- Multiple argument completion for ‘\bibliography’. In general, I ought to make , special for these kind of completions.
- Do not overwrite emacs warnings about existing auto-save files when loading a new file.
- Suggest ‘makindex’ when appropriate.
- Maybe the regexp for matching a TeX symbol during parsing should be ‘"\\\\\\([a-zA-Z]+\\|.\\)"’ — ‘<thiemann@informatik.uni-tuebingen.de>’ Peter Thiemann.
- AUC TeX should be able to parse LaTeX2e ‘.cls’ files. Here are the regexps by ‘<thiemann@informatik.uni-tuebingen.de>’ Peter Thiemann.

```
        ("\\\\DeclareRobustCommand{?\\\\\\([a-zA-Z]+\\)}?\\[\\([0-9]+\\)\\]\
\\[\\([^\]\\\\\n\r]+\\)\\]"
         (1 2 3) LaTeX-auto-optional)
        ("\\\\DeclareRobustCommand{?\\\\\\([a-zA-Z]+\\)}?\\[\\([0-9]+\\)\\]"
         (1 2) LaTeX-auto-arguments)
        ("\\\\DeclareRobustCommand{?\\\\\\([a-zA-Z]+\\)}?" 1 TeX-auto-symbol)
        ("\\\\DeclareFixedFont{?\\\\\\([a-zA-Z]+\\)}?"
         1 TeX-auto-symbol)
        ("\\\\Declare\\(Text\\|Old\\)FontCommand{?\\\\\\([a-zA-Z]+\\)}?"
         2 TeX-auto-symbol)
        ("\\\\DeclareMath\\(Symbol\\|Delimiter\\|Accent\\|Radical\\){?\\\\\\([a-zA-Z]+\\)}?"
         2 TeX-auto-symbol)
        ;;; it is also valid to declare just a single symbol, e.g. <,
        ;;; with \DeclareMathSymbol but it is not necessary to register that here
        ("\\\\DeclareText\\(Command\\|Symbol\\|Accent\\|Composite\\){?\\\\\\([a-zA-Z]+\\)}?"
         2 TeX-auto-symbol)
```

- Use index files (when available) to speed up *C-c C-m include* ⟨RET⟩.
- Option not to calculate very slow completions like for *C-c C-m include* ⟨RET⟩.
- AUC TeX should not parse verbatim environments.
- Font menu should be created from TeX-font-list.
- Installation procedure written purely in emacs lisp.
- Format LaTeX comment blocks.
- Included PostScript files should also be counted as part of the document.
- The argument to ‘\verb’ may be broken when filling if it contains a space. This should be fixed or documented. Suggested by several people.
- The parser should catch warnings about undefined crossreferences. Suggested by Richard Hirsch ‘i3080501@ws.rz.tu-bs.de’.
- A nice hierarchical by-topic organization of all officially documented LaTeX macros, available from the menu bar.

- Make ‘‘’ check for math context in `LaTeX-math-mode`. and simply self insert if not in a math context.
- Make `TeX-insert-dollar` more robust. Currently it can be fooled by ‘`\mbox`’’es and escaped double dollar for example.
- LaTeX formatting should skip `verbatim` environments.
- `TeX-command-default` should be set from the master file, if not set locally. Suggested by Peter Whaite ‘`<peta@cim.mcgill.ca>`’.
- Make AUC TeX work with ‘`crypt++`’. Suggested by Chris Moore ‘`<Chris.Moore@src.bae.co.uk>`’.
- Fix bug with `TeX-show-environment` from hidden document environment.
- Function to check if you are in math mode (between two dollar signs). Suggested by Jan Erik Odegard ‘`<odegard@dsp.rice.edu>`’
- The ‘`Spell`’ command should apply to all files in a document. Maybe it could try to restrict to files that have been modified since last spell check? Suggested by Ravinder Bhumbla ‘`<rbhumbla@ucsd.edu>`’.
- Make ⟨.⟩ check for abbreviations and sentences ending with capital letters.
- Use Emacs 19 minibuffer history to choose between previewers, and other stuff. Suggested by John Interrante ‘`<interran@uluru.Stanford.EDU>`’.
- Make features.

  A new command `TeX-update` (*C-c C-u*) could be used to create an up-to-date dvi file by repeatedly running BibTeX, MakeIndex and (La)TeX, until an error occurs or we are done.

  An alternative is to have an ‘`Update`’ command that ensures the ‘`dvi`’ file is up to date. This could be called before printing and previewing.
- Documentation of variables that can be set in a style hook.

  We need a list of what can safely be done in an ordinary style hook. You can not set a variable that AUC TeX depends on, unless AUC TeX knows that it has to run the style hooks first.

  Here is the start of such a list.

  `LaTeX-add-environments`

  `TeX-add-symbols`

  `LaTeX-add-labels`

  `LaTeX-add-bibliographies`

  `LaTeX-largest-level`
- Correct indentation for tabular, tabbing, table, math, and array environments.
- Optional special indentation after an ‘`\item`’.

  ```
  \begin{itemize}
  \item blabalaskdfjlas lajf adf
        lkfjl  af jasl lkf jlsdf jlf
  \item f lk jldjf lajflkas flf af
  \end{itemize}
  ```
- Completion for counters and sboxes.

- Outline should be (better) supported in TeX mode.

  At least, support headers, trailers, as well as TeX-outline-extra.
- `TeX-header-start` and `TeX-trailer-end`.

  We might want these, just for fun (and outlines)
- Plain TeX and LaTeX specific header and trailer expressions.

  We should have a way to globally specify the default value of the header and trailer regexps.
- Add support for original `TeX-mode` keybindings.

  A third initialization file ('`tex-mode.el`') containing an emulator of the standard `TeX-mode` would help convince some people to change to AUC TeX.
- Make `TeX-next-error` parse ahead and store the results in a list, using markers to remember buffer positions in order to be more robust with regard to line numbers and changed files. This is what `next-error` does. (Or did, until Emacs 19).
- When `LaTeX-environment` is given an argument, change the current environment. Be smart about '`\item[]`' versus '`\item `' and labels like '`fig:`' versus '`tab:`'.
- Check out if lightning completion from Ultra TeX is anything for us.
- Finish the TeXinfo mode. For one thing, many TeXinfo mode commands do not accept braces around their arguments.
- BibTeX mode.
- Support for AMSLaTeX style files.
- Hook up the letter environment with 'bbdb.el'.
- Make the letter environment hook produce 'documentstyle' too.

Appendix C Credit

A big smile and thanks should go to all the folks who cheered me up, during the long hours of programming. . . sorry folks, but I can't help including the list below, of comments I've got. . .

Kresten Krab Thorup

'`<monheit@psych.stanford.edu>`'
: I'd like to say that I'm very much enjoying using auc-tex. Thanks for the great package!

'`<georgiou@rex.cs.tulane.edu>`'
: I really enjoy working with auc-tex.

'`<toy@soho.crd.ge.com>`'
: Thanks for your great package. It's indispensable now! Thanks!

'`<ascott@gara.une.oz.au>`'
: Thanks for your time and for what appears to be a great and useful package. Thanks again

'`<hal@alfred.econ.lsa.umich.edu>`'
: Thanks for providing auc-tex.

'`<simons@ibiza.karlsruhe.gmd.de>`'
: I really enjoy using the new emacs TeX-mode you wrote. I think you did a great job.

'`<clipper@csd.uwo.ca>`'
: I am having fun with auc-tex already.

'`<ibekhaus@athena.mit.edu>`'
: Thanks for your work on auc-tex, especially the math-minor mode.

'`<burt@dfki.uni-kl.de>`'
: I like your auc-tex elisp package for writing LaTeX files - I am especially impressed by the help with error correction.

'`<goncal@cnmvax.uab.es>`'
: Thanks so much!

'`<bond@sce.carleton.ca>`'
: I >really< like the macro, particularly the hooks for previewing and the error parsing!

'`<ascott@gara.une.oz.au>`'
: All in all I am pleased with your package. Thanks a lot.

Key Index

Function Index

L

T

Variable Index

Concept Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

Short Contents

Table of Contents