AUC TEX

A much enhanced LaTEX mode for GNU Emacs.
September 1993, for AUC TEX Version 9.0

by Kresten Krab Thorup
updated for 6.1 to 9.0 by Per Abrahamsen

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, plus other separate programs (specifically `lacheck`). 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 oif AUC TEX, the latest changes can be found in the 'CHANGES' file. If you are upgrading from an older version, read the History cxhapter 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 an large range of handy Emacs macros, which in several different ways can help you write your LaTEX documents fast and painless.

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 build-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-format-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 '`\documentstyle`' 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 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 here 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 '`{\rm }`'), '`{\it \/}`' '`{\bf }`' 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 on 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 you document. In case TEX found 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 resent version is always available by ftp at

‘ftp://ftp.iesd.auc.dk/pub/emacs-lisp/auctex.tar.gz’

In case you don’t have access anonymous ftp, you can get it by email requests to ‘<ftpmail@decwrl.dec.com>’.

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

‘http://www.iesd.auc.dk/~amanda/aucTeX/’

1.7 Contacts

There has been established a mailing list for discussion of the evaluation of 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 ‘<auc-tex-request@iesd.auc.dk>’ in order to join. Articles should be send to ‘<auc-tex@iesd.auc.dk>’.

To contact the current maintainers of auc-TeX directly, email ‘<auc-tex_mgr@iesd.auc.dk>’.

AUC TeX development
c/o Kresten Krab Thorup

Mathematics and Computer Science
University of Aalborg
DK 9000 Aalborg
Denmark

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 avoiding 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 open 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 parially 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 {\rm ⋆} text.

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

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

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

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

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

`C-c C-f C-c`
: Insert SMALL CAPS '`{\sc ⋆}`' 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 '`\documentstyle`' 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 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.

2.4.2 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.

2.4.3 Tabular-like

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

2.4.4 Customizing environments

See Section 9.3 [Adding Environments], page 30, 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 mathematic 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 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 can be found in `TeX-style-path` and have been saved at least once after you enabled automatic parsing on save.

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 environment 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-format-paragraph` will reformat or ‘fill’ the current paragraph.

`C-c C-q C-e`
: `LaTeX-format-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-format-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-format-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

GNU Emacs earlier than version 19.19 does not have a useful outline mode. If you want to use outlines with old versions of emacs, please get the file ‘`outln-18.el`’ from ‘`ftp.iesd.auc.dk:/pub/emacs-lisp/outln-18.el`’. It is a port of the Emacs 19.19 outline mode to Emacs 18 and Lucid Emacs.

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 a previewer, printing the document, running BibTEX, making an index, or checking the document with `lacheck` 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 document 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 34 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.

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 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 14)). 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.

You must create the '`auto`' directory manually for this to work. If you don't do this, the other 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 26.

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 26. 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 not 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 expresions 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 useable for European (Latin, Cyrillic, Greek) based languages, but special versions of TeX and Emacs are needed for Korean, Japanes, 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 later, it has the adventage that the usual the standard emacs word movement and case change commands will work.

'`isolatin1.sty`'
: Support for ISO 8859 Latin 1. Available by ftp from the host `ftp.uni-stuttgart.de` as '`/pub/tex/macros/latex/contrib/misc/isolatin1.sty`'.

'`latin2.sty`'
: Support for ISO 8859 Latin 2. Available by ftp from the host `ftp.uni-stuttgart.de` as '`/pub/tex/macros/latex/contrib/latin2.sty`'.

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. I believe all emacs versions except plain Emacs 18 are capable of this. For GNU Emacs 19, see section "European Display" in *The GNU Emacs Editor*. Other relevant packages are:

'`remap`'
: Supports lots of different 7-bit and 8-bit character sets for GNU Emacs 19. Mostly useful if you have seldomly used character sets, or need to use different character set for keyboard, buffer, and display. An overkill if you just need ISO 8859 Latin 1. Currently in alpha test, but available by ftp from the host `ftp.iesd.auc.dk` in '`/pub/emacs-lisp`' or '`/pub/emacs-lisp/alpha`'.

  To get dead keys for TeX, install remap and insert the following in your '`.emacs`' or '`site-start.el`' file.

```
(require 'remap)

(defvar all-dead-keys "~'`^"
  "Dead keys used by remap")

(remap-define-map "Dead Key"
  (apply 'append (mapcar 'remap-dead-map all-dead-keys)))

(remap-define-map "TeX Dead Key"
  (remap-map "Dead Key" (remap-add "Ascii" "~TeX")))

(setq remap-setup-alist
      '(("7-bit"  "Raw"          "L1" "US" "Ctrl" "~TeX")
        ("8-bit"  "Raw"          "L1" "L1" "Ctrl" "Raw")
        ("Dead/7" "Dead Key"     "L1" "US" "Ctrl" "~TeX")
        ("Dead/8" "Dead Key"     "L1" "L1" "Ctrl" "Raw")
        ("TeX"    "TeX Dead Key" "L1" "US" "Ctrl" "Raw")))
```

  You can now enable TeX dead keys with

```
M-x remap-setup-choose RET TeX RET
```

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-tex.el'
: This file automatically converts between ISO 8859 Latin 1 encoding and LaTeX encodings of West European characters. It is available by ftp from `aida.intellektik.informatik.th-darmstadt.de` in the directory '/pub/gene/Emacs'.

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, and AUC TeX can be used with both, as well as with the two Japanese-aware Emacses, NEMACS and MULE.

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.

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 you compare 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

If not nil, search TEX directories recursively.

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 if 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 general 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 to 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:

(require 'tex-auto)
```

```
(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 new entry to `TeX-newmacro-regexp`. Declare variable to contain intermediate data during parsing. Add hook to be called before and after parsing. In this case, the hook before parsing just initialize the variable, and the hook after parsing collect the data from the variable, and add 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

Before you do anything else, make sure that you have the latest version of TEXinfo installed. It is available by ftp from '`prep.ai.mit.edu`'. You need at least version 2.16.

First, you should consult the file '`tex-site.el`' and edit it to fit your local site. Be sure to get the following two variables right, or you will not be able to complete the installation procedure:

TeX-lisp-directory User Option

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

TeX-macro-global User Option

Directories containing the site's TEX style files.

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.

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

The best way to change `TeX-macro-global`, `TeX-command-list`, `TeX-printer-list` and the other customization variables is to copy their definitions from '`tex.el`' to '`tex-site.el`' and change them there. This way, when you upgrade AUC TEX you can keep the old '`tex-site.el`' file so you don't need to customize it again.

Now copy '`tex-site.el`' to somewhere where emacs can find it. For FSF Emacs 19 the best place is in the '`.../emacs/site-lisp/`' directory. For Lucid Emacs and Emacs 18 you can copy it to the '`.../emacs/lisp/`' directory, just remember to move the file when you install a new version of Emacs.

Alternatively, you can insert the code from '`tex-site.el`' directly into your '`.emacs`' or '`site-start.el`' file, if you want to save the overhead of opening another file.

Next, edit the file '`Makefile`' in the AUC TEX directory to set up paths for installation of the files. Be sure that

`bindir` Is set to the directory where you want the LaCheck binary to be installed.

`infodir` Is set to the directory where you want the AUC TEX "info" documentation to be installed.

`aucdir` Is set to the directory where you want the lisp files for AUC TEX to be installed. This **must** be the same directory as you specified for `TeX-lisp-directory` in '`tex-site.el`'. If you unpacked AUC TEX in that directory (`TeX-lisp-directory`), you should set `aucdir` to '`.`' in order to avoid copying the byte compiled lisp files.

If you already have an old version of AUC TEX in that directory, you must delete it before installing the new version. Otherwise the wrong version of the files may be loaded during the byte compilation.

`mandir` Is set to the directory where you want to install the (unformatted) man page for LaCheck.

`CC` Is the compiler you want to use for compiling the LaCheck program. Is set to 'gcc' by default. Change this is 'cc' or whatever your system compiler is called if you do not have GCC installed.

Then type `make all`, and the whole thing will be 'made'. If this succeeds, type `make install`, and it will be installed.

Now, tell the users to insert

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

in their '`.emacs`' files if they want to use AUC TEX. If you use FSF Emacs 19, you can do this for them by inserting the above line in the '`.../emacs/site-lisp/site-start.el`' file.

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`'.

Appendix A The History of AUC TEX

This chapter is for old users, who want to see what has been added in this version, and for the authors, who get nostalgic at times.

A.1 News in 9.0

Coordinater: Per Abrahamsen, 1994.

Alpha testers (in order of appearance): Volker Dobler ‘<dobler@etu.wiwi.uni-karlsruhe.de>’, Piet van Oostrum ‘<piet@cs.ruu.nl>’, Frederic Devernay ‘<Frederic.Devernay@sophia.inria.fr>’, Robert Estes ‘<estes@ece.ucdavis.edu>’, Berwin Turlach ‘<berwin@core.ucl.ac.be>’, Tim Carlson ‘<imsgtcar@mathfs.math.montana.edu>’, Peter Thiemann ‘<thiemann@provence.informatik.uni-tuebingen.de>’, Kevin Scott ‘<scottkj@prl.philips.co.uk>’, Lawrence R. Dodd ‘<dodd@roebling.poly.edu>’, Johan Van Biesen ‘<vbiesen@wins.uia.ac.be>’, Marc Gemis ‘<makke@wins.uia.ac.be>’, Michelangelo Grigni ‘<mic@cs.ucsd.edu>’, Kevin Scott ‘<scottkj@prl.philips.co.uk>’, Peter Paris ‘<pparis@bass.gmu.edu>’, Peter Barth ‘<barth@mpi-sb.mpg.de>’, Andy Piper ‘<ajp@eng.cam.ac.uk>’, Richard Stanton ‘<stanton@haas.berkeley.edu>’, Christoph Wedler ‘<wedler@fmi.uni-passau.de>’, Graham Gough ‘<graham@cs.man.ac.uk>’, and probably more.

- Fixed problem with ‘filladapt’ defeating LaTEX mode’s own indentation algorithm. Thanks to Piet van Oostrum ‘<piet@cs.ruu.nl>’.
- Made environments and sections selectable from the menu bar.
- Support Emacs comment conventions. Comments starting with a single ‘%’ are indented at comment-column. Comments starting with ‘%%%’ are indented at column 0. Comments starting with ‘%%’ are indented like ordinary text. You can set the variables LaTeX-right-comment-regexp and LaTeX-left-comment-regexp to nil to disable this. See also LaTeX-ignore-comment-regexp. Rewrote it from Christoph Wedler ‘<wedler@fmi.uni-passau.de>’ from original code.
- TAB and LFD will not indent code in ‘verbatim’ environment if you set LaTeX-indent-environment-check to non-nil. Thise was also first implemented by Christoph Wedler ‘<wedler@fmi.uni-passau.de>’.
- You can now get get custom indentation for various environments. See LaTeX-indent-environment-list.
- C-c C-m left RET new automatically inserts a matching ‘\right’. See variables TeX-left-right-braces, TeX-braces-default-association, and TeX-braces-user-association. This feature was suggested by Jesse ‘<jes@math.msu.edu>’ and implemented by Berwin Turlach ‘<berwin@core.ucl.ac.be>’.
- Don’t automatically display the compilation buffer unless TeX-show-compilation in non-nil. Suggested by Stefan Schoef ‘Stefan.Schoef@arbi.informatik.uni-oldenburg.de’.
- Bundled ‘ltx-help.el’. Press C-h C-l to get the documentation for a LaTeX command.
- Fixed indentation of ‘\left’ and ‘\right’. Thanks to Christoph Wedler ‘<wedler@fmi.uni-passau.de>’.

- Installation procedure changed. 'tex-site.el' is now intended to survive AUC TEX upgraded. The distribution version only contains autoloads (eliminating the need for 'tex-load.el', the customization variables are moved to 'tex.el'. You should copy those variables you need to customize from 'tex.el' to 'tex-site.el'.
- Made it possible to use absolute paths when including bibliographies or style files.
- Fixed problem with parsing errors after running LaTEX on the region from the menu. Thanks to Peter Barth '<barth@mpi-sb.mpg.de>' for finding this one.
- The file 'doc/ref-card.texi' has been renamed to 'doc/tex-ref.texi' to avoid confusion with the reference card for GNU Emacs. Suggested by Michelangelo Grigni '<mic@cs.ucsd.edu>'.
- The 'README', 'CHANGES' and 'INSTALLATION' files are now generated from chapters of this manual, to ensure they stay in sync.
- M-TAB will now complete some macro arguments in addition to macro names. In particular, if you press M-TAB after '\cite{' or '\ref{' you will get completion for bibitems and labels, respectively.
- Merged a number of files. The major files are now 'tex.el' for plain TEX and common support, 'tex-buf.el' for running external commands, and 'latex.el' for LaTEX support.
- Unbundled 'outln-18.el'. Users of Emacs 18 or Lucid Emacs 19.9 or earlier must get 'outln-18.el' and install it as 'outline.el' if they want the outline commands to work.
- No longer bind C-c TAB to TeX-complete-symbol. Use M-TAB instead.
- Cleaned up the parser and parametrizised it. Now you can add now types of information to be maintained by the parser simply by calling TeX-auto-add-type. You still need to install the regexps with TeX-auto-add-regexp.
- Disable the automatic insertion of empty braces after macros with no arguments in LaTeX-math-mode. Added a variable TeX-insert-braces to disable it everywhere.
- Now complete with existing labels when asking for a label in a LaTEX environment. Suggested by Berwin Turlach '<berwin@core.ucl.ac.be>'.
- The variables TeX-private-macro, TeX-private-auto, and TeX-private-style are now initialized from the 'TEXINPUTS' and 'BIBINPUTS' environment variables.
- C-c C-f and C-c C-e will now put the template around the region if the region is active.
- Fixed C-u C-c C-e to handle environments ending with a star ('*'). Reported by Berwin Turlach '<berwin@core.ucl.ac.be>'.
- Don't use with-output-to-temp-buffer for compilation buffer. Fixed by Frederic Devernay '<Frederic.Devernay@sophia.inria.fr>'.
- New function TeX-command-buffer (C-c C-b) to run a command on the (visible part of) the current buffer. Requested by several people.
- Bundled the latest 'reporter.el', added TeX-submit-bug-report to menus.
- TeX-insert-braces now takes an argument like insert-parentheses. Thanks to Lawrence R. Dodd '<dodd@roebling.poly.edu>'.
- Fixed bug in '\put' and '\multiput' macros. Thanks to Kevin Scott '<scottkj@prl.philips.co.uk>'.
- Deleted ams-latex-mode, slitex-mode, and foiltex-mode. Instead, use LaTeX-command-style to determine the name of the external command to use.

- Deleted latex2e-mode. Instead set the LaTeX-version variable. This may be done automatically if you use '\documentclass' in the future.
- Fixed Lucid Emacs menu for TEXinfo mode. Thanks to Frederic Devernay '<Frederic.Devernay@sophia.inria.fr>',
- Added support for 'harvard.sty' by Berwin Turlach '<berwin@core.ucl.ac.be>'.
- Filling will not let display math equations '\[... \]' be on a line by themselves. Reported by Matthew Morley '<Matthew.Morley@gmd.de>'.
- Made words-include-escapes default to nil.
- Made TeX-expand-list expansions case sensitive. Thanks to Havard Rue '<Havard.Rue@sima.sintef.no>'.
- Fixed error in calculating indentation for lines starting with a brace. Thanks to Piet van Oostrum '<piet@cs.ruu.nl>'.
- Fixed bug in the 'addcontentsline', 'newtheorem', and 'pagenumbering' macros reported by Berwin Turlach '<berwin@core.ucl.ac.be>'.
- Doc fixes by Lawrence R. Dodd '<dodd@roebling.poly.edu>'.
- Indentation no longer fooled by '\\{', Thanks to Peter Thiemann '<thiemann@provence.informatik.uni-tuebingen.de>'.
- Bind M-C-e and M-c-a to LaTeX-find-matching-end and LaTeX-find-matching-begin. Suggested by Lawrence R. Dodd '<dodd@roebling.poly.edu>'.
- Added variable TeX-quote-after-quote which causes TeX-insert-quote to insert literal '"' except when after another '"', in which case it will expand to TeX-open-quote or TeX-close-quote. This code was contributed by Piotr Filip Sawicki '<pfs@mimuw.edu.pl>'.
- Added support for Polish style files 'plfonts.sty' and 'plhb.sty', contributed by Piotr Filip Sawicki '<pfs@mimuw.edu.pl>'.
- Added section with suggestions for how to handle European character sets.
- Created workaround for bug in the regexp handler in some Emacs 18 versions and older versions of Lucid Emacs. The workaround means you cannot use space in the documentstyle command in Emacs and Lucid Emacs earlier than version 19.9.
- 'powerkey.el' is removed since the functionality is integrated in GNU Emacs
- C-u " now inserts four literal '"', not just one. To insert a single '"' either press " twice or use C-q ".
- Allow non-string value for outline-minor-map-prefix. Reported by David Smith '<maa507@comp.lancs.ac.uk>'.
- Make the use of write-file-hooks more safe, and use local-write-file-hooks when possible. Suggested by David Smith '<maa507@comp.lancs.ac.uk>'.
- Don't indent '\begin{verbatim}' and '\end{verbatim}', since any space before '\end{verbatim}' is significant. Thanks to Peter Thiemann '<thiemann@informatik.uni-tuebingen.de>' for the patch.
- Show available fonts when you try to insert an non-existing font. Suggested by David Smith '<maa507@comp.lancs.ac.uk>'.
- The member function in 'tex-18.el' does not depend on TeX-member now. Thanks to Piet van Oostrum '<piet@cs.ruu.nl>'.

- Do not overwrite any global binding of `M-ret`. Suggested by Jens Petersen '<J.Petersen@qmw.ac.uk>'.
- Major modes for writing text are supposed to rebind `M-tab` to `ispell-complete-word`. Reported by Jens Petersen '<J.Petersen@qmw.ac.uk>'.
- Fixed problems with TeXinfo menus. Thanks to David Smith '<maa507@comp.lancs.ac.uk>' for reporting this.
- Code cleanup. Removed the '`format`' directory, as it did not make it easier to add new TeX modes, quite the contrary.
- Fixed name conflict in '`auc-tex.el`', reported by Rik Faith '<faith@cs.unc.edu>'.
- Fixed some spelling errors. Thanks to Lawrence R. Dodd '<dodd@roebling.poly.edu>'.
- Fixed bug prohibiting non-standard file extensions. Now recognize '`.ltx`' by default. Suggested by Lawrence R. Dodd '<dodd@roebling.poly.edu>'.
- Name of the AUC TeX info files changes once again to be useable under DOS. This time simply to '`auctex`'.
- Documented `TeX-outline-extra`.
- Could not select command on region from the menu before loading '`tex-buf`'. Reported by Uwe Bonnes '<bon@lte.e-technik.uni-erlangen.de>'.
- Make the hilit19 interface more robust. Thanks to William Dean Norris II '<wdn@dragonfly.cis.ufl.edu>'.
- More OS/2 Makefile fixes by Bodo Huckestein '<bodo@eu10.mpi-hd.mpg.de>'.
- Reimplemented comment support on top of `comment-region`, giving slightly different semantics.

A.2 News in 8.0

Coordinater: Per Abrahamsen, 1993.

Alpha testers (in order of appearance): Marc Gemis '<makke@wins.uia.ac.be>', Shinji Kobayashi '<koba@flab.fujitsu.co.jp>', Philippe Defert '<defert@gnuisance.cern.ch>', Richard Stanton '<stanton@haas.berkeley.edu>', Norbert Kiesel '<norbert@i3.informatik.rwth-aachen.de>', Roberto Cecchini '<CECCHINI@fi.infn.it>', Hanno Wirth '<wirth@igd.fhg.de>', Tim Carlson '<tim@math.montana.edu>', John Daschbach '<dasch@darkwing.uoregon.edu>', Bob Fields '<bob@minster.york.ac.uk>', Peter Whaite '<peta@cim.mcgill.ca>', Volker Dobler '<dobler@etu.wiwi.uni-karlsruhe.de>', Phil Austin '<phil@geog.ubc.ca>', Martin Maechler '<maechler@stat.math.ethz.ch>', Havard Rue '<Havard.Rue@sima.sintef.no>', Tim Geisler '<tmgeisle@faui80.informatik.uni-erlangen.de>', Tim Carlson '<imsgtcar@mathfs.math.montana.edu>', Sridhar Anandakrishnan '<sak@essc.psu.edu>', Peter Thiemann '<thiemann@provence.informatik.uni-tuebingen.de>', Pedro Quaresma '<pedro@mat.uc.pt>', Christian Lynbech '<lynbech@daimi.aau.dk>', Kevin Scott '<scottkj@prl.philips.co.uk>', Bodo Huckestein '<bodo@eu10.mpi-hd.mpg.de>', Cengiz Alaettinoglu '<ca@cs.UMD.EDU>', Jakob Schiotz '<schiotz@fysik.dth.dk>', and probably more.

- New variable `LaTeX-letter-sender-address` contains default address for use with the letter style. Set it to the address of your organization in '`tex-site.el`'. Thanks to Sridhar Anandakrishnan '<sak@essc.psu.edu>'.

- Makefile now works under OS/2 with GNU Make. Thanks to Bodo Huckestein 'bodo@eu10.mpi-hd.mpg.de'.
- Made it possible to install global auto files without having BibTeX mode installed. Thanks to Christian Lynbech '<lynbech@daimi.aau.dk>'.
- Minor documentation fixes. Thanks to Martin Maechler '<maechler@stat.math.ethz.ch>'.
- Added support for 'eqref' for the 'amsart' style. Thanks to Peter Whaite '<peta@cim.mcgill.ca>'.
- Use '-c' as the default shell command option under 'emx'. Eberhard Mattes '<mattes@azu.informatik.uni-stuttgart.de>' says it is better than '/c'.
- Made powerkey in the menus work better under OS/2. Thanks to Eberhard Mattes '<mattes@azu.informatik.uni-stuttgart.de>'.
- Made the reference cards print correctly on US letter format paper. Thanks to Magnus Nordborg '<magnus@fisher.Stanford.EDU>'.
- LaTeX-dead-mode removed. Read the file 'ISO-TEX' for alternative solutions.
- All minor modes unbundled. You can find them from ftp at 'ftp.iesd.auc.dk' in the directory '/pub/emacs-lisp'. Removed information about minor modes from this document.
- New hooks for changing ispell directory, see 'tex-site.el' for details.
- LaTeX2e mode now supported. Insert

  ```
  (setq TeX-default-mode 'latex2e-mode)
  ```

  in your '.emacs' file to get documentclass instead of documentstyle per default. The parser recognizes documentclass, usepackage, and newcommand with a default argument. There are also templates for all of them.
- Added Jakob Schiotz's '<schiotz@fysik.dth.dk>' help file for installing AUC TeX on OEMACS. It will probably also be of interest for DEMACS users.
- Minor changes to be more friendly for OEMACS, thanks to Jakob Schiotz '<schiotz@fysik.dth.dk>'.
- The control key bindings in LaTeX-math-mode now works, thanks to Frederic Devernay '<Frederic.Devernay@sophia.inria.fr>'.
- LaTeX outlines no longer matches '\partial' or other commands with a sectioning command as prefix. Thanks to Jakob Schiotz '<schiotz@fysik.dth.dk>'.
- LaTeX-fill-paragraph now handles the case where the previous line both contain an '\item' and an unmatched open brace. Thanks to Piet van Oostrum '<piet@cs.ruu.nl>'.
- Use abbreviated file name for TeX output buffers in Emacs 19. Thanks to Jens Gustedt '<gustedt@math.tu-berlin.de>'.
- Added lowercase alias for LaTeX-math-mode for use with Emacs file mode commands. Thanks to Olaf Burkart '<burkart@zeus.informatik.rwth-aachen.de>'.
- Added code to reuse old region in TeX-command-region if mark is not active. Thanks to Cengiz Alaettinoglu '<ca@cs.UMD.EDU>'.
- Now get keyboard accelerators on all menus rather than only AUC TeX menus, thanks to the 'powerkey.el' file by Lars Lindberg '<lli@sypro.cap.se>'.
- Added TeX-electric-macro for faster completion of TeX macros. See Section 3.2 [Completion], page 11.

- Comparing printer names are now case incentive. Thanks to Richard Stanton ‘<stanton@haas.berkeley.edu>’.
- Default shell fixed for OS/2. Thanks to Richard Stanton ‘<stanton@haas.berkeley.edu>’.
- Added functions to hide (LaTeX-hide-environment) and show (LaTeX-show-environment) the current environment.
- C-u C-c C-e will now modify the current environment instead of inserting a new environment. This is like the optional argument to the font commands.
- Added nabla to LaTeX Math Mode. Suggested by Bill Reynolds ‘<bill@goshawk.lanl.gov>’.
- Added commands for running TeX and LaTeX interactively. Thanks to David Carlisle ‘<carlisle@computer-science.manchester.ac.uk>’.
- The external commands will now insert there output *before* point in the output buffers. This allows you to follow the progress by putting point at the end of the file. Suggested by Jak Kirman ‘<jak@cs.brown.edu>’.
- When invoking an external command from a menu, the document will be automatically saved.
- There are now a printer menu for emacs 19.
- Redesigned dependecy checking. Now only checks dependencies for files loaded in the current emacs session. This is much faster, but will not catch files that are edited outside this emacs session, or files edited in killed buffers. **TeX-check-path must at least contain ‘.’ for saving to work**. If you have set ‘TeX-check-path’ in your ‘.emacs’ file, remove it. The default value is fast enough now.
- New variable TeX-save-query control if AUC TeX will query you for each modified buffer when you save the document. Set it to nil to get rid of these questions. Setting this variable also affect the automatic saving of the document that happens each time you start an external command.
- New command TeX-save-document will save all files in the current document, i.e. the document associated with the current buffer.
- Cleaned up all minor modes, also made them use ‘min-map.el’ or ‘min-mode.el’ instead of ‘min-bind.el’.
- Cleaned up release management.
- AUC TeX will not longer be confused when you rewrite a file under a new name.
- Lots of code cleanup, involving reformating the source and renaming all ‘-hook’ variables and functions to conform with the Emacs 19 guidelines.
- Can now parse Japanese characters in labels and macros when you use Japanese TeX. Thanks to Shinji Kobayashi ‘<koba@flab.fujitsu.co.jp>’.
- Made it safe to quit when AUC TeX asks for the name of the master file. It will simply assume the file itself is the master, and continue without inserting any file local variables.
- Support for epsf and psfig style files. Thanks to Marc Gemis ‘<makke@wins.uia.ac.be>’.
- Support for LaTeXinfo. Thanks to Marc Gemis ‘<makke@wins.uia.ac.be>’.
- Only examine the first 10000 bytes to find out what TeX mode to use.
- Added TeX-submit-bug-report command to submit bug reports. It uses the ‘reporter.el’ distributed with SuperCite, so it may not be available in some Emacs 18 installations.
- Speeded up parsing significantly by using a simpler regexp.

- Added variable `TeX-auto-untabify`. Set it to nil to prevent untabifying the buffer when it is saved. Several people wanted this.
- Changed defaults to *not* do any automatic parsing, nor prompt for a master file. See Chapter 6 [Parsing Files], page 21, Chapter 5 [Multifile], page 19, for information about how ot correct this. In short, insert the following in your '`.emacs`' file.

  ```
  (setq TeX-auto-save t)
  (setq TeX-parse-self t)
  (setq-default TeX-master nil)
  ```
- Some grammatical fixes to the '`PROBLEMS`' file. Thanks to Lawrence R. Dodd '`<dodd@roebling.poly.edu>`'.
- No longer install a separate `outline-minor-mode` by default, as the FSF Emacs 19.19 `outline-minor-mode` is adequate. The included file '`outln-18.el`' emulates the FSF Emacs 19.19 mode under Emacs 18.

  The FSF Emacs 19.19 `outline-minor-mode` use the `C-c` prefix instead of `C-c C-o` by default, and does not bind as many keys as the `outline-minor-mode` distributed with earlier versions of AUC TeX did. You can get the keybindings back together with other goodies by inserting

  ```
  (require 'out-xtra)
  ```

  in your '`.emacs`' file. '`out-xtra.el`' will probably be unbundled from AUC TeX in the future.
- Some fixes to AmS-TeX mode by Ulf Juergens '`<ulf@mathematik.uni-bielefeld.de>`'.
- Make '`plain-TeX-mode-menu`' work in Lucid Emacs. Thanks to Anthony Rossini '`rossini@hsph.harvard.edu`' for reporting this.
- First cut on a TeXinfo mode.
- More strict about parsing '`\bibitem`''s and BibTeX entries.
- Made it easier to write style files for environments that takes arguments and documented it. Suggested by Martin Wunderli '`<wunderli@inf.ethz.ch>`'.
- Parse optional argument to '`\newenvironment`'. Suggested by Martin Wunderli '`<wunderli@inf.ethz.ch>`'.
- Fixed '`parbox`' macro. Thanks to Shinji Kobayashi '`<koba@flab.fujitsu.co.jp>`'.
- Made the parser work better in outline minor mode. Thanks to Salvador Pinto Abreu '`<spa@khosta.fct.unl.pt>`'.
- Also save style information with `TeX-normal-mode` when buffer not modified.
- Use `$(MAKE)` instead of '`make`' to invoke `make` from the '`Makefile`'. Thanks to John Interrante '`<interran@uluru.Stanford.EDU>`'.
- Make last value default for `TeX-insert-macro`. Suggested by Matt Fairtlough '`<M.Fairtlough@dcs.sheffield.ac.uk>`'.
- Renamed info file to '`auc-info`' in order to fil DOS file limits. Please remember to update your '`dir`' file to reflect this change.
- Delete auto file instead of saving an empty file.

A.3 News in 7.3

Coordinater: Per Abrahamsen, 1993.

- More robust installation, especially for Lucid Emacs (I hope). Many people reported problems with this.
- Make 'easymenu' work when byte-compiled. Many people reported this bug.
- Minimally updated the 'README' file from version 6.0 (sigh). Thanks to Boris Goldowsky '<boris@cs.rochester.edu>' for reporting this.
- Added '@finalout' to manual. Reported by Henrik Drabol '<hvd@ens004.ens.min.dk>'.
- Fixed M-q to work after an '\end{. . .}'. It will not work at the end of the buffer, but there are usually the local variables so it should (hopefully) not matter. Thanks to Shinji Kobayashi '<koba@flab.fujitsu.co.jp>' again.
- New variables TeX-open-quote and TeX-close-quote determine what is inserted by TeX-insert-quote. The 'german' style file now use those variables instead of changing the keymap.
- Changes to the default settings in 'tex-site.el', in particular a 'Queue' command is added to display the print queue. Thanks to John Interrante '<interran@uluru.Stanford.EDU>' for code, and other members of the 'auc-tex@iesd.auc.dk' mailing list for ideas.
- Make sure all outline mode commands are bound in outline-minor-mode.
- Added autoload for TeX-command. Thanks to Hanno Wirth '<wirth@igd.fhg.de>' for reporting this.
- Added support for AmSTeX and AmSLaTeX. Currently they are identical to TeX and LaTeX except for another default command.
- Added VorTeX style matching of dollar sign. The style is guaranteed to be VorTeX, since I lifted the code directly from VorTeX. Thanks to Pehong Chen '<phc@renoir.berkeley.edu>' for writing the VorTeX code. Thanks to Jak Kirman '<jak@cs.brown.edu>' for pointing out this nice VorTeX feature.
- Added information about AUC TeX mail addresses to the manual. Thanks to Dave Smith '<maa507@computing.lancaster.ac.uk>'.
- Added menu to for plain TeX. Suggested by Tim Carlson '<imsgtcar@math.montana.edu>'.
- Made the menus depend on TeX-command-list.
- Made it possible to specify TeX-auto-regexp-list in the local variable section of each file.
- Added variable TeX-auto-parse-length to specify maximal length of text that will be parsed.
- Added automatic parsing of BibTeX files and 'bibitem' entries in order to get completion in 'cite'. This was inspired by an add on made by Sridhar Anandakrishnan '<sak@essc.psu.edu>'.
- Added variable TeX-byte-compile to disable automatic byte compilation of style files when loaded. This is needed when using different Emacs versions.
- Added variable TeX-translate-location-hook to translate file and line information before showing an error, as requested by Thorbjoern Ravn Andersen '<ravn@imada.ou.dk>'.
- Added variable TeX-auto-save to allow disabling the automatic saving of style information, either per file in the file local variables, or globally by using setq-default. Use TeX-normal-mode to force style information to be saved.
- Try to create 'auto' directory if it does not exists.
- Added chapter describing how to tune the TeX parsing.
- Allow (but do not encourage) a string value for LaTeX-default-options.

- Give ‘"’ word syntax when german.sty is loaded. Suggested by Tim Geisler ‘<tmgeisle@immd8.informatik.uni-erlangen.de>’.
- Many corrections to the grammar in the manual. Thanks to Manfred Weichel ‘<Manfred.Weichel@sto.mchp.sni.de>’.
- Bind TeX-home-buffer to C-c ^ instead of C-c C-h which are reserved in Emacs 19. Suggested by Chris Moore ‘<Chris.Moore@src.bae.co.uk>’.

A.4 News in 7.2

Coordinater: Per Abrahamsen, 1993.

- LaTeX-dead-mode works again. Thanks to Patrick O'Callaghan ‘<poc@usb.ve>’ for fixing it.
- Minor fixes to the documentation. Thanks to Shinji Kobayashi ‘<koba@flab.fujitsu.co.jp>’.
- Add ‘Compiling’ to the mode line of all buffers, while there is a AUC TeX compilation process running. This is similar to the behavior of compile in Emacs 19.
- TeX-normal-mode will now save the buffer first to make sure it gets reparsed.
- Labels with underscores are now recognized. Thanks to Wolfgang Franzki ‘<W.Franzki@kfa-juelich.de>’
- Fix to ‘ghostview’ printer specification. Thanks to Masayuki Kuwada ‘<kuwada@soliton.ee.uec.ac.jp>’.
- Recognize ‘abstract’, ‘center’, ‘titlepage’, ‘verse’, and ‘theindex’ environments. Thanks to Masayuki Kuwada ‘<kuwada@soliton.ee.uec.ac.jp>’.
- Fix to ‘newsavebox’ macro. Thanks to Shinji Kobayashi ‘<koba@flab.fujitsu.co.jp>’ for reporting this.
- Menu support for GNU Emacs 19 and Lucid Emacs. Thanks to Alastair Burt ‘<burt@dfki.uni-kl.de>’ for the initial Lucid Emacs version.
- C-c C-f C-d now deletes the current font. The current font is defined to be the innermost TeX group starting with a TeX macro that is terminated by a space.
- Giving C-c C-f a prefix argument will replace the current font, i.e. C-u C-c C-f C-b will change the current font to bold.

  The old functionality (putting the font around the region) has been removed. To make the region bold, type C-w C-c C-f C-b C-y instead.
- Chapter recognized as largest heading in the report style. Thanks to Shinji Kobayashi ‘<koba@flab.fujitsu.co.jp>’ for reporting this.
- More support for Japanese style files. Thanks to Shinji Kobayashi ‘<koba@flab.fujitsu.co.jp>’.
- No longer put ‘Outline’ in the mode line whenever selective-display is set. Thanks to Lawrence R. Dodd ‘<dodd@roebling.poly.edu>’ for reporting this.
- Support for inserting calligraphic letters in TeX-math-mode with ‘ c LETTER. Thanks to Olaf Burkart ‘<burkart@zeus.informatik.rwth-aachen.de>’.
- set-docstring in ‘tex-math.el’ should work better now. Thanks to Alastair Burt ‘<burt@dfki.uni-kl.de>’ and Olaf Burkart ‘<burkart@zeus.informatik.rwth-aachen.de>’.

- Support for dviout preview on PC-9801. Thanks to Shinji Kobayashi '<koba@flab.fujitsu.co.jp>'.
- Inserting environment in empty buffer should work now. Thanks to Alastair Burt '<burt@dfki.uni-kl.de>'.
- Default float for figures changed from 'tbp' to 'htbp'.
- LaTeX-format-environironment may work now. Thanks to Shinji Kobayashi '<koba@flab.fujitsu.co.jp>'.
- Better LaTeX-close-environment. Thanks to Thorbjoern Hansen '<thansen@diku.dk>'.
- Some support for Ispell 4.0.
- BibTEX in Emacs 19 need tex-insert-quote, make it autoload from AUC TEX instead of the standard tex-mode.
- TeX-auto-generate
  failed when repeated. Thanks to Peter Whaite '<peta@Thunder.McRCIM.McGill.EDU>' for reporting this.

A.5 News in 7.1

Coordinater: Per Abrahamsen, 1993.

- Allow multiple '%p' in print commands.

  Suggested by Cliff Krumvieda '<cliff@cs.cornell.edu>'.
- Improved backward compatibility in 'auc-tex.el'. Thanks to Ralf Handl '<handl@cs.uni-sb.de>'.
- New style hook for 'german.sty'.

  Disable smart quotes. Press C-c C-n to make it take effect.
- Allow files to have other extensions than "tex".

  But no longer allow files to have multiple dots. Sigh.
- Will no longer parse the buffer if it can use the saved state.
- New variable TeX-parse-self.

  Set it to nil if you never want to parse the buffer when you load it.
- Only offer to save files that belongs to the document.

  When you format the document with C-c C-c, AUC TEX will no longer offer to save your 'RMAIL', '.newsrc', or other files that does not belong to the document. Suggested by Jim Hetrick '<hetrick@phys.uva.nl>'.
- FoilTEX support.

  Thanks to Sven Mattisson '<sven@tde.lth.se>'
- Smarter about when you need to reformat.

  Thanks to Chris Callsen '<chris@iesd.auc.dk>'.
- Japanese TEX

  Now supports Japanese TEX. Thanks to Shinji Kobayashi '<koba@keisu-s.t.u-tokyo.ac.jp>'.
- Works again under OS/2 and other case insensitive file systems.

- DEMACS support.

  Thanks to Shinji Kobayashi '<koba@keisu-s.t.u-tokyo.ac.jp>'.
- Better `LaTeX-close-environment`.

  Thanks to Piet van Oostrum '<piet@cs.ruu.nl>'.
- Ispell support.

  Thanks to Piet van Oostrum '<piet@cs.ruu.nl>'.
- Support for Russian letters.

  Thanks to Justin R. Smith '<jsmith@king.mcs.drexel.edu>'.
- SliTEX fixes.

  Many people.
- Fixes for spelling errors.

  Many people.

A.6 Version 7.0

Coordinator: Per Abrahamsen, 1993.

Alpha testers (in order of appearance): Piet van Oostrum '<piet@cs.ruu.nl>', Sven Mattisson '<sven@tde.lth.se>', Tim Geisler '<tmgeisle@immd8.informatik.uni-erlangen.de>', Fran E. Burstall '<F.E.Burstall@maths.bath.ac.uk>', Alastair Burt '<burt@dfki.uni-kl.de>', Sridhar Anandakrishnan '<sak@essc.psu.edu>', Kjell Gustafsson '<kjell@sccm.Stanford.EDU>', Uffe Kjaerulff '<uk@iesd.auc.dk>', Kurt Swanson 'Kurt.Swanson@dna.lth.se', Mark Utting '<marku@cs.uq.oz.au>', Per Norman Oma 'perno@itk.unit.no', Naji Mouawad '<nmouawad@math.uwaterloo.ca>', Bo Nygaard Bai '<bai@iesd.auc.dk>', and probably more.

- New keymap.

  The keymap has been changed in order to make it more intuitive to new users, and because the old bindings did not work well with the new buffer manipulation commands in tex-buf.el. To use the new bindings, load 'tex-init.el' instead of 'auc-tex.el'.

  The file 'auc-tex.el' is still available and implements the old keybinding on top of the new code.

  Print out the reference card ('doc/tex-ref.tex') to see the new bindings.
- Completely redesigned the buffer handling.

  No part of the interface or the customization variables remain the same, unless you use the compatibility functions in 'auc-tex.el'. In that case the interactive commands remain similar in spirit, but the customization interface is still changed.

  The file 'tex-buf.el' has been completely rewritten, and there are major cleanup in 'tex-dbg.el', however the basic functionality remains the same in this file. The code for both 'tex-buf.el' and 'tex-dbg.el' should be much simpler now and easier to extent.

  'auc-tex.el' and 'tex-site.el' was updated to support the new interface. I actually believe the moral equivalent to `TeX-region` to work now `:-)`, at least I understand the code now.

  The two major functions are now `TeX-command-master` and `TeX-command-region`. Each function will prompt you for the command to execute. AUC TEX will make an educated guess on what command you want to run, and make that the default. The available commands are defined in the variable `TeX-command-list`.

`TeX-command-master` will run the specified command on the buffers master file. You can have one command running for each master file. `TeX-command-region` will run the specified command on the current region, getting the header on trailer from the master file.

You can have exactly one region command running, independent on how many master file commands that are running. Commands that operate on the active process (like `TeX-next-error`) will chose the process associated with buffers master file, unless the last region process is more recent than all master file processes.

AUC TEX now insist on knowing the master file for a buffer. If you do not specify it in the file variable section, and it is not obviously a master file itself, it will ask you. It will also add the master file name to the file variables, unless you disable this feature by setting `TeX-add-local` to nil. Furthermore, it will convert '`%% Master:`' lines to file variables, unless you disable it by setting `TeX-convert-master` to nil.

Functionality removed (for now, it might appear again latter) include all other functions to start a command (e.g. `LaTeX-BibTeX`), and alternative ways to specify headers and trailers. The only place to get the header and trailer is from the master file (I can easily change that, if anyone have such needs).

- Style specific code isolated.

  You can now add style specific information to AUC TeX by writing a style file somewhere in TeX-style-path.

  The main code is now organized around this principle.

- Automatically generate style files.

  AUC TEX can now automatically extract information from a TEX file, and will do this when you save a buffer.

- SliTEX mode.

  Just like LaTEX mode, except that the default command to format run on the buffer is '`slitex`'.

- `LaTeX-section` completely general.

  Rewrote '`ltx-sec.el`'.

  - Sectioning level, toc, and title queries can be individually turned off.
  - Label query can be turned on or off for selected sectioning levels.
  - Label prefix can be different for different sectioning levels.
  - If the title (or toc) is empty, point will be positioned there.
  - Users can add new hooks

- `TeX-insert-macro` much smarter.

  It will now prompt for the symbol with completions, and for many symbols it will also prompt for each argument. There are also completion on some of the arguments.

- Fixed center in figure environment.

  Thanks to Thomas Koenig '`<ig25@rz.uni-karlsruhe.de>`'.

- Changed `\M-` to `\e` in all keybindings in order to better support 8-bit input on some GNU Emacs. Thanks to Peter Dalgaard '`<pd@kubism.ku.dk>`'.

Please, implementors of 8-bit input extensions to GNU Emacs. `\M-x` does *not* means `x` *with the 8-bit set*. It means *pressing* `x` *while holding down the* `META` *key*. Some systems (such as X11) are able to tell the different. Thus, even if you implement 256 byte keymaps, `\M-x` should still expand `meta-prefix-char` followed by an `x` in the keymap. This allows you to distinguish pressing `x` while holding down the `META` key from entering a literal 8-bit character.

- Made the outline commands aware of the document style.

  That is, if the document style is '`article`', '`\section`' will be one level below the '`\documentstyle`', while if the style is '`book`', '`\section`' will be three levels below '`\documentstyle`'. This will make `show-children` work better at the top level.
- The makefiles are closer to GNU coding standard.

  They now understand 'prefix' and some other macros.
- Added hooks to be run after list of environments or list of completion names are updated, and also added a hook to be called after each file has been loaded. Thanks to Piet van Oostrum '`<piet@cs.ruu.nl>`'.
- Added '`*`' to lot of `(interactive)` declarations.
- The outline commands are now always accessible from LaTEX mode.
- Generalized the keyboard remapping and double modes.

  These are found in the file '`min-key.el`'.
- Smart Comments.

  Not really, but there are now two comment functions which use their arguments to determine what to do, instead of four functions ignoring their arguments.
- Add outline headers.

  It is now possible to add extra outline headers, by setting the variable `TeX-outline-extra`.
- Smart quotes even smarter.

  If you press `"` twice, it will insert an real double quote instead of two (or four) single quotes. This is consistent with how remapping in '`min-key.el`' is done.
- Automatically untabify buffer when you save it.

  Hands up, everyone who have produced a 'last revision' paper containing an unreadable list of data in the back, because TEX does not understands tabs.
- Call show-all when you change major mode.

  Thanks to Inge Frick's '`<inge@nada.kth.se>`' '`kill-fix.el`' enhancement, outline minor mode can now guarantee that all text is shown when you leave the minor mode, even if you leave the minor mode by changing the major mode.
- Updated documentation for 7.0.

  Also added key, variable, function, and concept indexes, as well as this history section and a new chapter on multifile documents (see Chapter 5 [Multifile], page 19).

A.7 Version 6.1

Coordinator: Per Abrahamsen, 1992.

- `TeX-region` might work now (heard that before?).

  Many people reported this one. Especially thanks to Fran Burstall ‘`<F.E.Burstall@maths.bath.ac.uk>`’ and Bill Schworm ‘`<bill@schworm.econ.ubc.ca>`’.
- The specification format for the TeX command is more general.

  See the documentation for `LaTeX-command` and `plain-TeX-command`.
- The specification format for the preview commands is more general.

  See their respective documentation.
- The specification format for the print command is more general.

  See the documentation for `TeX-print-command`.
- `TeX-args` is marked as obsolete.
- The ‘`"Emergency stop ..."`’ error.

  Some users of old TeX installations got might might be fixed now. Thanks to Philip Sterne ‘`<sterne@dublin.llnl.gov>`’.
- It is now possible to change the preview command.

  ... without loading TeX-site first. Thanks to Tim Bradshaw ‘`<tim.bradshaw@edinburgh.ac.uk>`’.
- New variable TeX-smart-quotes.

  Allow ‘`german.sty`’ users (and others) to disable the mapping of double quote (`"` to ‘``` `` ```’ or ‘`''`’). Thanks to Daniel Hernandez ‘`<danher@informatik.tu-muenchen.de>`’.
- Many minor corrections to the documentation.

  Thanks to Mainhard E. Mayer ‘`<hardy@golem.ps.uci.edu>`’.
- Make test for `HOSTTYPE` case insensitive.

  Thanks to Gisli Ottarsson ‘`<gisli@liapunov.eecs.umisc.edu>`’.
- `TeX-force-default-mode`

  Set to avoid AUC TeX's attempts to infer the mode of the file by itself.

A.8 Version 6.0

Coordinator: Kresten Krab Thorup, 1992.

Preliminary documentation is available in the directory ‘`doc`’. It isn't very well written, but I believe it covers most interesting points. Comments, suggestions, or even rewrites of sections are VERY WELCOME...

LaCheck has been incorporated in the package. The source code for it is available in the directory ‘`lacheck`’ along with the documentation for it. Lacheck may also be used from the command line. It is bound to `C-c $`.

Some minor changes in:

`TeX-region`
: Should work better with ‘`Master:`’ option.

`LaTeX-environment`
: Numerous new hooks added by Masayuki Kuwada.

TeX-command-on-region
: Removed. C-c C-o used for outline-minor-mode instead.

And some additional minor fixes...

A.9 Ancient History

The origin of AUC TEX is 'tex-mode.el' from Emacs 16. Lars Peter Fischer '<fischer@iesd.auc.dk>' wrote the first functions to insert font macros and Danish characters back in 1986. Per Abrahamsen '<abraham@iesd.auc.dk>' wrote the functions to insert environments and sections, to indent the text, and the outline minor mode in 1987. Kresten Krab Thorup '<krab@iesd.auc.dk>' wrote the buffer handling and debugging functions, the macro completion, and much more, including much improved indentation and text formatting functions. He also made the first public release in 1991, and was the main author and coordinator of every release up to and including 6.0.

Thanks should also go to all the people who have been a great help developing the AUC TEX system. Especially all the people on the 'auc-tex' mailing list, who have been very helpful commenting and pointing out weak points and errors.

Some of the contributors are listed below. Others are mentioned in the lisp files or in the History section.

'<dduchier@csi.UOttawa.CA>'
: Denys Duchier

'<ferguson@cs.rochester.edu>'
: George Ferguson

'<simons@ibiza.karlsruhe.gmd.de>'
: Martin Simons

'<smith@pell.anu.edu.au>'
: Michael Smith

'<per@iesd.auc.dk>'
: Per Hagen

'<handl@cs.uni-sb.de>'
: Ralf Handl

'<sven@tde.lth.se>'
: Sven Mattisson

'<kuwada@soliton.ee.uec.ac.jp>'
: Masayuki Kuwada

'<tb06@pl118f.cc.lehigh.edu>'
: Terrence Brannon

'<roseman@hustat.harvard.edu>'
: Leonard Roseman

Special thanks to Leslie Lamport for supplying the source for the LaTeX error messages in the 'tex-dbg.el' file.

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.

- 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.
- `LaTeX-fill-paragraph` gets confused when two blocks of text are separated by an empty comment only.
- 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.
- Explain how to get BibTeX completion for ‘`\cite`’ somewhere.
- Function to check if you are in math mode (between two dollar signs). Suggested by Jan Erik Odegard ‘<odegard@dsp.rice.edu>’
- Allow whitespace more places during parsing, e.g. between macro and the first argument, or between individual style options in ‘`\documentstyle`’.
- 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. Make `,` and `.` remove italic correction.
- 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 occur 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 lightening completion from Ultra TEX is anything for us.
- Finish the TeXinfo mode.
- BibTeX mode.
- Support for AMSLaTeX style files.
- Better LaTeX letter environment.

  Make a database system for addresses in the letter environment 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

$

"

C

L

M

T

Function Index

L

T

Variable Index

Concept Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Short Contents

Table of Contents