LaTeX Editing Support

Nelson H. F. Beebe
Center for Scientific Computing
Department of Mathematics
University of Utah
Salt Lake City, UT 84112
USA
Tel: +1 801 581 5254
FAX: +1 801 581 4148
E-mail: beebe@math.utah.edu

14 September 1993
Version 1.10

Contents

List of Tables

Licensing Information

The program currently being distributed that relates to LaTeX is contained in the file '`latex.el`' in the function `LaTeX-mode` and numerous other support functions. This program is *free*; this means that everyone is free to use it and free to redistribute it on a free basis.

Specifically, we want to make sure that you have the right to give away copies of the programs that relate to `LaTeX-mode`, 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 file '`latex.el`', 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 '`latex.el`'. 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 '`latex.el`' are found in the General Public Licenses that accompany them. The programs that are part of GNU Emacs are covered by the GNU Emacs copying terms (see section License in *The GNU Emacs Manual*), and other programs are covered by licenses that are contained in their source files.

Chapter 1

LaTeX support

1.1 Preliminaries

The material that you are now reading is contained in the distribution as two files, '`ltxmode.ltx`' for the TUGboat article, and '`ltxinfo.ltx`' for LaTeXinfo.

The first job is to arrange for `LaTeX-mode` to be selected automatically when required.

Because the '`latex.el`' file of Lisp code is not yet a standard part of the GNU Emacs distribution, its association with particular file extensions is unknown to Emacs. To remedy that, the Lisp code in Table 1 should be inserted into the '`.emacs`' file in your login directory; that file is processed each time Emacs starts up. The `load` command there tells where '`latex.el`' is found; it may need adjustment for your system.

The peculiar `if` commands attach `LaTeX-mode` to the `auto-mode-alist` variable, which is a list of pairs of file extensions and their mode names; the attachment is omitted if the file extensions are already in the list.

Once this startup code has been executed, Emacs will automatically select `LaTeX-mode` when you visit files with extensions '`.ltx`' (LaTeX document), '`.stx`' (SLiTeX document), or '`.sty`' (LaTeX style). Others can be easily added according to taste. File extension '`.tex`' is already attached to GNU Emacs `tex-mode` which comes with the system, so different extensions are needed for different editing modes.

If you don't have these associations of file extensions with editing modes in effect, you can always select the mode manually with the command `M-x LaTeX-mode`.

Experienced Emacs users may wonder why the `autoload` facility is not

```
;============================================
; Add mode for .ltx files if not there already
; for most people: (load-library "latex.el")
; for my private version:
; Force tex-mode.el to be loaded so latex.el
; can override it.
(load-library "tex-mode")
(load "~/emacs/latex" t t nil)
(if (not (assoc "\\.ltx$" auto-mode-alist))
    (setq auto-mode-alist
          (cons (cons "\\.ltx$" 'LaTeX-mode)
                auto-mode-alist)))
(if (not (assoc "\\.stx$" auto-mode-alist))
    (setq auto-mode-alist
          (cons (cons "\\.stx$" 'LaTeX-mode)
                auto-mode-alist)))
(if (not (assoc "\\.sty$" auto-mode-alist))
    (setq auto-mode-alist
          (cons (cons "\\.sty$" 'LaTeX-mode)
                auto-mode-alist)))

;============================================
; Private letter support
(load "~/emacs/letter" t t nil)
```

Table 1.1: '`.emacs`' file additions.

used instead of the explicit `load` command in the startup file; the reason lies in a conflict with the standard `tex-mode`. I expect that conflict will be removed when '`latex.el`' becomes part of the standard GNU Emacs distribution. A modification of '`tex-mode.el`' that eliminates all LaTeX-specific code has been prepared locally, and that new version removes the incompatibility with '`latex.el`', allowing `autoload` to be used.

In GNU Emacs Lisp, named functions, and variables defined outside functions, are known globally. There is no provision for file scoping or mode scoping to limit name visibility. Consequently, to avoid collision with the thousands of names from other libraries, each library usually includes a distinctive phrase in its global names. Naturally enough, '`LaTeX`' and '`SLiTeX`' are the phrases chosen in '`latex.el`'. They are almost always used as prefixes in variable names, and usually also in function names, although a few functions have embedded phrases for better readability (e.g. `show-LaTeX-labels`).

Tables 2 through 4 show the documentation summary for `LaTeX-mode`, which you can always reproduce online with the function `describe-mode` which is normally bound to the key `C-h m`.

Only major mode functions have such long documentation. Table 5 shows a more typical case, the documentation of the function `LaTeX-index`. You could produce it online with the function `describe-function` on `C-h d`.

Tables 6 and 7 show the key bindings in `LaTeX-mode`; the listings can be produced by the function `describe-bindings` which is normally attached to `C-h b`.

The name and documentation of the function attached to any key can be obtained by invoking the function `describe-key` attached to `C-h k`.

You can change existing key bindings, or add new ones, with the functions `local-set-key` and `global-set-key`, and otherwise modify the behavior of `LaTeX-mode`. See 1.19 [Customizing lists], page 35 for details.

```
LaTeX Mode: Major editing mode for LaTeX and
SLiTeX, with tex-mode underneath.

To create a new LaTeX document, use
make-LaTeX-document (on C-c d), or
make-SLiTeX-document (on C-c s).  With an
argument, they will provide some additional
helpful comments.  The functions will prompt
for document style and options.  Type ? to see
the standard ones; you can enter them with
name completion.  You can also enter your own,
since it is possible to have private styles
and options unknown to make-LaTeX-document or
make-SLiTeX-document.

The most frequent LaTeX construct is the
\begin{}...\end{} grouping; you can generate
it by LaTeX-begin-end-block (on C-c b).  With
an argument, a helpful comment may be printed.
Type ? to see the standard environments, or
enter your own.  If you type a \begin{...}
manually, you can later generate a matching
\end{...} with the function LaTeX-end (on C-c
n).

Other common constructs are the insertion of
labels, citations, cross-references, index
entries, verbatim strings, and additional
items in a list.  The functions
        C-c .   LaTeX-add-word-to-index
        C-c C-b LaTeX-bibitem
        C-c c   LaTeX-cite
        C-c k   LaTeX-counter
        C-c f   LaTeX-footnote
        C-c x   LaTeX-index
        C-c i   LaTeX-item
        C-c l   LaTeX-label
        C-c m   LaTeX-macro
        C-c p   LaTeX-pageref-with-completion
        C-c C-p LaTeX-protect
        C-c r   LaTeX-ref-with-completion
        C-c v   LaTeX-verb
        C-c d   make-LaTeX-document
        C-c s   make-SLiTeX-document
provide for these.
```

Table 1.2: Online summary of LaTeX mode.

```
LaTeX-tab (on C-c TAB) will indent a line to
the current \begin{}...\end{} nesting level.

Most major LaTeX macros can be entered by
LaTeX-macro (on C-c m).  This will supply the
correct set of following braces, brackets, and
parentheses, and with an argument, will insert
a short comment about the expected command
arguments.

For SLiTeX, the function renumber-slides can
be used to put a numbered comment on each
slide environment for handy reference in the
\onlynotes{...} and \onlyslides{...} commands.

You can move over \begin{} ... \end{} groups
with LaTeX-to-begin (on C-c a) and
LaTeX-to-end (on C-c e).

Insertion of paired angle brackets, braces,
square brackets, and parentheses is provided
by
        C-c <  LaTeX-angles
        C-c {  LaTeX-braces
        C-c [  LaTeX-brackets
        C-c (  LaTeX-parentheses
With an argument, the last three insert
backslash prefixes for literal braces, math
mode, and displaymath mode.

Unbalanced character pairs are found by
        C-c >  LaTeX-check-angle-balance
        C-c }  LaTeX-check-brace-balance
        C-c ]  LaTeX-check-bracket-balance
        C-c $  LaTeX-check-dollar-balance
        C-c )  LaTeX-check-parenthesis-balance

You can test for undefined labels with
check-LaTeX-labels, display the labels and
their line numbers with show-LaTeX-labels, and
fetch labels from an .aux file with
update-LaTeX-labels.
```

Table 1.3: More about LaTeX mode.

```
Environment nesting errors can be caught by
check-LaTeX-nesting.

Use indent-LaTeX-begin-end-groups to get
environments nested for improved visibility.

Use LaTeX-comment (on C-c %) to comment out
the current paragraph (no arg) or region
(arg); undo with M-X undo (C-_).  LaTeX-
uncomment (on C-c u) is the inverse function.

One or more words can be set in any standard
LaTeX font using the commands

        C-c C-c b       LaTeX-font-bf
        C-c C-c e       LaTeX-font-em
        C-c C-c i       LaTeX-font-it
        C-c C-c r       LaTeX-font-rm
        C-c C-c u       LaTeX-font-sc
        C-c C-c f       LaTeX-font-sf
        C-c C-c s       LaTeX-font-sl
        C-c C-c t       LaTeX-font-tt

Please report bugs, comments, and enhancements
by e-mail to
        beebe@math.utah.edu (Internet)
LaTeX-gripe (on C-c g) makes this easier.
```

Table 1.4: Even more about LaTeX mode.

```
LaTeX-index:
Insert \index{ ... } at point (on C-c x).

If a prefix argument is given, the string is
prompted for, and inserted both as text and as
an index entry, e.g. gnats and
gnus\index{gnats and gnus}.  All horizontal
space preceding \index is removed to prevent
an intervening page break causing an
off-by-one page number error.

If LaTeX-index-start-with-newline is non-nil,
insert a percent to start a comment, then put
\index at the start of a following new line.
If LaTeX-index-end-with-newline is non-nil,
follow the entry with a new line.
```

Table 1.5: Online help for `LaTeX-index`.

Local Bindings:	
key	binding
---	-------
_	LaTeX-subscript
^	LaTeX-superscript
.	LaTeX-dot
,	LaTeX-comma
$	LaTeX-dollar
ESC	Prefix Command
LFD	newline-and-indent
C-c	Prefix Command
"	LaTeX-quote
C-c ,	LaTeX-set-indentation
C-c C-c	Prefix Command
C-c =	LaTeX-equation
C-c }	LaTeX-check-brace-balance
C-c {	LaTeX-braces
C-c x	LaTeX-index
C-c v	LaTeX-verb
C-c u	LaTeX-uncomment
C-c s	make-SLiTeX-document
C-c r	LaTeX-ref-with-completion
C-c p	LaTeX-pageref-with-completion
C-c n	LaTeX-end
C-c m	LaTeX-macro
C-c l	LaTeX-label
C-c k	LaTeX-counter
C-c i	LaTeX-item
C-c g	LaTeX-gripe
C-c f	LaTeX-footnote
C-c e	LaTeX-to-end
C-c d	make-LaTeX-document
C-c c	LaTeX-cite
C-c b	LaTeX-begin-end-block
C-c a	LaTeX-to-begin
C-c 0	renumber-slides
C-c]	LaTeX-check-bracket-balance
C-c [	LaTeX-brackets
C-c .	LaTeX-add-word-to-index

Table 1.6: Local bindings in LaTeX mode.

C-c)	LaTeX-check-parenthesis-balance
C-c (	LaTeX-parentheses
C-c %	LaTeX-comment
C-c $	LaTeX-check-dollar-balance
C-c >	LaTeX-check-angle-balance
C-c <	LaTeX-angles
C-c C-p	LaTeX-protect
C-c C-n	LaTeX-news
C-c TAB	LaTeX-tab
C-c C-b	LaTeX-bibitem
ESC)	forward-list
ESC (	backward-up-list
C-c C-c t	LaTeX-font-tt
C-c C-c s	LaTeX-font-sl
C-c C-c f	LaTeX-font-sf
C-c C-c u	LaTeX-font-sc
C-c C-c r	LaTeX-font-rm
C-c C-c i	LaTeX-font-it
C-c C-c e	LaTeX-font-em
C-c C-c b	LaTeX-font-bf
C-c C-c }	LaTeX-check-brace-balance
C-c C-c]	LaTeX-check-bracket-balance
C-c C-c)	LaTeX-check-parenthesis-balance
C-c C-c $	LaTeX-check-dollar-balance
C-c C-c =	LaTeX-displaymath

Table 1.7: More local bindings.

1.2 Creating a new LaTeX file

When you visit a new LaTeX file, Emacs starts up with an empty buffer. The mode line at the bottom of the screen will look something like

```
---Emacs: foo.ltx (LaTeX Abbrev Fill)----All---
```

The first thing you want to do is get a standard template of a LaTeX file inserted; you do this by typing `C-c d` (`make-LaTeX-document`). This function will first prompt you with

```
Document style:
```

to which you can respond either with a complete style file name, or with the first few characters of a standard style, followed by a space to request Emacs to complete the name, and a `RETurn` to accept the completion. If you are unsure of what to supply, just type a query (?); Emacs will respond with something like

```
Possible completions are:
aaai                                amsart
amsbook                             amstex
aps                                 article
book                                deproc
letter                              ltugproc
refman                              report
siam                                slides
tugboat                             ucthesis
uoftoronto                          usafmemo
uuthesis                            xxxslides
```

Completion can also be requested on partial names: the input '`a?`' in the above command will respond with

```
Possible completions are:
aaai                                amsart
amsbook                             amstex
aps                                 article
```

Suppose you type '`article`' or '`ar<space>`', followed by a `RETurn`. Then Emacs will respond with

```
Document option (RET when done):
```

to which you can then supply an option name followed by a `RETurn`. As before, a query will show you what is available:

```
Possible completions are:
11pt                                    12pt
a4                                      a4wide
acm                                     acronym
agugrl                                  agujgr
alltt                                   amsbsy
amscd                                   amsfonts
amssymb                                 amssymbols
amstext                                 apalike
archmemo                                array
bezier                                  biihead
boxedmini                               changebar
chapterbib                              cite
clsc                                    clscmemo
concrete                                cropmark
ctagsplt                                cyrillic
dblspace                                doublespace
draft                                   drafthead
drop                                    dvidoc
eqsecnum                                espo
fleqn                                   format
fullpage                                geophysics
german                                  gnuindex
ifthen                                  intlim
ist21                                   leqno
listing                                 longtab
ltugboat                                makeidx
math                                    merge
mfr                                     mitthesis
moretext                                natsci
nl                                      nonamelm
nopageno                                nosumlim
openbib                                 pandora
preprint                                proc
psmavg                                  psmbkm
psmncs                                  psmpal
psmtim                                  remark
resume                                  revtex
righttag                                sc21
sc21-wg1                                sfwmac
showidx                                 showlabels
slem                                    spacecites
```

```
supertab                        suthesis
tablisting                      tapeseal
texindex                        texnames
tgrind                          theorem
thp                             threepart
titlepage                       trademark
troffman                        troffms
twocolumn                       twoside
underline                       uofutah
vdm                             verbatim
```

You can keep on selecting options until you finally just type a bare `RETurn`, at which point `make-LaTeX-document` completes, and you have a buffer that looks like this:

```
% -*-latex-*-
% Document name: /u/beebe/foo.ltx
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Sun Jul 28 08:44:46 1991
\documentstyle[11pt,makeidx]{article}
\newcommand{\X}[1]{{#1}\index{{#1}}}
\begin{document}
<cursor>
\end{document}
```

The cursor, represented by `<cursor>`, is left positioned on the line following the '`\begin{document}`', ready for text entry.

The initial comment with the tag '`-*-latex-*-`' is a special one. It requests an automatic mode selection when the file is freshly visited in Emacs, overriding any mode that might otherwise be selected based on the file name extension. `latex-mode` is made equivalent to `LaTeX-mode` in '`latex.el`' so that the comment tag can be written in lower-case letters, as is conventional.

The '`% Creator:`' comment information is obtained from Emacs' operating system interface, which in turn fetches the user's personal name and login name from system authorization files. On networked systems, the hostname is included as well, creating a valid electronic mail address.

Had you given `make-LaTeX-document` an argument, it would have been somewhat more verbose:

```
% -*-latex-*-
% Document name: /u/beebe/tex/tugboat/foo.ltx
% Creator: Nelson Beebe [beebe@math.utah.edu]
```

```
% Creation Date: Sun Jul 28 08:46:33 1991
%-------------------------------------------------
% EVERYTHING TO THE RIGHT OF A  %  IS A REMARK
% TO YOU AND IS IGNORED BY LaTeX.
%
% WARNING!  DO NOT TYPE ANY OF THE FOLLOWING 10
% CHARACTERS AS NORMAL TEXT CHARACTERS:
%       &   $   #   %   _    {   }   ^   ~   \
%
% The following seven are printed by typing a
% backslash in front of them:
%       $  &  #  %  _  {  and  }.
%-------------------------------------------------
\documentstyle[11pt,makeidx]{article}
\newcommand{\X}[1]{{#1}\index{{#1}}}
\begin{document}
<cursor>
\end{document}
```

The extra commentary is a useful reminder for beginning LaTeX users. Several other commands in `LaTeX-mode` use the presence of an argument to supply additional helpful commentary.

The '`\X`' command is one I have found helpful for preparing indexes; typing '`\X{gnats and gnus}`' will put that phrase in both the document and the index. Further discussion is given later; see 1.14 [Indexing], page 29.

The lists of standard styles and options are embedded in the '`latex.el`' code. They cannot be determined automatically from the file system for several reasons:

- Some options (e.g. '`11pt`') do not have a corresponding style file.
- Some style files do not represent a valid option (e.g. '`art10.sty`').
- Some style files found in the normal `TEXINPUTS` search path belong to other systems, such as AmSTeX.
- It is impossible to distinguish from the '`.sty`' file extension whether the file represents a major document style, or merely an option that modifies a major style.

It is regrettable that these difficulties exist, because they confuse users, as well as programmers of code like `LaTeX-mode`. It is often difficult to tell from its contents whether a particular file is a major style, or just a document option. It would have been wiser in the initial LaTeX design to

have chosen distinctive file extensions, and for each style and option to be associated with a unique file.

The present collection of twenty available styles and over one hundred options is daunting for a novice, particularly since many options can only be used with certain styles. Eventually I may find a satisfactory way to deal with this, and offer a better presentation of choices in the completion lists.

The style and option lists can be easily extended. See 1.19 [Customizing lists], page 35 for details.

1.3 LaTeX macros

Every one of the 589 LaTeX macros in the index to the LaTeX *User's Guide and Reference Manual* [14] is known to the function `LaTeX-macro` (on `C-c m`), and as before, Emacs command completion is available with the query (?) and space (`<space>`) characters. Thus, you can type

- `C-c m longr<space> RETurn` to get '`\longrightarrow`';
- `C-c m em RETurn` to get

  '`{\em <cursor>\/}`'

  with the cursor positioned ready for you to enter some text;
- `C-u C-c m newcom<space> RETurn` to get

```
\newcommand{<cursor>}[]{} % {\cmd}[n]{def} define new
                          % command with n arguments
```

- `C-c m e?` to find the names of macros that begin with `e`:

```
Possible completions are:
ell                         em
emptyset                    encl
end                         epsilon
equiv                       eta
evensidemargin              exists
exp                         extracolsep
```

When the cursor is left after an open brace in a generated command, the closing brace is by default placed at the start of the next line so that you have a clean line to type on. This reduces visually-distracting screen

updates, but means that you will need to delete the end-of-line character when you finish typing the argument. If you don't like this behavior, you can set the Emacs variable `LaTeX-newline-after-opening-delimiter` to a `nil` value, usually in your '`.emacs`' startup file:

```
(setq LaTeX-mode-hook
 '(lambda ()
    (setq LaTeX-newline-after-opening-delimiter
          nil)
    )
 )
```

The variable `LaTeX-newline-after-closing-delimiter` is used similarly for brace pairs inserted on their own, or after certain macros.

A *mode hook* is Lisp code to be executed whenever the corresponding mode function is executed. The hook is called just before the mode function returns, so it can override anything that the function did. The peculiar '`lambda ()`' is the Lisp way of declaring a nameless function. The single quote prefixing it means that `LaTeX-mode-hook` will be assigned the function itself, and not the value returned by the function.

In GNU Emacs, you can execute Lisp code in a file through `M-x load-file` and `M-x load-library` commands. Inline code must be executed in a buffer which is in `emacs-lisp-mode`. The minibuffer always has that mode, and can be temporarily accessed with the `eval-expression` function bound to `ESCape ESCape`.

If you just want to set an Emacs variable, you can also use the command `M-x set-variable`; it supports completion on the variable name, and prompts for the value.

The availability of all of the LaTeX macros with `LaTeX-macro` reduces the likelihood of spelling mistakes, and the command completion usually means that only a few leading characters need to be typed. The complete macro name is entered in the buffer: '`\abovedisplayshortskip`' is much more readable than a cryptic '`\adss`' that LaTeX users without such editing support might be inclined to define as a private macro to save typing effort.

If you have frequently-used private macros, there is a convenient way to make them known to `LaTeX-mode`. See 1.19 [Customizing lists], page 35 for details.

1.4 LaTeX environments

An important concept in LaTeX is the notion of *environments*, which are marked by surrounding '`\begin{...}`' and '`\end{...}`' commands. The

braced tag must be repeated, and if the tag name is long, spelling errors are more likely. The function `LaTeX-begin-end-block` on `C-c b` generates the command pair, indented according to the current nesting level, and leaves the cursor on the line between them. The indentation is controlled by the value of the Emacs variable `LaTeX-begin-end-indentation`; the default is two spaces.

If the variable `LaTeX-space-after-begin-end` is non-`nil`, a space will be inserted between the keyword and the following opening brace. The default value is `nil`. This formatting style is discouraged, because it cannot be applied uniformly; it must be suppressed for the '`verbatim`' environment. The code in '`latex.el`' knows about this vagary, and always omits the space for that environment.

The keystrokes `C-c b quote RETurn C-c b ite<space> RETurn C-c b verb<space> RETurn` produce

```
  \begin{quote}
    \begin{itemize}
      \item
\begin  {verbatim}
<cursor>
\end  {verbatim}
      \item
      \item
      \item
    \end{itemize}
  \end{quote}
```

with the cursor inside the '`verbatim`' environment. Notice that the normal indentation is automatically suppressed for that environment; the command knows that leading spaces in front of '`\end{verbatim}`' would otherwise generate an unwanted blank line in the typeset output.

If you have private environments, you can still use `C-c b`; you just have to type the complete environment name. If you use some private ones frequently in different documents, it is possible to extend the list of environments known to `LaTeX-mode`. See 1.19 [Customizing lists], page 35 for details.

In those environments that have '`\item`' commands, four are generated by default; that number can be changed by modifying the value of the Emacs variable `LaTeX-item-count`. Their indentation is defined by the variable `LaTeX-item-indentation`; the default is two spaces. Additional indentation of text under a '`\item`' command is set by the variable `LaTeX-item-info-indent`; the default is six spaces.

You can generate additional '`\item`' commands as needed with the function `LaTeX-item` on `C-c i`. With an argument, it supplies the alternate form '`\item[]`' instead.

If you forgot about the `C-c b` command, and manually typed a '`\begin {conjecture}`', you can get the matching '`\end{conjecture}`' generated automatically by typing `C-c n` (`LaTeX-end`). That is also sometimes helpful when you have forgotten what environment you are in: you can generate the '`\end{...}`', and then kill it.

You can type `C-c a` (`LaTeX-to-begin`) to move backward to the enclosing '`\begin`'; with an argument, you can move backward over a specified number of '`\begin{...}`'s. If the argument is negative, you move forward over '`\end{...}`'s instead. Usually, you would use `C-c e` (`LaTeX-to-end`) instead to move forward over '`\end{...}`'s; that command handles numeric arguments too, and reverses direction for negative arguments.

These two movement commands are analogous to the standard Emacs commands for moving backward and forward in sentences, normally bound to `M-a` and `M-e`. They also require that the '`\begin`' and '`\end`' macros be preceded on their lines only by optional whitespace, and immediately followed by open braces. This helps to enforce the discipline of keeping the '`\begin{...}`' ...'`\end{...}`' grouping properly nested and readable, and also avoids having to deal with more complex input parsing.

1.5 Common commands

Some LaTeX macros are used so often that it is desirable to have them bound directly to keys, instead of having to type their first few characters with `LaTeX-macro` as described earlier. You can find a list of these convenience functions in the fourth paragraph of Table 2.

For example, typing `C-c l` (that's the letter 'l') generates

```
~\label{<cursor>}
```

at the cursor position, leaving the cursor after the open brace.

1.6 Math mode

TeX's math mode uses paired dollar signs to delimit the math text. Single dollar signs mark inline math, and doubled ones mark display math. If you forget to type a closing dollar sign, or type the wrong number of dollar signs, TeX will complain bitterly.

The start and finish of these math regions are indistinguishable, making it hard for simple programs (and sometimes, readers) to distinguish the 'inside' from the 'outside'.

Consequently, LaTeX additionally supports two bracketing forms:'`\(`' ...'`\)`' for '`$ ... $`', and '`\[`' ...'`\]`' for '`$$ ... $$`', as well as more readable forms

```
\begin{math}
...
\end{math}

\begin{displaymath}
...
\end{displaymath}
```

for the single and double dollar sign pairs, respectively. All of these have unique opening and closing sequences.

`LaTeX-mode` supports all of these forms. The variants selected are determined by the variables `LaTeX-math-option` and `LaTeX-displaymath-option`. Values of 0, 1, and 2 select the dollar, backslashed delimiter, and '`\begin{...}`' ...'`\end{...}`' environment forms respectively; all other values are equivalent to 0. The default values select the single dollar and '`displaymath`' forms.

Normally, when you type a dollar sign, you get an inline math environment. If the dollar sign happens to be at the beginning of a line, it expands instead into a display math environment. On the other hand, if the preceding character was a backslash, you just get a plain dollar sign.

Spacing around the inserted text is controlled by two more variables. If `LaTeX-newline-after-opening-delimiter` is non-`nil` (the default), the opening delimiter is followed by a newline, with the cursor positioned immediately after the delimiter for the inline case, and on a new line for the display form. If `LaTeX-newline-after-closing-delimiter` is non-`nil` (the default), the closing delimiter is followed by a newline.

This pleasantness is our first example of having an alternate function bound to a normal printing character. Instead of running the usual `self-insert-command` attached to printing characters, '`$`' is bound to `LaTeX-dollar` which has been programmed to deal with the three different meanings of the dollar sign. Few editors outside the Emacs family could do this; the critical feature is that *any character* can run *any function*.

What if you really just wanted a single plain dollar sign, with no preceding backslash? Well, that is easy. In Emacs, any character can be *quoted*, which causes it to be inserted directly into the buffer; the `quoted-insert`

function is normally bound to `C-q`, but that too can be changed. Thus, `C-q $` will get you a literal dollar sign, if that is what you really want.

The function `LaTeX-equation` on `C-c =` generates

```
  \begin{equation}
<cursor>
  \end{equation}
```

with the cursor between them. `LaTeX-displaymath` on `C-c C-c =` inserts

```
  \begin{displaymath}
<cursor>
  \end{displaymath}
```

All math mode environments ('`array`', '`eqnarray*`', '`eqnarray`', '`math`', and '`theorem`') can be also generated by `LaTeX-begin-end-block` on `C-c b`, and additional ones can be easily added; see 1.19 [Customizing lists], page 35.

`LaTeX-subscript` on `_` (underscore) and `LaTeX-superscript` on `^` (caret) generate braced subscript and superscript sequences, leaving the cursor between the braces [20].

1.7 Dots, commas, and quotation marks

The dot or period (full stop, to some of you) is an overloaded character. Normally, it just means end of sentence. Sometimes, it ends an abbreviation in the middle of a sentence. It can also be a decimal point, as in 3.14159. Still other times, it appears in ellipses, as in centered dots ('`\cdots`'), diagonal dots ('`\ddots`'), ellipses ('`\ldots`'), and vertical dots ('`\vdots`').

In typography, these uses are distinguished by small changes in spacing after the dots. That is why TEX provides control sequences for the ellipses, and the LATEX book [14, p. 14] recommends that you type a '`\<space>`' sequence after a period that does not end a sentence.

TEX assumes that a period following an upper-case letter does not end a sentence (it might be someone's initial) [13, pp. 74, 311], and follows it with a smaller amount of space. The LATEX *User's Guide and Reference Manual* recommends insertion of '`\@`' before a sentence-ending dot that follows an upper-case letter in order to get additional space after the dot.

`LaTeX-mode` handles these intricacies by binding dot to the function `LaTeX-dot`. That function examines what is immediately before the dot in the buffer, and takes one of several actions:

- If the dot ends an ellipsis, the three dots are replaced by '`\ldots{}`'.

- If the dot is preceded by an italic correction at group end, '`\/}`', the italic correction is removed.

- If the dot ends an abbreviation, like *e.g.* or *i.e.*, it inserts a '`\<space>`' command or a comma, depending on whether the variable `LaTeX-comma-after-dotted-abbreviation` is `nil` or non-`nil`. Traditionally, such abbreviations are followed by a comma, but modern tendency is to omit the comma [19, p. 84], although Knuth disagrees [13, p. 74].

- If the previous characters are abbreviations like *Fig.*, *cf.*, *vs.*, and *resp.* [13, p. 74] [14, p. 18] that should be tied to the following word with a single unbreakable space, the function inserts a tilde to mark the tie.

- If the dot follows two or more upper-case letters, the dot is prefixed by '`\@`'.

To prevent confusion with words at end of sentence, abbreviations are not recognized if they are immediately preceded by a letter in the editing buffer.

The abbreviations known to `LaTeX-dot` are defined by the variable `LaTeX-standard-dotted-abbreviations`. More can be added by a suitable definition of the variable `LaTeX-extra-dotted-abbreviations`, such as this example:

```
(setq LaTeX-extra-dotted-abbreviations
      '(
        ("ad. lib.")
        ("cwt.")
        ("q.e.d.")
        )
      )
```

See 1.19 [Customizing lists], page 35 for more details.

The abbreviations for which ties are supplied are similarly defined by the variables `LaTeX-standard-tied-abbreviations` and `LaTeX-extra-tied-abbreviations`.

`LaTeX-dot` cannot tell whether a dot following an upper-case letter is a sentence-ending one or not, so it only inserts '`\@`' before a dot if there are *two or more* preceding upper-case letters. It is up to the user to remember to insert '`\@`' in the exceptional case of a single final upper-case letter.

What about a dot after a macro that expands to a word ending in an upper-case letter, like TeX itself? No '`\@`' is necessary there, because TeX treats that dot as a normal end of sentence.

These actions cover most of the common cases, but of course, for abbreviations, cannot be omniscient. Dictionaries that I consulted listed over a thousand abbreviations in use in English. Fewer than a dozen of the commonest ones are included in `LaTeX-standard-dotted-abbreviations`, but customization of `LaTeX-extra-dotted-abbreviations` provides a way to add new ones.

Emacs' `quoted-insert` function saves the day when you want to suppress all of this wizardry; just type `C-q .` to get a literal dot inserted.

`LaTeX-comma` bound to comma removes any immediately preceding italic correction, and then inserts a comma.

Italic corrections are somewhat of a nuisance, since you are supposed to have them after italicized text, *except* a period or a comma, which are so low that it doesn't matter if the character to their immediate left leans over them. Those two exceptions make it hard to write a TEX macro to determine whether the correction is needed or not. I'd rather let Emacs remember for me, so I programmed it to do so.

Good typography uses different opening and closing quotation marks; in TEX, you get "quoted text" by typing '`‘‘quoted text’’`'. Because many people are accustomed to using the double quotation mark character, '`"`', found on computer keyboards and typewriters, the function `LaTeX-quote` is bound to that character. It generates the correct paired opening grave accents, '`‘‘`', or closing apostrophes, '`’’`', as appropriate: the input '`"quoted text"`' expands to '`‘‘quoted text’’`' in the edit buffer. With an argument, the function inserts a bare double quotation mark; you can get the same effect from `quoted-insert` by typing `C-q "`.

1.8 Font changes

The `LaTeX-macro` function on `C-c m` lets you easily generate font changes to any of LATEX's eight standard styles, including supplying the italic correction in the three leaning font styles ('`\em`', '`\it`', and '`\sl`').

But what if you are revising a document, and decide to add a font change? Well, you could use `C-c m` to do it, but then you'd have to move the closing brace, and any preceding italic correction.

`LaTeX-mode` provides a better solution: customized functions *wrap* a font change around the current word, or if a numeric argument is given, around that many words starting from the current one. These functions are listed in Table 7. They are almost the only functions that require a second prefix character; `C-c C-c e` (`LaTeX-font-em`) is easy to type, and retains mnemonic significance. These functions handle the italic correction properly; leaning fonts get it unless the character following the last word is

a period or a comma.

1.9 Comments

TeX provides an inline comment mechanism with the percent sign. It discards text from that character to the end of the line, plus *all leading space on the next line up to, but not including, its end-of-line character.*

Sometimes, however, you want to temporarily suppress typesetting of a part of a document without actually removing it from the file. One way is to insert leading percents on each line of the region to be suppressed, but that is tedious to do manually. A second way is to use LaTeX's '`ifthen`' style option which added loops and conditionals, but appeared after the LaTeX *User's Guide and Reference Manual* was written, and so is not widely known. A third way is to define a LaTeX macro like

```
\newcommand{\comment}[1]{}
```

However, that won't work if the argument contains paragraph breaks. You could drop down to low-level TeX and write

```
\long\def\comment#1{}
```

but LaTeX users are not supposed to do that, except in style files. Also, both of these macros risk overflowing TeX's input buffer, since the complete argument might be rather long, even if it is subsequently discarded.

I often wish that LaTeX had provided a standard '`comment`' environment to do this; it could be implemented much like '`verbatim`', except that it must support properly nested '`comment`' environments. Rainer Schöpf has implemented a version of a '`comment`' environment in a new '`verbatim`' implementation for LaTeX [18], although nested comments are not yet supported.

The solution provided by `LaTeX-mode` is the function `LaTeX-comment` on `C-c %` which implements the first choice above. By default, it comments out the current paragraph, or with an argument, the current region. The comment prefix inserted at the beginning of each line is defined by the variable `LaTeX-comment-prefix`; its default value is '`%%:<space>`'. A unique prefix is desirable to distinguish such lines from ordinary comments. You should avoid using any regular-expression matching characters in it.

You can remove the comments inserted by `LaTeX-comment` by using the function `LaTeX-uncomment` on `C-c u`. It should normally be used with an argument to act on a region, instead of a paragraph. The reason for this is that paragraphs are recognized according to the regular expression in the Emacs variable `paragraph-separate`; the default setting is such

that a line containing only a comment ends a paragraph. Consequently, `LaTeX-uncomment` will do nothing because its 'paragraph' is always empty. Invoking it with an argument to process a marked region solves the problem.

1.10 Mismatched delimiters

TEX gets very unhappy when it finds mismatched braces, and in complex documents, it can sometimes be difficult to find them. The job is not made easier on low-resolution screen displays where braces and parentheses are virtually indistinguishable. Finding such mismatches is a job for a computer, not a human, and `LaTeX-mode` offers you help.

Braces are not the only things you might want assistance with. Checking for mismatched parentheses, square brackets, and angle brackets is also supported. These normally insert themselves when you type them, but `LaTeX-mode` has commands that insert pairs: typing `C-c {` gets you a pair of open and close braces, and similarly for `C-c [`, `C-c (`, and `C-c <`. These are the default bindings of the functions `LaTeX-braces`, `LaTeX-brackets`, `LaTeX-parentheses`, and `LaTeX-angles`.

If you get in the habit of using these instead of typing the delimiter pair yourself, you will rarely have mismatch problems. Emacs itself has native delimiter matching support: as you type a close delimiter, the cursor flashes briefly to the matching open delimiter.

But what if you didn't follow my advice, typed a lot of braces yourself, and then when TEX complained about unbalanced braces, you couldn't find the errors? Well, the function `LaTeX-check-brace-balance` bound to `C-c }` will scan the buffer from the current point to the end, checking for unbalanced braces, and brace pairs separated by unusually long strings, but ignoring backslashed braces. It complains if it finds a paragraph break between braces, unless you give it an argument. If mismatches are detected, you are so informed, and the positions of the mismatches are remembered on Emacs' stack of marks, so you can easily get to them.

Similar functions `LaTeX-check-angle-balance`, `LaTeX-check-bracket-balance`, and `LaTeX-check-parenthesis-balance` are provided on `C-c >`, `C-c ]`, and `C-c )`, for checking those delimiter balances. Note the key binding symmetry: `C-c open-delimiter` inserts the pair, and `C-c close-delimiter` checks the balance.

The characters '`<`' and '`>`' are likely to be used in math mode, and not require balancing. However, you might also have used them for other purposes. In such a case, you can just add matching delimiters inside comments in your math mode uses, and then `C-c >` will work satisfactorily.

There is also `LaTeX-check-dollar-balance` on `C-c $`. It looks for

dollar pairs separated by long strings, ignoring backslashed dollars. It also complains about paragraph breaks between dollar pairs, unless it is given an argument. There is no way for a computer program that is much less complex than TeX to tell the inside of math mode from the outside, so you can confuse this function by starting it inside math mode.

The Emacs variables `LaTeX-angle-interval`, `LaTeX-brace-interval`, `LaTeX-bracket-interval`, `LaTeX-dollar-interval`, and `LaTeX-parenthesis-interval` set the limiting between-delimiter string length before a warning is raised; their default values are each set to 500 characters.

1.11 Mismatched environments

If you have a complex document where '`\begin{...}`' ...'`\end{...}`' groups have been typed manually, you can also end up with mismatched environments that are hard to find.

The command `M-x check-LaTeX-nesting` scans the buffer to ensure the correctness of '`\begin{...}`' ...'`\end{...}`' nesting. If, for example, your buffer contains a nesting error like this

```
  \begin{verse}
...
    \begin{quote}
...
  \end{verse}
...
    \end{quote}
```

the command will terminate with the cursor in front of the '`\end{verse}`' with the error message

```
This \end{verse} is matched by preceding \begin{quote}
at mark.
```

The Emacs function `exchange-point-and-mark` on `C-x C-x` will move the cursor to the mismatching '`\begin{quote}`'. After fixing the error, you can rerun the nesting check by typing `C-x ESCape` (`repeat-complex-command`) until it finally reports

```
[done] -- all \begin{}-end{}s balance.
```

The command `M-x indent-LaTeX-begin-end-groups` indents '`\begin{...}`' ...'`\end{...}`' groups according to their nesting level, which helps to make the input file more readable.

You don't need these commands very often, so they do not have default key bindings.

1.12 Word abbreviations

Emacs provides a convenient word abbreviation facility that permits the automatic expansion of a string of letters when the following space or punctuation is typed. `LaTeX-mode` provides this by default with a set of abbreviations in a table that can be customized for personal use.

You can find the complete set of abbreviations by examining the value of the variable `LaTeX-mode-abbrev-table`, or by invoking the function `list-abbrevs`. You can make changes most easily with `M-x edit-abbrevs`.

The commonest abbreviations recognized in `LaTeX-mode` are for macro names that are awkward to type: the input '`latex `' expands to '`\LaTeX{} `', '`tex `' to '`\TeX{} `', and so on.

Note particularly that the expansion here is to '`\TeX{}`', and *not* '`\TeX\<space>`'! There are good reasons for this:

- If you terminate macro names with '`\<space>`' instead of with '`{}`', you have to remember to do that only where a space would really be permitted; in particular, you don't want the '`\<space>`' before punctuation. You can use '`{}`' consistently in both cases.

- If the '`\<space>`' appears at the end of a line, and trailing spaces are subsequently stripped, you now have a new macro equivalent to '`\^^M`', which might have a different meaning. Although both '`plain.tex`' and '`lplain.tex`' make the two macros equivalent, that might not always be the case.

  You should never count on end-of-line blanks being preserved; some e-mail systems, and some editors, may delete them. Text filling and justification as commonly used in Emacs can also remove them.

- In composite words, like '`\TeX{}book`', the braces keep the parts together for text fill operations, word counting, spell checking and so forth, while the form '`\TeX book`' will not.

The form '`{\TeX}`' has the almost same advantages over '`\TeX\<space>`' as '`\TeX{}`' does, but has one drawback: if you need to '`\protect`' it, then you must do so inside the braces. Use of a final empty brace pair therefore seems the best choice.

By Emacs convention, word abbreviation mode is never turned on automatically; you must explicitly toggle it on a case-by-case basis by executing

the command `M-x abbrev-mode`. If you always want it selected in `LaTeX-mode`, add it to your '`.emacs`' file in a mode hook:

```
(setq LaTeX-mode-hook
      '(lambda ()
         (abbrev-mode 1)
         )
      )
```

If you have other things to set in that mode hook, they should go on the lines immediately before or after the '`(abbrev-mode 1)`' line, since there can be only one mode hook for each editing mode.

1.13 Bibliographies

Literature citations are conveniently provided for on `C-c c` (`LaTeX-cite`), which generates a '`\cite{}`' command with the cursor after the open brace. With an argument, it produces '`\cite[]{}`' instead, with the cursor between the square brackets.

There is another function, `LaTeX-bibitem`, on `C-c C-b`, which generates a '`\bibitem{}`' for you; with an argument, it produces '`\bibitem[]{}`'. Normally, you should be using a program like BibTeX [14, Appendix B] or '`Tib`' [1, 2] for preparation of bibliographies from a data base and the citations recorded in the LaTeX '`.aux`' file. If you find yourself using this command, you are doing the job the hard and inflexible way.

`LaTeX-mode` automatically invokes the function `update-LaTeX-bibtags` to extract the list of bibliography files from the '`\bibdata`' command in the '`.aux`' file, or the '`\bibliography`' command in the LaTeX file. Each such file is then located in the `BIBINPUTS` search path, and scanned to extract citation tags which are stored as an association list in the variable `LaTeX-bibtags`. Each list entry contains a citation tag and the name of the file in which the tag was found.

If the `BIBINPUTS` environment variable is not set, then the Emacs variable `LaTeX-BIBINPUTS` is used instead to get a suitable search path.

You can view the citation tag list by invoking the function `show-LaTeX-bibtags`.

You can request completion on a citation tag inside a '`\cite`' command by using the function `LaTeX-bibtag-with-completion` normally bound to `C-c C`.

If you are visiting a subfile in a multi-file document, or have just modified the list of bibliography files in the '`\bibliography`' command, you

should manually run `update-LaTeX-bibtags` to update the variable `LaTeX-bibtags`.

1.14 Indexing

Index preparation is a tedious job that cannot be entirely automated [3, 4, 5, 6, 15]. Nevertheless, technical documents can benefit significantly from a good index, and support tools like '`makeindex`' or '`texindex`' can handle much of the drudge work of sorting and formatting the index entries.

`LaTeX-mode` provides two functions to support generation of index entries in the LaTeX file.

`LaTeX-add-word-to-index` on `C-c .` wraps an index entry around the word at the cursor position, either using a private indexing macro like '`\X`' described earlier (see 1.2 [Creating a new LaTeX file], page 12), or by duplicating the word inside a standard LaTeX '`\index`' command. The choice between the two is determined by the Emacs variable `LaTeX-index-macro`; a non-`nil` value supplies the indexing macro name. With a numeric argument, multiple words can be selected for indexing.

`LaTeX-index` on `C-c x` generates
'`\index{<cursor>}`'
so you can type an explicit index entry that will not appear at that point in the text.

For use in the X Window System on workstations and personal computers acting as X clients, '`latex.el`' provides additional functions, `x-LaTeX-index-start` and `x-LaTeX-index-end` that are bound to mouse actions; they cannot usefully be invoked from the keyboard.

To index a phrase in the document like *gnats and gnus*, simply hold down the shift key and click the right mouse button at the start of the first word, and then holding the button down, sweep the mouse across the entire phrase, lifting the button anywhere in the last word. Bingo: the buffer now contains '`\X{gnats and gnus}`'!

I find it useful to sit down with a typeset copy of a document, and use a colored highlighter pen to mark phrases to be indexed. With the marked-up copy, it is then straightforward to use the mouse to add the marked phrases to the index. For documents like this one, with lots of technical terms to be indexed, it pays to think about matters before you start writing. Define suitable macros that will typeset those terms in a particular font, and also automatically create an index entry for them. For example, I typeset and index an Emacs function using a private '`\FUNCTION{}`' macro.

The Emacs variable `LaTeX-index-start-with-newline` can be set to a non-`nil` value if you prefer to have the inserted index entry start a new

line. The variable `LaTeX-index-end-with-newline` can be set to a non-`nil` value to have a newline generated after the inserted index entry. The default for both is `nil`, so that no additional lines are generated.

1.15 Cross-references

LaTeX's '`\label`', '`\pageref`', and '`\ref`' commands provide a convenient way to generate cross-references to equations, figures, pages, sections, and tables. In `LaTeX-mode`, they are readily available on the keys `C-c l` (that's the letter 'l'), `C-c p`, and `C-c r`.

Since definitions with '`\label`' tend to be separated from their references, you may often find yourself unable to remember the exact name of the label you want. If you type the label name incorrectly, you won't discover the error until you have run LaTeX twice. To avoid this delay, two functions provide additional support:

- `show-LaTeX-labels` finds all labels in the buffer and displays them with their line numbers; I ran that function while I was writing a companion article, and got the output

  ```
  LaTeX labels for buffer ltxmode.ltx (in alphabetical \
  order).  Mouse-2 or C-c C-c goes to label.
  1862: "X-command"
  3793: "customizing-lists"
  1324: "fig-bindings-1"
  1455: "fig-bindings-2"
  1258: "fig-latex-index"
  1099: "fig-latex-mode-1"
  1188: "fig-latex-mode-2"
  4083: "fig-letter"
  1554: "fig-startup-code"
  748: "gnu-emacs-size"
  3159: "indexing"
  1487: "latex-support"
  2030: "mode-hook"
  2629: "quote"
  2976: "word-abbreviations"
  ```

  showing the labels in alphabetical order. With an argument, the labels are shown instead in line number order.

  The labels are displayed a buffer in Emacs `occur-mode`. You can position to any one of them, and then type `C-cC-c` to jump to the

location in the LaTeX buffer where the label is defined. With Emacs version 19 or later, you can also click the middle mouse button on an entry in the label buffer: the cursor in the LaTeX buffer will move to the point of definition and the label will be highlighted.

- `check-LaTeX-labels` scans the entire buffer, looking for labels that are referenced, but undefined. At each such label found, a recursive edit is entered to allow you to supply the missing label; when you exit the recursive edit, the search for undefined labels continues from where it left off.

These functions locate label references by searching for the string defined by the variable `LaTeX-label-reference`; it contains a regular expression that normally matches both '`\pageref`' and '`\ref`'. If you have defined private macros that also receive labels as arguments, you may wish to extend it. For example, the companion document has '`\FIGREF`' and '`\PAGEREF`' macros that simplify cross-referencing.

Similarly, these functions locate label definitions by searching for the string defined by the variable `LaTeX-label-definition`, another regular expression that normally matches '`\label`'. You may wish to customize it if private macros also generate labels.

Multi-file documents pose a challenge: labels defined in one file may be referenced in another file, and it would be helpful to be able to show a list of all currently-defined labels when one is needed for a cross-reference.

It is not feasible for the code in '`latex.el`' to attempt to figure out what files are used for a particular document, nor is it acceptable to ask the user to provide those file names. Yet we somehow need to scan those files to find all of the labels.

The solution is that LaTeX can do part of this job for us: LaTeX processing of the master file produces in the corresponding '`.aux`' file all defined labels embedded in '`\newlabel`' commands. The function `update-LaTeX-labels` does the remainder of the job: it prompts for the name of the master '`.aux`' file, and then scans it to extract a list of labels from the '`\newlabel`' commands. This list is then *appended* to the current contents of the variable `LaTeX-aux-file-label-tags`. Augmentation, rather than replacement, permits selective construction of list subsets by multiple invocations of `update-LaTeX-labels` on different files. `check-LaTeX-labels` includes the values from `LaTeX-aux-file-label-tags` in its list of defined labels.

For convenience, whenever `LaTeX-mode` is run (normally when a file is first visited), it will search the buffer for the first TeX macro which is preceded only by optional whitespace on its line. If that macro is '`\documentstyle`', then the file must be the master file for the document,

so `LaTeX-aux-file-label-tags` is set to `nil`, and `update-LaTeX-labels` is automatically called on the corresponding '`.aux`' file. This means that manual invocation of `update-LaTeX-labels` is needed only when you visit a sub-file without first visiting the master file.

In recent versions of '`latex.el`', `LaTeX-pageref` and `LaTeX-ref` have been augmented by `LaTeX-pageref-with-completion` and `LaTeX-ref-with-completion`. The new functions are bound to `C-c p` and `C-c r`, replacing the bindings to the older functions. The new ones use an internal function to dynamically scan the buffer for '`\label{}`' definitions, add the contents of `LaTeX-aux-file-label-tags` to the labels collected, and construct a list that can be used for command completion. They then prompt for a label, and if you type a query, they will display the current list of labels. You can use completion to select any one of them, or you can enter an arbitrary label that is not in the completion list.

Here is what the completion list looks like for this document:

```
Possible completions are:
X-command                      customizing-lists
fig-bindings-1                 fig-bindings-2
fig-latex-index                fig-latex-mode-1
fig-latex-mode-2               fig-letter
fig-startup-code               gnu-emacs-size
indexing                       latex-support
mode-hook                      word-abbreviations
```

Because the dynamic label scan that happens each time these functions are executed may be slow in large documents, the old functions remain available for rebinding to keys.

The Emacs variable `LaTeX-label-start-with-newline` can be set to a non-`nil` value if you prefer to have the inserted label entry start a new line. The variable `LaTeX-label-end-with-newline` can be set to a non-`nil` value to have a newline generated after the inserted label entry. The default for both is `nil`, so that no additional lines are generated.

1.16 French typography

English typographical conventions for spacing around punctuation are at odds with good French typography. TeX's '`\frenchspacing`' macro [13, p. 74] [14, p. 154] does not entirely satisfy French conventions: it only makes spaces after punctuation the same as interword spaces.

In French typography, the punctuation characters '`:  ; ?  !`' are *preceded* by an unbreakable interword space (in TeX language, a *tie*). Em-

dashes (—) are always surrounded by spaces; in English typography, such spacing is a matter of personal style. Quotations are made with guillemets as '`<< ... >>`', instead of as "..."; the open guillemet is followed by a space, and the close one is preceded by a space.

The GUTenberg organization has developed a standard TEX and LATEX style file, '`french.sty`', to provide suitable definitions of the guillemets as the control sequences '`"<`' (or '`\flqq`') and '`">`' (or '`\frqq`'), as well as more convenient accent primitives.

It is tedious to remember to type the TEX tie command everywhere that special French punctuation spacing is required, so the minor mode function `French-LaTeX-mode` is provided in '`latex.el`'. Each time it is invoked, it toggles between French and non-French conventions. It can be used in either TEX or LATEX files.

When French mode is selected, the Emacs mode line contains the string '`French`', and new functions `LaTeX-French-colon`, `LaTeX-French-exclamation`, `LaTeX-French-langle`, `LaTeX-French-minus`, `LaTeX-French-query`, `LaTeX-French-rangle`, and `LaTeX-French-semicolon` are bound to the appropriate keys. The quotation mark is rebound to `self-insert-command` so that the special handling normally provided by `LaTeX-quote` (see 1.7 [Dots and commas and quotation marks], page 21) is suppressed.

The French angle bracket functions insert an angle bracket, expand '`<<`' to '`"<`', and '`>>`' to '`">`', and then supply the necessary ties. Isolated angle brackets are inserted without further modification. Guillemets can thus be inserted either by typing their named control sequences, or by '`"<`' or '`<<`', and '`">`' or '`>>`'.

`LaTeX-French-minus` inserts a minus sign, and if it is the end of an em-dash, supplies surrounding spaces (*not* ties).

While it would be possible to have TEX provide for the French spacing requirements by suitable macro definitions of active characters, such redefinitions risk interfering with other macros. It seems best to provide for the spacing requirements in the input file itself, letting Emacs remember the necessary rules.

French mode can be customized by defining a suitable value of the `French-LaTeX-mode-hook` variable; see 1.3 [LaTeX macros], page 16.

1.17 Miscellaneous functions

`LaTeX-counter` on `C-c k` generates a counter name with completion, using the names stored in the variable `LaTeX-standard-counters`.

`LaTeX-footnote` on `C-c f` generates a '`\footnote{}`'. Not only

does this save typing, but it also prevents my fingers from producing '`\bootnote{}`', which looks reasonable enough, but makes TeX unhappy. Non-`nil` values of the variables `LaTeX-footnote-end-with-newline` and `LaTeX-footnote-start-with-newline` request that newlines be generated before and after the footnote; the default values are `nil`.

`LaTeX-news` on `C-c C-n` views the '`latex.el`' file to show the revision history recorded in it.

`LaTeX-tab` on `C-c Tab` indents to the current '`\begin{...}`' ...'`\end{...}`' nesting level.

`LaTeX-protect` on `C-c C-p` inserts a '`\protect`' macro at the current point. It is needed whenever fragile commands (e.g. any LaTeX command that has a star form [14, p. 27]) are present in a moving argument [14, p. 151]. In particular, all macros in '`\index{}`' entries should be preceded by '`\protect`'. Otherwise, you can get long macro expansions that cause buffer-overflow problems for indexing programs, and make the index file hard to read. Also, macro expansions can introduce special characters which cause problems for indexing programs like '`makeindex`'.

`LaTeX-set-indentation` on `C-c ,` sets the current indentation column to the value of its argument, or to the cursor column if there is no argument. It is only rarely necessary to use this function, because the indentation is reset by several other functions.

`LaTeX-verb` on `C-c v` generates a '`\verb|...|`' command with the cursor after the first vertical bar. With an argument, it creates the '`\verb*|...|`' form, which makes spaces visible. The delimiter character can be changed from a vertical bar by reassigning the Emacs variable `LaTeX-verb-delimiter`. For example, to set it to a plus sign instead of a vertical bar:

```
(setq LaTeX-verb-delimiter ?\+)
```

The funny '`?\`' prefix is one of Lisp's character quotes. As shown elsewhere (see 1.3 [LaTeX macros], page 16), this assignment should normally be placed inside a mode hook. It can also be executed dynamically inside the Emacs minibuffer. Alternatively, you can use `M-x set-variable`; it will prompt for the variable name and value.

There are several other functions in '`latex.el`' that I shall not document here; they are intended for internal use only, and are subject to change without warning. They all begin with the reserved prefix `internal-`.

1.18 Miscellaneous variables

There are a number of variables that we have not yet described. You should not have to modify any of them, but you occasionally might want to know what they are.

`LaTeX-current-indentation` tracks the current environment indentation as the number of spaces from the left margin. It is updated dynamically by many functions.

`LaTeX-mode-map` holds the map that associates keys with functions.

`LaTeX-mode-version` is a string containing the version number and date of the last revision to the code. You should cite its value when reporting bugs; `LaTeX-gripe` puts it in the e-mail subject line.

`LaTeX-standard-environments` contains the list of environment names used by `LaTeX-begin-end-block`.

`LaTeX-standard-fonts` holds a list of font names and sizes.

`LaTeX-standard-italic-fonts` is a list of the LaTeX control sequences for fonts that require italic corrections.

`LaTeX-standard-list-environments` is a list of LaTeX environments which can have '`\item`' entries.

`LaTeX-standard-options` is a list of options that can go in the square brackets of the '`\documentstyle`' command. It is used by `make-LaTeX-document` and `make-SLiTeX-document`.

`LaTeX-standard-styles` is a list of styles that can go in the braces of the '`\documentstyle`' command. It is used by `make-LaTeX-document`.

`LaTeX-standard-unindented-environments` is a list of environments that must not be indented.

Variables beginning with the reserved prefix `internal-` are for internal use only; they are subject to change without warning, and will not be documented here.

1.19 Customizing lists

It is undesirable for users to have to duplicate the long initializations of standard fonts, macros, environments, and so on, when all they want to do is add a few more items to the lists.

Therefore, for each variable `LaTeX-standard-xxx`, there is a corresponding variable `LaTeX-extra-xxx` that is intended for user customization. The extra values are appended to the standard ones to make new lists that are used for command completion. Here are the names of these customization variables: `LaTeX-extra-counters`, `LaTeX-extra-dotted-abbreviations`, `LaTeX-`

`extra-environments`, `LaTeX-extra-fonts`, `LaTeX-extra-italic-fonts`, `LaTeX-extra-list-environments`, `LaTeX-extra-macros`, `LaTeX-extra-options`, `LaTeX-extra-styles`, `LaTeX-extra-tied-abbreviations`, and `LaTeX-extra-unindented-environments`.

Suppose for example that you want to add new environments named '`conjecture`', '`postulate`', and '`wildguess`' to the standard list used by `LaTeX-begin-end-block`. In the `LaTeX-mode-hook` in your '`.emacs`' file, add this assignment:

```
(setq LaTeX-extra-environments
      '(
        ("conjecture")
        ("postulate")
        ("wildguess")
        )
      )
```

When you subsequently type `C-c b`, these three new names will appear in the environment name completion list. Such lists are automatically alphabetized during the completion process, so the names can be given in any order in the assignment.

There is no provision for deleting names from the standard lists; after all, they are *standard*! Also, remember that the new additions are made known only to Emacs; you still have to teach LaTeX how to typeset them.

You may have wondered in the example above why I wrote '`("conjecture")`' instead of just '`"conjecture"`' in the assignment of `LaTeX-extra-environments`. The extra parentheses make a list out of each element in the variable. In '`latex.el`', all variables that hold multiple strings store them as *lists of lists* of strings, rather than as simpler lists of strings. This choice was intentional, because it facilitates adding additional related strings in the form of Lisp *association lists*.

The variables `LaTeX-extra-environments`, `LaTeX-extra-macros`, `LaTeX-standard-environments`, and `LaTeX-standard-macros` already use association lists to hold trailing braces and brackets, and descriptive comments.

Chapter 2

SliTEX support

SLiTEX works much like LATEX, so little additional support is required. All of the commands in the preceding sections can be used, and the editing mode is still called `LaTeX-mode`.

When you select a new SLiTEX file to edit, you can create an initial template with `make-SLiTeX-document` on `C-c s`. It prompts only for document style options, since the major style is fixed at '`slides`', and produces something like this:

```
% -*-latex-*-
% Document name: /u/beebe/tex/tugboat/foo.stx
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Mon Jul 29 08:15:15 1991
\documentstyle[]{slides}
\begin{document}
  \onlyslides{1-9999}
  \onlynotes{1-9999}
% \colors{} % {red,green,blue,etc.} colors
% \colorslides{} % input file
  \blackandwhite{<cursor>} % input file
\end{document}
```

The cursor is left positioned in the brace pair of the '`\blackandwhite{}`' macro, where you can type the name of another file that you can create in `LaTeX-mode`, using '`slide`' environments generated by `C-c b sli<space> RETurn`.

It is convenient to number each slide with a comment, so that you have the numbers available if you need them for the '`\onlynotes`' and

'`\onlyslides`' commands. The function `renumber-slides` on `C-c 0` modifies relevant commands in the buffer to read something like this:

```
  \begin{slide}{} % slide number 1
...
  \begin{slide}{} % slide number 2
...
  \begin{overlay}{red} % overlay number 2-a
...
  \begin{overlay}{green} % overlay number 2-b
...
  \begin{note} % note number 2-1
...
  \begin{slide}{} % slide number 3
...
  \begin{note} % note number 3-1
...
  \begin{note} % note number 3-2
```

Any existing comments on the commands are discarded. Notice the special handling accorded the SLiTEX '`note`' and '`overlay`' environments; they receive a major number equal to that of the closest preceding slide, followed by a hyphen and a minor number or letter [14, pp. 136–137].

The Emacs variable `SLiTeX-begin-slide` defines the regular expression used to find the start of the '`note`', '`overlay`', and '`slide`' environments.

Chapter 3

Letter support

LaTeX [14, pp. 66ff] provides a '`letter`' document style, but there is a fair amount of constant 'boilerplate' that has to be typed each time you use it. This suggests that the best approach is to have private template files that already have a letter outline in them, and then arrange to start each letter with a copy of the template already inserted into the buffer. Emacs' programmability makes this straightforward.

The startup file code in Table 1 loads a small Lisp file, '`letter.el`'. It defines a `letter-mode` function, and associates it with files with extension '`.ltr`'. If I then visit a new file '`rb-jones.ltr`', I immediately get the buffer shown in Table 8, with the cursor positioned in the empty brace pair in '`\begin{letter}{}`', ready for entry of the address. The query in '`\opening{Dear ?}`' is a reminder to enter something there, and then the body of the letter can be typed in following the '`\opening`' line.

My personal style option '`nhfb-letter`' provides for a University and Department letterhead on the first page, and defines a few private macros that I often need in my correspondence.

Once the letter file has been saved, I suspend Emacs, or switch to another workstation window, and then process the letter by typing

```
make LET=rb-jones
```

The UNIX '`make`' utility is a wonderful tool; here it directs the execution of commands to expand tabs to blanks, changes the file protection to remove access to all but its owner, runs LaTeX, and then runs a PostScript DVI driver, sending the output to a nearby printer. Then it sends the output PostScript through a shell filter that modifies the '`showpage`' command to print the string *File Copy* in outline letters diagonally across each page,

```
% -*-LaTeX-*-
% Document name: /u/beebe/rb-jones.ltr
% Creator: Nelson Beebe [beebe@math.utah.edu]
% Creation Date: Mon Jul 29 20:06:18 1991

%------------------------------------------
% EVERYTHING TO THE RIGHT OF A  %  IS A REMARK
% TO YOU AND IS IGNORED BY LaTeX.
%
% WARNING!  DO NOT TYPE ANY OF THE FOLLOWING
% 10 CHARACTERS AS NORMAL TEXT CHARACTERS:
%       &   $   #   %   _    {   }   ^   ~   \
%
% The following seven are printed by typing a
% backslash in front of them:
%       $  &  #  %  _  {  and  }.
%------------------------------------------

\documentstyle[nhfb-letter]{letter}
\begin{document}
\setfilename{foo.info}
\begin{letter}{<cursor>}

\opening{Dear ?}

\closing{Sincerely,}
\ps{{$\cal NHFB$}/\LaTeX}
\end{letter}
\end{document}
```

Table 3.1: Letter template.

and queues that to the printer as well. On those occasions where I require multiple copies of the original letter, I just type something like

```
make LET=rb-jones C=3
```

This gives me one file copy, and three originals for signing and mailing.

With this support, preparation of typeset letters is just as easy as typewritten ones, and the result looks much better. And I can use mathematics and a variety of fonts too!

Chapter 4

Bug reporting

`LaTeX-gripe` on `C-c g` makes it easy to complain about `LaTeX-mode`; it puts you in an e-mail buffer with a mail header something like this:

```
To: beebe@math.utah.edu
Subject: LaTeX-mode gripe report {beta test 0.23 [08-May-1991]}
```

The standardized subject field makes it easy for the author to identify such messages in his voluminous e-mail correspondence, and also automatically identifies the code version.

If you don't have an electronic mail connection yet, you probably soon will. For now, just send me your comments in postal mail. Machine-readable submissions can be sent on IBM PC or Apple Macintosh floppy disks.

Chapter 5

Conclusion

`LaTeX-mode` editing support in GNU Emacs provides a very powerful, and pleasant, way to prepare LaTeX and SLiTeX input.

The original TOPS-20 version written in TECO was begun in October 1984, and used extensively at Utah until our DEC-20/60 retired on October 31, 1990. It consists of about 1000 lines of code defining 24 functions.

The power of Lisp over TECO made it possible to be much more ambitious in the functionality of the GNU Emacs version, and I consequently suspended further development of the TECO code in January 1988 when I began the Lisp coding. The current Lisp version is more than 4 times the size of the TECO one, both in lines of code, and in editing functions. Despite the increase in size, it is *much* easier to maintain.

The GNU Emacs implementation was released for beta testing in March, 1988. About 70 sites have participated in the beta test of '`latex.el`'. I thank these many people for their contributions of comments, and sometimes, even code.

Is it bug free? No computer program for a realistic problem ever is; there are simply too many details for humans ever to get completely right. The '`latex.el`' revision history shows mostly additions of new functionality, and code changes for better performance. The virtue of the Emacs editing environment is that new functions can be written, debugged, and tested in the editor itself, providing immediate feedback and confidence in the correctness of operation.

Much of the code is in small functions that are independent of the others, following the Lisp programming tradition. After removal of comments and empty lines, and ignoring the initializations of large tables, the average is about 22 lines of code and documentation per function. This is close to the 20-line average for all of GNU Emacs. Documentation contributions are

included here because Emacs functions, and some variables, carry their own documentation with them, even after compilation. That documentation is readily available for viewing in the editor.

I believe that this code could serve as a model for other Emacs-like editors that have an extension language in which new functions can be written. I had planned to write an EEL version for the '`epsilon`' editor on PC DOS, but alas, have not found the time to do so. This is not a small job, but it should be straightforward for someone familiar with both Lisp and EEL. The '`latex.el`' file is over 4200 lines long, with about 155KB of text; Emacs compiles it into about 100KB of byte codes. The latter figure should give some idea of the memory requirements for implementations in other editors. I hope some reader will prepare such a translation and make it freely available, as '`latex.el`' is. A large portion of the TEX community uses personal computers that are inadequate for GNU Emacs, but have smaller Emacs-like editors available, and such editing support would be a great boon for them.

If you plan to undertake such a translation effort, check with the author of '`latex.el`' first to find out if anyone else is already working on one.

Some other work in the TEX community for editing support has been reported in the TEX User Group TEXniques series [16, 17, 20] and in TUGboat [7, 8, 9, 10, 11, 12, 21]. I believe that '`latex.el`' is much more powerful, however.

Those who use UNIX workstations or VMS systems where GNU Emacs is already available should eventually find '`latex.el`' in the '`emacs/lisp`' directory, and until then, can obtain it directly from the author, provided they have Internet electronic mail or '`ftp`' access.

If you have a UNIX or VMS system, but do not yet have GNU Emacs, you can get Emacs from any site that already has it, or better, you can get the latest version directly from the Free Software Foundation. The latter is accessible either via Internet anonymous '`ftp`' to the machine '`prep.ai.mit.edu`', or through postal mail to

> Free Software Foundation, Inc.
> 675 Massachusetts Ave
> Cambridge, MA 02139
> USA

The postal mail approach carries some shipping and documentation charges, but the software itself is free.

Bibliography

[1] James Alexander. TIB: a reference setting package for TEX. *TUGBoat*, 7(3):138, October 1986.

[2] James Alexander. TIB: a reference setting package, update. *TUGBoat*, 8(2):102, July 1987.

[3] R. L. Aurbach. User's guide to the IdxTEX program. *TEXniques, Publications for the TEX community*, (3):i, 1–14, 1987.

[4] J. L. Bentley and B. W. Kernighan. Tools for printing indexes. *Electronic Publishing—Origination, Dissemination, and Design*, 1(1):3–18, April 1988.

[5] Pehong Chen and Michael A. Harrison. Automating index preparation. Technical Report 87/347, Computer Science Division, University of California, Berkeley, CA, USA, March 1987. This is an expanded version of [6].

[6] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software—Practice and Experience*, 19(9):897–915, September 1988. The LATEX text of this paper is included in the `makeindex` software distribution.

[7] Adrian Clark. An enhanced TEX-editor interface for VMS. *TUGBoat*, 10(1):14–15, April 1989.

[8] Victor Eijkhout. A new editor. *TUGBoat*, 11(4):605, November 1990.

[9] A. Hoenig and M. Pfeffer. TEX text editors for the IBM PC. *TUGBoat*, 7(1):51, March 1986.

[10] Anita Z. Hoover. Using WordPerfect 5.0 to Create TEX and LATEX Documents. *TUGBoat*, 10(4):549–559, December 1989.

[11] Kathy Hornbach. VAX Language Sensitive Editor Templates and Guide for use with LaTeX. *TUGBoat*, 7(2):99, June 1986.

[12] Karl Kleine. Customized editors for TeX. *TUGBoat*, 7(2):111, June 1986.

[13] Donald E. Knuth. *The TeXbook*. Addison-Wesley, 1984.

[14] Leslie Lamport. LaTeX—*A Document Preparation System—User's Guide and Reference Manual*. Addison-Wesley, 1985.

[15] Leslie Lamport. `makeindex`*: An Index Processor For* LaTeX, 17 February 1987. The LaTeX text of this paper is included in the `makeindex` software distribution.

[16] Kent McPherson. VAX Language-Sensitive Editor (LSEDIT) Quick Reference Guide. *TeXniques, Publications for the TeX community*, (1):ii, 1–9, 1988.

[17] Paul M. Muller. FASTeX: A PC text editor and front-end for TeX. *TeXniques, Publications for the TeX community*, (7):235–254, 1988.

[18] Rainer Schöpf. A new implementation of the LaTeX `verbatim` and `verbatim*` environments. *TUGBoat*, 11(2):284–296, June 1990.

[19] Margaret Shertzer. *The Elements of Grammar*. Collier Books, Macmillan Publishing Company, 1986.

[20] Stephan von Bechtolsheim. Using the Emacs editor to safely edit TeX sources. *TeXniques, Publications for the TeX community*, (7):195–202, 1988.

[21] Linda Williams and Linda Hall. Increased efficiency using advanced EDT editing features. *TUGBoat*, 11(3):421–424, September 1990.

Concept Index

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Control Sequence Index

"

L

T

Function Index

L

M

N

Q

R

S

T

U

X

Key Index

E

L

M

R

Person Index

P

S

V

W

Variable Index

P

S

T