GNU Emacs TeX Mode

— version 1.6 —

Pehong Chen*

Computer Science Division
University of California
Berkeley, CA 94720

February 1, 1987

* Sponsored in part by the State of California MICRO Fellowship, by the National Science Foundation under Grant MCS-8311787, and by the Defense Advanced Research Projects Agency (DoD), ARPA Order No. 4871, monitored by Naval Electronic Systems Command, under Contract No. N00039-84-C-0089.

Contents

1 Introduction

TEX-mode is part of an Emacs-based environment for editing TEX documents [2]. It is a GNU Emacs [6] interface to TEX [3], LATEX [4], and the basic execution of AMS-TEX [8]. It comprises the following subsystems: `TeX-mode.el`, `TeX-match.el`, `TeX-custom.el`, `TeX-misc.el`, `TeX-spell.el`, `TeX-bib.el`, and `TeX-index.el`. The idea is that only the most essential functions are defined in `TeX-mode.el` in order to minimize *TEX-mode* startup time. Other subsystems will be autoloaded as needed. The file `TeX-mode.el` includes primarily the function `tex-mode` which defines such attributes as syntax entry modifications, key bindings, and local variables for *TEX-mode*. Also included is the code for automatic matching of dollar signs ($'s) and double quotes ("'s). These functions will be loaded when the first file with suffix '`.tex`' is visited in an Emacs session.

The second file `TeX-match.el` contains an assorted set of facilities for matching non-standard delimiter pairs including the opening and closing of LATEX environments. It is also possible to introduce delimiter pairs of your own besides the ones supported by the system. The mechanism that involves this level of customization is defined in the file `TeX-custom.el`.

The next file `TeX-misc.el` is a collection of miscellaneous programs designed to shorten the edit-compile-debug cycle. Major features are document type checking, TEX/AMS-TEX/LATEX/SLITEX execution, previewing, printing, error positioning, etc.

The last three files deal with preprocessing and postprocessing which include spelling checking, bibliography making, and indexing. The file `TeX-spell.el` is a spelling interface specially tailored to TEX and LATEX documents with features such as dictionary lookup and searching under low speed connections ($\leq$ 2400 baud), in addition to spelling checking. The file `TeX-bib.el` is a mechanism for looking up citation entries in BIBTEX [4,5] databases and for constructing bibliographies for TEX and LATEX documents. Finally the file `TeX-index.el` makes use of the simple indexing facility available in LATEX for creating `\index` entries and preparing the actual index, the `.ind` file.

This document describes each user-level function available in version 1.6 of *TEX-mode*. It assumes the reader knows the basics of Emacs, TEX, and LATEX. The goal is to show you how to use the system with some instructive examples. The next chapter gives some general guidelines to making *TEX-mode* work under your environment. Chapter 3 describes *TEX-mode*'s basic abstractions in terms of document structure and command key bindings. Chapter 4 introduces the various user-level functions and their corresponding commands. Section 4.1 discusses several delimiter matching schemes available in the mode. Interfaces to spelling checking, bibliography making, and indexing are discussed in Sections 4.2, 4.3, and 4.4, respectively. Functions having to do with 'compile-debug-preview-print' are covered in Section 4.6. Finally all user level functions are summarized in Chapter 5, followed by two sets of indices in Chapters 6 and 7.

Disclaimer. Although *TEX-mode* has been extensively tested, there is no warranty that the functions described in this document are bug-free. The author does not accept any responsibility to anyone for the consequences of using it or for whether it serves any particular purpose or works at all.

Bugs/Comments. Bugs and comments on both the code and this document are welcome. Please send them via electronic mail to `phc@renoir.berkeley.edu`.

2 Installation and Startup

To get *TEX-mode* autoloaded, add the following two lines of code to your `.emacs`:

```
(setq auto-mode-alist (cons '("\\.tex\$" .  tex-mode) auto-mode-alist))
(autoload 'tex-mode "TeX-mode" "Major mode for editing TeX-based documents" t)
```

You can also enter *TEX-mode* manually by loading the file `tex-mode` first and then invoke the function `tex-mode` at `M-x` prompt.

2.1 TEX-mode Hook

The variable `tex-mode-hook` is the last object that gets evaluated whenever the function `tex-mode` is invoked, which means whatever defined in the hook overwrites the default. A typical `tex-mode-hook` is defined in the following way:

```
(setq tex-mode-hook
  (function
    (lambda ()
      <BODY>)))
```

For the convenience of version control (typing `tex-mode-version` at the `M-x` prompt returns the current *TEX-mode* version), the user is advised to put his local changes in the hook rather than modifying the various files of *TEX-mode* directly. The abbreviation table (see below) is one example that goes into the hook, your preferred key bindings may be another, and other TEX subsystems that you've developed can also be loaded from the hook.

2.2 Minor Mode for Abbreviations

There are no abbreviations defined in *TEX-mode*, but the user can define his own abbreviations in `tex-mode-hook`. For instance, a typical `.emacs` may contain a `tex-mode-hook` whose body is:

```
(define-mode-abbrev "tx"  "{\\TeX}")
(define-mode-abbrev "atx" "{\\AmSTeX})
(define-mode-abbrev "btx" "{\\BibTeX}")
(define-mode-abbrev "ltx" "{\\LaTeX}")
(define-mode-abbrev "stx" "{\\SliTeX}")
(abbrev-mode 1))))
```

Note that the `Abbrev` minor mode is set by invoking the function `abbrev-mode` with a positive number as argument. It can be reset by passing 0 as the argument.

The command **C-c C-a SPC** (`bibtex-abbrev-enable`) unconditionally enables the `Abbrev` minor mode. Conversely **C-c C-a DEL** (`bibtex-abbrev-disable`) disables it unconditionally.

2.3 Minor Mode for Auto-Filling

By default, *TEX-mode* has the `Auto Fill` minor mode turned off. If this minor mode is desired, put

```
(auto-fill-mode 1)
```

in your `tex-mode-hook`. When the mode is set one can keep typing beyond the right margin without any explicit **RET** or **LFD** and the line will wrap around automatically. The variable `fill-column` is set to 78 in *TEX-mode* instead of the Emacs default value of 70. This is the column beyond which automatic line-wrapping would happen when a space is hit. If instead N is desired column boundary, put

```
(setq fill-column N)
```

in your hook.

The standard Emacs line wrapping is modified a bit such that the wrapped line is indented by the amount of current indentation. Typing **LFD** (`tex-newline-indent`) in *TEX-mode* has the same effect. We shall find this feature useful in Section 4.1.4. Incidently, **TAB** is a self-inserting character in *TEX-mode* which simply moves the cursor to the next tab position.

The command **C-c LFD SPC** (`tex-autofill-enable`) enables the `Auto Fill` minor mode unconditionally. Conversely **C-c LFD DEL** (`tex-autofill-disable`) disables it unconditionally.

2.4 System Dependencies

There are some system-dependent spots in *TEX-mode*. First, a number of external programs are invoked inside the mode: `tex`, `amstex`, `latex`, `slitex`, `texdvi`, `amstexdvi`, `latexdvi`, `slitexdvi`, `dvitool`†, and `lpr` are used in `tex-misc.el`; `bibtex` is used in `tex-bib.el`, `tex-spell.el` uses `spell`, `detex`, and `delatex`, `tex-index.el` uses `makeindex`, etc. You need these programs to make *TEX-mode* fully functional.

Second, some of the default settings in *TEX-mode* may not be right for your local environments. You can redefine the variables involved in your `tex-mode-hook`. For instance, **C-c C-p all** `tex-print-all`) prompts you for the printer option. The default is the list `(ip, cx, dp, gp)` which represents the names of laser printers available to our group. If what you have is the list `(a, b, c)` with 'a' being the one you use most often, you could say

```
(setq tex-printer-list "(a, b, c)" "Printers available locally")
(setq tex-printer-default "a" "Default printer")
```

in your hook. Similarly if you have a different previewer called `previewtool` or if you are using a different printing scheme called `print`, you can alter the default values by saying

```
(setq tex-softcopy "previewtool" "My DVI previewer")
(setq tex-hardcopy "print" "My printing scheme")
```

in your `tex-mode-hook`.

† The programs `texdvi`, `amstexdvi`, `latexdvi`, and `slitexdvi` all call `dvitool`, a TEX DVI previewer running on the SUN workstation, after formatting. These programs are available through the Berkeley VORTEX distribution (`dist-vortex@berkeley.edu` or `ucbvax!dist-vortex`) while the formatting programs `tex`, `amstex`, `latex`, and `slitex` are available through the Unix TEX distribution at University of Washington.

2.5 Site Initialization

It is possible to setup a local *TEX-mode* environment by redefining the variables in a file called `tex-init.el`. This file does not come with *TEX-mode*. But if either the file or its compiled form (`tex-init.elc`) exists in `EMACSLOADPATH`, the command

```
(load "tex-init")
```

will be executed whenever the mode is invoked. Otherwise the default values remain intact unless they are redefined in the user's `tex-mode-hook`.

In effect this facility provides a site-wide initialization for everyone using the mode. Site-specific attributes like the speller, previewer, printers, etc., which would probably be different from the default settings but identical for most users in the community, can be redefined in this initialization file. For example, for site X the *TEX-mode* administrator can put

```
(setq tex-printer-list "(a, b, c)" "Printers available at site X")
(setq tex-printer-default "a" "Default printer of site X")
(setq tex-softcopy "previewtool" "DVI previewer used at site X")
(setq tex-hardcopy "print" "printing/spooling scheme at site X")
```

in the file `tex-init.el`, thereby alleviating users from having to deal with these attributes in their individual `tex-mode-hook`'s.

3 Basic Abstractions

3.1 Document Structure

Source Level

At the source level, *TEX-mode* makes the distinction between a *document* and a *file* by acknowledging that a TEX-based document may involve multiple files connected by `\input` or `\include` commands. *TEX-mode* views a document as a tree of files with edges being the connecting commands. The root of a document tree is called the *master file*. Operations involving the entire document must be started from the master file. The processing sequence is the preorder traversal of the tree. In *TEX-mode*, each individual file has a link to the master to assure any global commands initiated in its buffer will always start from the master. The link to master also makes it possible to separately format any component file or any part of it. The technique used in *TEX-mode* to do separate formatting is discussed in Section 4.5.1. The master pointer (default to the current file name) is to be specified by the user the first time an operation involving the entire document is invoked, as in

```
Master File:  foo.tex
```

where `foo.tex` is the current file name. If the entered file name does not correspond to any existing file, the user will be cautioned by

```
Master File:  goo.tex [no match]. Use it anyway? (y or n)
```

where `goo.tex` is the newly entered file name which does not exist yet. Answering **SPC** or '`y`' makes `goo.tex` the master file pointer anyway. Answering **DEL** or '`n`' aborts the action.

Once specified, the information is stored as a comment line at the beginning of the file and becomes implicit for future invocations. The command **C-c 0** (`tex-check-master-file`) can be used to check or change the master file pointer.

The next level of abstraction is a *file*, or when loaded in Emacs, a *buffer*. Objects of even smaller granularities include *regions* and *words*. A *region* is a piece of text, including any white space, bounded by a marker and the current cursor position (i.e. *point* in GNU Emacs). A *word* in *TEX-mode* is a piece of text with no white space in it.

DVI Level

At the output DVI level, the distinction is less complex. The only abstractions are the DVI file as a whole and subranges of one extracted out as another file. Normally DVI files themselves are not visited in Emacs, therefore in a buffer bound to the TEX source `foo.tex`, the implicit operand for operations such as *preview* and *print* is `foo.dvi` instead of `foo.tex`. With the abstractions, it is possible to preview or print a DVI file partially as well as in its entirety.

3.2 Document Type

TEX-mode maintains the notion of *document type* which may be either TEX, AMS-TEX, LATEX, or SLITEX in our current version. The type information is needed when the user tries to execute operations involving programs which are type-specific, such as the formatter (i.e. `tex`, `amstex`, `latex`, or `slitex`) and the document filter (i.e. `detex` or `delatex`). However, such information is implicit to the user except for the first time — once specified it will be saved as a comment line in the document to be read by later invocations. In other words, from the user's point of view, operations in *TEX-mode* are *generic*. For instance, an operation is known as *format* at all times instead of as `tex`, `amstex`, `latex`, or `slitex` under different situations. *TEX-mode* does operator overloading implicitly by consulting the type information.

The current document type is specified by inserting "`% Document Type: `" followed by either "`TeX`", "`AmSTeX`", "`LaTeX`", or "`SliTeX`" as the first line of the document. However, you don't have to type the line yourself. If the system fails to find that comment line, before it starts formatting it prompts you with:

```
Document Type?  (RET/t for TeX; a for AmSTeX; l for LaTeX; s for SliTeX)
```

which means you can answer either **RET** or '`t`' for a TEX document, and '`a`', '`l`' or '`s`' for an AMS-TEX, LATEX or SLITEX document respectively and the comment line will become the first line of your document as soon as your answer is entered. Once the document type is determined, the Emacs mode name changes accordingly.

The command **C-c l** (`tex-check-document-type`) can be used to check or change the current document type. Only the four types mentioned above will be accepted. The checking is case-insensitive.

3.3 Document Preamble and Postamble

The notion of master file plays an important role in separate formatting. Functions such as *format* can operate on either the entire document, a buffer, or a region in buffer. A document preamble and similarly a postamble can be associated with the master to contain the document's global context. To separately compile a component file or its subregion, a mechanism is available in *TEX-mode* that includes in a temporary file the document's preamble and postamble with the selected text inserted in between. The system will then run the formatter on this temporary file. This technique is primarily for debugging purposes as there is no provision for linking separately generated DVI files into one big DVI file. However, for users wanting only a quick look at a relatively small portion of a document in the debugging phase, this automatic facility turns out to be very valuable.

To create the preamble of a document for a master file (call it `foo.tex`), we put the global context in a file called `foo+.tex`. Normally this should include the definition of all macros used in the entire document and in AMS-TEX and LATEX/SLITEX, in particular, the opening of the document environment (i.e. `\document` and `\begin{document}`, respectively). The postamble must go into the file `foo-.tex` and should contain matching closing commands such as `\bye` or `\end` in TEX, `\enddocument` in AMS-TEX, and `\end{document}` in LATEX/SLITEX.

You can also set a region in an existing master file and have the pre- or post-amble file created automatically using the command **C-c C-\ SPC** (`tex-make-preamble`) or **C-c C-\ DEL** (`tex-make-postamble`). If any component buffer (file) or any region within a buffer is to be formatted, it will be copied to a temporary buffer enclosed by the pre- and post-ambles. This temporary buffer will have a master pointing to itself and will be formatted as if it were a stand alone document.

If the pre- or post-amble file is not found when a separate formatting command is issued, *TEX-mode* uses the following default values:

document type	*preamble*	*postamble*
TEX	(none)	`\bye`
AMS-TEX	`\document`	`\enddocument`
LATEX	(retrieved from master)	`\end{document}`
SLITEX	(see below)	`\end{document}`

In the TeX case, we assume no document preamble and the postamble is simply `\bye`. The $\mathcal{AMS}$-TeX case is straightforward: `\document` and `\enddocument` are pre- and post-ambles, respectively. The postamble in both LaTeX and SLiTeX is `\end{document}`. The preamble in these cases, however, is a bit complicated. For a LaTeX document, we retrieve everything before `\begin{document}`, including the statement itself, in the document master file as the preamble. If the statement `\begin{document}` is not found in the master, an error is raised. Finally the default preamble for SLiTeX is always

```
\documentstyle{slides}
\begin{document}
  \blackandwhite{noo}
```

where `noo.tex` is the buffer where separate formatting is issued.

3.4 Commands

The central issue for command naming and key binding is uniformity. Except for delimiter matching commands, *TeX-mode* by and large obeys the convention that a function name consists of three parts: (1) prefix (`tex-`), (2) generic operator, and (3) abstract object. The corresponding key binding will be the **C-c** prefix, followed by **C-** and the first letter of the middle part, then the first letter of the last part. One example is the functions `tex-format-document` and `tex-format-buffer` with the corresponding key bindings being **C-c C-fd** and **C-c C-fb**. The key binding scheme may vary slightly in some cases because of keyboard limitations or consideration of typing conveniences.

4 Functional Aspects

4.1 Delimiter Matching

In *TEX-mode*, automatic delimiter matching applies to parentheses, brackets, and braces (i.e. `(...)`, `[...]`, and `{...}`). That is, whenever a self-inserting closing delimiter (i.e. `)`, `]`, or `}`) is typed, the cursor moves momentarily to the location of the matching opening delimiter (i.e. `(`, `[`, or `{`). Matching standard delimiter pairs like these is relatively easy in Emacs: simply by modifying syntax entries.

TEX-mode has a facility to examine the matching of these delimiters. You can use the command **C-c (** (`tex-bounce-backward`) after a closing delimiter to check the matching opening delimiter with the cursor moved to its position for a short duration. Conversely, typing **C-c)** (`tex-bounce-forward`) before an opening delimiter checks for and bounces forward to the matching closing delimiter. If the bouncing point is invisible in the current window, the line containing it will be echoed in the minibuffer with the delimiter quoted.

The matching of other delimiters requires special treatment. Matching delimiters such as quotes (i.e. `` `...' ``, ``` ``...'' ```, and `"..."`) and TEX dollar signs ($...$) cannot be done automatically by syntax entry modification. For example, the symbol '`'`' is the right quote as well as the apostrophe. Modifying syntax entry in the normal way is inappropriate because we don't want the cursor to bounce in the case of apostrophes. Matching double quotes (`"..."`) and TEX dollar signs ($...$) is even harder because the opening and closing delimiters are identical in those situations. The rest of this section is devoted to the special methods available in *TEX-mode* that handle these odd cases.

4.1.1 Zone Matching (Semi-automatic)

The first method is based on the notion of *TEX zone*. A TEX zone is an Emacs region set with an alternate marker. The term *zone* is introduced here to emphasize that it is not an ordinary Emacs region, although the only difference is in the marker. The way non-standard delimiter matching works is to open a TEX zone by typing **C-c SPC** (`tex-zone-open`), followed by whatever to be included in the zone, and finally at the end, close it by typing **C-c** followed by **ESC** and the delimiter itself.

The following examples may help to clarify the idea.

C-c SPC `math mode` **C-c ESC-$**	⟶	`$math mode$`
C-c SPC `display math mode` **C-c ESC-d**	⟶	`$$display math mode$$`
C-c SPC `single quote` **C-c ESC-'**	⟶	`` `single quote' ``
C-c SPC `double quote` **C-c ESC-"**	⟶	``` ``double quote'' ```

In the same spirit, *TEX-mode* extends the notion of TEX zone to pieces of text included in `\hbox`'es, `\vbox`'es, `\centerline`'s, or useful font types such as `\bf`, `\it`, `\rm`, `\sl`, and `\tt`. A command belonging to this category can be entered by typing **C-c SPC**, followed by the text, and lastly **C-c** followed by either **ESC-c**, **ESC-h**, **ESC-v**, **ESC-b**, **ESC-i**, **ESC-r**, **ESC-s**, or **ESC-t**. Examples are the following:

C-c SPC `an hbox` **C-c ESC-h**	$\longrightarrow$	`\hbox{an hbox}`
C-c SPC `a vbox` **C-c ESC-v**	$\longrightarrow$	`\vbox{a vbox}`
C-c SPC `center` **C-c ESC-c**	$\longrightarrow$	`\centerline{center}`
C-c SPC `boldface` **C-c ESC-b**	$\longrightarrow$	`{\bf boldface}`
C-c SPC `italic` **C-c ESC-i**	$\longrightarrow$	`{\it italic\/}`
C-c SPC `roman` **C-c ESC-r**	$\longrightarrow$	`{\rm roman}`
C-c SPC `slanted` **C-c ESC-s**	$\longrightarrow$	`{\sl slanted\/}`
C-c SPC `typewriter` **C-c ESC-t**	$\longrightarrow$	`{\tt typewriter}`

A TEX zone is closed automatically when a command with **C-c ESC-** prefix is entered. The user may forcedly close a zone using the command **C-c DEL** (`tex-zone-close`). The zone marker position may be inspected by the command **C-c C-z** (`tex-zone-inspect`). The effect of the inspection is the cursor being moved to the marker position momentarily with its numbering displayed in the minibuffer. With positive prefix argument N, the N^{th} marker from top is inspected. A non-positive prefix argument will be converted to 1 implicitly.

If a zone starts or ends in a non-word position, the user will be prompted to confirm it. There are three possibilities at this point: (1) go to the left boundary of the word (by giving command '`l`'), (2) go to the right boundary (command '`r`'), or (3) confirm current point position (any other key). The word boundary here refers to a space (ASCII 32) rather than Emacs' non-word characters in the general sense. The following example demonstrates boundary checking. The sequence

`bold`**C-c SPC**`face type`**C-c ESC-b**

will trigger the boundary confirmation prompt:

`Confirm position 'bold|face' (l, r, else=yes)`

where each option produces a different result:

'`l`' (for *left*)	$\longrightarrow$	`{\bf boldface type}`
'`r`' (for *right*)	$\longrightarrow$	`boldface {\bf type}`
any other key stroke	$\longrightarrow$	`bold{\bf face type}`

The variable `tex-boundary-check-on` (default `t`) can be set to `nil` in `tex-mode-hook` to disable the checking. Alternatively, the command **C-c C-t ESC** (`tex-toggle-boundary-check`) toggles the checking in *TEX-mode*.

TEX zones can be opened and closed in a reverse order. This is convenient when moving the point backward. Suppose the sentence

C-c ESC-i█`This is to be italicized.`**C-c SPC**

is already in the buffer and point is tracing backward from the right and the bottom. Typing **C-c SPC** at the end of that sentence and closing it by **C-c ESC-i** at the beginning yields

`{\it This is to be italicized.\/}`

just as what it should be when going forward.

TEX zones can be nested. Each **C-c SPC** pushes a marker onto a stack and each closing command pops the stack, closing the topmost zone. For instance, the sequence

C-c SPC C-c SPC`nested`**C-c ESC-i**`{\TeX} zones.` **C-c ESC-b**

produces

`{\bf {\it nested\/} {\TeX} zones.}`

4.1.2 Word Matching (Semi-automatic)

The second method doesn't involve explicit regions. Typing **C-c** followed by one of the delimiters described above automatically puts a pair of matching delimiters around the previous word. Again, word boundaries are space and in-word positions are subject to confirmation unless the boundary checking mechanism is turned off. Examples:

`f(x)=y` **C-c d**	⟶	`$$f(x)=y$$`
`slanted` **C-c s**	⟶	`{\sl slanted\/}`

This set of commands accepts a number as the optional prefix argument (**C-u**). Hence

`f(x)=y, f(w)=z` **C-u 2 C-c d**	⟶	`$$f(x)=y, f(w)=z$$`

The prefix argument can be negative. The difference is that instead of going backward, a negative prefix argument forces it to search forward for the word boundary. To avoid using negative prefix, however, a set of mirror commands is bound to **C-c-4** (**4** for forward) followed by the delimiter. Thus **C-u -1 C-c d** is the same as **C-c-4 d**. Conversely, **C-u -1 C-c-4 t** is equivalent to **C-c t**.

4.1.3 Automatic Matching

Automatic matching of identical opening and closing delimiters is a difficult task. The situation is further complicated by the TEX dollar sign because single dollar pairs ($...$) denote *math mode* whereas double dollar pairs ($$...$$) represent *display math mode* in TEX. A correct mechanism not only has to know which self-inserting $ or $$ is an opening delimiter and which is a closing one but also must be clever enough so that the second $ in $$ does not match its preceding $.

TEX-mode is designed to handle all cases correctly. It maintains a buffer-specific list of markers (`tex-dollar-list`) as $'s and $$'s are inserted or deleted. The fundamental assumption is that text included in either math mode stays in one paragraph. Hence the scope of $ or $$ matching is restricted on a per paragraph basis. The list `tex-dollar-list` is reconstructed whenever a $ is entered in a different paragraph. Based upon this assumption a significant amount of overhead is avoided.

An extra $ will be inserted automatically if a $ is found to be matching against a $$. There are actually two scenarios:

1. Typing "`$$f(x)=y$`" with previous $'s in the paragraph well-balanced will become "`$$f(x)=y$$`" as soon as the $ immediately following '`y`' is entered.
2. Similarly, `$f(x)=y$$` becomes `$$f(x)=y$$` as soon as the $$ next to '`y`' is entered.

In particular, it still matches correctly if two dollar signs instead of just one were entered next to the letter '`y`' in the first scenario. No automatic insertion will be done in this case. In fact, *TEX-mode* does not allow three dollar signs ($$$) in a row. The mechanism always looks backward for the preceding two characters in the buffer. If $$ is found, the third $ will not be echoed. (It does cause bouncing, however, if the $$ is a closing delimiter.)

TEX treats an escaped dollar sign (\$) as a pure symbol instead of a math mode delimiter, so does *TEX-mode*. That is, a dollar sign immediately following a backslash is ignored by the matching engine and does not trigger bouncing.

Automatic matching may be disabled and enabled by toggling **C-c C-t $** (`tex-toggle-dollar`). Disabling the mechanism is sometimes useful because the echoing of dollar signs may become uncomfortably slow on a heavily loaded system. Doing everything right for the dollar sign has a certain amount of efficiency penality. When the penalty is intolerable in some situations, you have the option of turning it off, making the $ a plain self-insert command. The variable `tex-match-dollar-on` may be set to `nil` in your `tex-mode-hook` so that $ matching is off initially.

A somewhat parallel but not as complex case applies to '`"`'. That is, typing `"` by default invokes automatic matching and **C-c C-t "** toggles the matching mechanism. The variable `tex-match-quote-on` can be set `nil` in your hook to turn off `"` matching from the beginning.

4.1.4 LaTEX Environments

One of the most commonly used commands in LaTEX is a pair of `\begin` and `\end` which is normally used to embrace a large piece of text under a certain *environment*. Environments can be nested in the obvious way, just as in any block-structured language. With several levels of environments in place, proper indentations become essential to readability.

TEX-mode has a facility that opens and closes LaTEX environments automatically with proper indentations inserted. The command **C-c C-l SPC** (`tex-latex-open`) prompts you for the name of the environment. If there isn't one, simply type carriage return. Otherwise you will be prompted again for its associated arguments, if any. Typing **C-c C-l DEL** (`tex-latex-close`) matches and bounces backward momentarily to the innermost opening `\begin`.

The following example shows how it works. Typing **C-c C-l SPC** gets

```
LaTeX \begin{env}, specify env (RET if none):
```

Suppose the desired environment is `minipage`, it then inserts `\begin{minipage}` before point and prompts you with:

```
Arguments to environment {minipage} (RET if none):
```

to which you may answer say `[t]{1in}`. Now the part of buffer of concern reads:

```
\begin{minipage}[t]{1in}
  █
\end{minipage}
```

where point is designated by █. The indentation is determined by the variable `tex-latex-indentation` whose default value is 2. As usual, you can assign it another value in `tex-mode-hook`. Notice that unlike answers to the first prompt, brackets and braces must go with the text for the argument part.

If somehow `\end{...}` is lost, or if `\begin{...}` had been entered manually, you can use **C-c C-l DEL** (`tex-latex-close`) to close a LaTEX environment. This command will grab the environment name from the inner-most `\begin{...}` and line up with the `\begin` properly. However, the text in between will not be affected. *TEX-mode* does not have a beautifier for TEX code, but basic indent facilities available in GNU Emacs such as **C-x C-i** (`indent-rigidly`) can be used to adjust the text.

TEX-mode also has a number of LaTEX environments hard-wired in such a way that if the corresponding command is invoked, both its opening and closing delimiters (i.e. `\begin{...}` and `\end{...}`) are inserted at the same time with the cursor positioned in a new line in between. Some of the environments may require you to type in the arguments as in **C-c C-l SPC**. The following is a complete list of LaTEX environments currently supported.

Function Name	*Key Binding*	*Arguments?*
`tex-latex-array`	**C-c C-l a**	yes
`tex-latex-center`	**C-c C-l c**	no
`tex-latex-enumerate`	**C-c C-l e**	no
`tex-latex-figure`	**C-c C-l f**	no
`tex-latex-itemize`	**C-c C-l i**	no
`tex-latex-picture`	**C-c C-l p**	yes
`tex-latex-quote`	**C-c C-l q**	no
`tex-latex-tabbing`	**C-c C-l TAB**	yes
`tex-latex-table`	**C-c C-l t**	no
`tex-latex-tabular`	**C-c C-l C-t**	yes
`tex-latex-verbatim`	**C-c C-l v**	no

The command **C-c C-l LFD** (`tex-latex-skip`) can be used to start a new LaTeX environment outside the current nesting level. The idea is that since each of the environment commands listed above puts out `\end{...}` on the line below the cursor position, if a new environment is to be started at the same level, the skip command avoids the discrete motions needed to place the cursor to the right position. This is best explained by the following example:

```
\begin{enumerate}
  ...
  \item {\LaTeX} Environments LFD
  \item Customizing Delimiters C-c C-l LFD
\end{enumerate}
█
```

where █ denotes the current cursor position (point). Note that **LFD** is bound to `tex-newline-indent`, so the one at the end of line opens a new line and indents properly. The command **C-c C-l LFD** goes to the end of the innermost `\end{...}` and does a **LFD** there. As mentioned in Section 2.3, if the `Auto Fill` mode is on, the wrapped line will be indented with current indentation, as is done by **LFD**.

4.1.5 Customizing Delimiters

It is possible to customize *TEX-mode* so that other delimiters will work in a way consistent with the ones described above. Both semi-automatic matching (i.e. zone/word matching) and automatic matching delimiters as well as the delimiters for LaTeX environments are user-definable. Such customizations can either be set up statically in `tex-mode-hook` or be invoked interactively inside *TEX-mode*. The former makes new delimiter pairs available in every Emacs *TEX-mode* session whereas the latter is good only for that particular run of Emacs.

The command **C-c C-\ s** (`tex-make-semi`) defines a new pair of semi-automatic matching delimiters. The system will prompt you, in sequence, for the following four attributes: *l-sym*, *r-sym*, *name*, and *letter*. The first two attributes *l-sym* and *r-sym* are strings for opening and closing delimiters respectively. The next attribute *name* is a string for which you would like the pair be called. The last attribute *letter* is a string of a single letter which will be incorporated in a set of key bindings consistent with the default zone/word matching commands.

As a result of the command, one zone matching function `tex-zone-`*name*, and two word matching functions `tex-word-`*name*, and `tex-word-forward-`*name* will be generated and respectively be bound to the commands **C-c ESC-***letter*, **C-c** *letter*, and **C-c-4** *letter*. The user will be asked to confirm overwritting if any of these newly created functions or key bindings are bound already.

The list `tex-delimiters-semi` can be set in your `tex-mode-hook` to declare new delimiters permanently. It must get bound to a list whose elements are each a list of the four attributes mentioned above. A warning message will be issued if the number of attributes in any component list is incorrect.

Suppose you find yourself always using `\em` and `\large` in your LaTeX documents. You could say

```
(setq tex-delimiters-semi '(("{\\em " "\\/}" "em" "e")
                            ("{\\large " "}" "large" "l")))
```

in your hook and have six new commands permanently added to the arsenal of zone/word matching in *TEX-mode*. If you just want this for one Emacs session, invoke `tex-make-semi` interactively instead. Note that the backslash is escaped (i.e. \\) in the quoted string in order to produce a single backslash in the Emacs buffer. But if such attributes are specified interactively where the strings you type are implicitly quoted, a single backslash will suffice.

Similarly, the command **C-c C-\ a** (`tex-make-auto`) defines a new pair of automatic matching delimiters. The required attributes for this command are a *delimiter* and its *name* where *delimiter* is a string of one symbol whose name is *name*. Two functions `tex-`*name* and `tex-toggle-`*name* will be generated and be bound to the *delimiter* itself and the command **C-c C-t** *delimiter* respectively.

For example, in this document the typewriter font is heavily used and I have a TEX macro that creates a verbatim environment under `\tt` font. The environment is opened by a `|` and terminated by a second `|`. Automatic matching of delimiters like `|`'s in this case is very desirable, so I put

```
(setq tex-delimiters-auto '(("|" "bar")))
```

in my `tex-mode-hook`. And I can always turn off its automatic matching by using the command **C-c C-t** `|` (`tex-toggle-bar`) if I wanted to.

Customizing LATEX environment delimiters works under the same paradigm. The command **C-c C-\ e** (`tex-make-env`) can be used to define a new pair of LATEX environment delimiters interactively. The required attributes are *name*, *letter*, and *argp*, where *name* is the name of a LATEX environment, *letter* is the letter to be incorporated to the key binding, and *argp* is either true (by answering '`y`' or **SPC**) if the environment takes any arguments or false ('`n`' or **DEL**) if otherwise. The function `tex-latex-`*name* will be generated and be bound to **C-c C-l** *letter*.

The variable `tex-latex-envs` can be bound to a list in which each element is a list of the three attributes mentioned. Suppose you want to add the `minipage` environment to *TEX-mode*. In `tex-mode-hook` you can put

```
(setq tex-latex-envs '(("minipage" "m" t)))
```

to generate the function `tex-latex-minipage` and the command **C-c C-lm**.

4.2 Spelling Checking

The simplest form of spelling checking is word lookup. *TEX-mode* allows you to lookup words in the dictionary by typing the command **C-c C-s w** (`tex-spell-word`). It first prompts you with:

```
Lookup string as prefix, infix, or suffix?  (RET/p, i, s)
```

Upon receiving your answer, it then prompts you for the string to be searched for. The searching is case-sensitive. Answering **RET** or '`p`' to the first prompt means any word containing the string specified at the second prompt as a prefix will be a match, answering '`i`' has a similar meaning for the case of infix, and '`s`' for suffix. All matching words are displayed in the other window bound to a buffer called `--- TeX Dictionary ---`.

In addition to word lookup, *TEX-mode* has a general spelling checker that works on larger pieces of text. The command **C-c C-s r** (`tex-spell-region`) checks spelling for a prespecified region, **C-c C-s b** (`tex-spell-buffer`) does it for the current buffer and finally **C-c C-s d** (`tex-spell-document`) works for the entire document which may involve multiple files. The mechanism first filters out all TEX, AMS-TEX, or LATEX keywords and symbols using `tex-spell-filter` which may be set as either `tex-detex` or `tex-delatex` depending on the document type (the default value for `tex-detex` is "`/usr/local/detex`" and that for `tex-latex` is "`/usr/local/delatex`"). A plain TEX or AMS-TEX document will be filtered by the former while a LATEX or SLITEX document will be screened by the latter. The same procedure for verifying document type, as discussed earlier in Section 3.2, will be the first thing the spelling mechanism does. In the case of spelling checking for the entire document, all files included as children of the current file will be checked in a depth-first order. User confirmation is required before each file is checked, which avoids checking files such as an often used macro package that are know to be correct.

Suppose the misspelled word '`foo`' is found as a result of **C-c C-s b**. The first message will be:

```
Erroneous 'foo' (SPC, DEL, n, p, r, R, w, C-r, ?=help)
```

Below is a brief description of each of the commands:

- **SPC** — Ignore current erroneous word and advance to the next one, if any.
- **DEL** — Ignore current erroneous word and go back to the previous one, if any.
- '`n`' — Go to next instance of the word in buffer, wrap around if necessary.
- '`p`' — Go to previous instance of the word in buffer, wrap around if necessary.
- '`r`' — Replace all instances of the word below point. A repetition of current erroneous word appears at replacement prompt.
- '`R`' — Replace all instances of the word below point. If a word is specified in `--- TeX Dictionary ---`, it gets copied at prompt; otherwise nothing is repeated at replacement prompt.
- '`w`' — Dictionary lookup for words containing the specified substring. Results will be displayed in other window bound to a buffer called `--- TeX Dictionary ---`.
- `C-r` — Enter recursive edit. Return to spelling checking by **ESC C-c**.
- '`?`' — This help message. Displayed in a buffer called `--- TeX Spelling Help ---` in the other window.

The point starts at the beginning of buffer for each word currently active. The first two commands (**SPC** and **DEL**) scroll the list of erroneous words down and up respectively. If you try to go beyond either end, you will be asked to confirm exit. The next set of commands ('`n`' and '`p`') searches forward and backward for the next and previous instance of the current erroneous word. The first '`n`' positions you to the first instance of the word in buffer or region and the first '`p`' brings you to the last. It wraps around if the search fails at either end.

The two replacement commands ('`r`' and '`R`') both trigger query replacing from current point position which may be aborted by `C-c`. The only difference between the two is that the first option ('`r`') always repeats the current erroneous word at replacement prompt while the second ('`R`') in general does not. If the replacement is complete, the word gets deleted from the list and the next erroneous word becomes active. If, however, it aborts before completion, the point stays at last instance in buffer or region and the current word remains active.

The next command '`w`' invokes word lookup mentioned above. The matched words are displayed in a buffer called `--- TeX Dictionary ---` in the other window. Typing the command '`C-r`' enters *recursive edit* mode which makes it possible to scroll the dictionary window if not all words are visible. The command **ESC C-c** brings you back to spelling checking mode. A semi-automatic replacement facility is available here. First type '`w`' for dictionary lookup, followed by `C-r` which puts the cursor in the dictionary window, you then position point to the word you are looking for and type **ESC C-c** at that point. The word in `--- TeX Dictionary ---` will be selected and embraced in a pair of brackets, the cursor will jump back to the source window, and the spelling prompt returns to the state right before recursive edit was invoked. If you then enter the command '`R`', the selected word will be copied to the replacement prompt and the dictionary window will be deleted. Finally, '`?`' gives you the help message in the other window.

Searching under low speed connections ($\leq$ 2400 baud) is implemented for scrolling and replacing. That is, if an instance is not visible in the current window, only the line containing it is shown in a tiny window at the bottom to avoid the screen redisplay overhead.

4.3 Bibliography Making

BIBTEX is a system designed jointly by Leslie Lamport, Howard Trickey, and Oren Patashnik (implementation is due to Patashnik) as a bibliography preprocessor for LATEX documents [4,5]. It supports four bibliography styles and fourteen standard entry types (compatible with Scribe, but not with Unix *refer*). Customizations in both styles and types are possible. A BIBTEX database is a file with the name suffix '`.bib`' which contains one or more bibliography entries. In a LATEX document one makes citations with the command `\cite{foo}` or `\nocite{foo}` where `foo` is an entry name that appears in one of the bibliography files specified. The LATEX command `\bibliography{f1,f2,...,fn}` informs BIBTEX to lookup entries in files `f1.bib`, `f2.bib`, ..., `fn.bib` from the default load path `TEXBIB`. Moreover, the command `\bibliographystyle{...}` tells BIBTEX what style to use in representing the actual references where the ellipse could be one of `plain`, `unsrt`, `alpha`, and `abbrv`, the four standard styles, or any user-defined style known to the system.

To get the final output, we first run `latex` on the document `foo.tex` to produce a `foo.aux` file of reference information. We then run `bibtex` on `foo` to generate the bibliography file `foo.bbl`. The next `latex` looks up `foo.bbl` and incorporates the information into the `.aux` file. Finally the third run of `latex` gets the references correct.

TEX-mode has a bibliography interface that does entry lookups as well as citation substitutions for you. It bypasses the first `latex` and invokes `bibtex` directly. It works with multiple files involved in a document by recursively examining `\input` commands in plain TEX and AMS-TEX files and `\includeonly`, `\include`, and `\input` commands in LATEX/SLITEX files. Furthermore, it prompts the user for corrections at the places where citation errors are found. The implications are:

1. Due to the lookup facility the user does not have to memorize or type in the exact entry names for citations.
2. To get the final output, the user only has to invoke one or two `latex`'s manually — depending on non-citation symbolic references being present. This is due to the automatic invocation of `bibtex`, the error correcting facility, and the automatic substitution mechanism.
3. The same mechanism not only works for LATEX documents which BIBTEX was specifically designed for but for plain TEX and AMS-TEX documents as well. In fact, it will generally work on any TEX dialects.

The rest of the section discusses the features in some detail. Unless otherwise stated, it refers to both TEX and LATEX documents.

4.3.1 Lookup Facility

There are two ways to make a citation: either by the command **C-c C-b c** (`tex-bib-cite`) or the command **C-c C-b n** (`tex-bib-nocite`). Both `\cite` and `\nocite` will generate entries in the bibliography source. But unlike `\cite`, `\nocite` will not be replaced by actual references. Nonetheless, in terms of entry lookup, the two commands are essentially the same. The first prompt you see is

```
.bib filename (loadpath TEXBIB):
```

If a `.bib` has been specified previously, it will appear at the prompt as default. Suppose all your `.bib` files reside in `$HOME/biblib`. With your `TEXBIB` bound to "`.:$HOME/biblib`", you can answer `doc.bib`, if it is a file in the path (or simply answer `doc`, the system will attach the `.bib` extension at the end automatically). The next prompt asks for the lookup key:

```
Lookup string as a regexp (default browsing mode):
```

If there is no specific key to search for, typing **RET** enters the default browsing mode in which each entry in the file is considered a match. In either case, you get a message such as the following in the minibuffer:

```
Confirm {knuth:tex}?  (RET/y, SPC/n, DEL/p, s, f, k, C-r, C-c, ?=help)
```

where `{knuth:tex}` is an entry name in `doc.bib`. The meaning of the commands can be summarized as follows:

- **RET** or '`y`' — Confirm and exit. The string "`knuth:tex`" will either be inserted together with a `\cite` (or `\nocite`) command before point, or it will be merged with the existing citation which immediately precedes point. The `.bib` file which contains this entry (`doc.bib` in this example) will be merged to the `\bibliography` list at the bottom of the document's master file (such a list will be created if not found).
- **SPC** or '`n`' — Ignore current entry. Advance to the next entry which contains a match.
- **DEL** or '`p`' — Ignore current entry. Go back to the previous entry which contains a match.
- '`s`' — Show the actual entry in other window, if not already shown.
- '`f`' — Give up current `.bib` file. Prompt me for an alternate file name.
- '`k`' — Give up current search key. Prompt me for an alternate string.
- `C-r` — Enter recursive edit. Return to bibliography lookup by the command **ESC C-c**.
- `C-c` — Quit and abort to previous level, if there is one; otherwise abort to top level (equivalent to `C-g`).
- '`?`' — This help message. Displayed in a buffer called `--- TeX Bibliography Lookup Help ---` in the other window.

It is worth pointing out that you can make successive citations and have selected entries all merged in one list. For instance, suppose three **C-c C-b c**'s are made in a row with respective confirmed entries being `e1`, `e2`, and `e3`, the final citation will be `~\cite{e1,e2,e3}` rather than `~\cite{e1}~\cite{e2}~\cite{e3}`. This hack produces an actual reference list of `[1,2,3]` instead of the list `[1][2][3]`.

The command `C-r` invokes recursive edit which allows you to adjust point position before the final confirmation is made. You can also take this opportunity to modify the `.bib` file if some errors were discovered during the lookup session. The command **ESC C-c** returns the confirmation prompt to the one right before recursive edit was invoked.

Note that for each successful citation entry lookup (i.e. answering **RET** or '`y`'), the corresponding `.bib` file name will be merged to the per document `\bibliography` argument list. This list is inserted at the bottom of the document's master file rather than in any component files.

Rain Check. Notice that the very first lookup will take longer as the system spends some time loading BIBTEX-mode, a GNU Emacs interface to editing BIBTEX databases. The next version of *TEX-mode* will interact more closely with *BIBTEX-mode* [1] to provide a more powerful database query capability. It will be able to retrieve entries from multiple `.bib` files in `TEXBIB` which match a specified author name, title, journal, year, ..., etc., or their combinations.

4.3.2 Bibliography Preprocessing

In TEX and LATEX a document may span multiple files connected together by `\input` or `\include` commands. One of these files is the root of the document tree (the master file) with others being its children. Typing **C-c C-b d** (`tex-bib-document`) initiates the making of the document's bibliography and **C-c C-b b** does it only for the current buffer. Suppose this bibliography preprocessing initiates from file `foo.tex` and files `foo1.tex`, `foo2.tex`, ..., are also involved, the process consists of the following steps.

In step 1, it tries to recover all actual references back to symbolic citations based on the file `foo.ref`. This file was created in the previous run of bibliography making and is supposed to contain cross reference information necessary for the recovery. If such a file is non-existent, *TEX-mode* assumes it is the first run of bibliography making and no recovery effort will be attempted.

Step 2 collects all entries cited in files `foo.tex`, `foo1.tex`, `foo2.tex`, ..., etc. and creates the file `foo.aux` which will be used by BIBTEX in step 3. The user will be prompted to specify the `.bib` files and the bibliography style if it fails to find them. A first stage of error detection is performed in this step. If a citation contains blank space, which is by definition a BIBTEX error, an automatic error correcting mechanism will be triggered.

Suppose you have an erroneous citation `\cite{knuth tex}` in file `foo`, the first error correction message reads:

```
{knuth tex} in "foo" contains illegal white space...
```

followed by a list of possible actions:

```
Correcting {knuth tex} (SPC, DEL, r, l, c, d, ?=help)
```

The meaning of each of the options follows.

- **SPC** — Advance to next citation error, if any.
- **DEL** — Go back to previous citation error, if any.
- '`r`' — Replace current error by manually typing in a new string.
- '`l`' — Replace current error by invoking the lookup facility mentioned earlier.
- '`c`' — Comment it out.
- '`d`' — Delete it.
- '`?`' — This help message. Displayed in a buffer called `--- TeX Citation Correction Help ---` in the other window.

For the two scrolling commands (**SPC** and **DEL**), you will be asked to confirm exit or wrap around if they touch either end of the list. The '`c`' command will comment the erroneous entry out in one of the following ways: (1) If the error is not the only entry in the citation, it gets deleted from the list and is put in a new but commented-out citation in the line below. (2) If it is the only entry in the list, the citation itself gets commented out and a '`[?]`' will be inserted in its place automatically.

Step 3 passes "`bibtex foo`" to the inferior shell process and executes it there. As soon as the job starts in the shell window, the flow of control falls into the user's hands. The prompt

```
Continue act/sym substitution?  [Wait till finish if 'y'] (y or n)
```

remains in the minibuffer until a command '`y`' or '`n`' (**SPC** or **DEL**) is given. As the message warns: you must wait till the job is finished if your answer is '`y`'. You can type '`n`' any time to abort the BIBTEX job, however. In either case, if BIBTEX finds any errors in the `.bib` files or in your document, you will be asked to correct them.

There are two types of errors. The first type of errors are those that occur in the various `.bib` files. Correcting errors of this type is somewhat manual. If you answer '`y`' (or **RET**) to the prompt

```
Errors detected in .bib files, correct them?  (y or n)
```

it starts a new session of recursive edit. Each **C-c C-@** (`tex-goto-error`) locates the next error in a `.bib` file and displays the corresponding error message in the other window. At this point the command is actually invoking `bibtex-goto-error`, which is defined in *BIBTEX-mode* [1] but imported to *TEX-mode* by autoloading. The commands `bibtex-goto-error` and `tex-goto-error` have the same interface to the inferior shell window and the same key binding. Hence if there are multiple errors in more than one `.bib` file, it is not necessary to go back to the original `.tex` buffer to locate the next error. Instead, typing **C-c C-@** in any `.bib` will suffice. The last **C-c C-@** terminates recursive edit and resumes bibliography making. An early return is possible by typing **ESC C-c**.

If there are still errors of the second type, the next stage of error correcting, as discussed below, is triggered; otherwise it goes back and starts over again from Step 3. Errors of the second type are citation errors that occur in the document. If your answer to the prompt

```
Citation errors detected, correct them?  (y or n)
```

is positive, the automatic error correcting mechanism described in Step 2 is invoked. Note that recursive edit is not provided here for efficiency considerations. The mechanism does replacement by keeping track of a bunch of pointers to the various errors in the various files rather than by string matching. If recursive edit were allowed, it is likely that the user will mess up the pointer structure in any files by arbitrary insertions and deletions.

Once the error correction is completed, it goes back and repeats Step 2. If BIBTEX detects no errors and the user decides to proceed, it goes on to the next step.

Step 4 modifies `foo.bbl` generated by BIBTEX if it is to be used in a TEX document. It will do nothing if it is for LATEX. The string "`\input foo.bbl`" will be inserted at the end of `foo.tex` before `\end` or `\bye`, if it isn't there already. The user will have to manually insert a title for this bibliography (or reference) section since the file `foo.bbl` does not have any title. This step also prepares the cross reference information of symbolic and actual citations based on `foo.bbl` and saves it in the file `foo.ref`. The user should not modify the file `foo.ref` for the obvious reason of future recovery.

The last step does symbolic to actual substitution based on the information in `foo.bbl`. When done, all symbolic citations such as `\cite{knuth:tex}` and `\cite{lamport:latex,patashnik:bibtex}` will be replaced by actual references such as `[3]` or `[Kn84]` and `[4,5]` or `[La86,Pa85]`, or any other forms defined by a particular bibliography style. All `\nocite`'s, on the other hand, will simply be commented out.

Two more commands are related to the bibliography making. First, to recover symbolic references from their actual counterparts, use the command **C-c C-b r** (`tex-bib-recover`). Second, the command **C-c C-b s** (`tex-bib-save`) can be used to save all files modified by bibliography making. You will be prompted for each file involved. Answering '`!`' (or **RET**) will automatically save remaining files without further confirmation. Files which are part of the document but contain no citation entries will not be affected by this command.

4.4 Indexing

LATEX supports a very simple form of indexing. We have borrowed the indexing macros defined in `latex.tex` so that the same set of commands can be used in TEX and $\mathcal{AMS}$-TEX documents. In LATEX if the command `\makeindex` is given in the preamble, running `latex` will pick up the argument of each `\index` command and convert it to a `\indexentry` command concatenated with the current page number in a `.idx` file. For instance, the entry `\index{gnu}` on page 1 of file `foo.tex` becomes

```
\indexentry{gnu}{1}
```

in file `foo.idx`. Finally the user must process this `.idx` file manually to produce the actual index which contains the `theindex` environment, such as:

```
\begin{theindex}
  ...
  \item gnu 1
  ...
\end{theindex}
```

Note that the index entry has been transformed to a `\item` field in the actual index file.

4.4.1 Macro Package for Indexing: `idxmac.tex`

To provide TEX and $\mathcal{AMS}$-TEX documents with the same facility, the macro package `idxmac.tex` can be loaded. It contains the indexing commands excerpted from `latex.tex` plus some extensions. That is, in a TEX or $\mathcal{AMS}$-TEX document if the command `\makeindex` is given, the argument of `\index` and its page number will be picked up by `tex` or `amstex` and inserted as arguments of `\indexentry` in a `.idx` file.

The meaning of `\index` and the extensions supported by `idxmac.tex` are the following:

`\index{foo}` on page N	$\longrightarrow$	`\indexentry{foo}{`N`}`
`\indexbf{foo}` on page N	$\longrightarrow$	`\indexentry{foo}{{\bf `N`}}`
`\indexit{foo}` on page N	$\longrightarrow$	`\indexentry{foo}{{\it `N`\/}}`
`\indexsl{foo}` on page N	$\longrightarrow$	`\indexentry{foo}{{\sl `N`\/}}`
`\indexul{foo}` on page N	$\longrightarrow$	`\indexentry{foo}{\ul{`N`}}`

where the left-column commands can be used in a `.tex` source file and will be converted to the right-column entries in a `.idx` file. The macro `\ul` is for underlines. It has the following definition in `idxmac.tex`

```
\def\ul#1{$\underline{\rm #1}$}
```

The set of these `\index` variants makes it possible to have page numbers typeset in different fonts, thereby conveying more information about whatever is being indexed. For instance, one can declare that all page numbers in boldface represent the places of formal definitions and those in italics are instructive examples, etc. (see Appendix I of [3]). To use these extended commands in LATEX, load the file `idxmac.tex` before the `\makeindex` command in the preamble.

Other indexing commands like `\item`, `\subitem`, `\subsubitem`, and `\indexspace` which appear in the actual index file for formatting purposes are also defined in `idxmac.tex`. Also in it are the commands `\beginindex` and `\endindex` for TEX and $\mathcal{AMS}$-TEX in place of the LATEX `theindex` environment. However, in our current version they are only skeletons. The user has to determine his own style for the index file (such as title, heading, double column, ..., etc.) by modifying `\beginindex`, `\endindex` and related output routines (cf. pp. 261–264 of [3]).

4.4.2 Index Processor: `makeindex`

The process of transforming the `.idx` file generated by the formatter to the actual index file can be facilitated using the program `makeindex`†. This program takes at least one argument, namely the file name base of the `.idx` file, and produces a `.ind` file, the actual index. For instance, the entries

```
\indexbf{beta} on page 8
\index{{\tt alpha}} on page i
\index{{\tt alpha}} on page iv
\index{{\tt alpha}} on page 10
\index{beta !  gamma} on page 2
\index{beta!!!gamma} on page 3
\index{!!beta!gamma} on page 4
\index{beta!gamma!      delta} on page 3
\indexit{beta} on page 1
\index{beta} on page 20
\index{epsilon\!!{\it zeta\/}!eta} on page 30
```

† based on a much simpler version written by Mike Urban (`trwrb!trwspp!spp3!urban@berkeley.edu`) posted on the `unix-tex` mailing list (11/26/1985), which was based on an even simpler shell script posted earlier on the same list (11/20/1985) by Marshall Rose (`mrose%NRTC@usc-ecl.arpa`).

in file `foo.tex` get converted to

```
\indexentry{beta}{{\bf 8}}
\indexentry{{\tt alpha}}{i}
\indexentry{{\tt alpha}}{iv}
\indexentry{{\tt alpha}}{10}
\indexentry{beta !  gamma}{2}
\indexentry{beta!!!gamma}{3}
\indexentry{!!beta!gamma}{4}
\indexentry{beta!gamma!      delta}{3}
\indexentry{beta}{{\it 1\/}}
\indexentry{beta}{20}
\indexentry{epsilon\!!{\it zeta\/}!eta}{30}
```

in file `foo.idx` by the formatter. The program `makeindex` can then be used to further transform this file to the actual index file, such as `foo-.ind`. By default the document is assumed to be in LaTeX and the file `foo-.ind` will contain:

```
\begin{theindex}\makeatletter
  \item {\tt alpha}  i, iv, 10
  \indexspace
  \item beta  {\it 1\/}, {\bf 8}, 20
    \subitem gamma  2--4
      \subsubitem delta  3
  \indexspace
  \item epsilon!
    \subitem {\it zeta\/}
      \subsubitem eta  30
\end{theindex}
```

which indicates a number of interesting tasks accomplished by `makeindex`:

- A common prefix of up to two levels is recognized and is transformed from `\indexentry` to `\subitem` or `\subsubitem` properly.
- The exclamation point '`!`' is the prefix delimiter. Only the first two effective `!`'s (defined below) are regarded as prefix delimiters.
- The delimiter can be surrounded by any number of **SPC**'s or **TAB**'s, as shown above. Leading `!`'s are ignored (i.e. "`!!beta!gamma`" means "`beta!gamma`") and consecutive `!`'s are treated as a single delimiter (i.e. "`beta!!!gamma`" is actually "`beta!gamma`"). If '`!`' is needed as a symbol in the prefix, it can be escaped by a backslash, as in the "`epsilon\!`" case. Note that in the final `.idx` file, the `\` does not show up. Any `!`'s in the subsubitem, even unescaped, are treated as the symbol '`!`' itself.
- Entries in the index file are alphabetized according to their keywords. Thus `{\tt alpha}` is sorted using `alpha` as key, rather than anything else.
- An extra vertical space (`\indexspace`) is inserted before the first entry that starts a new letter.
- Page numbers are merged and sorted in one list if their respective keys are identical. The font information for page numbers are preserved. Both Arabic and Roman numerals are sorted. Roman numerals come in front of Arabics. Three or more pages in consecutive Arabic numbers are abbreviated as a range (e.g. `2--4` above). Consecutive Roman numerals are not abbreviated (e.g. `ii, iii, iv`), however.
- Even if there is no entry as its proper prefix, an entry like `\index{epsilon\!!zeta!eta}` is still converted to the correct levels of nesting in terms of `\item`, `\subitem`, and `\subsubitem`. In this case only the innermost level gets the page number.

The example above shows a somewhat pathological case where the `beta` entry on page 1 appears after the same entry on page 8. This will not happen very often in real situations because `\index` and its variants on lower page numbers will always be processed first. Also, a lot of redundant `!`'s, **SPC**'s, and **TAB**'s are introduced in the example. Again, this should not happen in the normal case. Our purpose here is to demonstrate the generality of `makeindex`.

Perhaps it's also time to say a few words about the prefix delimiter '`!`'. We use the exclamation point instead of the more obvious '`,`' because the comma may appear in the index key as a meaningful character. For instance, it is common to index people's names with their last names first and then the first names, separated by a comma. The exclamation mark, on the other hand, is less likely to be used in such a context.

4.4.2.1 Syntax

The actual argument syntax of `makeindex` is as follows:

makeindex `[-b` *base*`]` `[-o` *fn*`]` `[-d` *style* `]` `[-p` *page*`]` `[-[ta][es][yn]]` f_1 f_2 $\cdots$

where f_1, f_2, $\cdots$, are names of the `.idx` files, or their base names, if the extension is `.idx`.

4.4.2.2 Master File Name Base: `-b` *base*

This option specifies *base* as the document master's file name base, or its name itself, if the extension is not `.tex`. If this `-b` option is not given, the file name base of f_1 is used as default.

4.4.2.3 Output File Name: `-o` *fn*

This option declares *fn* to be the name of the output index file (i.e. the `.ind` file). If this `-o` option is not given, the file `foo-.ind` is used as output, where `foo` is the master file name base.

4.4.2.4 Document Style: `-d` *style*

This option specifies *style* as the document style. The style information makes sense only if `-s` is set (separate formatting). It is used as the argument to `\documentstyle` if the type is LaTeX (default). In such a case, the brackets and braces must be given along with the style info, as in

```
-d \[12pt\]\{book\}
```

Note that the brackets and braces must be escaped. If the document type is TeX or AMS-TeX, *style* is regarded as the name of the index style macro package, which will be used as the argument of `\input` in the `.ind` file.

If the `-d` option is not given but the `-s` option is set, in the LaTeX case the style information is retrieved from `foo.tex`, where `foo` is the master file name base. If `foo.tex` is not found, or somehow the style information is missing in `foo.tex`, the '`report`' style is assumed. In the TeX or AMS-TeX case, if `-d` is not given but `-s` is set, the default indexing package "`idxmac.tex`" is used.

4.4.2.5 Starting Page Number: `-p` *page*

This option specifies *page* as the starting page number for separate formatting (`-s` option). If this `-p` option is not set, the starting page number is determined by reading the last page number recorded in `foo.log` plus 1, where `foo` is the master file name base. If `foo.log` is not found, or somehow the page number is not found in the `.log` file, the field is left empty.

4.4.2.6 Document Type : `-t` or `-a`

By default, the document type is LaTeX. If the document type is TeX or AMS-TeX, give `makeindex` the '`-t`' or '`-a`' flag. With either flag set, the index file `foo-.ind` (or that specified by `-o`) will have:

```
\beginindex
  \item {\tt alpha} i, iv, 10
  \indexspace
  \item beta {\it 1\/}, {\bf 8}, 20
    \subitem gamma 2--4
      \subsubitem delta 3
  \indexspace
  \item epsilon!
    \subitem {\it zeta\/}
      \subsubitem eta  30
\endindex
```

instead of the `\begin{theindex}...\end{theindex}` the the LaTeX case.

4.4.2.7 Formatting Mode: `-e` or `-s`

The program `makeindex` also takes another set of flags which is orthogonal to the document type flag. This set of flags has to do with postprocessing of the `.ind` file and perhaps the source `.tex` file itself for formatting purposes. There are three possibilities:

1. `-e` *option*. The `.ind` file is included in the `.tex` source (i.e. by inserting `\input foo-.ind` at the bottom of `foo.tex`) and the program attempts to format the index file together with the entire document. The user will be prompted to decide whether to run the formatter (determined by the document type flag), and to restore the original source or not. In the LaTeX case, the difference between the original and the modified `.tex` source is that the latter has `\nofiles` in the preamble and `\input foo-.ind` right before `\end{document}`. In the TeX or AMS-TeX case, the only difference is the `\input foo-.ind` at the end of the file above the closing command `bye` or `end`. In either case, the user will be asked to confirm executing the formatter unless `-y` or `-n` is set.

2. `-s` *option*. In the LaTeX case, it gets the `\documentstyle` info from the `-d` option or the `.tex` file. The starting page number comes from the `-p` option or the `.log` file. It then patches the `.ind` file up with additional commands such as `\setcounter{page}{}` and `\begin{document}...end{document}` so that it becomes stand-alone and can be separately formatted. In the TeX or AMS-TeX case, the index macro package is either specified by the `-d` option or simply the default `idxmac.tex` and the starting page number is determined the same way as in the LaTeX case. In either case, the `.tex` source is intact. If the `-p` option is not set, or the `.log` file is not found, or it fails to find a page number in the `.log` file, the user must specify the page number before formatting, unless the `-n` option is set. If everything is successful, the user will be asked whether to run the formatter on `foo-.ind` or not (again, unless `-y` or `-n` is set.)

3. *default*. No additional information is added to the `.ind` file after the conversion from `.idx`. The `.tex` source file is intact.

The `-s` option deserves more attention here. Suppose the LaTeX file `foo.tex` has a `book` style at 12 points and `foo.log` shows that the last page formatted is 100, with the command

```
makeindex -s foo
```

`foo-.ind` will be:

```
\documentstyle[12pt]{book}
```

```
\setcounter{page}{101}
\begin{document}
\begin{theindex}
  ...
\end{theindex}
\end{document}
```

Similarly, for TeX, the command

```
makeindex -ts foo
```

or for $\mathcal{AMS}$-TeX,

```
makeindex -as foo
```

produces

```
\input idxmac
\pageno=101
\beginindex
  ...
\endindex
\bye
```

The advantage of the `-e` option is that there will be only one `.dvi` file. The disadvantage is that reformatting the document body, which must have been done at this point, is a somewhat unnecessary overhead. The `-s` option seems to be a remedy in this regard. But in this arrangement, the document is made of two `.dvi` files instead of one. It is now clear why the default index file name is called `foo-.ind` instead of simply `foo.ind`. In the separate formatting case (`-s`), the **DVI** output of `foo.ind` will clobber the output of `foo.tex` whereas that of `foo-.ind` will be in `foo-.dvi`. This is due to the internal file name processing of TeX which we can do nothing about. Finally the vanilla default option does not touch the `.tex` file, nor does it add any global formatting information to the `.ind` file. The idea is that in most situations the `.ind` file may need some editing and fine tuning before it is ready for formatting.

4.4.2.8 Noninteractive Mode: `-y` or `-n`

The `-y` or `-n` option disables the formatting confirmation prompt. If the `-y` flag is on and either `-e` or `-s` is set, the index file will be formatted with no questions asked. The `-n` option, on the other hand, will not format the file, which means if `-e` or `-s` is set, the files involved are returned as in the `-y` case, but no formatting will be invoked. This allows the user to edit the index file before producing the final output.

4.4.3 Indexing Facility in TeX-mode

The purpose of the *TeX-mode* indexing commands is to somewhat automate the index making process. There are a number of ways to create index entries in *TeX-mode*. The command **C-c C-i w** (`tex-index-word`) copies the previous word `foo` and inserts `\index{foo}` before point. A positive prefix argument N does it for the previous N words. The command **C-c C-i r** (`tex-index-region`) expects a region as its implicit argument. The text between mark and point will be copied and inserted along with `\index{}` at the right end of the region.

4.4.3.1 Indexing Mode

By default, `\index` is used and no index prefix is assumed, as indicated by the initial `nil` value of the two *TEX-mode* flags `tex-index-variant-on` and `tex-index-prefix-on`. These flags, along with a third called `tex-index-keyptrn-on`, constitute the *mode* of indexing as a three-bit binary code:

    variant-prefix-keyptrn.

By default the code is `000`, which means none of the flags are set. To change the default, all you need to do is set the coresponding flags `t` in your `tex-mode-hook`. Also available in *TEX-mode* is the ability to toggle these flags. The commands **C-c C-i v** (`tex-index-variant-toggle`), **C-c C-i p** (`tex-index-prefix-toggle`), and **C-c C-i k** (`tex-index-keyptrn-toggle`) toggle the three flags, respectively. Alternatively, the command **C-c C-i c** (`tex-index-chmod`) can be used to set the three at the same time by giving a 3-bit code to

    Change mode (variant-prefix-keyptrn, 3-bit binary): █

where a non-zero digit sets the corresponding flag; 0 resets it.

Variant Selection

If `tex-index-variant-on` is true, the following prompt appears when an index command is issued:

    Index variant: RET=default, b=boldface, i=italic, s=slanted, u=underline

Answering **RET** inserts the good old `\index{...}` at point while answering '`b`', '`i`', '`s`', and '`u`' respectively inserts `\indexbf{...}`, `\indexit{...}`, `\indexsl{...}`, and `\indexul{...}` at point.

Prefix Specification

If `tex-index-prefix-on` is true, the user will be prompted to enter the index prefix, as in

    Index prefix (max 2 levels, ! as delimiter, RET if none):

Simply answer **RET** if there is none. A prefix string of level 2 must contain a separating '`!`'. For instance, suppose the entry to be included in `\index{}` is `gamma`, and if we answer `beta` at the prefix prompt, in the text the actual entry becomes

    \index{beta!gamma}.

Answering `beta!` produces the same result. Now suppose the text selected is `delta`. For a prefix of level 2, one should answer something like `beta!gamma` in order to get

    \index{beta!gamma!delta}

inserted. This time the '`!`' between `beta` and `gamma` is essential.

Auto Key-Pattern Saving

If `tex-index-keyptrn-on` is true, the following prompt appears in the minibuffer after an index command has been inserted:

    Save index regexp: █

which asks you for a search pattern as a regular expression. Given a non-empty answer, it then continues and prompts you for an auxiliary file name in which the `[key, pattern]` tuple is to be saved. If a file has been specified before, it is listed as default. The tuple is lexicographically merged into the list of existing tuples in the file. If the same key already exists in the file with a different pattern, you will be asked to confirm overwritting. If overwritting is declined, the tuple is inserted anyway.

A related command **C-c C-i s** (`tex-index-save`) unconditionally saves a `[key, pattern]` tuple in the specified auxiliary file (i.e. regardless if `tex-index-keyptrn-on` is set or not).

The central idea here is to save these `[key, pattern]` tuples in an auxiliary file so that the same pattens in the remaining text can be indexed automatically with the corresponding keys. This auxiliary file can be incrementally maintained and can be processed directly by *TEX-mode* (see Section 4.4.3.2 below).

4.4.3.2 Higher-level Indexing

To enter index entries in the source file more systematically, **C-c C-ib** (`tex-index-buffer`) is provided by *TEX-mode* which prompts for the actual key to be indexed:

```
Index key: █
```

It then asks for a search pattern as a regular expression, taking the specified key as default:

```
Regexp: goo█
```

where `goo` is the key specified at the first prompt. For each instance of the pattern found in the buffer, with `goo` as the given index key, the following menu will appear in the minibuffer:

```
Insert \index{goo}? (SPC/y, DEL/n, LFD/p, ?  for more options)
```

and the user can answer one option and proceeds to the next or previous instance of the pattern. The pattern appearing in the argument list of `\index` (or its variants) is not considered a match because it does not make sense to index an index key. If the user attempts to pass either end of the buffer where no more next or previous instance is found, he will be asked to confirm exit. If no instances of the pattern are found to begin with, it aborts prematurely.

A similar command **C-c C-id** (`tex-index-document`) does the same with every file included in the document, including the bibliography file `foo.bbl` referenced by the `\bibliography` command in a LaTeX document. Using the same interface as in `tex-spell-document`, for each file in the document, the user will be asked to confirm visiting that file. The advantage of this is to give the user the convenience of bypassing component files unlikely to be indexed, such as macro packages. Given `goo` as the index key, when file `foo.tex` is to be visited a typical confirmation prompt looks like

```
Ok to index 'goo' in "foo.tex"? (SPC/y, DEL/n, RET/!)
```

where the first two options are self-explanatory, and the third option means answering **RET** or `!` will visit the current and all remaining files without requiring any further confirmations. As in the buffer case, any attempt to pass either end of the current file will trigger an exit confirmation prompt. If the user answers positively, the next file in the document will be processed, otherwise the searching wraps around. If no instances of the pattern are found in the current file, it is bypassed automatically.

With prefix argument, the two commands can be used to process more than one `[key, pattern]` tuple listed in a file. The command **C-u C-c C-ib** takes a file `foo.key` and goes through each tuple in it and attempts to index the pattern in current buffer. Similarly **C-u C-c C-id** does it for the entire document. An typical entry in a key file looks like the following

```
"key"
"\\bkey\\b\\|key idea"
```

which is a tuple of a key and a regular expression pattern. Unlike the `regexp` specified interactively, which requires only a single backslash to escape the empty symbol (`\b`), the OR sign (`|`), etc. the one in `foo.key` is represented as a Lisp string object and must use double backslashes. The `.key` file can be prepared manually, or automatically using **C-c C-is** discussed above, or by turning on the flag `tex-index-keyptrn-on` before any insertions are issued. The single backslash you give at prompt will be converted to double backslashes in `foo.key` by *TEX-mode* automatically.

4.4.3.3 The Menu

When an instance of the search pattern is found in the current buffer, a menu of the three most commonly used options, plus the question mark is displayed. When '`?`' is typed, a complete menu, together with the meaning of each option available is displayed in buffer `--- TeX Indexing Help ---`. The following is a copy of the menu:

- **SPC** or '`y`' — Confirm index insertion and advance to next instance, if any.
- **DEL** or '`n`' — Ignore current instance and advance to the next, if any.
- **LFD** or '`p`' — Ignore current instance and advance to the previous, if any.
- **RET** or '`m`' — Confirm index insertion with a possible mode change. Advance to the next instance, if any.
- '`M`' — Global mode change.
- '`k`' — Change index key for the current instance.
- **C-k** — Change index key for all remaining instances in buffer.
- '`K`' — Change index key for all remaining instances in document.
- '`@`' — Quietly insert index entries in the remaining buffer.
- '`!`' — Quietly insert index entries in the remaining document.
- **ESC** — Quit working with current key on current buffer. Try next file in document, if any.
- **C-c** — Quit working with current key on current document. Try next key in the key file, if any.
- **C-r** — Enter recursive edit. Return by ESC C-c.
- '`?`' — Help message. Displayed in a buffer called `--- TeX Indexing Help ---` in the other window.

Note that the indexing mode has a global scope over these operations, unless a request for *quiet insert* is made ('`@`' or '`!`'), in which case the mode becomes `000` (i.e. no questions asked). Indexing mode can also be changed for a single instance by typing **RET** or '`m`', or permanently altered using '`M`', which is equivalent to `tex-index-chmod`.

There are two ways to abort. The first is **ESC** which quits working with the current key in current buffer. If there is a `.key` file involved, the next tuple in that file is processed. Similarly, the **C-c** option aborts the current key for the rest of the document, including the remainder of current buffer. Again, if the command is invoked with **C-u** prefix, the process restarts with the next tuple in the `.key` file.

4.4.3.4 Indexing Author Names

TEX-mode has a facility to index every author name in a `.bbl` bibliography file. The command **C-c C-i a** (`tex-index-authors`) positions the cursor after each author name of a `\bibitem` entry in the specified file (default `foo.bbl`, where `foo` is the master file name base or the base of current file name), and requests a confirmation on the name.

A typical `\bibitem` entry in the `.bbl` file looks like the following:

```
\bibitem{knuth:tex}{
Donald~E. Knuth.
\newblock {\it The {\TeX} Book}.
\newblock Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.}
```

TEX-mode tries to process author names starting from the line next to `\bibitem`. However, there may be entries with no authors at all. *TEX-mode* has no way of telling whether there is an author name in the line or not. What it does is to position the cursor at the line in question, prompting with

```
Ok to process current line?  (SPC/y, DEL/n, RET/!)
```

If the current line is the list of authors, answering **SPC** or '`y`' triggers the name processing facility. If the current line is just the title of a work without any specific authors, you should answer **DEL** or '`n`' instead. In that case, the next `\bibitem` is processed. Since the convention is that anonymous papers or books are listed first, once an author name appears in an entry, the rest will be the same. The third option (**RET** or '`!`') tells *TEX-mode* to process author names in the current and remaining entries without any further confirmations.

The name processing facility can locate each author name in the current entry precisely. For each author name located, a prompt like

```
Author name:  Knuth, Donald~E.█
```

will be given, where the name is rearranged so that last name comes in front of the first and other parts of the name, separated by a comma. If the name is correct, typing **RET** will insert `\index{Knuth, Donald~E.}` after the name. So the text in the `.bbl` file becomes

```
\bibitem{knuth:tex}{
Donald~E. Knuth.\index{ Knuth, Donald~E.}
\newblock {\it The {\TeX} Book}.
\newblock Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.}
```

In most cases, the prompted author name will be correct. But it uses an ad hoc heuristic to determine the last name: only the last "word" in the name field is taken as one's last name. Thus a name like "Michael Van De Vanter" will be prompted as

```
Author name:  Vanter, Michael Van De█
```

which is, of course, wrong. In this case, it is the user's responsibility to correct the name so that it becomes

```
Author name:  Van De Vanter, Michael
```

before confirming it.

Special words such as "`and`" and "`et al.`" which are common in the author name list will be ignored. If there is already a `\index{...}` entry after the name, the newly confirmed overwrites the old. Finally the notion of our indexing mode is carried over to the insertion of `\index{...}`.

4.4.3.5 Interface to `makeindex`

The command **C-c C-i m** (`tex-index-make`) is an interface to the index processor `makeindex` discussed in Section 4.4.2. Remember that the `.idx` file must be created by the formatter before `makeindex` can do anything useful. If the `.idx` file is missing, you will be asked to run the formatter first. If the `.idx` file exists, the user will be prompted with

```
Index formatting option:  e=entire document, s=separate index, else=nothing
```

which means answering '`e`' or '`s`' corresponds to the '`-e`' or '`-s`' option of `makeindex`, respectively, while typing any other key means the default option. The command **C-c C-i m** gets the document type information by the standard way discussed in Section 3.2 and attaches the appropriate flags (i.e. '`-t`', '`-a`', or nothing) to `makeindex`. The command

`makeindex` <*flags*> `foo`

is then sent to inferior shell process (bound to buffer `*shell*`) and executed. There may be further interactivity originated from `makeindex` and it will take place in the shell buffer.

4.5 Format-Debug-Preview-Print

A number of TEX related programs can be invoked from *TEX-mode*. These external programs are executed uniformly in Emacs' inferior shell process. The generic operators are *format*, *display*, *view*, and *print*. The first two are overloaded based on the document type. The *display* operator is a pipeline of formatting followed by previewing (i.e. `texdvi`, `amstexdvi`, `latexdvi`, or `slitexdvi`). The *view* operator is bound to a previewer such as `dvitool`. The other two operators are self-explanatory. There are also commands specifically designed to aid debugging, such as locating errors and commenting out regions. This section describes all these commands in detail.

4.5.1 Format

Typing **C-c C-fd** (`tex-format-document`) pops to the master buffer of the current document and formats the whole document using the program `tex-formatter`. This program can be either `tex`, `amstex`, `latex`, or `slitex`, depending on the document type. It is executed under the inferior shell process in the other window. The user will be asked to save the file if the buffer has been modified since last time.

Separate Formatting

A component buffer (file) or a region within a buffer may be separately formatted using the command **C-c C-fb** (`tex-format-buffer`) or **C-c C-fr** (`tex-format-region`). Suppose the command is issued from buffer `noo.tex` whose master is `foo.tex`, in the region case (**C-c C-fr**) the text will be copied into a buffer called `noo#.tex` and the document pre- and post-amble files (`foo+.tex` and `foo-.tex`), if any, will be included properly. If these files are not found, the default setup discussed in Section 3.3 will be inserted. The buffer case (**C-c C-fb**) is almost the same except that the current file is included in `noo#.tex` by a `\input` command instead of being copied to between the pre- and post-ambles.

The file `noo#.tex` will have a master of itself and will be formatted as a stand-alone document. The master pointer and document type information will not necessarily be recorded in this temporary buffer (depending on whether the selected region contains this header information or not). But *TEX-mode* knows what it is implicitly anyhow. When this temporary file is being formatted its buffer will be displayed side by side with the original file (i.e. window will have a horizontal split).

As a special case, if a file has a master pointer to itself, the command **C-c C-fb** will be equivalent to **C-c C-fd** and the file will be formatted as itself instead of through the temporary `#.tex` file. Yet another special case happens in the separate formatting of inidivdual slides in SLITEX. Since it is a common practice to put each slide in an individual file, *TEX-mode* coerces any **C-c C-fr** to **C-c C-fb** implicitly. That is, no matter what region is selected for formatting in the current buffer, it always formats the entire buffer. The underlying rationale for this is that a slide in SLITEX is dependent upon the `slide` environment, which presumably is available in each file (slide). Allowing arbitrary regions to be formatted requires the knowledge of whether the `slide` environment is included or not. Because the overall text in a slide is relatively short, the overhead does not justify the effort.

4.5.2 Print and View

To print the DVI file generated by the previous formatting program, try **C-c C-p SPC** (`tex-print-all`). It asks you to specify a printer name (default `tex-printer-default`) then goes off and executes `tex-hardcopy` (default "`lpr -d`"). If you have a DVI previewer, use the command **C-c C-v SPC** (`tex-view-all`) to view your output on your workstation. The variable `tex-softcopy` (default "`dvitool -E`") can be redefined in your hook if yours is different.

Separate Printing/Viewing

A DVI file can be printed and viewed in its entirety. *TEX-mode* can also invoke the program `dviselect`‡ so that arbitrary pages within a DVI file may be extracted and only these selected pages will be previewed

‡ written by Chris Torrek of the University of Maryland. Available through Unix TEX distribution.

or printed. The commands **C-c C-p DEL** (`tex-print-partial`) and **C-c C-v DEL** (`tex-view-partial`) both prompt the user for the pages to be selected. Suppose the command is issued in the buffer bound to file `foo.tex`, as mentioned in Section 3.1 the implicit operand is `foo.dvi`. The selected pages will be put in a temporary file called `foo%.dvi` and passed to the program `tex-hardcopy` or `tex-softcopy`. This is another useful tool for avoiding unnecessary work in a batch oriented environment like TEX, in addition to saving paper expenditure.

4.5.3 Display

The command **C-c C-d d** (`tex-display-document`) invokes the program bound to `tex-displayer` which may be either `texdvi`, `amstexdvi`, `latexdvi`, or `slitexdvi`. As mentioned earlier in Section 2, the four programs all invoke `dvitool` automatically after finishing the formatting job. Hence the net effect of this command is functionally equivalent to **C-c C-f d** followed by, when it's finished, **C-c C-v SPC**, if `tex-softcopy` is bound to `dvitool`.

Similar to the separate formatting case, the commands **C-c C-d b** (`tex-display-buffer`) and **C-c C-d r** (`tex-display-region`) both copy the text into a temporary buffer and execute `tex-displayer` there.

4.5.4 Executing Other Programs

TEX-mode has an automatic bibliography making facility (see Section 4.3), therefore you need not call BIBTEX explicitly. However, if you really want to do so, you can use **C-c C-e** (`tex-execute`) and enter the string "`bibtex`" to its prompt. In fact, any program can be invoked inside *TEX-mode*: simply by entering the program name and its associated switches at **C-c C-e**'s first prompt and the file name at the second prompt.

4.5.5 Other Facilities for Debugging

TEX-mode has an error positioning mechanism that greatly facilitates the debugging cycle. Each error which appears on the shell window during the formatting is recorded. If you issue the command **C-c C-@** (`tex-goto-error`) in the document's source buffer when the formatter is finished or has come to a pause state (e.g. at the TEX prompt '?'), *TEX-mode* will locate the error by positioning the cursor to the line and column in the file where it occurs. Meanwhile the shell window will recenter itself to show you the error message as much as possible. The next **C-c C-@** brings you to the next error, and then the next, and so on. This mechanism works even if the current formatting session is not started from *TEX-mode* using **C-c C-f** *letter*.

Notice that a formatting job can be started either automatically by *TEX-mode* or manually by the user himself in the shell buffer. The error positioning mechanism works in both cases. If the job is initiated by *TEX-mode*, the command **C-c C-@** can also be issued from the shell buffer, thus saves the user from having to pop back to a TEX buffer. If, however, the job is manually started, the user has to go back to a source buffer to issue the first error positioning command. But this is required only for the very first time; once **C-c C-@** is invoked, the command itself will be known to the shell key map.

Because external files can be included in a TEX-based document, a **C-c C-@** issued in a buffer bound to file A may end up in one bound to file B. If there are no more errors, the last **C-c C-@** will move the cursor to the shell window where the formatting job was left off so that you can continue from that point. Or you can just take care of a bug and type **C-c C-f d** to restart formatting without having to worry about the state of the previous job. In fact, the old job gets killed in the shell process before the new job is started. The same command can also be used to locate errors in BIBTEX databases, as was discussed earlier in Section 4.3.

Notice that on most keyboards **C-SPC** also generates the **C-@** signal. It is much easier to type **C-SPC** than to type **C-@** because the latter in general involves an extra shift key.

Warning. This error positioning mechanism is a kludge. Instead of getting the information from TEX,† it retrieves the file name which contains the current error by performing pattern matching on the error messages generated by the formatter in the shell buffer. This is fine for most cases, but there is no guarantee it would always work. So if you encounter any pattern that screws up **C-c C-@**, please save that particular shell buffer snapshot and send a bug report to the author.

Another function that might be useful to debugging is the one that comments out an area of text and recovers it later. The command **C-c C-c** (`tex-comment-region`) works on a preset (ordinary) region and inserts a TEX comment sign (`%`) in front of each line in the region. Conversely, the command **C-c C-u** (`tex-uncomment-region`) deletes the leading `%`, if any, from each line of a region. More precisely speaking, a region here is implicitly coerced to a rectangle because comments are line-based in TEX. You can set the mark (**C-@**) anywhere in a line to start the region and do the same with the two commands to get all lines in between affected. For instance,

```
This is the C-@ first line.
% This is the second line and is already commented out.
This is the third line. C-c C-c
```

produces

```
%This is the first line.
%% This is the second line and is already commented out.
%This is the third line.
```

and **C-c C-u** deletes the leading `%` from each line in the same region. With a positive prefix argument N, the two commands inserts or deletes that many `%`'s for each line in the region. That is, the command **C-u 2 C-c C-c** inserts `%%` in front of every line. Conversely, **C-u 2 C-c C-u** erases them. Any prefix arguments less than 1 are converted to 1 by default.

4.5.6 Design Decision

Some people may wonder why *TEX-mode* executes external programs as subjobs of Emacs' inferior shell process instead of starting a dedicated process for each of them. There are pros and cons for either approach. A major disadvantage of running everything in the inferior shell process is that the very same shell is subject to user commands totally out of *TEX-mode*'s control. For instance, the user may change the working directory arbitrarily in the shell. To ensure that output files (`.dvi` in particular) generated by a formatting job reside in the same directory where the source file (`.tex`) is, for each **C-c C-f** *TEX-mode* has to verify the shell's working directory and `cd` to the right one if necessary. That is what's going on behind the scenes when you see a message like "`Sending 'tex foo' to shell...`".

But there are at least two reasons that support our design decision. First, with the shell's buffer, a history of external program invocations is maintained which may or may not be useful in some occasions. The second reason is more important. The inferior shell process offers a standard set of control commands such as `interrupt-shell-subjob` and `stop-shell-subjob`. It is simply convenient to adopt these protocols as a standard, not to mention the redundancy and overhead of reimplementing them for each dedicated process.

† If you know how to query TEX to get the current file name, please let me know.

5 Final Remarks

In summary, *TEX-mode* commands and their key bindings obey the following convention:

Command	*Meaning*
C-c ESC-*letter*	zone matching
C-c *letter*	word matching (backward)
C-c-4 *letter*	word matching (forward)
C-c C-b *letter*	bibliography making
C-c C-i *letter*	indexing
C-c C-l *letter*	LATEX environments
C-c C-m *letter*	customizing delimiters
C-c C-s *letter*	spelling checking
C-c C-t *letter*	toggling matching
C-c C-*letter*	miscellaneous global operations

While in *TEX-mode*, the command **C-c C-h** (`tex-mode-help`) displays in the other window a table of the commands available. A version of this document will be translated to TEXInfo [7], the official GNU Emacs documentation format. When that is available you will be able to run the `info` system in Emacs to consult this manual interactively.

Acknowledgements. I would like to thank the following people for making constructive suggestions on *TEX-mode* at various stages of the development: Mike Harrison, Art Werschulz, Rusty Wright, Paul Rubin, Fred Douglis, and Jim Larus.

6 References

[1] Peehong Chen. *GNU Emacs BIBTEX-mode.* Technical Report, Computer Science Division, University of California, Berkeley, California. To appear.

[2] Peehong Chen, Michael A. Harrison, John Coker, Jeffrey W. McCarrell and Steve Procter, "An improved user environment for TEX", in *Proc. of the 2nd European Conference on TEX for Scientific Documentation*, Strasbourg, France, June 19–21, 1986. To be published by Springer-Verlag.

[3] Donald E. Knuth. *The TEX Book.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1984.

[4] Leslie Lamport. *LaTEX: A Document Preparation System. User's Guide and Reference Manual.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

[5] Oren Patashnik. *BIBTEXing.* Computer Science Department, Stanford University, Stanford, California, March 1985.

[6] Richard M. Stallman. *GNU Emacs Manual*, 4th Edition, Version 17, Free Software Foundation, Cambridge, Massachusetts, February 1986.

[7] Richard M. Stallman. *TEXInfo, The GNU Documentation Format.* first edition, Free Software Foundation, Cambridge, Massachusetts, June 1985.

[8] Michael D. Spivak. *The Joy of TEX*, American Mathematical Society, Providence, RI., 1985.

7 Summary

7.1 Installation and Startup (cf. Chapters 1, 2, and 5)

`tex-mode.el` **file**

A *TEX-mode* file which defines basic attributes and key bindings for *TEX-mode*. Also included is the code for doing automatic matching of dollar signs and double quotes and some supporting functions shared by other subsystems. To begin with only this file is loaded.

`tex-match.el` **file**

A *TEX-mode* file which defines several delimiter mattching schemes. This file is autoloaded whenever a function defined in it is invoked.

`tex-misc.el` **file**

A *TEX-mode* file which defines a number of interfacing facilities to external programs such as formatting, previewing, displaying (formatting plus previewing), printing, etc. It also contains some debugging aids and help functions for document and file processing. This file is autoloaded whenever a function defined in it is invoked.

`tex-spell.el` **file**

A *TEX-mode* file which defines an interactive spelling checker. This file is autoloaded whenever a function defined in it is invoked.

`tex-bib.el` **file**

A *TEX-mode* file which defines an interface to BIBTEX in terms of citation entry lookups as well as a general-purpose bibliography processing facility. This file is autoloaded whenever a function defined in it is invoked.

`tex-index.el` **file**

A *TEX-mode* file which defines an automatic indexing mechanism. It also contains an interface to the index processor **`makeindex`**. This file is autoloaded whenever a function defined in it is invoked.

`tex-init.el` **file**

A file which may be created locally to redefine site-specific attributes. This file is loaded whenever the function `tex-mode` is invoked.

`tex-mode` **major mode function**

Major mode for editing TEX-based documents.

`tex-mode-version` **function**

Return the current *TEX-mode* version.

`tex-mode-help` **C-c C-h**

Display a summary of *TEX-mode* commands in the other window.

`tex-mode-hook` **variable**

Variable to be bound to `(function (lambda () <body>))` where `<body>` is a sequence of statements having to do with abbreviations, redefinition of key bindings, non-default settings of *TEX-mode* variables, loading of other functions, etc.

`abbrev-mode` **minor mode function**

An Emacs minor mode which enables the expansion of abbreviated text. By default this mode is turned off in *TEX-mode*. Invoking this function with a positive integer turns the mode on.

`tex-abbrev-enable` **C-c C-a SPC**

Unconditionally enables the `Abbrev` minor mode.

`tex-abbrev-disable` **C-c C-a DEL**

Unconditionally disables the `Abbrev` minor mode.

`auto-fill-mode` **minor mode function**

An Emacs minor mode which enables auto line wrapping when a space is typed beyond column `fill-column`. In *TEX-mode* `fill-column` is set to 78 but this mode is turned off by default. Invoking this function with a positive integer turns the mode on.

`tex-autofill-enable` **C-c LFD SPC**

Unconditionally enables the `Auto Fill` minor mode.

`tex-autofill-disable` **C-c LFD DEL**

Unconditionally disables the `Auto Fill` minor mode.

7.2 Basic Abstractions (cf. Chapter 3)

`tex-check-master-file` **C-c 0**

Check or change the current master file pointer. A reconfirmation message will be given if the entered name does not correspond to any existing file.

`tex-check-document-type` **C-c 1**

Check or change the document type. Only the four types `TeX`, `AmSTeX`, `LaTeX`, and `SliTeX` are currently supported. The entered string will be matched against these built-in types. The matching is case-insensitive. The action is aborted if the type entered is unknown.

`tex-make-preamble` **C-c C-\ SPC**

Save region in the document preamble file. Suppose the current buffer is bound to file `foo.tex`, then the text between mark and point is replaced by the command `input foo+` and is yanked in file `foo+.tex` (what's was original in that file gets overwritten).

`tex-make-postamble` **C-c C-\ DEL**

Save region in the document postamble file. Suppose the current buffer is bound to file `foo.tex`, then the text between mark and point is replaced by the command `input foo-` and is yanked in file `foo-.tex` (what's was original in that file gets overwritten).

7.3 Delimiter Matching (cf. Section 4.1)

`tex-boundary-check-on` **variable**

Boundary checking in delimiter matching will be disabled if this flag is `nil`. Default value is `t`.

`tex-toggle-boundary-check` **C-c C-t ESC**

Toggle the boundary checking mechanism in delimiter matching.

`tex-bounce-backward` **C-c (**

Bounce backward to check the opening delimiter.

`tex-bounce-forward` **C-c)**

Bounce forward to check the closing delimiter.

7.3.1 Zone Matching (cf. Section 4.1.1)

`tex-zone-open` **C-c SPC**

Open a TEX zone.

`tex-zone-close` **C-c DEL**

Close a TEX zone explicitly. The topmost zone marker will be popped and no delimiters will be inserted.

`tex-zone-inspect` **C-c C-z**

Inspect the marker position of TEX zone. With positive prefix argument N, inspect the N^{th} marker in the stack. A non-positive prefix argument is converted to 1 implicitly.

`tex-zone-math` **C-c ESC-$**

Embrace the innermost TEX zone with a pair of $'s. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-display-math` **C-c ESC-d**

Embrace the innermost TEX zone with a pair of $$'s. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-single-quote` **C-c ESC-'**

Embrace the innermost TEX zone with a left single quote (`) and a right single quote ('). The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-double-quote` **C-c ESC-"**

Embrace the innermost TEX zone with left double quote (``) and right double quote (''). The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-centerline` **C-c ESC-c**

Embrace the innermost TEX zone by `\centerline{...}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-hbox` **C-c ESC-h**

Embrace the innermost TEX zone by `\hbox{...}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-vbox` **C-c ESC-v**

Embrace the innermost TEX zone by `\vbox{...}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-bf` **C-c ESC-b**

Embrace the innermost TEX zone by `{\bf ...}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-it` **C-c ESC-i**

Embrace the innermost TEX zone by `{\it ...\/}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-rm` **C-c ESC-r**

Embrace the innermost TEX zone by `{\rm ...}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-sl` **C-c ESC-s**

Embrace the innermost TEX zone by `{\sl ...\/}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-zone-tt` **C-c ESC-t**

Embrace the innermost TEX zone by `{\tt ...}` with `...` being the text between zone marker and point. The zone marker is popped from the stack. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

7.3.2 Word Matching (cf. Section 4.1.2)

`tex-word-math` **C-c $**

Embrace the previous word with a pair of $'s. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-math` **C-c-4 $**

Embrace the next word with a pair of $'s. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-display-math` **C-c d**

Embrace the previous word with a pair of $$'s. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-display-math` **C-c-4 d**

Embrace the next word with a pair of $$'s. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-single-quote` **C-c '**

Embrace the previous word with a pair of left and right single quotes (`` `...' ``). With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-single-quote` **C-c-4 '**

Embrace the next word with a pair of left and right single quotes (`` `...' ``). With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-double-quote` **C-c "**

Embrace the previous word with a pair of left and right double quotes (` ``...'' `). With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-double-quote` **C-c-4 "**

Embrace the next word with a pair of left and right double quotes (` ``...'' `). With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-centerline` **C-c c**

Embrace the previous word by `\centerline{...}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-centerline` **C-c-4 c**

Embrace the next word by `\centerline{...}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-hbox` **C-c h**

Embrace the previous word by `\hbox{...}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-hbox` **C-c-4 h**

Embrace the next word by `\hbox{...}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-vbox` **C-c v**

Embrace the previous word by `\vbox{...}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-vbox` **C-c-4 v**

Embrace the next word by `\vbox{...}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-bf` **C-c b**

Embrace the previous word by `{\bf ...}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-bf` **C-c-4 b**

Embrace the next word by `{\bf ...}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-it` **C-c i**

Embrace the previous word by `{\it ...\/}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-it` **C-c-4 i**

Embrace the next word by `{\it ...\/}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-rm` **C-c r**

Embrace the previous word by `{\rm ...}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-rm` **C-c-4 r**

Embrace the next word by `{\rm ...}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-sl` **C-c s**

Embrace the previous word by `{\sl ...\/}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-sl` **C-c-4 s**

Embrace the next word by `{\sl ...\/}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-tt` **C-c t**

Embrace the previous word by `{\tt ...}` with `...` being the word. With positive prefix argument N, embrace previous N words; or with negative prefix argument N, embrace next N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

`tex-word-forward-tt` **C-c-4 t**

Embrace the next word by `{\tt ...}` with `...` being the word. With positive prefix argument N, embrace next N words; or with negative prefix argument N, embrace previous N words. Position confirmation is required if either end touches any non-blank symbol unless the boundary checking mechanism is disabled.

7.3.3 Automatic Matching (cf. Section 4.1.3)

`tex-toggle-dollar` **ESC-$**

Toggle automatic matching of $'s and $$'s.

`tex-match-dollar-on` **variable**

A flag if set `nil` disables $ matching. Default value is `t`.

`tex-toggle-quote` **ESC-"**

Toggle automatic matching of double quotes.

`tex-match-quote-on` **variable**

A flag if set `nil` disables " matching. Default value is `t`.

7.3.4 LaTeX Environments (cf. Section 4.1.4)

`tex-latex-open` **C-c C-l SPC**

Open a LaTeX environment. *TEX-mode* will prompt you for the environment name (e.g. `env`) and its associated arguments (e.g. `[foo]`). Type **RET** to either prompt if none. The string `\bgein{env}[foo]` will be inserted before point and the cursor will be positioned in a new line below with an indentation of `tex-latex-indentation` relative the `\begin`.

`tex-latex-close` **C-c C-l DEL**

Close a LaTeX environment. The string `\end{...}` will be inserted where `...` is the name of the matching environment. The cursor is positioned in a new line below.

`tex-latex-skip` **C-c C-l LFD**

Skip the next line (presumably `\end{...}`) and open a new line with a proper indentation below it.

`tex-newline-indent` **LFD**

Open a new line below the current line with an indentation of its current indentation.

`tex-latex-indentation` **variable**

Indentation under current LaTeX environment. Default value 2.

`tex-latex-array` **C-c C-l a**

Invoke the delimiters of LaTeX environment `array` with its arguments to be specified interactively. The cursor is positioned in an empty line between `\begin{array}` and `\end{array}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. Negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-center` **C-c C-l c**

Invoke the delimiters of LaTeX environment `center`. The cursor is positioned in an empty line between `\begin{center}` and `\end{center}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-enumerate` **C-c C-l c**

Invoke the delimiters of LaTeX environment `enumerate`. The cursor is positioned in an empty line between `\begin{enumerate}` and `\end{enumerate}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-figure` **C-c C-l c**

Invoke the delimiters of LaTeX environment `figure`. The cursor is positioned in an empty line between `\begin{figure}` and `\end{figure}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-itemize` **C-c C-l c**

Invoke the delimiters of LaTeX environment `itemize`. The cursor is positioned in an empty line between `\begin{itemize}` and `\end{itemize}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-picture` **C-c C-l c**

Invoke the delimiters of LaTeX environment `picture`. The cursor is positioned in an empty line between `\begin{picture}` and `\end{picture}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-quote` **C-c C-l c**

Invoke the delimiters of LaTeX environment `quote`. The cursor is positioned in an empty line between `\begin{quote}` and `\end{quote}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-tabbing` **C-c C-l TAB**

Invoke the delimiters of LaTeX environment `tabbing` with its arguments to be specified interactively. The cursor is positioned in an empty line between `\begin{tabbing}` and `\end{tabbing}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. Negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-table` **C-c C-l c**

Invoke the delimiters of LaTeX environment `table`. The cursor is positioned in an empty line between `\begin{table}` and `\end{table}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-tabular` **C-c C-l C-t**

Invoke the delimiters of LaTeX environment `tabular` with its arguments to be specified interactively. The cursor is positioned in an empty line between `\begin{tabular}` and `\end{tabular}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. Negative prefix argument is converted to the value of `tex-latex-indentation`.

`tex-latex-verbatim` **C-c C-l c**

Invoke the delimiters of LaTeX environment `verbatim`. The cursor is positioned in an empty line between `\begin{verbatim}` and `\end{verbatim}` with an indentation of `tex-latex-indentation` relative to `\begin`. With non-negative prefix argument N, indent N columns relative to `\begin` instead. A negative prefix argument is converted to the value of `tex-latex-indentation`.

7.3.5 Customizing Delimiters (cf. Section 4.1.5)

`tex-delimiters-auto` **variable**

A list that may be set in `tex-mode-hook` to declare new automatic matching delimiters permanently. It must get bound to a list whose components are each a list of two elements, a single letter string and its name.

`tex-make-auto` **C-c C-\ a**

Define a new pair of automatic matching delimiters with attributes *delimiter* and *name* where *delimiter* is a string of one symbol whose name is *name*. This function can be invoked from `tex-mode-hook` or called interactively in a TEX/LATEX editing session. Two functions `tex-`*name* and `tex-toggle-`*name* will be generated and be bound to the *delimiter* itself and the command **C-c C-t** *delimiter* respectively.

`tex-delimiters-semi` **variable**

A list that may be set in `tex-mode-hook` to declare new semi-automatic matching delimiters permanently. It must get bound to a list whose components are each a list of four elements: opening delimiter string, closing delimiter string, name, and a single letter string.

`tex-make-semi` **C-c C-\ s**

Define a new pair of semi-automatic matching delimiters with attributes *l-sym*, *r-sym*, *name*, and *letter*. The first two attributes *l-sym* and *r-sym* are strings for opening and closing delimiters, respectively. The third attribute *name* is a string for which you would like the pair be called. The forth attribute *letter* is a string of a single letter which will be incorporated in a set of key bindings consistent with the default zone/word matching commands. The function can be used in `tex-mode-hook` or invoked interactively in a TEX editing session. The functions `tex-zone-`*name*, `tex-word-`*name*, and `tex-word-forward-`*name* will be generated and be bound to **C-c ESC-***letter*, **C-c** *letter*, and **C-c-4** *letter*, respectively.

`tex-latex-envs` **variable**

A list that may be set in `tex-mode-hook` to declare new LATEX environment delimiters. It must be bound to a list whose components are each a list of three elements: name, letter, and either `t` or `nil`.

`tex-make-env` **C-c C-\ e**

Define a new pair of LATEX environment delimiters with attributes *name*, *letter*, and *argp*. The first attribute *name* is the name of a LATEX environment. The second attribute *letter* is a string of a single letter which will be incorporated in the key binding. The last attribute *argp* is `t` if the environment takes arguments, `nil` if not. This function can be invoked from `tex-mode-hook` or called interactively in a LATEX editing session. The function `tex-latex-`*name* will be generated and be bound to the the command **C-c C-l** *delimiter*.

7.4 Spelling Checking (cf. Section 4.2)

`tex-spell` **variable**

Name of the program that checks the spelling. Default is "`/usr/bin/spell`".

`tex-dict-words` **variable**

Name of the file upon which `tex-word-spell` is based. Default is "`/usr/dict/words`".

`tex-detex` **variable**

Name of the program that filters out commands and keywords from TEX documents. Default is "`/usr/local/detex`".

`tex-delatex` **variable**

Name of the program that filters out commands and keywords from LATEX documents. Default is "`/usr/local/delatex`".

`tex-spell-document` **C-c C-s d**

Check spelling for entire document. Prompt the user for document type (TEX, AMS-TEX, LATEX, or SLITEX) if unknown. Depending on its type, filter the document with `tex-detex` or `tex-delatex` before running `tex-spell`. Do this for every file involved in this document in a depth-first order.

`tex-spell-buffer` **C-c C-s b**

Check spelling for current buffer. Prompt the user for document type (TEX, AMS-TEX, LATEX, or SLITEX) if unknown. Depending on its type, filter the document with `tex-detex` or `tex-delatex` before running `tex-spell`.

`tex-spell-region` **C-c C-s r**

Check spelling between mark and point. Prompt the user for document type (TEX, AMS-TEX, LATEX, or SLITEX) if unknown. Filter document with `tex-detex` or `tex-delatex` (depending on document type) before running `tex-spell`.

`tex-spell-word` **C-c C-s w**

Lookup and display all words which contain the specified key as a substring. The key may be considered as either a prefix, infix, or suffix of the matching words. The searching is case-sensitive.

7.5 Bibliography Making (cf. Section 4.3)

`tex-bib-cite` **C-c C-b c**

Lookup bibliography entries from BIBTEX databases. Prompt the user for the target `.bib` file name and the string to be searched for. Browse each entry if search string is null. Insert the confirmed entry as a citation before point. Merge with previous `\cite` list if it immediately precedes point.

`tex-bib-nocite` **C-c C-b n**

Lookup bibliography entries from BIBTEX databases. Prompt the user for the target `.bib` file name and the string to be searched for. Browse each entry if search string is null. Insert the confirmed entry as a pseudo citation before point. Merge with previous `\nocite` list if it immediately precedes point.

`tex-bib-document` **C-c C-b d**

Make a bibliography for the entire document. Pop to the document master if not already in it. First recover symbolic references from actual ones in all files. Then invoke BIBTEX and do error recovery if any errors were found. Create the bibliography (reference) file and interpolate it at the end of current buffer. Finally substitute actual references for the symbolic ones.

`tex-bib-buffer` **C-c C-b b**

Make a bibliography for the file bound to current buffer First recover symbolic references from actual ones in all files. Then invoke BIBTEX and do error recovery if any errors were found. Create the bibliography (reference) file and interpolate it at the end of current buffer. Finally substitute actual references for the symbolic ones.

`tex-bib-recover` **C-c C-b r**

Recover symbolic citations from actual references in all files associated with current file.

`tex-bib-save` **C-c C-b s**

Save all files associated with current file interactively. Answering '`!`' (or **RET**) automatically saves the remaining files without further confirmation.

7.6 Indexing (cf. Section 4.4)

`idxmac.tex` **macro package in TEX**

A collection of indexing macros excerpted from `latex.tex`, plus some extensions. In can be loaded in a LaTEX document to have index page numbers typeset in different fonts. It can also be loaded in a TEX or AMS-TEX document so that the same indexing facility available in LaTEX can be used for the other two types of documents (i.e. the `.idx` file can be automatically generated and the `.ind` file can be properly formatted.)

`makeindex` **external program**

The program which transforms the `.idx` file produced by the formatter to the actual index, the `.ind` file. Default document type is LaTEX. The '`-t`' or '`-a`' flag declares the document to be TEX or AMS-TEX, respectively. The postprocessing option for formatting purpose is either '`-e`' (for formatting the entire document), '`-s`' (for separately formatting the `.idx` file only), or nothing (for creating only the `.ind` file for editing and tuning). Check Section 4.4.3.3 for more options.

`tex-index-make` **C-c C-i m**

Make the actual index file by calling the program `makeindex`. Prompt the user for formatting option as is required by `makeindex`. Give `makeindex` the document type info automatically.

`tex-index-variant-on` **variable**

The flag that enables the variant selection query. Default is `nil`. Can be toggled by the function `tex-index-variant-toggle`. Index variants currently supported are `\index` (default), `\indexbf`, `\indexit`, `\indexsl`, and `\indexul`.

`tex-index-variant-toggle` **C-c C-i v**

Toggle the flag `tex-index-variant-on`.

`tex-index-prefix-on` **variable**

The flag that enables the prefix specification query. Default is `nil`. Can be toggled by the function `tex-index-prefix-toggle`. Up to two levels of prefix is allowed. Prefix delimiter is '`!`'.

`tex-index-prefix-toggle` **C-c C-i p**

Toggle the flag `tex-index-prefix-on`.

`tex-index-keyptrn-on` **variable**

The flag that enables the saving of a `[key, pattern]` tuple in a file. Default is `nil`. Can be toggled by the function `tex-index-keyptrn-toggle`.

`tex-index-keyptrn-toggle` **C-c C-i k**

Toggle the flag `tex-index-keyptrn-on`.

`tex-index-chmod` **C-c C-i c**

Change the current indexing mode (`variant-prefix-keyptrn`). Specify the mode in a 3-bit binary code.

`tex-index-save` **C-c C-i s**

Save a `[key, pattern]` tuple in a file.

`tex-index-word` **C-c C-i w**

Insert `\index{}` before point and copy the previous word in the braces. With positive prefix argument N, copy the previous N words. Select a variant of `\index` if the corresponding flag is on. Furthermore, if the prefix flag is on, enter the index prefix (up to two levels, separated by '`!`', so that they can be transformed to `\subitem`'s or `\subsubitem`'s in the actual index file) at prompt; answer **RET** if there is none.

`tex-index-region` **C-c C-i r**

Insert `\index{}` at the right end of the current region and copy the text in the region in the braces. Select a variant of `\index` if the corresponding flag is on. Furthermore, if the prefix flag is on, enter the index prefix (up to two levels, separated by '`!`', so that they can be transformed to `\subitem`'s or `\subsubitem`'s in the actual index file) at prompt; answer **RET** if there is none.

`tex-index-buffer` **C-c C-i b**

Insert `\index{KEY}` after each instance of `REGEXP` in current buffer where the strings `KEY` and `REGEXP` are to be specified at their respective prompts. With prefix argument **C-u**, process each tuple of `[key, pattern]` in a specified key file.

`tex-index-document` **C-c C-i d**

Insert `\index{KEY}` after each instance of `REGEXP` in every file included in current document where the strings `KEY` and `REGEXP` are to be specified at their respective prompts. For each file included, the user will be asked to confirm visiting, making it possible to bypass files like macro packages which are unlikely to be indexed. With prefix argument **C-u**, process each tuple of `[key, pattern]` in a specified key file.

`tex-index-authors` **C-c C-i a**

Process each author name and insert `\index{AUTHOR}` in the specified `.bbl` bibliography file. Prompt each name appearing in the `\bibitem` entry (but ignoring any one that's in `\index{...}`) for confirmation. The prompted name will be last name first, followed by a comma, and then the other parts of the name. Names like "Michael Van De Vanter" will be regarded as "Vanter, Michael Van De", which is of course wrong. However, this can be modified before the final confirmation is made (i.e. typing **RET**)."

7.7 Format-Debug-Preview-Print (cf. Section 4.5)

7.7.1 Format (cf. Section 4.5.1)

`tex-format-document` **C-c C-f d**

Execute the program `tex-formatter` with the master file of the current document as its argument in the inferior shell process. Pop to the master buffer, if not already there. Before the job is started, the user will first be asked to enter the master pointer, if not already specified. vThe document type (TEX, AMS-TEX, LATEX, or SLITEX) will be also checked and file saving confirmation is required if buffer has been modified.

`tex-format-buffer` **C-c C-f b**

Separately format the file bound to the current buffer. Suppose the current buffer is `noo.tex` with its master being `foo.tex`, the content of the current buffer will be copied to a temporary file `noo#.tex` and the document preamble `foo+.tex` and postamble `foo-.tex`, if any, will be interpolated. This file `noo#.tex` will have a self master pointer and will be run as a stand alone document.

`tex-format-region` **C-c C-f r**

Separately format the current region. Suppose the current buffer is `noo.tex` with its master being `foo.tex`, the content of the current region will be copied to a temporary file `noo#.tex` and the document preamble `foo+.tex` and postamble `foo-.tex`, if any, will be interpolated. This file `noo#.tex` will have a self master pointer and will be vrun as a stand alone document.

7.7.2 Print and View (cf. Section 4.5.2)

`tex-hardcopy` **variable**

Name of the DVI printing and spooling scheme (default "`lpr -d -Pxp`", where "`xp`" is the printer name given by the user).

`tex-printer-list` **variable**

List of available printers (default "`(ip, cx, dp, gp)`").

`tex-printer-default` **variable**

Name of the default printer (default "`gp`").

`tex-extractor` **variable**

The program to be used to extract pages from a DVI file. Default is `dviselect` written by Chris Torrek of Maryland.

`tex-print-all` **C-c C-p SPC**

Print the entire DVI file using the program `tex-hardcopy`. Suppose this command is issued in buffer `foo.tex`, then implicitly this command takes `foo.dvi` as its operand.

`tex-print-partial` **C-c C-p DEL**

Print selected pages of a DVI file. Suppose this command is issued in buffer `foo.tex`, then implicitly this command takes `foo.dvi` as its operand. The user will be asked to specify the pages and the program `tex-extractor` will then be run. The extracted pages will be put in file `foo%.dvi` and will be printed.

`tex-softcopy` **variable**

Name of the DVI previewer (default "`/usr/local/dvitool -E`").

`tex-view-all` **C-c C-v SPC**

View the entire DVI file using the preview program `tex-softcopy`. Suppose this command is issued in buffer `foo.tex`, then implicitly this command takes `foo.dvi` as its operand.

`tex-view-partial` **C-c C-v DEL**

View selected pages of a DVI file. Suppose this command is issued in buffer `foo.tex`, then implicitly this command takes `foo.dvi` as its operand. The user will be asked to specify the pages and the program `tex-extractor` will then be run. The extracted pages will be put in file `foo%.dvi` and the file will be viewed..

7.7.3 Display (cf. Section 4.5.3)

`tex-display-document` **C-c C-d d**

Execute the program `tex-displayer` (i.e. either `texdvi`, `amstexdvi`, `latexdvi`, or `slitexdvi`) with the master file of the current document as its argument in the inferior shell process. Pop to the master buffer, if not already there. Before the job is started, the user will first be asked to enter the master pointer, if not already specified. The document type (TEX, AMS-TEX, LATEX, or SLITEX) will be also checked and file saving confirmation is required if buffer has been modified. This function is equivalent to doing **C-c C-f d** followed by, when it's finished, **C-c C-v SPC**, if `tex-hardcopy` is bound to `dvitool`.

`tex-display-buffer` **C-c C-d b**

Display the file bound to the current buffer separately. Suppose the current buffer is `noo.tex` with its master being `foo.tex`, the content of the current buffer will be copied to a temporary file `noo#.tex` and the document preamble `foo+.tex` and postamble `foo-.tex`, if any, will be interpolated. This file `noo#.tex` will have a self master pointer and will be run as a stand alone document.

`tex-display-region` **C-c C-d r**

Display the current region separately. Suppose the current buffer is `noo.tex` with its master being `foo.tex`, the content of the current region will be copied to a temporary file `noo#.tex` and the document preamble `foo+.tex` and postamble `foo-.tex`, if any, will be interpolated. This file `noo#.tex` will have a self master pointer and will be run as a stand alone document.

7.7.4 Executing Other Programs (cf. Section 4.5.4)

`tex-execute` **C-c C-e**

Execute an external program in the inferior shell process. Specify the program name and its switches at the first prompt and give the file name at the second prompt.

7.7.5 Other Facilities for Debugging (cf. Section 4.5.5)

`tex-goto-error` **C-c C-@**

Go to the next error generated by the formatter or BIBTEX. The user can start a new formatting job (**C-c C-f**) from the source buffer once the errors are corrected. If the previous job is at a halt state (e.g. at the TEX prompt '?') it is terminated before the new job is started. If there are no more errors, point is placed at the shell window where the previous formatting job was left off.

This command can also be issued at the shell buffer, if the formatting job is initiated by *TEX-mode*. If, however, the job is started manually by the user in the shell buffer, the first **C-c C-@** must be issued at a buffer which is in *TEX-mode*. But this only has to be done for the very first time. Once the command is invoked, it is known to the shell key map.

If this command is invoked during the second stage of error correction in bibliography making (cf. Section **4.3.2**, Step 2), it will position the cursor to the next error in a `.bib` file detected by BIBTEX. At this point the command is actually invoking `bibtex-goto-error`, which is imported from BIBTEX-mode by autoloading and therefore can be invoked in any `.bib` files to get to the next error (as opposed to going back to the original `.tex` source file). If there are no more errors, the recursive edit started earlier is terminated and bibliography making is resumed.

`tex-comment-region` **C-c C-c**

Insert a TEX comment sign (`%`) in front of each line between mark and point. With positive prefix argument N, insert that many `%`'s. Otherwise, insert just one. Any prefix arguments less than one are converted to 1 implicitly.

`tex-uncomment-region` **C-c C-u**

Delete the leading `%` in front of each line between mark and point. A line is unchanged if its leading character is not a `%`. With prefix argument N being positive, delete that many `%`'s, if any. Any prefix arguments less than one are converted to 1 implicitly.

8 Index to Function Names and Variables

9 Index to Key Bindings

— Basic Matching —

(See Sections **4** and **7.3**)

C-c C-t ESC	`tex-toggle-boundary-check`
C-c (	`tex-bounce-backward`
C-c)	`tex-bounce-forward`

— Zone Matching —

(See Sections **4.1.1** and **7.3.1**)

C-c SPC	`tex-zone-open`
C-c DEL	`tex-zone-close`
C-c C-z	`tex-zone-inspect`
C-c ESC-$	`tex-zone-math`
C-c ESC-'	`tex-zone-single-quote`
C-c ESC-"	`tex-zone-double-quote`
C-c ESC-d	`tex-zone-display-math`
C-c ESC-c	`tex-zone-centerline`
C-c ESC-h	`tex-zone-hbox`
C-c ESC-v	`tex-zone-vbox`
C-c ESC-b	`tex-zone-bf`
C-c ESC-i	`tex-zone-it`
C-c ESC-r	`tex-zone-rm`
C-c ESC-s	`tex-zone-sl`
C-c ESC-t	`tex-zone-tt`

— Word Matching (Backward) —

(See Sections **4.1.2** and **7.3.2**)

C-c $	`tex-word-math`
C-c d	`tex-word-display-math`
C-c '	`tex-word-single-quote`
C-c "	`tex-word-double-quote`
C-c c	`tex-word-centerline`
C-c h	`tex-word-hbox`
C-c v	`tex-word-vbox`
C-c b	`tex-word-bf`
C-c i	`tex-word-it`
C-c r	`tex-word-rm`
C-c s	`tex-word-sl`
C-c t	`tex-word-tt`

—Word Matching (Forward) —

(See Sections **3.1.2** and **7.3.2**)

C-c-4 $	`tex-word-forward-math`
C-c-4 d	`tex-word-forward-display-math`
C-c-4 '	`tex-word-forward-single-quote`
C-c-4 "	`tex-word-forward-double-quote`
C-c-4 c	`tex-word-forward-centerline`
C-c-4 h	`tex-word-forward-hbox`
C-c-4 v	`tex-word-forward-vbox`
C-c-4 b	`tex-word-forward-bf`
C-c-4 i	`tex-word-forward-it`
C-c-4 r	`tex-word-forward-rm`
C-c-4 s	`tex-word-forward-sl`
C-c-4 t	`tex-word-forward-tt`

— Automatic Matching —

(See Sections **4.1.3** and **7.3.3**)

$	`tex-dollar`
C-c C-t $	`tex-toggle-dollar`
"	`tex-quote`
C-c C-t "	`tex-toggle-quote`

— LaTeX Environments —

(See Sections **4.1.4** and **7.3.4**)

C-c C-l SPC	`tex-latex-open`
C-c C-l DEL	`tex-latex-close`
C-c C-l LFD	`tex-latex-skip`
LFD	`tex-newline-indent`
C-c C-l a	`tex-latex-array`
C-c C-l c	`tex-latex-center`
C-c C-l e	`tex-latex-enumerate`
C-c C-l f	`tex-latex-figure`
C-c C-l i	`tex-latex-itemize`
C-c C-l p	`tex-latex-picture`
C-c C-l q	`tex-latex-quote`
C-c C-l TAB	`tex-latex-tabbing`
C-c C-l t	`tex-latex-table`
C-c C-l C-t	`tex-latex-tabular`
C-c C-l v	`tex-latex-verbatim`

— Customizing Delimiters —

(See Sections **4.1.5** and **7.3.5**)

C-c C-\\ a	`tex-make-auto`
C-c C-\\ s	`tex-make-semi`
C-c C-\\ e	`tex-make-env`

— Spelling Checking —

(See Sections **4.2** and **7.4**)

C-c C-s d	`tex-spell-document`
C-c C-s b	`tex-spell-buffer`
C-c C-s r	`tex-spell-region`
C-c C-s w	`tex-spell-word`

— Bibliography Making —

(See Sections **4.3** and **7.5**)

C-c C-b b	`tex-bib-buffer`
C-c C-b c	`tex-bib-cite`
C-c C-b d	`tex-bib-document`
C-c C-b n	`tex-bib-nocite`
C-c C-b r	`tex-bib-recover`
C-c C-b s	`tex-bib-save`

— Indexing —

(See Sections **4.4** and **7.6**)

C-c C-i d	`tex-index-document`
C-c C-i b	`tex-index-buffer`
C-c C-i r	`tex-index-region`
C-c C-i w	`tex-index-word`
C-c C-i m	`tex-index-make`
C-c C-i s	`tex-index-save`
C-c C-i a	`tex-index-authors`
C-c C-i c	`tex-index-chmod`
C-c C-i v	`tex-index-variant-toggle`
C-c C-i p	`tex-index-prefix-toggle`
C-c C-i k	`tex-index-keyptrn-toggle`

— Format-Debug-Print-Preview —

(See Sections **4.5** and **7.7**)

— Useful Variables —