User's Guide
Alpha LaTeX Macros

Version 5.0 early draft

Original manual by Tom Scavo
Updates by the AlphaTcl community

October 2003

Contents

Chapter 1

Overview

Welcome to the *Alpha* LaTeX macros, a library of Tcl code which forms a mode for the text editor *Alpha*, designed to ease the input and processing of LaTeX documents when using any of the Alpha family of editors.

This document is still in need of a major update to document changes which have been implemented with versions 5.0 and above.

1.1 Features

- automatically invokes TeX mode when opening or saving a `.tex` document
- single-keystroke typesetting of LaTeX documents (will even typeset an untitled or unsaved window, or portions of windows)
- through the creation of a TeX fileset, provides full support for 'master documents' composed of many separate .tex files (which are 'input' or 'included' into the master document). This support carries as far as automatically routing all document-centric commands (e.g. 'typeset') from sub-documents to the master document.
- with Alpha 7.6, Alpha 8 (on MacOS classic), and Alpha X or Alphatk (on MacOS X), it works with all known Macintosh TeX implementations, including OzTeX, *Textures*, CMacTeX, Euro-OzTeX, DirectTeX, DirectTeX Pro, TeXshop, iTeXMac.
- with Alphatk on any platform (Windows, Unix, MacOS X), it works with all known command-line TeX tools including teTeX, MikTeX, Yap, Xdvi, including the tools running under 'cygwin' on Windows.
- with Alphatk or Alpha X it can act as an interactive shell for your TeX processing.

- where the TeX system supports it, synchronising back and forth between the source .tex document and a dvi viewer is provided (*Textures*, Yap, Xdvi, at least).
- additional support provided for *Textures* version 1.8
- dynamic menus and menu items
- choice of long or short LaTeX menu; optional floating menu, as well
- color syntax highlighting of LaTeX keywords
- intelligent treatment of highlighted text (called "wrapping")
- easily creates LaTeX document templates, complete with indentation and tab stops
- handy text-to-LaTeX conversion utilities
- quickly navigate and select commands, environments, subsections, sections, and chapters
- "smart quotes" and "smart dots" with on-the-fly escape; "smart" subscripts and superscripts as well
- command-double-click feature that chases references and citations, or opens `\input` and `\include` files; also opens `\bibliography`, `\includegraphics`, `\usepackage`, and `\documentclass` files
- a handy mark menu for navigating large LaTeX documents
- follows closely the terminology and organization of Leslie Lamport's *LaTeX: A Document Preparation System* [Reading, MA: Addison-Wesley, 1985, 1994 (ISBN 0-201-52983-1)]
- closely integrated with new features of AlphaTcl 7.x and 8.x (for example the user can easily add menu items to most TeX menus, and can set user-defined key-bindings to most menu items)
- sophisticated support for Electric Completions and Expansions packages.
- smart indentation of pasted text if the smartPaste package is activated.

1.2 Documentation

Pull down the System help menu and choose the command LaTeX Help. This will open a new window with a brief introduction to the *Alpha* LaTeX macros. From this window, you can access numerous other documentation files using *Alpha*'s built-in hypertext capability. Just click on any of the following links in LaTeX Help:

User's Guide: an introduction to the *Alpha* LaTeX macros (this file)
LaTeX Menus: commands and bindings (organized by menu)
LaTeX Key Bindings: commands and bindings (organized by command key)

Note that the User's Guide is available in both LaTeX and HTML formats. (The latter is online at URL

`no-current-url`

and was created by the program `TeX4ht`.

In addition to the above *Alpha* LaTeX documents, there are also the more general Web pages

An Introduction to LaTeX
`http://www.latex-project.org/intro.html`
An Introduction to LaTeX and $\mathcal{A}_{\mathcal{M}}\mathcal{S}$-LaTeX
`http://whereisthis`

1.3 LaTeX 2_ε

The *Alpha* LaTeX macros support LaTeX 2_ε. *Alpha* no longer directly supports the 2.09 macros (although many are unchanged in LaTeX 2_εso if you must you could probably still manage). LaTeX 2_ε will typeset a 2.09 document automatically, using what is called "compatibility mode". Most of the LaTeX 2.09 commands and environments have been preserved in 2_ε, making the transition from 2.09 to 2_ε relatively painless (from the user's point of view, at least).

1.4 Installation

Alpha is configured to use the LaTeX macros right out of the box, so there is no installation process per se. However, there are a number of flags and variables that control the inner workings of `latex.tcl` that may be changed at the user's discretion.

The following TeX mode flags and variables may be accessed by pulling down the **Config** menu and opening the Preferences dialog on the **Current Mode** submenu. See the Alpha Manual (on the System help menu under the question mark) for more information about *Alpha*'s global preferences.

1.4.1 Flags

buildPkgsSubmenu The **Packages** submenu is an optional submenu containing a list of all `.tex` and `.sty` files in your TeX search path. Choosing a filename from the list inserts the corresponding `\usepackage` command into the preamble of the current document. By default, however, the **Packages** submenu is not built when the LaTeX macro package is loaded. To build this submenu on-the-fly, enable the flag buildPkgsSubmenu as described above, and then choose Rebuild Documents Submenu on the **Documents**

submenu (see section 2.1.3). Thereafter, the **Packages** submenu will be built automatically along with the LaTeX menu.

deleteObjNoisily One of the basic `latex.tcl` operations is to insert an object into the current document. If, at the time the insertion command is issued, there is a selection (i.e., text is highlighted), then the program behaves differently depending on the value of the flag deleteObjNoisily. If set to true, the user will be prompted before any selected text is deleted. If, on the other hand, this flag is false, then the selection is replaced quietly and without warning (although it may be undone). By default, deleteObjNoisily is set to true. NOTE: Not all objects are "inserted" into the document since sometimes there is an attempt to "wrap" the current selection. See section 1.6 for more information.

deleteEnvNoisily Before an environment is inserted into the document, the program checks to see if there is a selection. If so, and the flag deleteEnvNoisily is set to true, the user is asked whether or not the current selection should be replaced; if false, the current selection is deleted without warning. Note that the default value of deleteEnvNoisily has been set to true. Like objects, environments may wrap, so sometimes the current selection is treated differently. See section 1.6 for details.

promptNoisily Some environment commands prompt the user for input. As mentioned below, if useStatusBar is set to true, the prompt is displayed on the thin status bar at the bottom of the screen. This is less obtrusive than a dialog, but may go unnoticed at first, so if promptNoisily is set to true (which it is, by default) and useStatusBar is enabled, the program beeps prior to displaying the prompt. You can turn off this annoying sound by invoking the Flags command on the **Current Mode** submenu of the **Config** menu and removing the check on promptNoisily.

runTeXInBack If true, typesetting will occur in the background. This flag is false by default. The main **Process** menu contains options for both foreground and background typesetting. This preference setting simply defines which of the two is bound to a ⟨Cmd-T⟩ keypress, and which to ⟨Shift-Option-T⟩.

searchNoisily Many commands cause `latex.tcl` to search the current document. If a search fails, and searchNoisily is set to true, the program displays a message on the status bar and beeps. If, on the other hand, searchNoisily is set to false, only the message is displayed. By default, searchNoisily is set to true.

smartDots By default, `latex.tcl` replaces three consecutively typed dots (`...`) with the LaTeX command `\ldots`. To escape the effect of smartDots, press the ⟨Delete⟩ key on-the-fly.

smartQuotes If this flag is set to true, pressing the single quote key ⟨'⟩ will generate ‘ or ’ automatically depending on the context. Similarly, pressing the double quote key ⟨"⟩ generates “ or ”, whichever is required. Set smartQuotes to false if you want the single and double quote keys to insert 'and " literally, or press the ⟨DELETE⟩ key to escape the effect of smartQuotes on-the-fly.

smartScripts When this flag is enabled (which it is by default), the ^ and _ keys on a U.S. keyboard are bound to the commands superscript and subscript, respectively, on the **Formulas** submenu of the LaTeX menu (see section 2.1.5). Press the ⟨DELETE⟩ key to escape the effect of smartScripts on-the-fly.

useBrackets In LaTeX, the `displaymath` environment is equivalent to `\[...\]`. If you prefer to use the latter, set useBrackets to true. By default, useBrackets is set to false, that is, the `displaymath` environment is used to construct multi-line math displays. Note: By default, `latex.tcl` *always* uses `\[...\]` inline (unless useDollarSigns is set to true—see below). This setting applies whenever LaTeXmode inserts math environments (usually due to a menu selection).

useDollarSigns Support is provided for both the TeX and LaTeX methods of invoking inline math mode (see the **Math Modes** submenu in section 2.1.5 for the various options), but only one of these is bound to command keys (namely, ⟨CTL CMD M⟩ and ⟨CTL OPT CMD M⟩, by default). This is what the flag useDollarSigns does. If set to true, `latex.tcl` uses dollar signs to delimit inline math mode (`$...$` and `$$...$$`), whereas if it is false, LaTeX notation will be used (`\(...\)` and `\[...\]`). By default, useDollarSigns is set to false—the LaTeX way of doing things.

useStatusBar This flag determines whether or not the status bar is used when prompting for user input. (The status bar is a long, thin message area at the bottom of your screen.) Use of *Alpha*'s status bar is enabled in TeX mode, by default. See the related flag promptNoisily above.

wordWrap If this flag is set to true, the program automatically inserts a carriage return as the cursor nears the end of a line (the length of which is defined by the variable fillColumn described in section 1.4.2 below); otherwise, the line extends indefinitely to the right (until the ⟨RETURN⟩ key is pressed, of course). By default, wordWrap is turned on in TeX mode. See the Alpha Manual on the System help menu (under the question mark) for more information.

1.4.2 Variables

boxMacroNames This TeX mode variable contains a list of names of box-making macros used in the body of a figure environment. The standard

LaTeX commands `\includegraphics` and `\includegraphics*` (both part of the LaTeX 2_ε `graphics` package) are included in this list by default.

citeCommands Any command listed as a "cite" command is command-double-clickable (see section 3.2). The standard LaTeX commands `\cite` and `\nocite` are included in this list by default.

fillcolumn, leftFillColumn See the Alpha Manual on the System help menu (under the question mark) for more information about these variables.

funcExpr In TeX mode, funcExpr is a regular expression used to search for a subsection header (see the commands Next Subsection and Prev Subsection described in section 2.1.2) and to build the **funcs** pop-up menu (see section 2.2).

funcExprAlt In TeX mode, funcExprAlt is a regular expression used to search for a section header (see the commands Next Section and Prev Section described in section 2.1.2).

parseExpr The variable parseExpr is a regular expression used to build the **funcs** pop-up menu. See section 2.2 for more information.

prefixString This variable is used in conjunction with the Comment Line command on *Alpha*'s **Text** menu. In TeX mode, this string is set to "%␣" by default, which makes comment line very useful for commenting out large blocks of LaTeX code.

refCommands Any command listed as a "ref" command is command-double-clickable. The standard LaTeX commands `\ref` and `\pageref` are included in this list by default.

TeXSearchPath Whenever LaTeX mode searches for files, it examines each directory in this user-defined list. Such a search is normally triggered by the user command-clicking on a citation, reference or input command in a LaTeX document. In such cases, *Alpha* attempts to find the appropriate document and open it for the user.

wordBreak This variable holds a regular expression that define a "word" in TeX mode. (A "word" is any text string that is double-clickable.) See the Alpha Manual in the help menu for more information.

wrapBreak, wrapBreakPreface These variables are similar to wordBreak and wordBreakPreface above, except that they are used by *Alpha* to wrap lines, not delineate words. See the Alpha Manual on the System help menu (under the question mark) for more information.

1.4.3 Tips and tricks

A useful installation trick that you might want to put in your `prefs.tcl` file (opened by choosing Edit Prefs on the **Global** submenu of the **Config** menu) is the following key binding:

```
Bind 'n' <cs> {TeX::newLaTeXDocument}
```

With this binding, it's easy to bring up a new LaTeX document no matter where you are or what you're doing. Regardless of the file you're currently editing, simply press ⟨Shf Cmd N⟩ to open a new LaTeX document (see the New Document command in section 2.1.3 for details). If you use *Alpha*'s 'New Document' package, this capability is automatically part of the standard 'New Document' dialog.

Alpha is completely customizable, but it's not a good idea to modify its Tcl files directly. Instead, put your modifications in preferences files designed specifically for this purpose. Besides the global preferences file `prefs.tcl` mentioned above, there are also mode-specific preferences files. TeX mode, for instance, has its own prefs file called `TeXPrefs.tcl`. To edit this prefs file, open a `.tex` file and choose the command Edit Prefs File on the **TeX Mode Prefs** submenu of the **Config** menu. All TeX-related modifications should be placed in this preferences file.

Note: All preferences files are stored separate to *Alpha* and are not touched when you upgrade your version of *Alpha*.

The LaTeX menu can be modified to suit your needs. Three different methods allow you to insert new commands into any of the **Text** or **Math** submenus, presented here in increasing order of difficulty.

The first method will define a simple LaTeX "command" item that is followed by braces. Select the Add Menu Template Item from the **LaTeX Menu** submenu. Choose a menu that should contain the new item. A new dialog appears with three text fields. In the first field enter the name of the item as you want it to be presented in the menu. The second field contains text looking like **XXX•{•** — replace **XXX** with the name of the command as it should appear in the window when the menu item is called. If you want to include any default text within the braces that will follow this item, you can insert that in either the second or the third text fields. Click on the "OK" button to save the changes. All menu items added in this way can be changed later using the Edit Template Item command, or deleted using the Delete Template Item command.

The second method requires a little working knowledge of AlphaTcl. Suppose, for example, that you want to insert a new My Template item to the **Arrows** menu. Open your `TeXPrefs.tcl` file by selecting the Edit Prefs File item in the **Config** >**TeX Mode Prefs** submenu. Paste in a command that looks like this:

```
menu::insert "Arrows" items 0 "My Template"
```

Now you need a Tcl procedure that will be evaluated when this item is selected. It should be named **TeX::MyTemplate**, i.e. scrunch the name together by removing all spaces, but retain upper and lower cases to match the

name of the menu item. Place this new procedure in your `TeXPrefs.tcl` file, and "evaluate" this file by pressing ⟨COMMAND-L⟩. The new item should now appear in the menu. If it does, save the preferences file, and close it. This file will be "sourced" automatically the next time that Alpha is started and a .tex file has been opened.

The third method requires more extensive knowledge of AlphaTcl. You must turn on the package "Smarter Source", and place a copy of the file `latexMenus.tcl` in your "Smarter Source" folder. Look for the definition of the menu in question, and modify it to include the items that you want. Restart Alpha, and your customized LaTeX menu should include the items you added. Unfortunately, you're not done yet. You must now modify the "menu procedure" that is called by the given menu in order for your new item to actually do anything! This procedure is listed in the definition of the menu, and if it is not in the file `latexMenus.tcl` then you'll have to Command-Double-Click on the name of the procedure to find it, and then copy *that* file to your "Smarter Source" folder as well, and then modify it accordingly.

An alternative to this third method would be to simply use the "Macros" package to define a new macro, edit it to do what you want, and then assign it a keyboard shortcut so that it can be called whenever you want. Not as elegant, perhaps, but in the end just as effective and less complicated.

Feel free to ask for more help in the AlphaTcl mailing lists.

1.5 TeX Filesets

A very important capability of *Alpha*'s LaTeX mode is the seamless handling of documents composed of many separate files on disk. Through the one-off action of creating a TeX-fileset you inform *Alpha* of the relationship between these parts and it automatically uses that information from that point onwards. This means, for example, when you attempt to typeset a sub-document, *Alpha* notices, retrieves the master document instead and processes it for typesetting. Here is a partial list of the actions which are automatically aware of multi-part documents:

1. Typesetting, BibTeX'ing and all other actions in the **Process** menu.
2. Double-clicking on references to jump to their matching label.
3. Searching (the standard search dialog provides option to search all files in any defined fileset).
4. Synchronicity features (.tex to .dvi and .dvi to .tex synchronisation).

In addition all parts of the multi-part TeX/LaTeX document are listed in *Alpha*'s **Fileset** menu which therefore provides quick and easy access to editing these files.

To create a TeX fileset, select **Fileset** >**Utilities** >**New Fileset**. When prompted for the type of the new fileset, make sure you select TeX.

1.6 Basic operations

The *Alpha* LaTeX macros revolve around two basic operations called insertObject and wrapObject. Basically, insertObject is a call to the primitive procedure insertText preceded by the automatic deletion of previously selected text (this behavior is easily changed, however, by resetting the flag deleteObjNoisily described in section 1.4.1). For example, if there is no current selection, choosing the command alpha from the **Greek** submenu of the LaTeX menu inserts the LaTeX command `\alpha` at the insertion point; otherwise, if there is a selection, it is replaced with the string "`\alpha`". In other words, insertObject works just like the familiar Paste command on the **Edit** menu. It turns out that a large number of commands in `latex.tcl` rely on insertObject, but sometimes it's faster to type the desired LaTeX command directly. (Even faster is to use the corresponding command keys, but more on that later.) On the other hand, if you forget the syntax of a particular LaTeX command, it's sometimes easier to look it up on the LaTeX menu than it is in a reference manual.

The complementary operation to insertObject is called wrapObject. The difference between the two is the way the latter treats the current selection, that is, wrapObject inserts its argument at the insertion point (just like insertObject), but if there is a selection, wrapObject cuts it out (without effecting the state of the Clipboard) and inserts it in the middle of the chosen command. For example, consider the LaTeX menu command footnote on the **Miscellaneous** submenu (see section 2.1.4). This command inserts the string "`\footnote{}•`" (without the double quotes, of course) into the document, positioning the insertion point between the pair of braces. The user then types the text to be footnoted and presses the ⟨Tab⟩ key, after which the tab stop macro finds (and deletes) the bullet • at the end of the string. (Note: Use ⟨Opt Tab⟩ to insert a literal tab character into the document.) On the other hand, if a selection exists at the time the footnote command is issued, the selection itself is surrounded by LaTeX's `\footnote` command, and the insertion point is brought to the end of the selection automatically. Some commands, for better or worse, even go so far as to insert the selection into one of several competing positions within the command string. The fraction command on the **Formulas** submenu (section 2.1.5) is a good example of this type of behavior. It assumes the current selection (if there is one) is the numerator of the fraction to be typeset, cutting and pasting accordingly.

The concept of wrapping is carried one step further in the case of environments. Suppose you want to center an existing `tabular` environment, for example. Just select the `tabular` environment to be centered and choose the center command from the **Environments** submenu on the LaTeX menu (see section 2.1.4). The resulting `center` environment will completely surround the existing `tabular` environment, indenting the latter one tab stop to the right.

Not all environments wrap, however. Those environments whose body is very structured (such as `enumerate`, `itemize`, `description`, `thebibliography`, `tabular`, `array`, `eqnarray`, and `eqnarray*`) do not. Instead, these environments simply insert text into the document. If there happens to be a selection

at the time one of these commands is issued, an alert appears asking if the selection should be deleted. To turn this alert off, simply toggle the flag deleteEnvNoisily (see section 1.4.1) in the Flags dialog on the **Current Mode** submenu on the **Config** menu.

1.7 Whitespace

Before continuing, let me say a few words about whitespace. In virtually all cases, superfluous whitespace in command strings has been deleted. For example, objects inserted with insertObject (a sizable portion of `latex.tcl`'s functionality) do not routinely insert a trailing space character. Instead, the user must decide whether or not space should immediately follow a particular LaTeX control word, since sometimes it's needed and sometimes it's not. Disabling the indentOnReturn preference for LaTeX mode also prevents a lot of extraneous whitespace from being inserted into your document... but then it won't look so good!

1.8 Customising

LaTeX mode contains a vast array of preferences which can be rather confusing. Here are some not so obvious ways to customise it.

Always opening the log file? Add a key-binding (here with ⟨Opt Cmd O⟩):

```
Bind 'o' <co> {TeX::mp::Processmenu "" "Open .log"} TeX
```

Want to add some optional parameters to an automatically inserted latex environment? (E.g. you like `enumerate` with a parameter in square brackets '[]'), then add the following line to your `TeXPrefs.tcl` (where the stars are bullets):

```
set TeXbodyOptions(enumerate) "\[*a|i*\]"
```

1.9 Bugs

Comments, suggestions, and bug reports are certainly welcome. In fact, many of the improvements and features in this version of `latex.tcl` were suggested by *Alpha* LaTeX users. (Some of them even sent me code!).

Alpha is privileged by an active and helpful user community whose main communication channels are two mailing lists you are invited to subscribe to. The developers' list is for discussing technical issues, while the users' list is for all sorts of questions and hints — it is read also by the developers. Questions are usually answered within a day, and often they lead to improvements in subsequent versions of Alpha and its documentation. Go to:

`http://lists.sourceforge.net/lists/listinfo/alphatcl-users`
`http://lists.sourceforge.net/lists/listinfo/alphatcl-developers`

In addition there is the bugzilla bug tracking system which should be used to report bugs:

`http://www.maths.mq.edu.au/~steffen/bugzilla/`

The following are known bugs in `latex.tcl`:

- When wrapping, the frac command does not remove redundant parentheses.
- The options commands assumes the `\documentclass` command already has an optional parameter (which it does if the document template was inserted via the LaTeX menu). Moreover, the options command does not check for duplicate options.
- The commands Prev Command Select With Args and Next Command Select With Args will not select LaTeX commands whose argument(s) contain braces.
- The **Goto** submenu could be better organized (and will be, if I can ever think of a good set of command keys!).
- The results of Delete Comments can not be undone (but it works!).
- The Any TeX File command on the **Process** submenu does not open the correct folder.
- The at-symbol @ is not recognized as a LaTeX command character in `.sty` files.
- The command keys for subscript and superscript are not compatible with international keyboards. However they can be set by the user manually.
- The marking algorithm should ignore comments.
- The **Process** submenu gets confused if the current file has a name containing meta-characters. The only known fix is to avoid the characters `<`, `(`, `;`, `!`, `^`, and `/` in filenames.

1.10 Acknowledgments

Tom Scavo wrote most of LaTeX mode and this documentation. The present author, Vince Darley can take very little credit.

Here are Tom's original acknowledgements.

Numerous people have made significant contributions to the *Alpha* LaTeX macros. You will find their names and initials scattered throughout this and other *Alpha* documents. Tom Pollard pollard@cucbs.chem.columbia.edu and Vince Darley vince@santafe.edu have been especially helpful and deserve a lot of credit. Of course, none of this would have been possible without the support and encouragement of *Alpha*'s author,

Pete Keleher keleher@cs.umd.edu

whom I heartily thank.

Chapter 2

Menus

2.1 The LaTeX menu

Upon entering TeX mode, either

1. *manually* (by choosing TeX from the pop-up mode menu on the status bar at the bottom of your screen); or
2. *automatically* (whenever a `.tex` or `.sty` file is opened or saved);

a new menu appears in the menu bar. The LaTeX menu provides access to scores of procedures loaded automatically the first time TeX mode is entered.

The bulk of the LaTeX menu contains items for marking up "Text" and "Math" items. You can condense the submenus in these two categories by selecting the Compress Text/Math Menus command in the **LaTeX Menus** submenu. All of the commands will be available even if the menus have been compressed, these preferences only affect the path that your mouse will have to take in order to access them.

The LaTeX menu follows closely the organization and terminology of Lamport's *LaTeX: A Document Preparation System* [second edition, Addison-Wesley, 1994], especially chapter 3. Many people agree that the *LaTeX* book is still the definitive LaTeX reference. In conjunction with *The LaTeX Companion* by Goossens, Mittlebach, and Samarin [Addison-Wesley, 1994 (ISBN 0-201-54199-8)], these two books constitute the "official" LaTeX 2_ε documentation. These books, as well as Knuth's classic *TeXbook* [Addison-Wesley, 1986 (ISBN 0-201-13448-9)], should be on every serious LaTeX user's desk.

The LaTeX menu is organized into four parts: general commands, document-related commands, paragraph mode commands (that is, text commands), and math mode commands. Each group of commands is separated by a thin grey line on the LaTeX menu. The order of the commands on any given submenu is significant insofar as possible. For example, the various commands on the **Environments** submenu mirror the corresponding command keys, while other

submenus follow the ordering found in the LaTeX book. We'll try to point out these organizational aids as we go along.

A brief description of each available command follows. See section 1.2 for pointers to other help documents.

2.1.1 Menu Keyboard Shortcuts

Most of the LaTeX menu items can be accessed by keyboard shortcuts that are specific to TeX mode. Many of these shortcuts can be changed to any combination that might seem more intuitive.

Before you change a menu item keyboard shortcut, you should probably make sure that the new modifier/keystroke combination is not already used by some other function. Select the Describe Binding menu command in the **Config** menu to determine if this is the case.

Once you have decided upon some keyboard shortcuts that you want to change, use the Assign Menu Bindings command in the **LaTeX Menu** submenu to first select the relevant menu, and then change the bindings that appear in the dialog by clicking on the relevant "Set" buttons. When you have finished, click on "OK", and then either choose another menu or select "Finish" and press "OK". All changes will take effect immediately, and will be remembered between editing sessions.

2.1.2 General commands

The *Alpha* LaTeX mode can interact with a very large number of other applications for typesetting, viewing, converting documents, etc.

In particular recent versions of Alphatk can be used as external editors for command-line LaTeX tools.

Most of this interactions occurs through the **Process** submenu.

Process submenu

Typeset file.tex	⟨Cmd T⟩
View file.dvi	⟨Shf Cmd V⟩
Print file.dvi	⟨Shf Cmd P⟩

If you use *Textures*, OzTeX, CMacTeX, or DirectTeX, or any command-line tool on MacOS X, Windows or Unix, you'll be happy to know that *Alpha* and LaTeX work well together. To typeset the file you're currently editing in *Alpha*, simply choose Typeset from the **Process** submenu or press ⟨Cmd T⟩. *Alpha* first checks to make sure that any changes to the file have been saved; if not, the user is prompted for the appropriate action. Note that it is not necessary to save the document to process the window. Just click the "No" button when asked to save the current window, whereupon *Alpha* will pass the contents of the window to the LaTeX application and typeset the file automatically. If the flag runTeXInBack is set to true, typesetting will occur in the background.

Inverse search — jumping to a source file The inverse operation, switching from LaTeX to *Alpha*, depends on which LaTeX application you're using. OzTeX users, for example, simply choose the Edit command from OzTeX's **Edit** menu or press ⟨Cmd E⟩ to return to *Alpha*. (By the way, typing 'e' in response to a LaTeX error message in the OzTeX window throws you back into *Alpha* at the offending line. The same trick works in CMacTeX and DirectTeX, too.)

Many TeX-related applications allow the possibility of jumping back to the source file corresponding to a certain situation (an error in processing or a mouse click in a dvi window). This is sometimes known a "inverse search"

For example, the xdvi viewer works very well in OS X. If one clicks on an xdvi window then xdvi attempts to execute a unix command that one can specify, e,g. a unix shell script. Typically, this is used to open an editor with a specified file and at a specified place. Thus, if using Alphatk on MacOS X one can set the environment variable XEDITOR or EDITOR to

```
/Applications/Alphatk.app/Contents/MacOS/Alphatk +%l %f
```

or on Windows:

```
C:/Apps/alphatk.exe +%l %f
```

and clicking on an xdvi window opens the tex source file in alphatk at the right spot.

If operating with the cygwin environment on Windows (using Xdvi running in Xwin on top of cygwin), then one should use

```
C:/Apps/alphatk.exe -cygwin +%l %f
```

so that Alphatk knows to interpret the path as a cygwin-style path instead of an ordinary Windows path.

MikTeX When using MikTeX, the following should be included in the "miktex.ini" file:

```
Editor = C:/Apps/alphatk.exe +"%l" %f
```

View synchronization This is also known as "forwards search" or "synchronicity". When editing any particular .tex file, one can jump to that position in the typeset document (in a .dvi view, or in *Textures* view). This is known to work with *Textures* on MacOS, with Yap and Xdvi on Windows. Normally ⟨Control `⟩ is bound to this action.

The commands that Alphatk uses for this, internally, are:

```
yap.exe -1 -s "%l %n" "%P.dvi"
xdvi -sourceposition "%l %n" "%P.dvi"
```

where %l is the linenumber, %n is the tail of the name of the source file, and %P is the name of the dvi file (the tex master file).

More process menu commands

Typeset Clipboard ⟨Shf Cmd T⟩
Typeset Selection

To typeset the contents of the Clipboard, choose the Typeset Clipboard command from the **Process** submenu or press ⟨Shf Cmd T⟩. This command is handy for typesetting and viewing TEX or LATEX code copied to the Clipboard from other applications such as terminal emulators or e-mail clients.

It's also possible to typeset a portion of a document. Simply select (i.e., highlight) the LATEX code you'd like to typeset and choose Typeset Selection from the **Process** submenu. *Alpha* will construct a temporary document from the current document's preamble and the highlighted text, and pass this virtual document to the TEX application to be typeset automatically.

dvips file.dvi
Open file.ps
View file.ps
Print file.ps

To convert a `.dvi` file to a `.ps` file, choose the dvips command on the **Process** submenu. Assuming you have the necessary applications installed on your Macintosh, choose View file.ps or Print file.ps to view or print the resulting `.ps` file.

Once a `.ps` file has been created, you may open a window containing the raw PostScript code by choosing Open file.ps on the **Process** submenu. To see this command, press the ⟨Opt⟩ key while the **Process** submenu is down.

bibtex file.aux
Open file.bbl
makeindex file.idx
Open file.ind

To run BibTEX or *MakeIndex*, choose the corresponding command from the **Process** submenu. While the **Process** submenu is down, press the ⟨Opt⟩ key and choose Open file.bbl or Open file.ind to open the file created by BibTEX or *MakeIndex*, respectively.

Open file.log
Open file.aux
Open file.toc
Open file.lof
Open file.lot
Open file.idx
Open file.blg
Open file.ilg
Open Any TeX File... ⟨Shf Cmd O⟩

The **Other Files** submenu on the **Process** submenu provides convenient access to other LaTeX auxiliary files. Choose Open Any TeX File on the **Other Files** submenu to open *any* file in the current directory.

Remove Auxiliary Files...
Remove Temporary Files

The utility Remove Auxiliary Files interactively removes all auxiliary files (`.aux .bbl .dvi .glo .idx .ind .lof .log .lot .toc .blg .clg .ilg .ps`) in the current directory. Two additional buttons have been added to the dialog: the button labeled "rm ext" removes all files with the same extension as the file displayed in the dialog, and "rm all" removes all auxiliary files from the current directory without prompting.

Alpha writes all temporary files to `$PREFS:tmp:`, which makes them easier to remove. All temporary files are removed once, at launch; however, the command Remove Temporary Files on the **Process** submenu removes all temporary files immediately.

Experimental interactive processing Not included in the process menu is some new experimental commands which allow you to execute interactive LaTeX runs from inside Alpha(tk).

Use ⟨Shift Command R⟩ to launch such an interactive session inside one of Alpha's windows. If errors occur, you can use 'e' to switch directly to the offending text.

Once this code is robust and tested, and known to work well across Windows and MacOS X, it will be integrated into the standard process menu.

Goto submenu

LaTeX	⟨Shf Cmd S⟩
BibTeX	
MakeIndex	

These commands launch and switch to the corresponding application *without* saving and typesetting the current document. The LaTeX command, for instance, is identical to the old latex command in `latex.tcl` v2.0.

Next Template Stop	⟨Tab⟩
Prev Template Stop	⟨Shf Tab⟩

As you write your document using the various commands on the LaTeX menu, templates are inserted into the text along with tab stops (represented by bullets, which may also be inserted with ⟨Opt 8⟩). The idea is to type an argument at the current tab stop, press ⟨Tab⟩ to go to the next tab stop, enter another argument, press ⟨Tab⟩ again, and so on. That's what Next Tab Stop and Prev Tab Stop do: they move around from tab stop to tab stop. Since Next Tab Stop and Prev Tab Stop are bound to the ⟨Tab⟩ and ⟨Shf Tab⟩, respectively, the

menu commands aren't as convenient as simply pressing the tab key, but they're included on the LaTeX menu for completeness.

NOTE: Press ⟨Opt Tab⟩ to insert a literal tab into the document.

Prev Command	⟨Kpad4⟩
Next Command	⟨Kpad6⟩
Prev Command Select	⟨Shf Kpad4⟩
Next Command Select	⟨Shf Kpad6⟩
Prev Command Select With Args	⟨Shf Opt Kpad4⟩
Next Command Select With Args	⟨Shf Opt Kpad6⟩

The Prev Command and Next Command commands move the cursor to the beginning of the previous or next LaTeX command, while Prev Command Select and Next Command Select select the previous or next LaTeX command. Similarly, Prev Command Select With Args and Next Command Select With Args select the previous or next LaTeX command, along with any command arguments that may be present. Required arguments containing nested braces will not be selected, however. See section 1.9 for more information about this and other *Alpha* LaTeX bugs.

Prev Environment	⟨Cmd Kpad4⟩
Next Environment	⟨Cmd Kpad6⟩
Prev Environment Select	⟨Shf Cmd Kpad4⟩
Next Environment Select	⟨Shf Cmd Kpad6⟩

Like Prev Command and Next Command, these commands either move the cursor to the beginning of the previous or next LaTeX environment, or select the previous or next LaTeX environment. They are useful for locating or relocating environments.

Prev Section	⟨Cmd Kpad8⟩
Next Section	⟨Cmd Kpad2⟩
Prev Section Select	⟨Shf Cmd Kpad8⟩
Next Section Select	⟨Shf Cmd Kpad2⟩

The Prev Section and Next Section commands may be used to navigate large files with many sections. They use the regular expression funcExprAlt (which, of course, may be modified) discussed in section 1.4.2. The Prev Section Select and Next Section Select commands select the previous or next section, that is, all the text from one `\section` command to the next, and are useful for relocating large blocks of text.

Prev Subsection	⟨Kpad8⟩
Next Subsection	⟨Kpad2⟩
Prev Subsection Select	⟨Shf Kpad8⟩
Next Subsection Select	⟨Shf Kpad2⟩

The Prev Subsection and Next Subsection commands are similar to Prev Section and Next Section except that they also stop at each `\subsection` and `\subsubsection` as well. They use the variable funcExpr discussed in section 1.4.2. In TEX mode, these commands take the place of *Alpha*'s generic Next Func and Prev Func commands, which are bound to ⟨KPAD3⟩ and ⟨KPAD1⟩, respectively, in other modes. Like Prev Section Select and Next Section Select, the Prev Subsection Select and Next Subsection Select commands select the previous or next `\section`, `\subsection`, or `\subsubsection`.

LaTeX Utilities submenu

Choose Command ⟨SHF CMD C⟩

This command has been removed from LATEX mode version 4, but will hopefully be replaced in the future.

This command provides access to each and every command on the LATEX menu via the keyboard. It's a multi-step process, where the number of steps depend on whether you're using the long or short LATEX menu: first, press ⟨SHF CMD C⟩ and choose a submenu from the list (using the arrow keys or by pressing the first letter of a submenu name). Next, choose another submenu from the list or the desired command, whichever is appropriate. Continue descending the LATEX submenus until the desired command is found.

Delete Tab Stops ⟨CMD TAB⟩
Delete Comments

The Delete Tab Stops command deletes all tab stops (bullets) from the current document (or the current selection, if there is one). The Delete Comments command deletes all *unnecessary* comments from a LATEX document (which is more difficult than you think). Using the Find dialog, the following three-step manual operation (try it!) will remove all comments from the current document:

	search string	replace string
step 1:	`^[␣\t]*%[^\r\n]*\r`	null
step 2:	`[␣\t]+%[^\r\n]*`	null
step 3:	`([^\\](\\\\)*)%[^\r\n]*`	`\1%`

The utility Delete Comments simply automates this process. Thanks to Craig Platt platt@cc.umanitoba.ca for posting this algorithm in the newsgroup `comp.text.tex`.

WARNING! The effects of Delete Comments can not be undone.

Convert Quotes
Convert Dollar Signs

If there is a selection, Convert Quotes converts all straight quotes to curved quotes (LATEX-style) within the selection; otherwise, it converts the entire document.

Plain TEX uses dollar signs to delimit math mode and displaymath mode. Since LATEX inherits most, if not all of plain TEX's functionality, dollar signs

work in LaTeX documents, too. Identical left and right delimiters are difficult to parse, however, and so any error messages will be misleading at best. That is why LaTeX has its own math mode delimiters and that's why they should be used. The Convert Dollar Signs command replaces all dollar signs in the current document (or the current selection, if there is one) with appropriate LaTeX syntax. It does this by making two passes over the code, and is therefore somewhat slow on large documents.

Compress Text/Math Menus

The Compress Text/Math Menus commands are toggleable menu items that affect the appearance of the LaTeX menu. the discussion on page 15 for details.

2.1.3 Document-related commands

Documents submenu

New Document ⟨Shf Cmd N⟩

Use this command to open a new window in TeX mode. Choose New Document or press ⟨Shf Cmd N⟩ to bring up a dialog with a pop-up menu of standard document types. This will create a new TeX window, insert a document of the requested type, and automatically run the options command (which is still on the **Documents** submenu). The old commands article, letter, etc. will be found on the **Insert Document** subsubmenu (discussed below). Each such command behaves as it did before, that is, it inserts a document template into an empty window or wraps the entire contents of the current window.

article
report
book
letter
slides
generic...

Choosing one of these document templates from the **Documents** submenu either inserts the desired template at the insertion point, or if there is a current selection, the selection is wrapped up inside the chosen template. In either case, the insertion point is positioned at the beginning of the template where the user may enter any specific document class options that may be required (standard options include `11pt`, `twoside` and `twocolumn`, for example). If none are desired, simply skip over this part of the template (it's okay to leave the square brackets empty). See the options command below for more information on document class options.

options...
usepackage ⟨Ctl Opt U⟩

The options command presents the user with a dialog box and a list of standard document class options. Choosing one of these options or typing a name into the text box of the dialog inserts the chosen option into the current document at the appropriate place. See the bug list in section 1.9 for a caveat, however.

When you insert a LaTeX document template into the current window, you get one `\usepackage` command by default. To insert another `\usepackage` immediately after the `\documentclass` command, choose the usepackage command on the **Documents** submenu or press ⟨Ctl Opt U⟩.

filecontents...
filecontents All

To facilitate file transfer, LaTeX 2_ε now has a `filecontents` environment that contains the source of a LaTeX auxiliary file or input file. Issuing this command brings up a standard file dialog. After locating the file to be included, `latex.tcl` wraps the file inside a `filecontents` environment and inserts it at the beginning of the document.

There is also a filecontents All command that scans the current document and prepends one `filecontents` environment for each custom package or class file in the current folder. Local files read by `\input` or `\include` are also attached, as well as `.bib` and `.bst` files.

Rebuild Documents Submenu

This command rebuilds the **Documents** submenu on-the-fly. It's a temporary fix until I think of a better way to handle the **Packages** submenu. The **Packages** submenu contains a list of all packages known to the TeX application. Choosing one of these packages inserts the corresponding `\usepackage` command into the preamble of the current document. To build this submenu, enable the flag buildPkgsSubmenu as described in section 1.4.1, and then choose Rebuild Documents Submenu on the **Documents** submenu.

Page Layout submenu

maketitle

LaTeX's `\maketitle` command formats a title page with information provided by the user. Choosing this command from the LaTeX menu inserts a title page template into the current document just after the `\begin{document}` command.

abstract
titlepage

The `abstract` and `titlepage` environments contain the text of an abstract and title page, respectively. The latter differs from `\maketitle` in that the user is totally responsible for the format of the title page.

pagestyle...
thispagestyle...
pagenumbering...

The pagestyle and thispagestyle commands control what appears in the header and footer of the current document. The user is presented with a list of standard formats from which to choose. The pagenumbering command is for choosing the style of the page numbers, and is also interactive.

twocolumn
onecolumn

These are simple declarations that tell LaTeX to begin formatting the output in two or one column format, respectively.

Sectioning submenu

part
chapter
section
subsection
subsubsection
paragraph
subparagraph

All LaTeX sectioning commands are available from the **Sectioning** submenu, the most common commands being the chapter, section, and subsection commands. The corresponding LaTeX command is inserted at the insertion point. The current selection, if there is one, is assumed to be the name of the section and wrapped up inside curly braces. The resulting declaration is *not* automatically followed by a carriage return since the user has the option of putting a label (or whatever) on the same line.

appendix

Unlike the other sectioning commands, this command does not have an argument. It simply tells LaTeX to start numbering differently. The `\appendix` declaration only makes sense in the context of a long document such as a book.

2.1.4 Paragraph mode commands

Text Style submenu

The following text style commands each take an argument, namely, the text to be formatted in the given style. For large amounts of text, use the corresponding declarations listed on p. 37 of the LaTeX book.

emph	⟨Ctl Opt E⟩
underline	⟨Ctl Cmd U⟩

Short for "emphasized", the emph command is perhaps the most often used LaTeX text style. If the surrounding text has the upright shape (see below), then LaTeX typesets emphasized text in italics. If the surrounding text is italicized, then emphasized text will be upright. The so-called "italic correction" is handled automatically by this command.

Although underlined text is not used much anymore, the corresponding command is included here for completeness. The underline command may also be used in math mode and therefore also appears on the **Grouping** submenu. See section 2.1.5.

textup	
textit	⟨Ctl Opt I⟩
textsl	⟨Ctl Opt S⟩
textsc	⟨Ctl Opt H⟩

These four commands specify the *shape* of their respective arguments. They call for upright text, italics, slanted text, and small caps, respectively. Upright is the default.

textmd	
textbf	⟨Ctl Opt B⟩

These commands specify an attribute called the *series* of the corresponding font. They call for medium and boldfaced text, respectively. Medium is the default.

textrm	⟨Ctl Opt R⟩
textsf	⟨Ctl Opt W⟩
texttt	⟨Ctl Opt Y⟩

The third and final component of any given font is the *family*. There are three families: roman, sans serif, and typewriter. Roman is the default.

textnormal	

Regardless of the surrounding text, the argument of `\textnormal` is typeset in the default style, that is, upright, medium, and roman.

em	⟨Shf Ctl Opt E⟩
upshape	
itshape	⟨Shf Ctl Opt I⟩
slshape	⟨Shf Ctl Opt S⟩
scshape	⟨Shf Ctl Opt H⟩
mdseries	
bfseries	⟨Shf Ctl Opt B⟩
rmfamily	⟨Shf Ctl Opt R⟩
sffamily	⟨Shf Ctl Opt W⟩
ttfamily	⟨Shf Ctl Opt Y⟩
normalfont	

These commands are the declarative counterparts of the previously mentioned text style commands. Typically, they are used for large chunks of text, say, entire paragraphs. (Note: the declarative versions do not apply an italic correction. See the LaTeX manual for usage and examples.) To access these commands, press the ⟨Shf⟩ key with the **Text Style** submenu down.

Text Size submenu

tiny	⟨Ctl Opt 1⟩
scriptsize	⟨Ctl Opt 2⟩
footnotesize	⟨Ctl Opt 3⟩
small	⟨Ctl Opt 4⟩
normalsize	⟨Ctl Opt 5⟩
large	⟨Ctl Opt 6⟩
Large	⟨Ctl Opt 7⟩
LARGE	⟨Ctl Opt 8⟩
huge	⟨Ctl Opt 9⟩
Huge	⟨Ctl Opt 0⟩

These commands declare the text font size. They affect the entire document unless surrounded by braces, so the menu commands automatically insert braces. If you want the entire document set in a certain font size, insert a class option with the options command (see section 2.1.3).

International submenu

`latex.tcl` implements about half of LaTeX's full palette of international symbols and accents (if you can think of ways to get the rest of these on the LaTeX menu, please let me know!). See Tables 3.1 and 3.2 on pp. 38–39 of the LaTeX book for a complete list.

Environments submenu

itemize...	⟨Opt F7⟩
enumerate...	⟨Shf Opt F7⟩
description...	⟨Ctl Opt F7⟩
thebibliography...	
slide	⟨Opt F8⟩
overlay	⟨Shf Opt F8⟩
note	⟨Ctl Opt F8⟩
figure	⟨Opt F9⟩
table	⟨Shf Opt F9⟩
tabular...	⟨Ctl Opt F9⟩
verbatim	⟨Opt F10⟩
quote	⟨Shf Opt F10⟩
quotation	⟨Ctl Opt F10⟩
verse	
center	⟨Opt F11⟩
flushleft	⟨Shf Opt F11⟩
flushright	⟨Ctl Opt F11⟩
general...	⟨Opt F12⟩

One of the most useful of `latex.tcl`'s many features is its ability to insert skeletal templates for multi-line environments (that is, LaTeX constructs delimited by a `\begin...\end` pair). These may be inserted anywhere in the document (even in the middle of a line), complete with tab stops and appropriate indentation. In some cases (like itemize), the user is asked to specify the number of rows desired, after which the program generates the corresponding environment body complete with indentation and tab stops. Some environment commands (like tabular) also prompt the user for the desired number of columns. There's even a general command for inserting user-defined environments on-the-fly.

The figure command deserves special mention. Choosing this command from the **Environments** submenu (or by pressing ⟨Opt F9⟩) brings up a dialog with a pop-up menu of box-making macros, one for every macro name stored in the TeX mode variable boxMacroNames (see section 1.4.2). With the mouse or arrow keys, choose one of these macro names and click "OK" to insert the corresponding `figure` environment at the insertion point, or leave the text box blank to wrap a `figure` environment around the current selection (if there is one). If only one macro name is stored in boxMacroNames, the dialog is automatically circumvented and the `figure` environment is inserted at the insertion point without prompting.

Note: The **Environments** submenu seeks to mimic the corresponding command keys. Each group of environments on this submenu has been assigned a different function key, beginning with ⟨F7⟩. The general environment, for instance, is bound to ⟨Opt F12⟩. See section 3.1 for more information.

Boxes submenu

mbox ⟨Ctl Opt M⟩
makebox
fbox
framebox

Perhaps the most useful box-making command is `\mbox`, which formats its argument in LR mode, a restricted form of paragraph mode impervious to line breaks. The `\mbox` command is especially useful for inserting a bit of plain text in the middle of a math formula (see the LaTeX book for examples). The `\makebox` command is a generalized form of `\mbox`, which takes the width and height of the box as additional arguments.

The commands `\fbox` and `\framebox` are analogous to `\mbox` and `\makebox` except that a rectangular frame is drawn around the box.

newsavebox
sbox
savebox
usebox

A "savebox" is a bin for storing text, graphics, formulas, or whatever. The argument to `\sbox` or `\savebox` is typeset *once* and may be recalled later, any number of times, via `\usebox`.

raisebox

This box-making command takes a vertical offset as one of its arguments.

parbox
minipage

The primary argument of LaTeX's `\parbox` command or `minipage` environment is typeset in paragraph mode. `\parbox` is for small amounts of text, while the `minipage` environment is for large blocks of text.

rule

The `\rule` command makes a box filled with ink. For example,

```
\newcommand{\filledsquare}{\rule[0.125ex]{1.3ex}{1.3ex}}
```

makes a black square approximately the same size as LaTeX's open `\Box`. (There is an analogous command called `\blacksquare` defined in the AMS symbol package.)

Miscellaneous submenu

verb	⟨Ctl Opt V⟩
footnote	⟨Ctl Opt F⟩
marginal note	⟨Ctl Opt N⟩

All the above commands wrap the current selection, if there is one.

label	⟨Ctl Opt L⟩
ref	⟨Ctl Opt X⟩
pageref	⟨Ctl Opt P⟩
cite	⟨Ctl Opt C⟩
nocite	⟨Shf Ctl Opt C⟩

These commands do more than simply insert the corresponding LaTeX command. For instance, press ⟨Ctl Opt X⟩ or ⟨Ctl Opt P⟩ to insert a `\ref` or `\pageref` command, respectively. The inserted command will contain the argument of the nearest `\label` command. Continue pressing ⟨Ctl Opt X⟩ or ⟨Ctl Opt P⟩ to cycle through all the `\label` commands in your document.

item	⟨Ctl Opt J⟩

Simply press ⟨Ctl Opt J⟩ inside an `itemize`, `enumerate`, `description`, or `thebibliography` environment to insert an item of the appropriate type at the insertion point.

quotes	⟨Ctl Opt '⟩
double quotes	⟨Shf Ctl Opt '⟩

The `latex.tcl` macro package incorporates a "smart quotes" feature originally implemented by an unknown author (see the code in `latexSmart.tcl`) that makes the typing of quoted material totally transparent. Just use the quote key as you would for plain text files. Consequently, the quotes and double quotes commands are primarily used for quoting existing text.

ellipsis
en-dash
em-dash
TeX logo
LaTeX logo
Latex2e logo
date

These are a few of the text-related LaTeX commands that I've found useful from time to time.

dag
ddag
section mark
paragraph mark
copyright
pounds

The previous six commands may be used in any mode, including math mode.

2.1.5 Math mode commands

Math Mode submenu

TeX math	
TeX displaymath	
LaTeX math	⟨Ctl Cmd M⟩ or ⟨Ctl Cmd 4⟩
LaTeX displaymath	⟨Ctl Opt Cmd M⟩ or ⟨Ctl Opt Cmd 4⟩

Math mode may be invoked in a number of ways. Many TEXnical typists rely exclusively on TEX's use of dollar signs and almost always key in their documents horizontally from left to right. Others have adopted LATEX's tendency to prefer vertical constructions (environments). Still others have settled on some combination of these, using whichever seems comfortable or convenient at the time. Whatever your approach to mathematical typesetting, there's something for everybody in `latex.tcl`, designed to simplify the input of complex mathematical formulas.

Four math modes are available for normal, left-to-right input. These are called TeX math `$...$` and TeX displaymath `$$...$$`, along with their corresponding LATEX equivalents called LaTeX math `\(...\)` and LaTeX displaymath `\[...\]`. The LATEX versions are logically equivalent to the multi-line `math` and `displaymath` environments (see below). The latter have the advantage that 1) they are often more readable in source form, and 2) they are more easily changed (by simply replacing keywords) as the document evolves.

NOTE: The above command keys automatically switch from LaTeX math and LaTeX displaymath to TeX math and TeX displaymath, respectively, when the flag useDollarSigns is set to true.

Tip: Get into the habit of pressing ⟨Ctl Cmd M⟩ or ⟨Ctl Opt Cmd M⟩ when composing in-line equations, since there is less chance of inadvertantly omitting a dollar sign if you do.

Math Style submenu

mathit	⟨Ctl Opt Cmd I⟩
mathrm	⟨Ctl Opt Cmd R⟩
mathbf	⟨Ctl Opt Cmd B⟩
mathsf	⟨Ctl Opt Cmd W⟩
mathtt	⟨Ctl Opt Cmd Y⟩
mathcal	⟨Ctl Opt Cmd C⟩
displaystyle	⟨Ctl Opt Cmd D⟩
textstyle	⟨Ctl Opt Cmd T⟩
scriptstyle	⟨Ctl Opt Cmd S⟩
scriptscriptstyle	

The next submenu on the LATEX menu is called **Math Style**, with commands for math italic, roman, boldface, sans serif, typewriter and calligraphic typefaces,

as well as declarations for `\displaystyle`, `\textstyle`, `\scriptstyle`, and `\scriptscriptstyle`. The latter command quartet are sometimes needed to override LaTeX's default math style. (The `array` environment, for example, insists on enabling `\textstyle` regardless of the surrounding environment.)

Math Environments submenu

math	⟨Opt F5⟩
displaymath	⟨Shf Opt F5⟩
equation	⟨Ctl Opt F5⟩
eqnarray*	⟨Shf Opt F6⟩
eqnarray	⟨Ctl Opt F6⟩
array	⟨Opt F6⟩
general	⟨Opt F12⟩

Besides the `math` and `displaymath` environments discussed in section 2.1.5, other multi-line math environments (`equation`, `array`, `eqnarray`, and `eqnarray*`) are also available. Each is mutually exclusive (that is, one may not be nested inside the other) except for the `array` environment which *must* be nested inside some other math environment. (It took me a long time to come to grips with this apparent anomaly). There's also a general environment command, which is exactly the same command found on the **Text Style** submenu.

Formulas submenu

subscript	⟨_⟩ (if smartScripts is true)
superscript	⟨^⟩ (if smartScripts is true)
frac	⟨Ctl Cmd F⟩
sqrt	⟨Ctl Cmd R⟩
nth root	
one parameter	⟨Ctl Cmd 1⟩
two parameters	⟨Ctl Cmd 2⟩

The **Formulas** submenu contains LaTeX commands commonly used to build up even the simplest mathematical expressions. There are commands for typesetting subscripts and superscripts, fractions (which used to be difficult to typeset), square roots, and arbitrary nth roots. There are also one and two-parameter LaTeX commands, which allow the user to type in a command name on-the-fly. Next to `latex.tcl`'s environment commands, the formula commands are most useful. (In fact, it pays to memorize their command key equivalents.)

While we're talking about the **Formulas** submenu, let me say a little bit about `latex.tcl`'s ability to parse fractions. How many times have you found yourself wanting to recast a horizontally typeset fraction such as

```
$x = (-b \pm \sqrt{b^2 - 4ac})/(2a)$
```

in a corresponding "vertical" form

```
$$x = \frac{(-b \pm \sqrt{b^2 - 4ac})}{(2a)}$$
```

Obviously, such an operation involves a lot of cutting and pasting, and I used to avoid it like the plague. Well, now all you have to do is select the text you want converted (in this case, all the text inside the dollar signs except "`x =` ") and then choose the frac command from the **Formulas** submenu on LaTeX menu. The rest is automatic. (Now if only I could get it to automatically remove the redundant parentheses...)

Greek submenu

One of the longest of `latex.tcl`'s submenus contains the entire Greek alphabet, including both lower and upper-case letters (hold down the ⟨Opt⟩ key while the **Greek** submenu is down to see the latter), plus a handful of lower-case "italicized" letters (`\varepsilon`, `\vartheta`, `\varpi`, `\varrho`, `\varsigma`, and `\varphi`). To type a **Greek** command at the keyboard, press ⟨Ctl M⟩ ⟨letter⟩, where ⟨letter⟩ is the same key assigned to that letter by the Macintosh Symbol font. See the file `latex_bindings.tex` for a useful summary.

NOTE: There are two **Greek** submenus. While one is down, press a modifier key (such as ⟨Opt⟩) to see the alternate menu.

Binary Operators and Relations submenus

Plain TeX defines an incredible variety of mathematical symbols, each transparently available to the LaTeX user. All of these symbols have been implemented in this version of `latex.tcl`.

There are two **Relations** menus. While one is down, press a modifier key (such as ⟨Opt⟩) to see the alternate menu.

Arrows, Dots, and Symbols submenus

A quick glance at the LaTeX book shows a wide assortment of arrows, dots, and miscellaneous mathematical symbols. Starting with v2.2, all of these have been implemented in `latex.tcl`. See the **Arrows**, **Dots**, and **Symbols** submenus for exhaustive lists of available commands.

NOTE: There are two **Arrows** menus. While one is down, press a modifier key (such as ⟨Opt⟩) to see the alternate menu.

Functions submenu

All of TeX's so-called "log-like" functions (`\exp` and `\sin`, for instance) have been implemented in this version of `latex.tcl`. Some of these commands (lim, inf, sup, liminf, limsup, max, and min) automatically insert a subscript. Only lim has a command key, namely, ⟨Ctl Cmd L⟩.

Large Operators submenu

sum	⟨Ctl Cmd S⟩
prod	⟨Ctl Cmd P⟩
coprod	
int	⟨Ctl Cmd I⟩
oint	
bigcup	
bigcap	
bigsqcup	
bigvee	
bigwedge	
bigodot	
bigotimes	
bigoplus	
biguplus	

The `latex.tcl` macro package also provides support for TEX's so-called "large operators". Commands such as sum ⟨Ctl Cmd S⟩, prod ⟨Ctl Cmd P⟩, int ⟨Ctl Cmd I⟩, bigcup, bigcap, bigvee, and bigwedge may be found on the **Large Operators** submenu.

Delimiters submenu

parentheses
brackets
braces
vertical bars
other delims...
half-open interval
half-closed interval
big parentheses
big brackets
big braces
big vertical bars
other big delims...
big left brace
other mixed big delims...

TEX is particularly adept at "delimiting" arbitrary-sized mathematical expressions. Examples include parenthesized equations, matrices, and determinants. Since the left and right delimiters need not be of the same type, there are a host of options from which to choose, which presents an interesting design problem. A workable compromise was achieved by implementing a handful of common delimiters explicitly, and then providing access to other more esoteric combinations via dialogs. Consequently, commands for big parentheses, big brackets, big braces, and big vertical bars (i.e., absolute value signs) will be found on the

Delimiters submenu, along with a big left brace (commonly used to define multi-part functions or systems of equations), as well as commands called other big delims and other mixed big delims. The latter two commands are interactive—the user either types the delimiter name directly into a text box or chooses the desired name from a pop-up menu of available options. Also on the **Delimiters** submenu are normal-sized parentheses, brackets, braces, vertical bars, and other fixed-size delimiters.

multi-line big parentheses
multi-line big brackets
multi-line big braces
multi-line big vertical bars
other multi-line big delims...
multi-line big left brace
other multi-line mixed big delims...

All of the big delimiters have multi-line counterparts (i.e, a vertical, as opposed to a horizontal construct). To access these commands, press the ⟨Opt⟩ key while the **Delimiters** submenu is down.

Math Accents submenu

acute	⟨Ctl Cmd A⟩
bar	⟨Ctl Cmd B⟩
breve	
check	⟨Ctl Cmd C⟩
dot	⟨Ctl Cmd D⟩
ddot	
grave	⟨Ctl Cmd G⟩
hat	⟨Ctl Cmd H⟩
tilde	⟨Ctl Cmd T⟩
vec	⟨Ctl Cmd V⟩
widehat	
widetilde	
imath	
jmath	

Math accents (not to be confused with diacritical marks used in paragraph mode) are accessed from a submenu of the same name. There are commands for hats, bars, tildes, vectors, dots, etc., plus wide hats and tildes, most of which have command keys. There are also commands for dotless versions of the letters "i" and "j" used in conjunction with these accents. Insofar as possible, the macros check to make sure that only single characters are being accented, or in the case of wide accents, three or fewer characters.

Grouping submenu

underline	⟨Ctl Cmd U⟩
overline	⟨Ctl Cmd O⟩
underbrace	⟨Ctl Opt Cmd U⟩
overbrace	⟨Ctl Opt Cmd O⟩
overrightarrow	
overleftarrow	
stackrel	

The **Grouping** submenu has commands for underlining and overlining, and related commands that produce underbraces and overbraces. There's also a command called stackrel used to construct compound operators via vertical stacking (see p. 50 of the LaTeX book for more information).

Spacing submenu

neg thin
thin
medium
thick
quad
qquad
hspace
vspace
hfill
vfill
smallskip
medskip
bigskip

The **Spacing** submenu provides for various types of horizontal and vertical spacing. There are commands for negative thin, thin, medium, and thick amounts of whitespace, and additional commands for inserting the traditional typesetter's quad (1em) and double quad. Arbitrary horizontal whitespace, defined via LaTeX's `\hspace` command, and vertical whitespace via `\vspace`, may also be inserted from the **Spacing** submenu. LaTeX's "fill" commands will also be found on this submenu, as well as `\smallskip`, `\medskip`, and `\bigskip`.

2.2 The **funcs** menu

The pop-up menu activated by pressing the "{}" icon on the tool bar at the right of each window is called the **funcs** menu. In TeX mode (and other modes as well), the **funcs** menu gives an outline of the current document and provides a way to quickly navigate a long file. Simply press the "{}" icon to build the **funcs** pop-up menu on-the-fly. The current document will be scanned and the titles of all sections and subsections will be placed on the menu.

2.3 The **mark** menu

The pop-up menu activated by pressing the "M" icon on the tool bar at the right of each window is called the **mark** menu. In TEX mode (and other modes as well), the mark menu gives an outline of the current document and provides a way to quickly navigate a long file. Simply press the "M" icon and choose the Mark File command. The contents of the current document will be scanned and the titles of all chapters, sections, and subsections will be placed on the menu. Files that are `\include`'d or `\input`'ed will also appear on the **mark** menu. Note that the mark menu is static, that is, if you change the structure of the current document, you must choose the Mark File command again.

Note: Currently, there is considerable overlap between the **funcs** menu and the **mark** menu in terms of functionality. This will change in future versions of *Alpha*.

Chapter 3

Command Keys

Menus are great at first, but eventually the tendency is to move away from menus towards commands keys. This can significantly speed the input process. Few of us, however, are inclined to memorize more than a couple dozen such keystrokes unless continually prompted with reminders. Thus most command key equivalents are displayed on the LaTeX menu in full view.

3.1 Tips

A few remarks will help you remember the many command key sequences. All paragraph mode commands (`\textbf`, `\footnote`, etc.) begin with ⟨Ctl Opt⟩ or ⟨Shf Ctl Opt⟩, whereas all math mode commands begin with ⟨Ctl Cmd⟩ or ⟨Ctl Opt Cmd⟩. All environments are bound to some modified function key. Sometimes the ⟨Shf⟩ key reverses the orientation of an existing key (as in the case of Next Tab Stop and Prev Tab Stop described in section 2.1.2) or acts as a selection key (see, for example, Next Environment and Next Environment Select in section 2.1.2). Knowing these simple facts helps tremendously.

See section 1.2 for pointers to useful command key summaries.

3.2 Double-clicking

The arguments of certain LaTeX commands are command-double-clickable, that is, you hold down the ⟨Cmd⟩ key while double-clicking the argument of certain LaTeX commands. These commands are underlined and therefore easily recognized in your document. When you command-double-click the required argument of a `\ref` command, for example, the cursor jumps to the corresponding `\label`. (Note: Press ⟨Ctl .⟩ to return to the original cursor position.) Similarly, when you cmd-dbl-click the required argument of a `\cite` command, the cursor jumps to the corresponding `\bibitem` if the document contains a `thebibliography` environment; otherwise, the arguments of a `\bibliography`

command are sequentially searched until the `.bib` file containing the target item is found. This `.bib` file is then opened and the cursor jumps to the target item.

Tip: By default, `\ref`, `\pageref`, `\cite`, and `\nocite` commands are cmd-dbl-clickable. If you use a package that defines other `\ref`-like or `\cite`-like commands, modify the TEX mode variables refCommands or citeCommands discussed in section 1.4.2.

Other LATEX commands may also be command-double-clicked, for example, `\input`, `\include`, `\includegraphics` (or rather the commands specified in the TEX mode variable boxMacroNames mentioned in section 1.4.2), `\bibliography`, `\usepackage`, and `\documentclass`. Note that the required arguments of these commands are files, and so command-double-clicking such an argument opens the corresponding file. Unless the filename includes a Macintosh path (which is not recommended, since it's not portable), the current folder is searched first. If the file is not found in the current folder, the algorithm next checks the hierarchy of folders under the user-specified "TeX Inputs Folder", which is optionally set by choosing App Paths on the **Config** menu. If the file is still not found, all folders whose name contains the string "inputs" in the TEX application folder are checked next. For example, all folders in the TEX folder with names such as `TeX-inputs`, `TeX-inputs2`, and `My-TeX-inputs` will be searched.

Tip: A command-double-click operation may be simulated with a keystroke. With the cursor inside the required argument of a cmd-dbl-clickable LATEX command, press ⟨F6⟩ to activate the algorithm.

Alpha has another modified double-click that will be of interest to LATEX users. You may already know that double-clicking a delimiter (parenthesis, bracket, or brace) selects the text between it and its matching delimiter. Moreover, if you hold down the ⟨Ctl⟩ key while double-clicking a delimiter, the text *and* the delimiters will be selected. These commands are very handy for cutting and pasting blocks of delimited text, especially in a LATEX document where braces, for example, run rampant.

Chapter 4

Technical tips

Tip: To see what TEX applications are currently supported for typesetting, viewing, or printing, type

```
array names texAppSignatures
```

or

```
array names viewDVIAppSignatures
```

or

```
array names printDVIAppSignatures
```

respectively, in the Tcl shell. (To invoke the shell, choose the Shell command from *Alpha*'s **File** menu or press ⟨CMD Y⟩.)

Tip: To see what applications are currently supported for creating, viewing, or printing `.ps` files, type

```
array names dvipsAppSignatures
```

or

```
array names viewPSAppSignatures
```

or

```
array names printPSAppSignatures
```

respectively, in the Tcl shell.

Tip: To see what BIBTEX or *MakeIndex* applications are currently supported, type

```
array names bibtexAppSignatures
```

or

```
array names makeindexAppSignatures
```

respectively, in the Tcl shell.