About TeXShop

Richard Koch

September 3, 2006

Contents

1 Basic Help

1.1 Preliminaries

TeXShop is a Mac OS X program which can typeset TeX and LaTeX files and preview the output on the screen. The program usually typesets using pdftex or pdflatex instead of tex or latex. These programs output pdf files rather than dvi files. Since pdf is the native graphics format of Mac OS X, it is easy to display the output on the screen. Pdftex and pdflatex were written by Han The Thanh, Petr Sojka, and Jiri Zlatuska.

The pdf files created by TeXShop are standard pdf files which can be given to other people and displayed on a variety of computer systems. For example, they can be displayed by Adobe Acrobat.

TeXShop does not include TeX. Instead it uses the standard TeX distribution teTeX, a wonderful collection of the entire suite of TeX programs, files, and utilities created by Thomas Esser. This collection (the standard version of TeX on Linux systems) has been compiled for Mac OS X by Gerben Wierda. Instructions for obtaining it are given below.

1.2 Getting and Installing teTeX

The MacTeX working group of the TeX Users Group (TUG) has produced an Apple install package containing everything needed to run TeX on Mac OS X. The package installs in a couple of minutes with almost no user intervention. It contains a very complete version of Gerben Wierda's TeX redistribution, TeXShop, BibDesk, the Excalibur spell checker, and Gerben's i-Installer. For up to date links to the package, see

http://www.uoregon.edu/∼koch/texshop/obtaining.html

You can also obtain Gerben Wierda's TeX redistribution separately (this is the distribution contained in the above package). The latest version of this distribution is compiled for both Intel and PowerPC processors and requires system 10.3 or higher. It is based on the latest teTeX for MacOSX and includes updated programs from the TeX User's Group's distribution TeXLive.

To obtain these files and programs, go to Wierda's site

http://www.rna.nl/tex.html

and obtain i-Installer, the TeX installer. You'll find it in the downloadable package II2.dmg. Run the installer and install the following packages in the following order:

- FreeType 2
- libwmf
- GNU gettext libraries
- PNG Library
- ImageMagick
- Ghostscript 8
- FontForge
- Fondu Mac Font cli Tools
- TeX

The MacTeX package installs some other optional packages which you can also obtain with i-Installer:

- CB Greek
- CM Super
- MusixTeX
- ConTeXt updater
- LaTeX updater
- XeTeX (this is in Jonathan Kew's directory)

More detailed installation instructions can be found on Wierda's site, and on the TeXShop web site listed later.

1.3 Getting and Installing TeXShop

If you installed TeX with the MacTeX install package, you already have TeXShop. In that case, you only need read the rest of this section if you had an earlier version of TeXShop before using the MacTeX install package. The earlier TeXShop was probably installed in /Applications, but MacTeX installs it in /Applications/TeX. Remove the earlier version so this new version will be used.

To get TeXShop from scratch, go to

www.uoregon.edu/∼koch/texshop/texshop.html

and click the TeXShop link to download the file TeXShop.dmg. Double click on this file; a folder will appear containing TeXShop and other files which can be read or discarded. Drag TeXShop to the Applications folder. If you intend to use it often, drag its icon to the Dock.

TeXShop has a feature called pdfsync: clicking on a spot in the preview window activates the corresponding source window with the appropriate source line selected. This feature requires that files named "pdfsync.sty," "pdfsync.tex", and "pdfsync4context.tex" be installed in teTeX. This is done automatically if you install from the MacTeX install package. Otherwise, find these files in the TeXShop distribution and drag them to ~/Library/texmf/tex/latex, ~/Library/texmf/tex/plain, and ~/Library/texmf/tex/context. Here ~/Library is the Library folder in your home directory. You may have to create some or all of the folders texmf, tex, latex, and plain.

TeXShop 1.35 came with additional applescripts by Will Robertson and Claus Gerhardt. You will have these automatically if you installed a later version of TeXShop for the first time. But if you upgraded from a version earlier than 1.35, you will not automatically see those new scripts.

If you never edited the default macro set, you can obtain the new macros by going to the folder ~/Library/TeXShop and removing the entire Macros folder inside. The next time TeXShop starts, it will recreate this folder with the latest macros. It is not enough to remove the contents of the Macros folder; the entire folder must be removed.

If you have edited the default macros and want to add the new macro set, remove the Macros folder temporarily and let TeXShop create a new macro set. Then add your old macros using the Macro Editor.

TeXShop 1.35 came with new templates by Will Robertson. These templates will not automatically be installed. You can obtain them by going to the folder ˜/Library/TeXShop and moving the entire Templates folder inside to the Desktop. The next time TeXShop starts, it will recreate this folder with the latest Templates. You can then move old Templates you have edited from the folder on the desktop to the newly created folder ˜/Library/TeXShop/Templates.

1.4 Typesetting Documents

To use TeXShop, type your LaTeX input in the editing window it provides. Then push the ¨Typeset¨ button at the top of the window. The input will be saved and pdflatex will run. A second window will open displaying messages from tex. If there are errors, pdflatex will halt; type ¨return¨ to skip the errors one by one, or one of the standard TeX inputs to quit, run continuously, etc. After the document has been typeset, a new window will appear showing the resulting pdf file. You can switch between the two windows by typing command-1.

Select the magnify icon in the pdf window's toolbar, and press the mouse on a section of the pdf output display to magnify the region near the cursor. Double click to magnify a larger region; triple click to magnify a still larger region. Notice that the default region size can be selected using an element of the toolbar at the top of the page.

To typeset a TeX document, push the pulldown menu labeled "LaTeX" to the right of the "Typeset" button, and select TeX instead of LaTeX. If you usually use TeX, choose it as your default program in the Preferences Dialog.

If you notice an error, you can immediately fix it in the editing window and push the "Typeset" button again without halting the first invocation of pdflatex. The program will kill that invocation and run another.

TeXShop remembers the source lines of the first twenty errors. To cycle through these lines in the source file, choose"Go To Error" in the Edit menu or type the keyboard shortcut for this menu item.

When you fix errors and typeset again, the viewer will remember the page you were previously viewing.

If a line in the source begins with the characters "%:", TeXShop interprets the remaining word or words on the line as a ¨tag¨ and adds these words to the Tag pulldown menu. Choosing a tag entry in this menu will scroll to the appropriate line in the source file. Lines which begin with the words \section, \subsection, \subsubsection, or \chapter are automatically added to the Tag menu. However, this behavior can be changed by typing the following command in Apple's Terminal program:

```
 defaults write TeXShop TagSections NO
```

TeXShop can show multiple documents. If you choose “New” or “Open”, your original document will remain and additional windows for the new document will open. If you open a TeX file in a folder which contains a pdf file with the same name, this pdf will also appear.

Some users like to divide their source into multiple files controlled by a master file using an \input command. TeXShop always saves the source file before typesetting. If the preference item “During File Save, Save Related Files” is checked, the master file will be searched before typesetting and any input file open in TeXShop will also be saved. This feature is due to John Nairn.

You can start TeXShop by double clicking on a document with extension“.tex”. The program will open that input document and (if it exists) the associated pdf document.

You can close the pdf window at any time. It will reappear when the document is typeset again. If you close the input window then both the input and output windows will close (windows from other TeX documents will remain open). Therefore, if an input source window is cluttering the screen, hide it instead of closing it.

pdftex and pdflatex are unix programs derived from Knuth's TeX program. Until recently, TeX would not process files whose names contained spaces. If you have collaborators working on other machines, you may wish to avoid spaces in filenames. However, the TeX in Gerben Wierda's distribution allows spaces in filenames, and TeXShop itself has never had problems with spaces in filenames.

The preview window's behavior can be changed using the "Magnification" and "Display Format" items in the Preview menu. For example, you can configure the scroller to scroll through all of the document pages; the preview window can show two pages at a time; resizing the window can be made to change the preview magnification. Experiment until you find the mode you prefer, and then select this mode in preferences to make it permanent.

Use the magnification tools in the Preview window's toolbar to magnify portions of text. By double or triple clicking before holding down the mouse, a larger portion of the text will be magnified. The magnified region can be changed during magnification by pushing the apple, option, control, and shift keys.

Use the pdf selection tool to select a portion of the preview window and drag it to the desktop or another program's window. The selection can also be copied and then pasted into another document. By default, this selected region contains only foreground text in pdf format. This is useful, for example, when dragging text to Keynote; the slide's background will show behind the text and the text can be resized without losing clarity.

Use the text selection tool to select text in the preview window. This text can be copied and then pasted to a word processor. If the preview contains links, these links can be activated by clicking with the text selection tool. Use the Back/Forward commands to return to the original page. Use the Find command in the window's drawer to search for text in the pdf document.

You can jump between the source and preview windows by holding down the Apple command key while clicking on a word or phrase. If the phrase is in the source window, the preview window will scroll to the appropriate spot and the typeset phrase will be circled in red. If the phrase is in the preview window, the source window will open (if necessary) and scroll to the appropriate spot, and the source phrase will be highlighted in yellow.

1.5 Alternate Typesetting Mode

There is another way to typeset with TeXShop; in this alternate mode, eps illustrations can be input directly without conversion. To use the alternate method, choose "TeX and Ghostscript" in the Typeset menu. Then TeXShop will typeset by calling tex or latex to produce a ".dvi" file, calling dvips to convert it to a postscript file, and calling ps2pdf to convert the postscript file to pdf. The method chosen in the typeset menu will only affect the topmost file; other documents will continue to be typeset with pdftex or pdflatex. The primary method used when a document is first opened can be selected in the preference dialog.

The"TeX and Ghostscript" method should be used for old projects with many eps illustrations, and for TeX files with postscript special commands, and for TeX files that include bitmapped fonts which do not display correctly when typeset with pdflatex. Experimentation will show which is the preferable typesetting engine.

There is a way to permanently set the typesetting method of a document regardless of preference choices. If one of the first twenty lines of the source file is

```
%!TEX TS-program = tex
```

then tex + ghostscript will be used. If one of the first twenty lines is

```
%!TEX TS-program = latex
```

then latex + ghostscript will be used. If one of the lines is

```
%!TEX TS-program = pdftex
```

or

```
%!TEX TS-program = pdflatex
```

then pdftex or pdflatex will be used. If one of the lines is

```
%!TEX TS-program = personaltex
```

or

```
%!TEX TS-program = personallatex
```

then the personal script will be used as set in the preference dialog.

1.6 Checking Spelling

Cocoa programs automatically inherit spell checking technology by Apple.

TeXShop supports continuous spell checking, which can be toggled on or off with a menu command. A preference item selects the initial position of this toggle.

Aspell is ¨a more intelligent Ispell¨ for Unix machines written by Kevin Adkinson. His program has been ported to Mac OS X and made into a spelling service by Anton Leuski. This service installs and then is available to all Mac OS X programs as an “alternate dictionary” for Apple’s spelling services. The new speller supports LaTeX, recognizing that words like “documentclass” are valid LaTeX commands which should not be flagged as misspelled. To obtain it, go to

http://cocoaspell.leuski.net/

An alternate LaTeX spell checker named Excalibur is often used. This spell checker was written by Rick Zaccone; it can open TeXShop files. To obtain it, go to

http://www.eg.bucknell.edu/∼excalibr/excalibur.html

1.7 Latex Panel

Under the Windows menu, there is an item named ¨Latex Panel...¨ Choosing this item will open a panel containing a large number of mathematical symbols, Greek letters, international characters, and typesetting environments. Clicking on a symbol will insert the corresponding text into your LaTeX source file. This wonderful panel is the work of Geoffroy Lenglin, who can be reached at geoffroy.lenglin@m4x.org.

Some symbols in the panel require the line

```
\usepackage{amssymb}
```

at the top of the source file.

It is possible to customize the behavior of the panel. When TeXShop first runs, it creates a file "/Library/TeXShop/LatexPanel/completion.plist" inside your personal Library folder. This file is an ordinary text file which can be opened and edited by TeXShop. The file contains all strings inserted into the source code when a Palette item is chosen. If such a string contains #SEL#, then the current selection will replace this expression in the inserted string. If a string contains #INS#, then the cursor will be set to this position when the string is inserted. To customize panel behavior, edit this file.

Editing plist files is slightly tricky; see the item about them at the end of this help file. Be sure to edit and save in UTF-8 format if you use Unicode characters.

Users can add up to sixteen additional items to the Latex Panel by editing completion.plist. These are displayed on the "Custom" subpanel of the Latex Panel. See comments in completion.plist for details about adding such items.

1.8 Matrix Panel

The Windows menu contains an item "Matrix Panel...". Choosing this item will open a panel allowing easy creation of matrices. Only a small space is provided for each matrix entry, but larger items can be edited elsewhere and pasted into the matrix. This wonderful panel is the work of Jonas Zimmermann, who can be reached at zimmerleut@gmx.de.

In TeXShop 1.35 or above, the Matrix Panel can also create tables. The panel can be configured using the file "matrixpanel_1.plist" in ~/Library/TeXShop/MatrixPanel. The structure of this file was revised in 1.35; an old "matrixpanel.plist" file may remain from TeXShop 1.34.

There is a hidden preference item to set the default size of matrices created by the panel. To change the default

```
defaults write TeXShop matrixsize 12
```

1.9 Macros

There is an alternate mechanism to insert commonly used TeX commands in your source document. Find the command in the Macro menu or the Macro button on the toolbar and choose it. In addition, some macros run AppleScript commands. A Macro Editor is provided which allows you to examine current macros, modify them, and add your own macros. For details, see the "About Macros" document in the Help menu. The TeXShop Macro commands and the Macro Editor were created by Mitsuhiro Shishikura. Default macros were created by Mitsuhiro Shishikura and Hirokazu Ogawa.

Macros are stored in the file ~/Library/TeXShop/Macros/Macros.plist. This is an ordinary text file which can be copied and sent to others; thus you can distribute macros you have created. The Macro Editor has a command to read a plist file and add macros in it to existing macros.

1.10 Toolbar and Applescript

The selection and location of tools at the top of the TeX Source and PDF Preview windows can be modified using the menu item "Customize Toolbar."This toolbar support is entirely the work of Anton Leuski. Thanks!

TeXShop supports Applescript. This is also the work of Anton Leuski.

1.11 Including Graphics

The programs pdftex and pdflatex can use graphic files produced in pdf, jpg, png, or mps format. If you are using the default latex template and installed the graphic conversion packages from Gerben Wierda's distribution, you can also use graphic files produced in eps or tif format; they will automatically be converted to pdf or png formats during typesetting. One peculiarity is that tiff files must have extension "tif" rather than "tiff". The native graphics format of Mac OS X is pdf (portable document format) and such files print well at any size. It is likely that most future Mac graphics programs will output pdf.

If you used TeX in the past, your illustrations may be in eps format. These files must be converted to pdf format before being typeset with pdftex and pdflatex. As explained above, this will happen automatically if you use the default latex template. You can also convert an eps illustration by opening it in TeXShop. The illustration will appear in a graphic window and TeXShop will simultaneously write the corresponding pdf file to disk. Ghostscript also contains a command line program to convert; indeed TeXShop calls this program or Apple's distill program depending on a choice in TeXShop Preferences. To convert myfile.eps to myfile.pdf within Terminal, type

```
epstopdf myfile.eps
```

The authors of pdflatex and the authors of the graphics package graphicx have made it easy to include graphic files in a LaTeX document typeset with pdflatex, even if the document will later be typeset by standard latex and converted to a dvi file for distribution to other people. At the top of such a LaTeX input file, include the line:

```
\usepackage{graphicx}
```

When you wish to include a graphic file, say "f1.pdf", use the command

```
\includegraphics[width=2in]{f1}
```

This command will cause tex to input the graphic file ¨f1.pdf¨ when the text is typeset with pdflatex, but input the file ¨f1.eps¨ when the text is typeset with latex.

If you upgraded from a previous version of TeXShop and want to use automatic conversion of eps and tif files during typesetting, make certain that you have installed Ghostscript 8 and ImageMagick with Gerben Wierda's installer, the pdflatex program preference is “pdflatex –shell-escape”, and the LaTeX header contains the following lines

```
\usepackage{graphicx}
\usepackage{epstopdf}
\DeclareGraphicsRule{.tif}{png}{.png}{‘convert #1 ‘basename #1 .tif‘.png}
```

1.12 Printing

To print a TeX output file, select the “Print” menu item. To print TeX source, select the “Print Source” menu item.

1.13 Setting Preferences

Several items can be changed using the Preference panel: the default font for the input window, the default magnification for the output window, and the default position of these windows when they first appear. You can configure the console so it always appears when typesetting, or only appears when there is an error. You can configure the source window so an initial click in the window only activates the window, or this initial click also sets the text insertion point.

The preference dialog can also be used to change the commands executed when the “TeX” and “LaTeX” buttons are pushed. The substituted command must produce a pdf output file. If the new programs are in the teTeX binary directory, it is enough to name them in the preference dialog; the dialog will also accept fully qualified path names for files which live elsewhere.

TeXShop editing is done with Apple’s Cocoa editing class, which uses Unicode for internal work. When TeXShop files are written to disk, they are usually converted to 8-bit ascii because TeX expects to receive such a file. There are several ways to do the conversion: Mac OS Roman, Iso Latin 1, Iso Latin 2, and others. A preference item selects the method used. For many users, the choice will make no difference. Some TeX packages allow users to type accented European characters directly on the keyboard; these packages require the Iso Latin conversion. Thanks to Martin Heusse for providing the original code for this preference. The conversion method can also be selected directly from the Open and Save panels.

Users in Japan and Korea will use other conversion preferences, also provided.

It is possible to reset the background color of the source window. The preference dialog does not have an interface to make this change. To set the background to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal:

```
defaults write TeXShop background_R 0.42

defaults write TeXShop background_G 0.39

defaults write TeXShop background_B 0.77
```

It is also possible to set the text color in the source window. This preference will only be recognized if syntax coloring is on. To set the forground color to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal:

```
defaults write TeXShop foreground_R 0.42

defaults write TeXShop foreground_G 0.39

defaults write TeXShop foreground_B 0.77
```

The color of the insertion point in the source window can be changed. For example, to set this insertion point color to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal:

```
defaults write TeXShop insertionpoint_R 0.42

defaults write TeXShop insertionpoint_G 0.39

defaults write TeXShop insertionpoint_B 0.77
```

By using the previous three sets of commands in combination, the source window can be made to display white text on a black background or other coloring schemes as desired.

There are hidden preferences to set the transparency of the source, preview, and console windows:

```
defaults write TeXShop ConsoleWindowAlpha 0.75

defaults write TeXShop SourceWindowAlpha 0.75

defaults write TeXShop PreviewWindowAlpha 0.75
```

Here an alpha value of 0.00 is completely transparent and an alpha value of 1.00 is completely opaque. Use these commands cautiously!

A pulldown menu at the bottom of the Preferences Panel can be used to reset preferences to their default values. This is mainly useful in Japan, since different defaults are available for users of various pTeX distributions.

1.14 Modifying the Templates Menu

When TeXShop runs for the first time, it creates a folder

~/Library/TeXShop/Templates

and places the files Graphics.tex and LatexTemplate.tex (and a folder named "More" of additional files) in this folder. The names of these files are shown under the Templates pull down menu at the top of the input window. Choosing one of these items inserts the contents of the template file at the current cursor position in the source. The Graphics and LaTeX templates were recently revised by Will Robertson to follow modern TeX practices; Robertson provided the templates in the More subdirectory.

Template files are standard TeX files which can be opened and edited by TeXShop. Additional TeX files can be inserted into the Templates folder; they will also appear under the Templates pull down menu provided their names end with the extension ".tex" The original template files can be removed if desired.

To create a hierarchical structure in the Templates menu, place some of the template files in subfolders of ~/Library/TeXShop/Templates.

To restore the original template files, delete the Templates folder and run TeXShop again. It will recreate the folder and restore its original contents.

When users upgrade TeXShop, the templates folder is not updated because the user may have modified it. If an upgrade changes default templates, it is a good idea to move the ~/Library/TeXShop/Templates folder to the desktop, let TeXShop recreate the folder from scratch, and then repopulate the folder with additional templates from the old Templates folder.

The commands \usepackage{amssymb} and \usepackage{graphicx} mentioned earlier are included in recent copies of the LatexTemplate, but not in copies from very old versions of TeXShop.

1.15 Editing Tricks

Double clicking on a bracket of the form (, {, [,], }, or) will select everything between that bracket and its matching bracket. Double clicking on a bracket with the option key down will select the bracket alone.

TeXShop has a new Find panel by Isao Sonobe. This panel supports regular expressions. Users can switch between this panel and the original one in Preferences. The Find panel depends on OgreKit, a Cocoa framework for handling regular expressions by Sonobe. See

http://www-gauge.scphys.kyoto-u.ac.jp/∼sonobe/OgreKit

OgreKit is distributed using a slightly modified version of the BSD license; this license can be found in the TeXShop source code distribution. OgreKit requires Panther, so the new panel will only appear on machines running system 10.3 or later.

There are many nice features of this new Find panel. OgreKit modifies the "Find" submenu of the TeXShop edit menu, replacing it with a more extensive menu. The Find panel presents buttons controlling how it will find words; the settings of these buttons will be remembered from session to session. Adjust them until Find words as expected and then relax.

Groups of lines can be commented out using a menu command, and later activated again. Groups of lines can be indented with a menu command, and later moved back. If a line starts with a series of tabs or spaces, new lines will start at the same position until the behavior is cancelled by backspacing. This behavior was added to TeXShop by Nicolás Ojeda Bär in Argentina, who can be reached at lojedaortiz@interlink.com.ar. Thanks!

The source window can be split in two pieces using the split tool at the top of the window. When it is split, two views of the same source are shown. You can type in either piece and your changes will immediately appear in the other as well. Thus you can modify one section of source while reading a second part of the source.

TeXShop can be instructed to create a backup file every time a file is saved or typeset. This is set using a hidden preference. To create backup files, open Terminal and type the following command:

```
defaults write TeXShop SaveBackup YES
```

Change YES to NO to stop creating backups.

A Statistics panel lists the number of words, line, and characters in a document. Internally, this panel calls

```
detex myfile | wc
```

The detex command removes tex commands, but the word count is still only approximate. Input and include files are counted automatically by this command.

File icons can be dragged and dropped onto the source text. If the file is a tex file, an \input command will be added with a relative path to the file. If the file is a graphic file, an \includegraphics command will be added instead. The following graphic types are recognized: pdf, jpg, jpeg, tif, tiff, eps, ps.

Dropping a cls file produces \documentclass and a file reference, dropping a sty file produces \usepackage, dropping a bib file produces \bibliographystyle, and dropping other text files produces \input.

Drag and drop resolves aliases. For example, if a graphic file is really an alias, the file can be drag-and-dropped to the source and this source will typeset fine. TeX itself cannot resolve aliases, although it can resolve symbolic links. So if the name of the graphic alias is typed directly into the source, it will cause an error during typesetting.

The behavior of drag and drop is user customizable. Customization is done with the Macro menu. Insert a new submenu titled "Drag & Drop". Inside this submenu place items named with the extension whose behavior you wish to customize, and place the code that should be inserted in the body of this item. For instance, if the item is .pdf, the code might be \includegraphics[#INS#]{%r}. In these inclusions, the following abbreviations may be used:

%F	full path of an dropped file
%f	dropped filename
%r	relative path of the dropped file
%n	filename without extension

```
    %e    extension
```

If an extension is not mentioned in the Macro Editor, then it is handled in the default manner mentioned earlier. Thus for most people, no macro"Drag & Drop" submenu is needed.

1.16 Auto Completion

Greg Landweber has added auto completion to the TeXShop editor. Typing a double quote will produce a pair of double quotes with the cursor positioned between them. Typing the ˆ symbol will produce ˆ{ } with the cursor positioned between the brackets. Etc. Thirty-nine such completions are currently available.

Auto completion can be turned off or on with a preference item. The default preference is off. To try auto completion, turn it on.

Auto completion is user configurable. To configure, open the file

˜/Library/TeXShop/Keyboard/autocompletion.plist

with TeXShop. Read the comments at the top, edit appropriately to redefine Landweber's choices or add your own, and save. Be sure to edit and save in UTF-8 format if you use Unicode characters.

1.17 Command Completion

The source editor supports command completion. Type the first few letters of a word and hit the escape key. The remaining letters will be entered. Hitting escape again will cycle through all possible completions.

Initially this will have limited usefulness because the completion dictionary is almost empty. To add a word to the dictionary, select the word and choose "Add Word" under the Format menu. Notice that this menu command has a key equivalent. To see and edit the entire dictionary, choose "Open Completion File" under the Format menu. Notice that the completion can be more complicated than just a single word, and the cursor can be placed correctly within this completion. Examine the samples provided in the default file for details.

This feature will be improved in later versions of TeXShop.

1.18 Sending Bug Reports

To file bug reports for TeXShop and suggest improvements, contact

```
Richard Koch
Mathematics Department
University of Oregon
Eugene, Oregon 97405
koch@math.uoregon.edu
```

or

```
Dirk Olmes
dirk@xanthippe.ping.de
```

Reports for teTeX/TeXLive should go to

```
Gerben Wierda
Gerben_Wierda@rna.nl
```

1.19 Useful Web Sites

TeXShop Web Site:

http://www.uoregon.edu/~koch/texshop/

teTeX/TeXLive Web Site:

http://www.rna.nl/tex.html

Gary Gray's MacOSX TeX/LaTeX Web Site:

http://www.esm.psu.edu/mac-tex/

Unicode Extensions to support TeXShop's UTF-8 Unicode file preference option:

http://www.ctan.org/tex-archive/macros/latex/contrib/supported/unicode/

Context Web Site:

http://www.pragma-ade.com

MetaPost Web Site:

http://cm.bell-labs.com/who/hobby/MetaPost.html

Comprehensive TeX Archive:

`http://www.ctan.org`

http://www.ctan.org/tex-archive

TeX Users Group (TUG):

http://www.tug.org

XeTeX and XeLaTeX:

http://scripts.sil.org/xetex

1.20 License

TeXShop is provided under the GNU General Public License (GPL). This means that you are free to use, copy, and modify the program. If you give your modifications to others, the modifications must also be provided under the GPL.

The source code is available on the web site:

http://darkwing.uoregon.edu/~koch/texshop/texshop.html

2 Advanced Help

2.1 Adding Personal Sty and Macro Files to teTeX

Users sometimes create additional sty, cls, bib, and tex files for common use in several different projects. These files will certainly be seen by tex and latex if they are placed in the same folder as the tex source file being typeset, but then multiple copies of the files must be kept on the computer.

It is possible to make these files visible to all teTeX projects, whether typeset from TeXShop or from the Terminal. To do so, create a subfolder of the Library folder in your home directory named texmf. Create subfolders named tex and bibtex. Within the tex folder, create a subfolder named latex. Store your personal files in these folders.

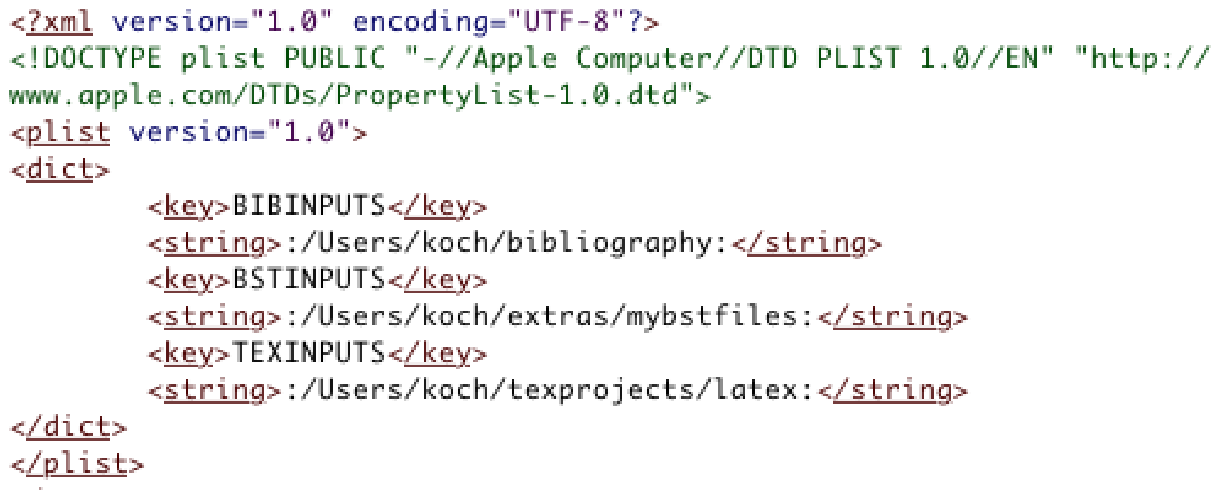
Tex will find any file in ~/Library/texmf/tex or a subfolder of this directory. Latex will find any file in ~/Library/texmf/tex/latex or a subfolder of this directory, and bibtex and makeindex will find any file in ~/Library/texmf/bibtex and ~/Library/texmf/makeindex respectively, or a subfolder. It is not necessary to run texhash after adding files to these folders.

On other Unix machines, tex is sometimes notified of the location of personal input files by setting the environment variables TEXINPUTS, BSTINPUTS, and BIBINPUTS. This is generally not necessary in Gerben Wierda's teTeX distribution.

These environment variables can be set in unusual circumstances. Mac OS X allows users to notify programs about environment variables by creating a file in their home directory named ~/.MacOSX/environment.plist. These environment variables get picked up by all user shells and all Carbon and Cocoa programs, so caution should be used when creating the file.

Below is a sample environment.plist file sent me by Nathan Potter; thanks! For further details, consult

http://developer.apple.com/qa/qa2001/qa1067.html



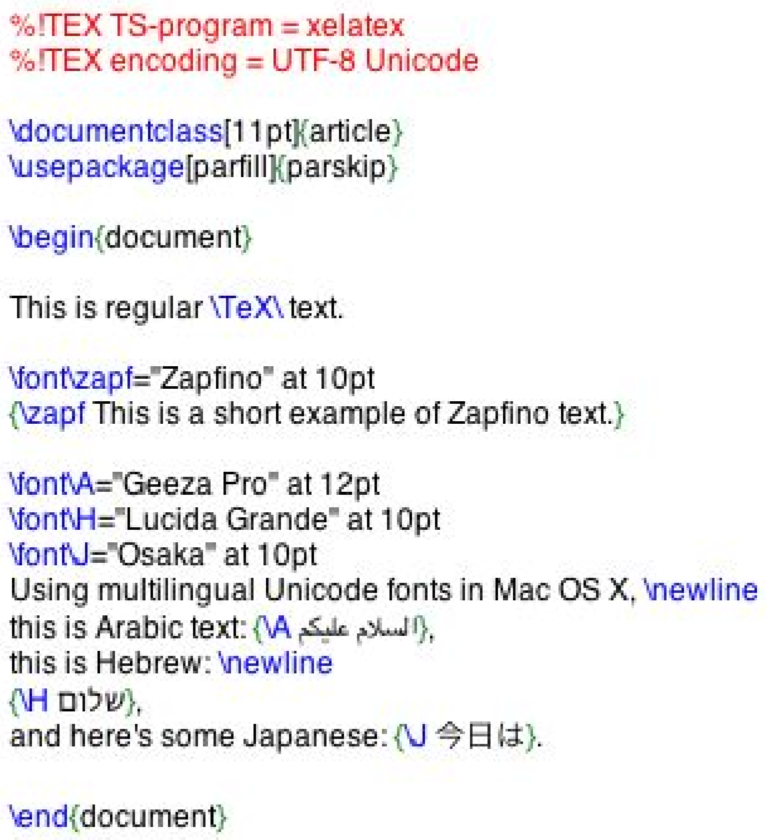
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://
www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>BIBINPUTS</key>
        <string>:/Users/koch/bibliography:</string>
        <key>BSTINPUTS</key>
        <string>:/Users/koch/extras/mybstfiles:</string>
        <key>TEXINPUTS</key>
        <string>:/Users/koch/texprojects/latex:</string>
</dict>
</plist>
```

2.2 Additional Typesetting Engines

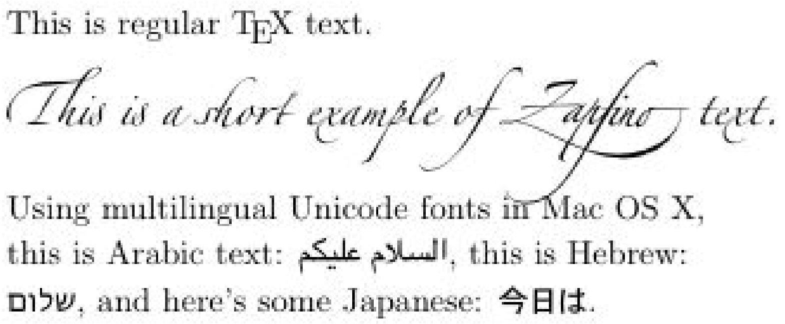
Recently, Jonathan Kew introduced XeTeX and XeLaTeX. These programs can use Macintosh fonts directly within a TeX document. XeTeX and XeLaTeX accept unicode source and thus can deal with any unicode character; for example, with them you can type Arabic or Hebrew in the source document and later see Arabic or Hebrew in the output. TeXShop's editor will correctly input these characters from right to left. XeTeX is not part of the standard TeX distribution from Gerben Wierda, but it can be obtained using Wierda's i-Installer as an optional install directly from Jonathan Kew. It is automatically installed as part of the MacTeX install package. For additional details, consult

http://scripts.sil.org/xetex

To understand how this works, here is a brief XeLaTeX source document, and the resulting output. The Arabic and Hebrew characters were obtained by accessing the International Panel in System Preferences, selecting the Input Menu tab, and checking Arabic and Hebrew; a small flag appeared on the menu bar and a user could switch keyboards by selecting an appropriate flag.



```
%!TEX TS-program = xelatex
%!TEX encoding = UTF-8 Unicode

\documentclass[11pt]{article}
\usepackage[parfill]{parskip}

\begin{document}

This is regular \TeX\ text.

\font\zapf="Zapfino" at 10pt
{\zapf This is a short example of Zapfino text.}

\font\A="Geeza Pro" at 12pt
\font\H="Lucida Grande" at 10pt
\font\J="Osaka" at 10pt
Using multilingual Unicode fonts in Mac OS X, \newline
this is Arabic text: {\A السلام عليكم},
this is Hebrew: \newline
{\H שלום},
and here's some Japanese: {\J 今日は}.

\end{document}
```



This is regular TEX text.

This is a short example of Zapfino text.

Using multilingual Unicode fonts in Mac OS X,
this is Arabic text: السلام عليكم, this is Hebrew:
שלום, and here's some Japanese: 今日は.

TeXShop now supports XeTeX and XeLaTeX. XeTeX and XeLaTeX are available in the pull-down typesetting menu on the source window A user can make XeTeX or XeLaTeX the default typesetting option in preferences. If one of the first twenty lines of the source has the form

```
        %!TEX encoding = UTF-8 Unicode
```

then that source will be loaded and saved with UTF-8 Unicode encoding regardless of the default encoding setting in preferences If one of the first twenty lines of the source has the form

```
        %!TEX TS-program = xetex
```

or

```
        %!TEX TS-program = xelatex
```

then the appropriate program will be used regardless of the typesetting option chosen in the menu.

These XeTeX features are a special case of a new general method for adding typesetting engines to TeXShop. There is a folder in ˜/Library/TeXShop named Engines; the files in this folder are shell scripts which call typesetting programs. When TeXShop first starts, it examines this folder and adds the script names of files it contains to the pull-down typesetting menu. Choosing one of these items and pushing the Typeset button calls the script. Users can write their own scripts and add them to the Engines folder. Each such script must have a name without spaces, and extension ".engine", and have the executable bit set.

Items in ˜/Library/TeXShop/Engines can be chosen as the default typesetting method in TeXShop Preferences. Notice that when a method is listed in Preferences, its name is given without the extension.

The default program can be set on one of the first twenty lines of the source code by writing a line like

```
        %!TEX TS-program = xelatex
```

The encoding used to open or save any file can be set by writing a line of the following form as one of the first twenty lines of a source document. Any supported encoding is allowed. To bypass this behavior, hold down the option key while opening a file.

```
        %!TEX encoding = UTF-8 Unicode
```

There must be one space after %!TEX and one space before the equals sign; the spaces in the name of the encoding must match the spaces in the name listed below.

Below is a list of allowed encoding names:

```
MacOSRoman
IsoLatin
IsoLatin2
IsoLatin5
IsoLatin9
MacJapanese
DOSJapanese
SJIS_X0213
EUC_JP
JISJapanese
MacKorean
UTF-8 Unicode
Standard Unicode
Mac Cyrillic
DOS Cyrillic
DOS Russian
Windows Cyrillic
KOI8_R
Mac Chinese Traditional
Mac Chinese Simplified
DOS Chinese Traditional
DOS Chinese Simplified
GBK
GB 2312
GB 18030
```

2.3 Removing AUX Files

When documents are typeset, auxiliary files are created with extensions .aux, .log, .bbl, etc. Occasionally these files can become corrupt and lead to unexplained typesetting errors. TeXShop has a menu command “Trash AUX Files” which will remove all such files so typesetting can proceed. There is a similarly named button on the Console window. If either of these items is activated, TeXShop will move to the trash all files in the current source directory with the same name as the source file and extension aux, blg, brf, ccs, ent, fff, glo, idx, idv, ilg, ind, ioa, lg, log, lot, mte, mlf, out, pdfsync, toc, ttt, wrm, xref, 4ct, or 4tc.

Additional extensions can be added to this list. To add "dvi" to the list, activate the Terminal and type

```
defaults write TeXShop OtherTrashExtensions -array-add "dvi"
```

To remove all additions and return to the original default list, tppe

```
defaults write TeXShop OtherTrashExtensions -array
```

Sometimes more extensive cleanup is needed. For example, if a book is controlled by main.tex, and chapters are in subfolders accessed with commands like

```
\include{chapter1/chapter1}
```

then typesetting the book will create main.aux, main.pdfsync, and main.log in the main folder, and chapter1.aux in the chapter1 folder.

The required extensive cleanup can be done by holding down the option key while choosing "Trash AUX Files". In this case: "%!TEX root" and Root File information will be used to find the root document and its folder All files with appropriate extensions in this folder or any subfolder will be moved to the trash, regardless of the name of the file.

There is a way to make this behavior the default behavior for "Trash AUX Files" even if the option key is not down:

```
defaults write TeXShop AggressiveTrashAUX YES
```

2.4 Using an External Editor

TeXShop can be configured for an external editor in several ways. Some features below will be useful and others not depending on how your editor works. In the future, your editor may come with configuration instructions.

There is a menu item named "Open for Preview.." When this item is chosen, you will be asked to select a .tex source file. But only the associated pdf preview window opens. If the source has not yet been typeset or if the pdf is out of date, TeXShop will typeset the file as it is opened. You can open your source file in any editor. When it is time to typeset, save the changes in your editor, switch to the preview window, and typeset. In this mode, TeXShop never opens or modifies the source file. It simply passes this file to the Unix TeX or LaTeX process.

The preview window can be configured to contain a typesetting button. If such a button is not there, use “Customize Toolbar...” in the Windows menu to add one.

For users who prefer external editors most of the time, there is a preference item called “Configure for External Editor.” When this preference is chosen, the“Open” and “Open Recent...” menus open tex source files in the above manner for editing with an external editor. Moreover, the “Open for Preview...” menu becomes “Open for Editing...” and opens the source file in TeXShop’s internal editor for those rare occasions when the internal editor is desired. The “Open” and “Open Recent...” menus can also open jpg, tiff, ps, pdf, dvi, log, and other files; selecting the new preference does not change this behavior.

Some editors are able to call Unix typesetting commands directly. A new preference item, “Automatic Preview Update”, has been added to improve your experience with these editors. When this item is active and a .tex file is opened using “Open for Preview”, the pdf preview display automatically updates whenever the pdf file is rewritten. Thus you can typeset with an external editor and the preview window will automatically show changes. The preference also applies to .pdf files opened in TeXShop, and if TeXShop is configured to use an external editor, the preference applies to .tex files opened with “Open”.

The following applescript commands have been added to TeXShop so editors can call TeXShop directly:

```
typesetinteractive
texinteractive
latexinteractive
contextinteractive
bibtexinteractive
makeindexinteractive
metapostinteractive
taskdone
refreshpdf
open_for_externaleditor
```

The first seven commands activate TeXShop typesetting commands. Applescript will return immediately after these calls without waiting for typesetting to complete.

The "taskdone" call can be used to test completion; it returns NO while typesetting is being done, and YES when it is finished. Each of these calls refreshes the preview window at the end of typesetting. If your editor calls TeXShop to typeset, the "Automatic Preview Update" preference can be turned off.

If the external editor modifies the pdf output directly, it can call "refreshpdf" to refresh the pdf display. This is only needed if "Automatic Preview Update" is off. Finally "open_for_externaleditor" opens a tex file by calling "Open for Preview".

The Tcl code to make these calls from AlphaX in included in the AlphaTcl folder inside the TeXShop_Folder of the TeXShop distribution. Read the first lines of this file to see how to configure AlphaX for this purpose.

2.5 Copy-Paste and Drag-Drop from the Preview Window

It is possible to select a portion of the pdf preview display and copy and paste the resulting graphic into another program. Drag and drop of the selection is also supported. Preference items allow users to select the file type of the copy (pdf, png, jpg, tiff, etc.) and the foreground and background colors and transparency.

Preview selection combined with drag and drop can be used with Keynote. Using the default copy preferences, a text selection can be dropped on a Keynote slide; the background of the text will be transparent, so the slide background will show between the letters. This text can be resized since the default copy format is pdf.

The Edit menu's "Select All" command can be used to select the entire pdf page for copying. However, in Multi-Page and Double-Multi-Page modes Select All only works if the document has 20 or fewer pages, since otherwise the selected pdf could be enormous and copying could bring the machine to a crawl.

2.6 Setting a Project Root File

It is common to split large input files into several smaller files controlled by a root file. For instance, the root file of a book project might have the form

```
\documentclass[11pt]{book}
\includeonly{Chapter2/two}
\textwidth = 6..5 in
\textheight = 9 in
```

```
\begin{document}
\include{Chapter1/one}
\include{Chapter2/two}
\include{Chapter3/three}
\end{document}
```

This root file contains formatting commands for the entire project, but the subject matter is contained in files one.tex, two.tex, and three.tex. The second line above tells TeX to typeset chapter two only, speeding up typesetting while chapter two is being written. When the book is complete, this line can be commented out and the entire book can be typeset a final time.

While chapter two is being written, it would be natural to open the file two.tex in TeXShop. But after changes are made to this input file, TeXShop should not typeset two.tex; instead it should typeset the root file. TeXShop can be told to do so in two ways. The first of these is easier and is the preferred method. Specify the root file in one of the first twenty lines of each chapter file by writing

```
%!TEX root = ../Main.tex
```

The second method uses the menu command ¨Set Project Root...¨ This menu command presents a panel with a text box where the name of the root file can be entered.

If the root file is in the same folder as the input file, it is enough to give its name, including the ¨.tex¨ extension. For instance, if the root file is named Main, you can enter Main.tex in the first source line or the dialog window.

If the root file is in a different directory, its name can be given relative to the location of the input file. In the above example, the various chapters are contained in subfolders within the folder containing the root file. In that case, you could enter ../Main.tex in the first source line or dialog window, showing the location of the root file relative to the chapter input file.

Finally, the name of the root can be given with an absolute name, as in

```
/Users/me/Main.tex
```

If the entire TeX source is contained in a single file, it is not necessary to set the root project name.

If the "Set Project Root.." command is used to indicate a root file, TeXShop remembers the name of the root file by writing the information to a file with the same name as the input file and the extension ".texshop". For instance, if the input for chapter two is in two.tex, TeXShop writes the name of the root in two.texshop in the same directory as two.tex. If the file two.texshop is later thrown away, TeXShop will revert to typesetting two.tex rather than Main.tex. No extra ".texshop" file is required if the root file is specified in the first line of the source file.

If a root file is active, typing command-1 when a given source window is at the front will activate the corresponding preview window. Typing command-1 again will activate the source window. Since a preview window may correspond to several source windows, command-1 will activate the source window which previously activated the preview window, or the root source if no previous command-1 was entered.

2.7 Pdfsync

Synchronization is an important feature of Textures, a commercial implementation of TeX by Blue Sky TeX Systems. Using this feature, an author can click in the preview window and immediately be taken to the corresponding spot in the source window, or click in the source window and be taken to the preview window. The implementation of synchronization in Textures is widely admired for its ability to precisely locate the corresponding spot in the source code.

Since this implementation, a number of TeX systems have provided a roughly similar feature using an additional style file which causes TeX to write extra information to the dvi file. None of these approximations (including the TeXShop methods to be described next) is as accurate as the Textures method.

TeXShop can provide synchronization in two ways. The default method uses a new ability in Mac OSX 10.4 to search for strings in pdf files. No special style files need be included to use this method. Click on a word or phrase in the source window while holding down the Apple control key; the preview window will scroll to the appropriate spot and the corresponding typeset phrase will be circled in red. Similarly, click on a word or phrase in the preview window. The appropriate source file will open (if necessary) and scroll to the corresponding source phrase, which will be highlighted in yellow.

This method is independent of the engine used to typeset the file, so it will work with pdftex and pdflatex, with TeX + Ghostscript and LaTeX + Ghostscript, with XeTeX, and with other engines.

When using this facility, it helps to know the underlying mechanism. Suppose you click on a spot in the source file. TeXShop obtains the string 20 characters wide centered about the click, and searches for the corresponding string in the pdf file. If it finds this string exactly once, it circles the pdf string in red and declares success. But often, source strings contain formatting commands and do not match output strings; this is certainly true when typesetting mathematics. So if the search fails, TeXShop backs up 5 characters, obtains a new string 20 characters wide, and tries again. It repeats this operation 20 times if necessary, and if none of the tries succeeds it repeats the operation 20 more times looking forward past the original string rather than backward. If none of these search operations succeeds, TeXShop gives up.

Obviously, then, it is best to click on sentences rather than formulas. Paragraphs with heavy markup may not synchronize well. Experiments show, however, that searches usually succeed and clicking can be done without really thinking about selecting an appropriate spot.

Synchronizing from the preview window to the source window works the same way, but there is an additional complication. Projects may contain several source files, inserted using the \include or \input methods. TeXShop handles this complication by assuming that all \include and \input statements occur in the root file, using one of the commands \include{this file}, \input{this file}, or \import{this file}. In the initial implementation, it makes a list of the first sixty such files and searches all of them, declaring success if a string occurs exactly once in exactly one of the files. If the string is found in a file which is not yet open, TeXShop opens the file. Then it scrolls to the appropriate spot and highlights the resulting phrase in yellow. (In future implementations, the number sixty may be raised; write if you need more files searched.)

ConTeXt uses a different method of inputting files, so a search for \include and \input statements will not find related ConTeXt files. There is a manual method to indicate files to be searched for synchronization; this method works with any typesetting method and is required in ConTeXt. At the top of the root file, list files to be searched via

```
%!TEX projectfile =
```

Here are examples:

```
%!TEX projectfile = /Users/koch/MyDoc/chapter1.tex

%!TEX projectfile = chapter2.tex

%!TEX projectfile = ../chapter3.tex
```

Earlier versions of TeXShop used an alternate synchronization method, invented in 2003 by Jérôme Laurens and Piero D'Ancona. They wrote a style file for pdflatex; their package, pdfsync.sty, causes pdftex to write an extra file during typesetting with information needed to correlate the output pdf and input source files. Laurens' and D'Ancona's work depends on internal pdftex commands which output the x and y coordinates of certain typesetting operations, so it only works with pdftex, pdflatex, and context.

TeXShop can still use this method. A preference is available in the Misc tab to select the synchronization method; "pdfsync" is the original method and "pdfsearch" is the new method. It is also possible to choose to use "pdfsearch" but fall back on "pdfsync" when "pdfsearch" fails to find a match.

The "pdfsync" method requires that you install the latest version of pdfsync.sty and related files. Read "How do I configure TeXShop?: Did You Install TeXShop Correctly" to make certain that you installed these files.

To use pdfsync with LaTeX, add the following line to the preamble of your tex source code before the \begin{document} line:

```
\usepackage{pdfsync}
```

The package will be ignored if you typeset with tex + ghostscript, but if you typeset with pdflatex, an extra file will be created with extension .pdfsync. For example, if the source file is named main.tex, the extra file will be named main.pdfsync. This extra file contains the information needed to correlate spots in the pdf file with spots in the source file. Most files should typeset without errors using pdfsync, but in a small number of cases, this package might introduce extra errors. Such dangerous code can be enclosed between a pair of commands \pdfsyncstop and \pdfsyncstart. For added accuracy, extra synchronization points can be added using the command \pdfsync. The pdfsync style files sometimes changes line breaks in a document, so the standard recommendation is to typeset using it and then typeset a final time without it.

Once a file has been typeset with pdfsync, synchronization works exactly as with "pdfsearch" described above. Note that if you include pdfsync.sty but choose "pdf-search" as the TeXShop synchronization preference, then the .pdfsync file will just be ignored.

Warning: if a root document uses \include to add other source files but these files are not linked back to the root document with a "%!TEX root" line or a root file, then command-click will move from the preview window to an included source, but not from the included source back to the preview. It is easy to remedy this by adding a "%!TEX root" comment to the top of each source file.

As explained above, pdfsync works with files read by \include{myfile}. It also works with the input command provided the syntax used is \input{myfile}; the alternate syntax \input myfile will not work.

The file pdfsync.sty is used for LaTeX; similar files pdfsync.tex and pdfsync4context.tex exist for tex and context synchronization.

Synchronization works by writing data to mysource.pdfsync corresponding to certain synchronization spots in the preview window. Roughly speaking, each data entry contains

> the page number of the preview document where the point occurs
>
> the location of the point on this page
>
> the name of the source file producing this particular output
>
> the line number in this source file for this particular output

There is a way to get TeXShop to display these synchronization points. The preview window toolbar has a checkbox item called SyncMarks. By default, this item is not shown; use Customize Toolbar in the Window menu to select it. When the checkbox is checked, synchronization points are shown.

When the Preview window first appears, this item is not checked. A hidden preference item can change this:

```
defaults write TeXShop ShowSyncMarks YES
```

2.8 Unicode

Unicode is a format which can simultaneously encode characters from languages across the world: Arabic, Chinese, Greek, Hebrew, Japanese, Roman, Russian, and a host of others. Editors written with Apple's Cocoa use Unicode internally and can write external Unicode files. In particular, TeXShop can do so. To experiment, open the International preference in the System Preferences program, choose the Input Menu tab, and add additional selections to the items already selected. It is illuminating to select Greek (because it is familiar), Hebrew (because it is written from right to left), Arabic (because it is written from right to left and makes extensive use of ligatures, so characters have different shapes at the ends of words than in the middle) and Chinese. A new menu item will appear with a flag indicating the current input language.

Type some characters, switch to another language, and type some more.

There are several formats for Unicode files. A preference item allows users to select two of them. Standard OSX Unicode is the usual code used by Mac OSX, but I do not know if this is useful in the TeX world. UTF-8 Unicode is a popular format because ordinary ascii characters appear as they usually do. If a file is saved in UTF-8 format, TeX can process it, but it will convert Unicode characters to question marks.

There is a TeX package which accepts UTF-8 Unicode input files; see

http://www.ctan.org/tex-archive/macros/latex/contrib/supported/unicode/

2.9 BibTeX

BibTeX is a tool used to create bibliographies in LaTeX documents. The tool assumes that you have a large database of references which you often quote. Simple commands allow you to cite certain of these references in your LaTeX source, and BibTeX then creates a bibliography for your article containing only the items cited.

Here is a brief example taken from The LaTeX Companion by Goossens, Mittelbach, and Samarin. Consult this book for many additional details. Suppose the database is a file called ¨mybibliography.bib¨ containing the text shown below. In this text, the entries ¨Felici:1991, Knuth:WEB,¨ and ¨Liang:1983¨ are key values used to cite the articles in the LaTeX source.

```
@article{Felici:1991,
author ={James Felici},
title ={{PostScript versus TrueType}},
journal ={Macworld},
volume =8, pages ={195--201},
month =sep, year =1991}

@techreport{Knuth:WEB,
title ={{The \textsf{WEB} System of
Structured Documentation}},
month =sep, year =1983,
author ={Donald E. Knuth},
address ={Stanford, CA 94305},
number ={STAN-CS-83-980},
institution ={Department of Computer
Science, Stanford University} }

@phdthesis{Liang:1983,
author ={Franklin Mark Liang},
month =jun, year = 1983,
school ={Stanford University},
address ={Stanford, CA 94305},
title ={{Word Hy-phen-a-tion by
Com-pu-ter}},
note ={Also available at Stanford
University, Department of
Computer Science Report
No. STAN-CS-83-977} }
```

Suppose the LaTeX source file is called "myfile.tex" and contains the following text:

```
\begin{document}
Consider the argument of Felici~\cite{Felici:1991} and others.
\nocite{Liang:1983} We provide further remarks later.
\bibliographystyle{plain}
\bibliography{mybibliography}
\end{document}
```

When this source is typeset, a reference to Felici's article will appear in the text, and a bibliography will be created at the end of the text containing the articles of Felici and Liang, but not the article of Knuth.

TeXShop can be used with this example in the following way. First edit and typeset the document "myfile.tex" as usual. Citations will appear in the output as "[?]" and the bibliography will be missing. Then select "BibTeX" under the Program button and run BibTeX. Next select "LaTeX" and typeset again. Citations will still appear as "[?]", but the bibliography will be added to the output. Typeset a final time, and citations will have their correct values.

The file "mybibliography.bib" can be opened and edited by TeXShop. If you use

TeXShop to create the file “mybibliography.bib” in the first place, use the pulldown tag labeled “file format” to save the file as a bib file rather than as a tex file.

2.10 MakeIndex

MakeIndex is a tool used to add an index to a LaTeX document. To create such an index, you should

- Add the command “\usepackage{makeidx}” to the top of the source file
- Put a “\makeindex” command in the document preamble
- Put a “\printindex” command where the index should appear, often just before “\end{document}”
- Put index references in the source document, as in the example below:

  ```
  There are many animals in the world\index{animal}. Examples include
  bears\index{animal!bear} and tigers\index{animal!tiger} and various
  insects\index{insert|see{animal}}.
  ```

To create the index, typeset the document as usual. Then run MakeIndex. Then typeset again. Many additional details can be found in The LaTeX Companion by Goossens, Mittelbach, and Samarin.

2.11 Making and Using New TeX Formats

2.11.1 Using formats

When TeX typesets a document, it must process the document header, which may contain a large number of \input, \include, and \usepackage statements. Every time the document is typeset, this process is repeated even though large portions of the header did not change.

TeX has a built-in mechanism to speed up this process. TeX can be instructed to process lines of source and output the result to a “format file.” When TeX typesets after that, it can rapidly read the format and then typeset the document. LaTeX is constructed in this manner; the format file is created when TeX is installed on your machine and the Terminal command “latex file” tells TeX to read this format and then typeset “file”.

When machines were slower, users often created their own format files to speed up typesetting. This is done less often today, but dealing with format files may still be useful. For instance, some organizations create and distribute a format to be used by everyone working on a common project. In this section we'll explain how to use a format file provided by someone else, and how to create a format.

A typical format file has extension "fmt". Suppose a department of the University of Oregon has provided a format "uo.fmt". To use this format, follow the steps below:

- Move uo.fmt to "~/Library/texmf/web2c/uo.fmt". You may need to create some of these directories.
- Create a new "engine file" by going to ~/Library/TeXShop/Engines and duplicating the file XeLaTeX.engine. Engine files need to have the execute bit set, and this duplication step automatically does that.
- Rename this new file. The name need not match the name of the format. Since the name will appear in the TeXShop interface, it should make sense to a user. We'll choose "Oregon.engine".
- Open Oregon.engine in TeXShop and edit it to read as follows. The first two lines may already be present.

```
#!/bin/tcsh
set path= ($path /usr/local/teTeX/bin/`uname -p`-apple-darwin-current //
     /usr/local/bin)
pdflatex -fmt uo  "$1"
```

This completes the installation. The next time you start TeXShop, the pulldown menu beside the typeset button on the Source Window Toolbar will list "Oregon" as an option. Choose this to typeset using the uo.fmt format. If you want this typesetting method to be the default, go to TeXShop preferences under the Typesetting tab, and in the Default Command box select "Command Listed Below" and fill in the edit box with the word Oregon.

It is also possible to select the Oregon format for a particular document without changing the typesetting option. To do that, add the following line to the top of the source file:

```
        %!TEX TS-program = Oregon
```

Then uo.fmt will be used for that document regardless of the typesetting option chosen.

The above instructions assume you have a format for pdflatex. You can also make formats for plain tex; in that case change "pdflatex" to "pdftex" in the engine file.

It is also possible to use formats when you are typesetting with tex + ghostscript. TeXShop assumes that an engine file contains one or more command line instructions and ultimately produces a pdf file. So the engine file must contain the commands which convert the dvi file to a pdf file. Here is a typical engine file for uo.fmt in that case:

```
#!/bin/tcsh
set path= ($path /usr/local/teTeX/bin/`uname -p`-apple-darwin-current //
   /usr/local/bin)
latex -fmt uolatex  "$1"
set filename = "$1"
dvips "${filename:r}.dvi"
pstopdf "${filename:r}.ps"
```

2.11.2 Making formats

Format files contain the internal binary representation of typeset lines of source. This representation depends on the processor and particular TeX implementation. Thus it is rarely possible to use a format file from someone else unless that person has the same machine and TeX installation that you do.

However, it is common for organizations to distribute the source lines needed to construct a format. In that case you'll be told to make the format using"initTeX". For example, suppose this source text is "uo.tex". To make the format, open Apple's Terminal program and change to the directory containing the source for the format. Then type

```
 pdflatex -ini
```

You will get a ** prompt. Type the following line at the prompt

```
 &pdflatex uo
```

and press return. The format file will be created. At the end you may have to issue a "\dump" command if the format source doesn't contain it. This will produce the required “uo.fmt”.

You might like to try this with the following “uo.tex” file:

```
\documentclass[11pt]{article}
\usepackage{geometry}
\geometry{letterpaper}
\usepackage[parfill]{parskip}
\usepackage{gra[hicx}
\usepackage{amssymb}
\usepackage{epstopdf}
\DeclareGraphicsRule{.tif}{npg}{.png}{‘convert #1 ‘dirname #1‘/‘basename #1
    .tif‘.png}
\dump
```

2.12 Opening Other Files with TeXShop

TeXShop can open most files for editing. This facility has been provided so users can read TeX-related files with extensions ¨.log¨, ¨.aux¨, etc. Files with extensions ¨.jpg¨, ¨.tif¨, ¨.eps¨, or ¨.pdf¨ are opened as graphic files.

TeXShop can also open ¨.dvi¨ and ¨.ps¨ files. In these cases, it converts the file to pdf and displays this pdf file.

TeXShop can open ¨.ins¨, ¨.dts¨, ¨.sty¨, ¨.cls¨, ¨mf¨, ¨.def¨, ¨.fd¨, ¨.ltx¨, ¨.clo¨, and ¨mp¨ files for processing. It can open any text file.

It is not a good idea to use TeXShop as a general editor. TeXShop can read files written in UniCode and other formats, but it saves files in the format chosen in Preferences. Consequently, information may be lost if TeXShop is used to edit files which really aren’t related to TeX.

2.13 Mathematica

Mathematica can save images in eps format. These files can be used by either typesetting option; as explained above, they are automatically converted to pdf if typeset with pdflatex.

The eps files produced by Mathematica use the Macintosh line feed conventions. Earlier versions of TeXShop and teTeX required that these files be converted to use Unix line feeds before converting them to pdf format, but that is no longer necessary.

2.14 Localizations

TeXShop contains a Dutch localization Maarten Sneep, a French localization by Jerome Laurens, a German localization by Keith J. Schultz, Sascha Beverungen, Martin Kerz, and Max Horn, an Italian localization by Giuseppe Carlino, a Japanese localization by Seiji Zenitani, a Spanish localization by Juan L. Varona, and a Portuguese localization by Paulo T. Abreu. Thanks!

In Japan, modified versions of TeX and LaTeX called ptex and platex are sometimes used. TeXShop has been modified by Makoto Inoue, Seiji Zenitani, and Mitsuhiro Shishikura to deal with these versions. See files in the TeXShop distribution folder on my web site. See

http://macwiki.sourceforge.jp/cgi-bin/wiki.cgi/TeXShop

for further details. Ptex and platex are available at

http://www.ascii.co.jp/pb/ptex/

Dvipdfmx is available at

http://project.ktug.or.kr/dvipdfmx/

A dvi previewer for Japanese is available at

http://macptex.appi.keio.ac.jp/˜uchiyama/macptex.html

2.15 Syntax Colors

When syntax coloring turned on in the source window, comments will be colored red, commands will be colored blue, and the symbols $, {, and } will be colored dark green. A few users may wish to change these colors. Each color is determined by setting its red, green, and blue components; these components are real numbers between 0.0 and 1.0. Suppose we wish to change the color of $, {, and } to bright green, a color with components (r, g, b) = (0.0, 1.0, 0.0). To do so, open the Terminal window and type the following commands

```
defaults write TeXShop markerred 0.0

defaults write TeXShop markergreen 1.0

defaults write TeXShop markerblue 0.0
```

The corresponding preference items for comments are commentred, commentgreen, commentblue; the items for commands are commandred, commandgreen, commandblue.

2.16 Shell Escape Protection

The default commands for pdftex and pdflatex are now "pdftex –shell-escape" and "pdflatex –shell-escape". The"shell-escape" portion of this text tells pdftex that it is legal to run other programs during typesetting. This is useful because if tex finds a graphic file in an unsupported format, it can automatically call another program to convert it to supported format. For example, the default Latex template automatically converts tif files to png and automatically converts eps files to pdf.

This creates one difficulty that may worry some users. The "shell-escape" flag allows pdftex to run ANY program. Thus a disgruntled student could send you a tex source file by email and when you typeset it you would discover that some of the files in your directory had been erased.

I believe the danger is slight. A source file which did harm would have to be created deliberately, but sending a "virus" via tex source seems somewhat esoteric. Nevertheless, there are two ways that you can protect yourself. The first and easiest is to remove the letters"–shell-escape" from the two spots they occur in Preferences and then convert all of your graphic files by hand.

TeXShop now provides a different protection. A preference item under the Engine tab is labeled "Shell Escape Warning". This item is off when TeXShop is delivered. If the item is on and "shell-escape" is active, then the first time a file is typeset during a TexShop session, a warning dialog will appear allowing you to turn shell-escape off for that particular file. This dialog will not appear again during the session for that particular file. Thus you can typeset your own files using "shell-escape" and typeset files received in the mail without "shell-escape".

2.17 Colored Text in TeX

It has always been possible to color TeX output. To do so, add the line

```
\usepackage{color}
```

to the preamble, and insert a line like

```
\color{blue}
```

before typesetting begins. To color a limited section of text, use a command like

```
{\color{red} This is very important!}
```

To define custom colors, use a command like

```
\definecolor{mycolor}{rgb}{0.2, 0.7, 0.8}
```

Users may be familiar with a list of defined colors with names like "BrickRed"; these named colors were defined automatically for the dvips driver. To use them generally, use the syntax

```
\usepackage[dvipsnames,usenames]{color}
```

When output is colored, the output may be difficult to read on the preview screen. TeXShop has hidden preferences to set the background color of the preview window. This background color will not appear when the document is printed. To set the background to gray, issue the following commands in Terminal:

```
defaults write TeXShop Pdfbackground_R 0.5

defaults write TeXShop Pdfbackground_G 0.5

defaults write TeXShop Pdfbackground_B 0.5
```

Change 0.5 to 1 to get back to white. If you find yourself changing the background often, create an Applescript macro named "PDF background gray" with the Macro Editor using the following text:

```
--AppleScript

do shell script "defaults write TeXShop Pdfbackground_R 0.5"

do shell script "defaults write TeXShop Pdfbackground_G 0.5"

do shell script "defaults write TeXShop Pdfbackground_B 0.5"
```

and create a similar macro named "PDF background white".

2.18 More About teTeX

teTeX is installed in the directory /usr/local/teTeX, which is not visible from the Finder. But the installer creates a symbolic link to this directory in /Library/teTeX. Therefore, go to /Library/teTeX to inspect files in teTeX.

The documentation for teTeX is in /Library/teTeX/share/texmf/doc/tetex. In particular, examine the files

TETEXDOC.pdf

teTeX-FAQ

teTeX comes with a program which looks up documentation and displays it in an appropriate viewer for Mac OS X. The program is called "texdoc". For examples of its use, type one or two of the following commands in Apple's Terminal program:

```
texdoc latex

texdoc kpathsea

texdoc pdftex
```

Additional documentation for Gerben Wierda's Mac OSX compilation is in

/Library/teTeX/README.macosx

It is possible to store additional style files, fonts, etc., in teTeX itself. But teTeX is owned by root and lives in /usr/local/teTeX, which isn't visible in the Finder. Moreover, it is a good idea to keep this directory pure so it can be upgraded easily when later versions are released. It is better to construct a mirror image of the teTeX directory structure inside your Library folder and store your personal files there.

The files must be in appropriate directories. teTeX uses a directory structure invented by the TeX working group. Suppose you want to add a extra style file named new.sty to the style files used by LaTeX. Look inside teTeX and notice that a natural spot for this file would be

/usr/local/teTeX/share/texmf/tex/latex/misc/

So inside your home directory's Library folder, make a series of subdirectories as follows:

Library –> texmf –> tex –> latex –> misc

and store “new.sty” inside this misc directory.

2.19 Coexisting with Fink

Mac OS X can also run X-windows applications using free third party software. These applications run side by side with regular Mac OS X programs. Fink is a system which can download and compile these applications on Mac OS X. For details, see http://fink.sourceforge.net. In particular, Fink can install ghostview, xdvi, and xemacs.

Fink installs software in /sw/bin. One of the packages Fink can install is teTeX. In the past this caused problems because users had duplicate copies of teTeX, configured one of the copies, and then found that the configuration did not affect the other copy. The latest version of Fink gives the option of using Gerben Wierda’s teTeX with symbolic links in /sw/bin rather than installing a second copy. We recommend that you choose that option.

2.20 Coexisting with Other TeX Distributions

Problems *may* arise when multiple TeX distributions are available on a system, especially if they set environment variables. This may lead to hard to detect problems where other files are used than the ones you expect. However, provided you only have one TeX distribution, TeXShop should work fine with it. You will need to reset the TeXShop preferences under the “Engine” tab for path settings to the TeX binaries and the Distiller binary.

2.21 ConTeXt and MetaPost

ConTeXt is a general purpose TeX macro package by Hans Hagen; for some, it will be a serious alternative to LaTeX. For details, see

http://www.pragma-ade.com/

MetaPost is a MetaFont like system by John Hobby which can output postscript and pdf files. The package can be used to draw elaborate postscript illustrations. For more details see

http://cm.bell-labs.com/who/hobby/MetaPost.html

Interesting metapost examples can be found at many web sites; for instance see

http://www.cs.ucc.ie/˜dongen/mpost/mpost.html

Here is a sample MetaPost file:

```
prologues:=2;
color yellow; yellow = green + red;

def star (expr size, n, pos, color)=
for a = 0 step 360/n until 360:
draw (origin -- (size/2,0))
rotatedaround (origin, a)
shifted pos withcolor color;
endfor ;
enddef;

beginfig(1);
pickup pencircle scaled 2mm; star (2cm,5,origin,red);
endfig;

beginfig(2);
pickup pencircle scaled 2mm; star (2cm,7,origin,yellow);
endfig;

beginfig(3);
pickup pencircle scaled 2mm; star (2cm,11,origin,blue);
endfig;

beginfig(4);
pickup pencircle scaled 2mm; star (2cm,13,origin,blue);
endfig;

end
```

Suppose this file is named “metademo.mp”. When the file is processed by Meta-Post, it will generate four different postscript files, named metademo.1, metademo.2, metademo.3, metademo.4. These names are determined by the number parameter of “beginfig()”. If this number is nonnegative, like beginfig(0) or beginfig(10), the resulting file will be named metademo.0 or metademo.10. If this number is negative, like beginfig(-10), the resulting file will be named metademo.ps, overwriting any earlier metademo.ps file created by the source.

In its default configuration, TeXShop assumes that one of the numbers is zero. TeXShop calls the script pstopdf, which runs MetaPost and thus creates all of these postscript files. The script then converts each postscript file to a pdf file. Finally, the script renames the zeroth pdf file to the name of the source with extension “pdf”, for instance metademo.pdf and displays this figure in the preview screen.

When you are editing a MetaPost file, change the number of the figure being edited from positive to zero. TeXShop will then display this figure as it is being debugged. When you are satisfied with this figure, change its number back to positive and change the number of another figure from positive to zero.

Once MetaPost files have been created, then can be displayed like any other illustration. Pdflatex can be used if the illustrations are converted to pdf form, or TeX and Ghostscript can be used to include the postscript illustrations without conversion. For example, the four illustrations created by the above MetaPost file can be displayed by typesetting the following file with TeX and Ghostscript:

```
\documentclass[11pt]{article}
\usepackage{graphicx}
\begin{document}
Here are some illustrations.
\vspace{.2in}
\includegraphics[width=1cm]{metademo.1}
\hfill
\includegraphics[width=1cm]{metademo.2}
\hfill
\includegraphics[width=1cm]{metademo.3}
\hfill
\includegraphics[width=1cm]{metademo.4}
\hfill
\end{document}
```

It is also possible to embed MetaPost source code directly in a Latex document using the package mfpic. When this method is used, the MetaPost preference should be set to “mpost” rather than “mptopdf” in the Preferences dialog so MetaPost will run directly when the MetaPost engine is selected. A document containing MetaPost source code is first typeset with pdflatex or latex, creating a mp source file with all of the document's illustrations. This file is then compiled with MetaPost. Finally, the document is typeset again with pdflatex or latex to show the resulting illustrations.

Below is an example created by Claus Gerhardt. Save this example as "MetaPostTest". Notice the line "\opengraphsfile{MetaPostTest}" in the source. Although the mfpic package permits any name for this graph file, its name must be the same as the document name to use the following procedure in TeXShop. Typeset the document once, switch to MetaPost and typeset again, switch back to LaTeX and typeset a final time. In this process, either pdflatex or latex + ghostscript can be used for the first and third steps.

```
% This example is a shortened version of an example provided by
% Claus Gerhardt.

\documentclass[11pt]{article}
\usepackage[metapost]{mfpic}
\usepackage{amsmath}
\opengraphsfile{MetaPostTest}

\title{Brief Article}
\author{The Author}
\begin{document}
\maketitle

\begin{mfpic}[20]{-0.5}{11}{-0.5}{11}
{\drawcolor{red}\function{0,10,0.05}{10-x}}
{\drawcolor{blue}\function{0.99,10,0.05}{10/x}}
{\drawcolor{green}\dashed\lines{(0.0,4),(10,4)}}
\tlabelcolor{black}
\drawcolor{black}\ymarks[4pt]{4}
\headcolor{black}
\drawcolor{2*black}\axes
\tlabel{(4,6.5)}{$P_{\negthickspace c}$}
\tlabel{(6,6.5)}{$P_{c}$}
\tlabel(5,3.5){$A$}
\tlabel{(-.6,3.9)}{$4$}
```

```
\end{mfpic}

\begin{center}
\begin{mfpic}[15]{-2.2}{5}{-2.2}{2.2}
\store{a}{\circle{(0,0),2}}
\store{b}{\circle{(2 *sqrt 2,0),2}}
\store{c}{\arc[p]{(0,0),-45,45,2}}
\gfill[0.7white]\lclosed\mfobj{a}
\gfill[white]\lclosed\mfobj{b}
\draw\mfobj{a}\draw\mfobj{b}
\tlabel(-1,-0.3){ $A$ }
\tlabel(3,-0.3){ $B$ }
\end{mfpic}
\end{center}

\closegraphsfile
\end{document}
```

The following example, also provided by Claus Gerhardt, shows the power of MetaPost.

```
% This example was provided by Claus Gerhardt
% Most of the figures and the text are taken from G.'s book
% "Analysis I" published by International Press, Boston,
% which will appear at the beginning of 2004.

\documentclass[11pt]{amsart}
\usepackage[metapost]{mfpic}
\usepackage{amsmath}
\usepackage{amsthm}
\RequirePackage{amssymb}
\RequirePackage[mathscr]{eucal}
\opengraphsfile{MetaPostTest}

\DeclareMathOperator*{\Au}{\forall}
\DeclareMathOperator*{\Eu}{\exists}
\newcommand{\msc}{\protect\mathscr}
\newcommand\su{\subset}
\newcommand{\pri}[1]{#1^\prime}
```

```
\newcommand{\tit}[1]{\textit{\ignorespaces #1\ignorespaces}}
\newcommand{\Cc}{{\protect\mathbb C}}
\newcommand\ra{\rightarrow}
\newcommand{\abs}[1]{\lvert#1\rvert}
\newcommand{\fv}[2]{#1\hspace{0pt}_{|_{#2}}}
\newcommand{\set}[2]{\{\,#1\colon #2\,\}}
\newcommand\inn[1]{{\overset{\msp[9]\circ}{#1}}}
\newcommand{\msp}[1][1]{\mspace{#1mu}}
\newcommand{\Si}{\varSigma}

\theoremstyle{remark}
\newtheorem*{definition}{\bf Definition}
\theoremstyle{theorem}
\newtheorem*{theorem}{Theorem}

\title{An Example of Using MetaPost with mfpic}
%\author{The Author}

\begin{document}
\maketitle
\thispagestyle{empty}

\bigskip
\begin{mfpic}[20]{-0.5}{11}{-0.5}{11}
{\drawcolor{red}\function{0,10,0.05}{10-x}}
{\drawcolor{blue}\function{0.99,10,0.05}{10/x}}
{\drawcolor{green}\dashed\lines{(0.0,4),(10,4)}}
\tlabelcolor{black}
\drawcolor{black}\ymarks[4pt]{4}
\headcolor{black}
\drawcolor{2*black}\axes
\tlabel{(4,6.5)}{$P_{\negthickspace c}$}
\tlabel(5,3.5){$A$}
\tlabel{(-.6,3.9)}{$4$}
\end{mfpic}

\bigskip
\begin{definition}
```

```
Let $E,\pri E$ be metric spaces and $f:E\rightarrow \pri E$ a map.
$f$ is called   \tit{continuous} at $x_0\in E$ if
\begin{equation}\notag
\Au_{\pri U\in \msc U(f(x_0))}\; \Eu_{U\in \msc U(x_0)}\quad f(U)\su \pri U.
\end{equation}
$f$ is called continuous in $E$ if $f$ is continuous at every point of $E$.
\end{definition}

\bigskip
\begin{center}
\begin{mfpic}[15]{-4.2}{16}{-4.2}{4.2}
\store{R}{\rect{(-4,-4),(4,4)}}
\store{U}{\cyclic[.75]{(-2,-2),(0,-1.5),(2,-2.4),(1.8,2),(0.5,1.8),
     (-2.3,1.7)}}
\store{FU}{\shiftpath{(12,0)}\cyclic[.75]{(-1.5,-1.5),(0,-1.2),(2,-1.7),
     (1.8,2),(0,1.6),(-2,1)}}
\store{UU}{\shiftpath{(12,0)}\cyclic[.75]{(-2.8,-3),(0,-2),(3,-2.4),
     (2.8,2.8),(0.5,2.4),(-2.9,1.7)}}
\gfill[0.6white]\mfobj{U}
\gfill[0.8white]\mfobj{UU}
\gfill[0.6white]\mfobj{FU}
\draw\mfobj{U}
\draw\mfobj{UU}
\draw\mfobj{R}
\draw\mfobj{FU}
\arrow\curve[1]{(3,2),(6,3),(9,2)}
\point{(0,0),(12,0)}
\shiftpath{(12,0)}\mfobj{R}
\tlabel[tc](0,3.5){$E$}
\tlabel[tc](12,3.5){$E'$}
\tlabel[tl](-2,0){$U$}
\tlabel[tl](10.1,1){$f(U)$}
\tlabel[tl](9,-1){$U'$}
\tlabel[tl](0.1,0){ $x_0$ }
\tlabel[tl](12.1,0){ $f(x_0)$ }
\tlabel[tc](6.1,3.8){$f$}
\end{mfpic}
\end{center}
```

```
\noindent
\parbox[c]{7.51cm}
{The picture on the right shows the intersection of two
sets $A$ and $B$. Notice that this intersection consists of
all points which belong to both sets.}
\hfill
\begin{minipage}{40mm}
\begin{mfpic}[15]{-2.2}{5}{-2.2}{2.2}
\store{a}{\circle{(0,0),2}}
\store{b}{\circle{(2 *sqrt 2,0),2}}
\store{c}{\arc[p]{(0,0),-45,45,2}}
\store{de}{ \arc[p]{(2 *sqrt 2,0),135,225,2}}
\store{dd}{\lclosed\connect\mfobj{de}\mfobj{c}\endconnect}
\gfill[0.7white]\mfobj{dd}
\draw\mfobj{a}\draw\mfobj{b}
\tlabel(-1,-0.3){ $A$ }
\tlabel(3,-0.3){ $B$ }
\end{mfpic}
\end{minipage}

\bigskip
\begin{definition}[Complex logarithm\index{complex logarithm}]
The \tit{complex logarithm}, $\log: \Cc^* \ra S_l$, is defined by
\begin{equation}\notag
\log z=\log\abs z+i\arg_lz.
\end{equation}
It is the inverse of $\fv\exp{S_l}$, the so-called \tit{ main branch}
of the exponential function.
\end{definition}

The region of discontinuity
is now the axis
$\set{z\in\Cc^*}{\arg z=\pi}$. Thus, the exponential function
is not only bijective in the
open strip
$\inn S_l$,
but also a differentiable homeomorphism onto
```

```
$\Si=\set{z\in\Cc^*}{\arg z\neq \pi}$ with
$\pri\exp z=\exp z\neq 0$, and therefore, in view of the previous theorem,
we may conclude

\begin{theorem}
The complex logarithm is infinitely often differentiable in $\Si$
and
$\pri\log z=\frac{1}{z}$.
\end{theorem}

\begin{mfpic}[15]{0}{20}{-2.5}{3}
\gfill[0.6white]\rect{(12,-2),(20,2)}
\gfill[0.6white]\circle{(4,0),2}
\arrow[l 5pt]\lines{(4,0),(8,0)}
\arrow[l 5pt]\lines{(4,-2.5),(4,3)}
\arrow[l 5pt]\lines{(12,0),(20,0)}
\gfill[white]\circle{(4,0),0.05}
\arrow\curve[1]{(7,2.5),(10,3.5),(13,2.5)}
\penwd{1pt}
\draw[white]\lines{(0,0),(4,0)}
\penwd{0.5pt}
\arrow[l 5pt]\lines{(16,-2.5),(16,3)}
\tlabel[cr](17.2,2.3){$\pi$}
\tlabel[cr](17.2,-2.3){$-\pi$}
\tlabel[cc](10,4){ log}
\tlabel[cc](2,2.5){ $\Si$}
\end{mfpic}%

\closegraphsfile
\end{document}
```

2.22 Plist Files

Mac OS X makes extensive use of xml files; xml is a structured language closely related to html.

TeXShop uses five xml files for configuration: completion.plist, autocompletion.plist, KeyEquivalents.plist, Macros_Latex.plist, and Macros_Context.plist. These files reside inside your personal library in a folder called ~/Library/TeXShop created when TeXShop first runs. If you have modified the default TeXShop configuration and want to move TeXShop to another machine preserving your modifications, copy ~/Library/TeXShop to the new machine.

The files are used to configure the Latex Panel, Auto Completion, the Keyboard Menu Shortcuts, and the Macro menu. Details are given elsewhere in this document.

Files of type plist are ordinary text files, so they can be opened and edited with TeXShop, TextEdit, or other text editors. Editing the file is straightforward, but somewhat tedious.

If a plist file contains unicode characters, it needs to be edited and saved in UTF-8 format. Before opening such a file in TeXShop, change the TeXShop encoding preference to UTF-8. Then edit and save the file. Then change the encoding preference back to the original value. The default value is MacOSRoman if you did not reset it earlier.

If you installed the Developer distribution of Mac OS X and you double click a .plist file, it will open in a program named Property List Editor. This program is useful for editing plist files, but it is buggy and does not display all file information. In particular, comments are missing. It is better to use TeXShop.

2.23 Redefining Keyboard Menu Shortcuts

It is possible to redefine all keyboard menu shortcuts used by TeXShop. To do so, open the file ~/Library/TeXShop/Menus/KeyEquivalents.plist with TeXShop, read the comments at the top of the file, and edit the file to redefine selected menu shortcuts. Be sure to edit and save in UTF-8 format if you use Unicode characters.

3 Macros Help

3.1 Preliminaries

TeXShop macros and the Macro Editor are by Mitsuhiro Shishikura. Default Latex macros were created by Mitsuhiro Shishikura and Hirokazu Ogawa. Default Context macros were created by Hans Hagen.

The available Macros depend on the typesetting engine chosen. If this engine is LaTeX, then a series of macros suitable for LaTeX will be displayed. If the engine is ConTeXT, then macros for ConTeXt will be displayed. When TeXShop is first installed, it has macros for only these two engines; the LaTeX macros will be displayed for all engines except ConTeXt.

Macros come in two kinds. Some insert strings into the TeX source file; these macros are similar to buttons in the Latex panel. Other macros run Applescript scripts.

Macros are stored in

```
~/Library/TeXShop/Macros/Macros_Latex.plist
```

and

```
~/Library/TeXShop/Macros/Macros_Context.plist
```

If these files are missing when TeXShop runs, default macro files are created there. These files are ordinary text files, so it can be opened and inspected with TeXShop. This is usually not necessary because the files are best manipulated with the Macro editor.

The Macro editor modifies macros stored in Macros_Latex.plist and Macros_Context.plist when one of these typesetting engines is chosen. But if another typesetting engine like TeX is chosen, then the Macro editor will replace the default Latex macro set with a set defined by the user for TeX, storing the results in Macros_Tex.plist and leaving Macros_Latex.plist unchanged.

When you define useful Macros and wish to give them to others, simply open the Macro Editor and choose ¨Save selection to file...¨ in the Macro menu. This will create a file in any location you wish. To distribute your macros, send this file.

To add macros created by others to your list of macros, open the Macro Editor and choose ¨Add macros from file...¨. Then use the Macro Editor to arrange the macros as you desire.

3.2 Understanding Default Macros

The best way to understand default macros is to examine their definition with the Macro Editor. For example, consider the macro titled “Begin/End”. Suppose you wish to use an environment like the theorem environment. Type the word “theorem” and select it. Then choose the “Begin/End”macro. The word“heorem” will be replaced with the text

```
\begin{theorem}

\end{theorem}
```

with the cursor placed on the line between this pair.

Now examine the begin/end macro code:

```
\begin{#SEL#}
#INS#
\end{#SEL#}
```

Text in the macro will be inserted into the source file. Each occurrence of the string #SEL# will be replaced by the text selected when the macro was invoked. If no text was selected, #SEL# will be replaced with an empty string. The cursor will be placed at the end of the inserted text unless the text contains the string #INS#, in which case the cursor will be placed at that location.

Using this knowledge, it is easy to understand and modify the default macros.

3.3 Rearranging the Macro Menu

Open the Macro Editor and examine the macro menu on the left. It is shown in an outline view similar to the view of the file system obtained by choosing the middle button of the View tab in the Finder. To rearrange items, drag them from one spot to another. Notice that items can be placed at different levels by sliding left and right. To rename an item, select it in the outline view and type a new name in the field at the top right.

3.4 Defining New Macros

New items, new submenus, and new separators can be created by the buttoms at the bottom left. A little practice illustrates the basic behavior of the editor. Submenus can be created to any level.

Any particular item can be assigned a keyboard equivalent using the buttons on the right side of the menu. The result will immediately appear in the outline view. However, these assignments cannot duplicate key combinations already used by TeXShop menus. If they do, the new keyboard shortcut will be ignored.

3.5 AppleScript Macros

If a macro begins with the string

```
--AppleScript
```

or

```
--AppleScript direct
```

then the resulting applescript code will run when the Macro is chosen. For example, consider the macro titled "View pdf with Acrobat". Choosing this macro when the source file is active (and the pdf file has been created) will start Adobe Acrobat Reader and open the output pdf file in that program. The corresponding applescript code reads

```
--AppleScript direct
tell application "Acrobat Reader 6.0"

activate
open POSIX file #PDFPATH#

end tell
```

Commands beginning with "–AppleScript direct" are run directly by TeXShop. During the time this applescript is running, TeXShop's event loop is not active. Consequently, the script cannot call TeXShop to perform an action which might require user input. For example, it cannot ask TeXShop to run LaTeX, because if the La-

TeX file has an error, the console will appear and wait for user input, but such input would not be recognized.

Commands beginning with "–AppleScript" are run by a small auxiliary program in the TeXShop application bundle. Such commands can ask TeXShop to perform actions which might require user input, because in that case the auxiliary program will temporarily halt but TeXShop will remain active.

Thus the syntax "–AppleScript direct" is appropriate for routine macros, but"–AppleScript" may be needed for fancy ones.

3.6 Default AppleScript Macros

TeXShop comes with an extended collection of Applescripts by Will Robertson, Claus Gerhardt and others. Some of these scripts automate workflow when a series of typesetting commands must be issued in sequence. By copying and modifying the scripts, users can construct scripts appropriate for their own workflow.

In this section, we will describe some of these scripts.

Column Macros, Insert Reference, Open Quickly: These macros are by Will Robertson. The first provides a very convenient way to construct an arbitrary matrix or table. The second searches through the current file for \label{...} commands and then pops up a list from which you may insert a reference label, wrapped in an (optional) customizable LaTeX command. The final macro allows you to rapidly open any file in the directory of the current source file.

Convert to Mac, Convert to Unix, Convert to Windows: In the early days, teletype terminals were used for communication. The ascii character set still has remnants from those days; for instance 0x07 rings the teletype bell. The character 0x0a was a line feed which turned the carriage to the next line and the character 0x0d was a carriage return which pulled the carriage back to the start of the line. To get a line feed, one used the sequence 0x0d 0x0a.

After the age of teletypes ended, computer manufacturers selected different subsets of these characters to indicate a line feed. In the Unix world, 0x0a was used, in the Windows world 0x0d 0x0a was used, and in the old Macintosh Classic world 0x0d was used. Apple's guidelines for Mac OS X state that programs should be able to automatically open files using any of these conventions. Most programs including TextEdit, TeXShop, etc., follow these guidelines. In TeXShop, new files are created using Unix conventions, but if you load a file created with Mac OS Classic and then add extra lines, old portions of the document will be saved with the Classic convention and new sections will have the Unix convention.

These line feeds cause no trouble until you send a file to a friend using a different operating system. Many editors now understand multiple line feed conventions, so often you'll have no problems. But if you do, use the scripts above. Suppose the source file for a document is named MyFile.tex. Then "Convert to Mac" will create a new file named MyFile_Mac.tex with the same source code and Macintosh line feed conventions. "Convert to Unix" and "Convert to Windows" work the same way. All of these scripts call a binary program by Craig Stuart Sapp named "flip" in ~/Library/TeXShop/bin. See http://ccrma-www.stanford.edu/~craig/utility/flip/ for details.

Other Scripts-Bibliography: Processing a file with a bibliography requires multiple typesetting operations. First Latex is run to create an .aux file. Then Bibtex is run and uses this file to create .bbl and .blg files. Latex is run again to add the bibliography to the document. Latex is run a final time to update the references to the bibliography in the text.

The "Bibliography" command does all of these things one by one. First it saves the file. Then it runs

```
latex->bibtex->latex->latex
```

Finally it updates the preview display.

htlatexc, htlatexr: Tex4ht is a TeX program which converts a latex document into a web page. The source can be a standard Latex file and can include eps illustrations. The end result is an html page and a large number of gif files. The script command "htlatexc" saves the source documents, runs htlatex, and opens the resulting html file in Safari. Thus it behaves like a new TeXShop typesetting command, except that it was constructed by a user without waiting for new TeXShop code!

When using these scripts, it is not necessary to use the package tex4ht since this package will automatically be loaded. The script htlatexc calls htlatex without additional options. The script htlatexr calls it with the option "-r", which tells the program to recreate any .gifs which already exist.

pdfselectc: This script runs the shell script pdfselect to create pdf files containing only certain selected pages of a document. When it is run, a dialog appears asking for the number of pdf files to be created. Suppose we answer 3. Next a dialog appears asking for the range for the first document. Suppose we answer 5:8. A dialog appears asking for the range of the second document. Suppose we answer 10. A dialog asks for the range of the third document. Suppose we answer 20:30. We will then obtain three documents, one containing pages 5 - 8 of the original, one containing page 10, and one containing pages 20 - 30.

mpostc, mpostcpl, latex-makeindex-mpost: The script mpostc runs mpost and then pdflatex; the script mpostcpl runs pdflatex, and then mpost, and then pdflatex again. See the TeXShop help section on Context and Metapost for an example of the use of these three commands in sequence.

The script latex-makeindex-mpost saves the source and runs pdflatex, makeindex, mpost, and pdflatex again, opening relevant log files in the process.

3.7 Defining AppleScript Macros

AppleScript syntax is a subject all its own. To learn more about it, read one of several books on the subject. If you installed the developer tools which come with Mac OS X, there is an online book about AppleScript in the Developer folder.

When TeXShop interpretes an applescript macro, it first replaces any string

```
#FILEPATH#, #PDFPATH#, #DVIPATH#, #PSPATH#, #LOGPATH#, #AUXPATH#
```

with the complete path name of the source tex file, pdf file, dvi file, ps file, log file, or aux file respectively. Similarly, the strings

```
#INDPATH#, #BBLPATH#, #HTMLPATH#
```

are replaced with the complete path name of the ind file, bbl file, or html file.

In addition, any string

```
#NAMEPATH#
```

is replaced with the complete path name of the source tex file minus its extension, and any string

```
#DOCUMENTNAME#
```

is replaced with the display name of the current document. This last replacement is somewhat subtle; it gives the title of the document as shown at the top of the source window. If a document was saved with the "hide extension" box checked, #DOCUMENTNAME# will contain only the document name. But if the document was saved without checking "hide extension", #DOCUMENTNAME# will contain the document name and extension. This information can be used to locate the calling document for Applescript code as follows:

```
tell document #DOCUMENTNAME# of application "TeXShop"
latex
end tell
```

There are at least two ways to write Applescript commands. Applescript can run shell commands, so after preliminary processing with an applescript, a shell command can be called to do the actual work. TeXShop comes with several examples of this technique; some of these examples will be explained in a later help section. Applescript commands can also call built-in TeXShop commands and thus work directly. A later section will give examples of this technique.

TeXShop understands the following commands:

```
typeset
latex
tex
context
bibtex
makeindex
metapost
typesetinteractive
latexinteractive
texinteractive
contextinteractive
bibtexinteractive
makeindexinteractive
metapostinteractive
```

```
taskdone
refreshpdf
refreshtext
goto line
```

The first seven commands call TeXShop typesetting routines. These commands typeset continuously without stopping at errors. The next seven commands also call TeXShop typesetting commands, but this time if there is an error, the user is allowed to interact with the console. When a typesetting command is called, control returns to the applescript immediately without waiting until the operation is complete. The "taskdone" call returns NO if typesetting is still running, and YES when it is done. The calls "refreshpdf" and "refreshtext" cause pdf and text documents to display the latest version of their files on the screen. The "goto line" command tells the editor to select a given line; for example:

```
tell document #DOCUMENTNAME# of application "TeXShop"
goto line 37
end tell
```

3.8 Creating Dialogs

AppleScript can create dialogs and act on user input. For example, insert the following script and test its behavior.

```
-- AppleScript
-- Use a dialog to enter user defined information into an applescript.
-- There are two feedbacks possible 'text' and 'choice of buttons'
-- The returned information can be used to define corresponding variables.
-- The number of buttons may not exceed three.
-- In the three examples below it is shown how to use text return,
--      button return, and both.

activate
display dialog "Test dialog: type something below"
     default answer "Hello World!"
     buttons {"A", "B", "B"} default button "B"
     set theText to (the text returned of the result)
```

```
display dialog "Test dialog: type something below"
     default answer "Hello World!"
     buttons {"A", "B", "B"} default button "B"
     set theButton to (the button returned of the result)

display dialog "Test dialog: type something below"
     default answer "Hello World!"
     buttons {"A", "B", "B"} default button "B"
     set {theText, theButton} to {the text returned of the result,
          the button returned of the result}
```

One command in this example, “activate”, requires comment. When TeXShop is asked to run an AppleScript, it calls a separate program to run the script. That program, Scriptrunner, is in the TeXShop bundle. Any dialogs created by the script will be displayed by Scriptrunner rather than by TeXShop. The “activate” command brings Scriptrunner to the front so dialogs will appear on top of other windows.

3.9 Writing Scripts with TeXShop Typesetting Commands

If a TeX project contains a bibliography, a sequence of typesetting commands must be run to update the bibliography. Latex is run first to create an .aux file. Bibtex is run to create .bbl and .blg files from this .aux file. Latex is run again to add the bibliography to the document. Latex is run a final time to update the references to the bibliography in the document.

Your projects may contain similar sequences of typesetting commands. It is possible to use applescript to automate these sequences.

To see how to do that we’ll examine the OtherScripts-Bibliography command which comes with TeXShop. Here is the body of that command

```
--Applescript

set fileName to #FILEPATH#
if fileName is equal to ""
activate
display dialog "Please save the file first" buttons {"OK"} default button "OK"
```

```
return
end if

set frontName to #DOCUMENTNAME#
tell application "TeXShop"
save document frontName
end tell

tell document frontName of application "TeXShop"

     latexinteractive

     repeat

          delay 2
          if taskdone
          exit repeat
          end if

     end repeat

     bibtex

     repeat

          delay 2
          if taskdone
          exit repeat
          end if

     end repeat

     latex

     repeat

          delay 2
          if taskdone
```

```
                exit repeat
                end if

            end repeat

            latex

        end tell
```

The first line of this command indicates that this is an AppleScript macro. The next lines check #FILEPATH#, a parameter which gives the full path to the tex source. This parameter is an empty string when a new document has been created and not yet saved. In that case the user is asked to save the document and the script ends.

The next line tells TeXShop to save the document. Notice that we use #DOCUMENTNAME# to refer to the document in question.

The remaining commands run latexinteractive, bibtex, latex, and latex. Recall that control returns to applescript immediately after calling a typesetting command before the typesetting job is over. The repeat loop tells the script to check whether the job is complete before running another typesetting task. The line "delay 2" causes applescript to pause rather than continually asking if the task is done and thereby slowing the entire computer down.

3.10 Writing Scripts with Shell Commands

We will use the AppleScript “pdflatexc” by Claus Gerhardt to illustrate the principles involved in calling a Unix shell script from an Applescript. This particular shell script isn’t very interesting; it adds the location of the teTeX binary files to the $PATH variable and calls pdflatex to typeset.

The shell script itself is independent of TeXShop and can be run from the Terminal by typing “pdflatexc myfile.tex” provided the directory holding pdflatexc is in the search path for binaries. Here is that shell script:

```
#!/bin/tcsh
# pdflatexc
# Claus Gerhardt
#
# Usage
# pdflatexc filename.tex
set path= ($path /usr/local/teTeX/bin/powerpc-apple-darwin-current /usr/local/bi

pdflatex --shell-escape "$1"
```

Of course you are likely to write a more complicated script which performs several operations in sequence. That is when these techniques become useful.

The AppleScript used to call this shell script is more interesting. Here it is:

```
--Applescript
-- Apply only to an already saved file.
-- Claus Gerhardt, Nov. 2003
set scriptPath to
     (do shell script "dirname " & "~/Library/TeXShop/Scripts/ex")
set scriptPath to scriptPath & "/setname.scpt"
set scriptName to POSIX file scriptPath as alias
set scriptLiB to (load script scriptName)
tell scriptLib
set frontName to setname(#NAMEPATH#,#TEXPATH#)
end tell

set fileName to #TEXPATH#
set n to (number of characters of contents of fileName)
set fileNamequoted to quoted form of fileName
set baseName to do shell script "basename " & fileNamequoted
set m to (number of characters of contents of baseName)
set dirName to quoted form of
     (characters 1 thru (n - m - 1) of fileName as string)


set shellScript to "cd " & dirName & ";"
set shellScript to shellScript &
     "~/Library/TeXShop/bin/pdflatexc " & baseName
do shell script shellScript
```

```
tell document frontName
refreshpdf
end tell
```

Ignoring the introductory comments, the first seven lines of this script are a magic recipe by Claus Gerhardt to save the source file and find the name of the document in question, setting “frontName” to this name. This recipe uses a compiled script named “setpath.scpt” in ~/Library/TeXShop/Scripts to do all the hard work. Careful reading shows that these lines find the path ~/Library/TeXShop/Scripts/setname.scpt and call this script with parameters #NAMEPATH# and #TEXPATH#.

In many AppleScripts all of this could be accomplished in an easier way using the commands:

```
set frontName to #DOCUMENTNAME#
tell document frontName of application "TeXShop"
save
end tell
```

However, Gerhardt’s script has two advantages. First, his script can be called when the front document is a log file, say /Users/koch/Examples/myfile.log, that is, in a case when the front document is not the document which should receive later commands. Second, if the file has never been saved, Gerhardt’s script returns an error when trying to save, while the “save” command would cause TeXShop to hang after it put up a save dialog (see the help section “Writing Scripts with TeXShop Typesetting Commands” for details.)

The next six lines of the script define the variables dirName and baseName. If the source file is“/Users/koch/This directory/Stuff/myfile.tex”, dirName is

```
‘‘ ’/Users/koch/This directory/Stuff’ ’’
```

including the single and double quotation marks, and baseName is “myfile.tex”. A lot of this work is required so spaces can occur in the names of folders.

The next three lines call the shell script. The result is the same as typing “cd dirName; ~/Library/TeXShop/bin/pdflatexc baseName” in a Terminal, although, to be absolutely precise, dirName would have to be replaced by its unquoted form, i.e., if we use the example above, by ’/Users/koch/This directory/Stuff’ including the single quotes because of the space in the name of one directory.

Shell commands in AppleScript are issued in the form

```
do shell script "command input"
```

If one wants to combine several shell scripts, it is better to write the command in an equivalent form

```
do shell script "cmd " & "input"
```

Notice the space behind cmd and the quotes. The ampersand is a binary concatenation operator, i.e.,

```
"cmd " & "input" = "cmd input"
```

When the shell is called via an applescript, the default working directory is the root directory. The command

```
do shell script  "cd " & dirName
```

changes the directory to the directory specified in dirName; dirName is already quoted so that additional quotes are not needed.

If one would like to keep the working directory and issue further commands, then these commands may not be stated in the form "do shell script", since then a new shell would be invoked. In the terminal one would separate consecutive commands by semicolons, in AppleScript one does it by concatenating";", i.e.,

```
do shell script "cmd(1) " & "input(1)" & ";" & "cmd(2) " & "input(2)"
```

This is done in the above example: cmd(1) = cd, "input(1)" = dirName, cmd(2) = ˜/Library/TeX/bin/pdflatexc - calling the shell script -, and "input(2)" = baseName.

The lines

```
set shellScript to "cd " & dirName & ";"
set shellScript to shellScript & "~/Library/TeX/bin/pdflatexc " & baseName
```

are simply a convenient way to concatenate this sequence of commands and input into one variable.

The final three lines update the Preview window.

To be sure, several of these lines are complicated, but these lines can be copied without change into new scripts.

4 How do I configure TeXShop?

4.1 Preliminaries

Please read only the first two sections below if you are a new user installing TeXShop for the first time. If you are upgrading from a previous version, read these two and the next four sections. If you are familiar with TeXShop and want to customize it, read the remaining sections.

When TeXShop first runs, it places default preference items, macros, templates, and typesetting engines in the user's library folder. If the program is later updated, these preference items, macros, etc. are not modified because the user may have edited them. In a few cases, important changes were made in the default items; these changes are listed here so users of previous versions can modify preferences and templates appropriately.

TeXShop has a few hidden preferences for power users. Such users can also modify program behavior by installing various plist files. Details are provided below.

New versions of TeXShop sometimes require additional menu items and other interface components. These new items appear in English even when the rest of the interface has been localized for another language. If users write me with appropriate translations, I will immediately make the changes in the copy of TeXShop on my web site. Write koch@math.uoregon.edu.

4.2 Did You Install Correctly?

Starting with version 1.34, TeXShop has a feature called pdfsync: clicking on a spot in the preview window activates the corresponding source window with the appropriate source line selected. This feature requires that files named "pdfsync.sty", "pdfsync.tex", and "pdfsync4context.tex" be installed in ~/Library/texmf/tex/latex, ~/Library/texmf/tex/plain, and ~/Library/texmf/tex/context, respectively. If you install with the MacTeX install package, these files are already installed, but in /usr/local/tetex/share/texmf.local. Otherwise remove the old "pdfsync.sty" from ~/Library/texmf/tex/latex if it exists. Then find these new files in the TeXShop distribution and drag them to the appropriate folders. Here ~/Library is the Library folder in your home directory. You may have to create some or all of the folders texmf, tex, latex, plain, context.

TeXShop 1.35 comes with additional applescripts by Claus Gerhardt and Will Robertson. If you upgrade from a previous version, you will not automatically see those new scripts.

If you never edited the default macro set, you can obtain the new macros by going to the folder ~/Library/TeXShop and removing the entire Macros folder inside. The next time TeXShop starts, it will recreate this folder with the latest macros. It is not enough to remove the contents of the Macros folder; the entire folder must be removed.

If you have edited the default macros and want to add the new macro set, remove the Macros folder temporarily and let TeXShop create a new macro set. Then add your old macros using the Macro Editor.

TeXShop 1.35 comes with new templates by Will Robertson. These templates will not automatically be installed. You can obtain them by going to the folder ~/Library/TeXShop and moving the entire Templates folder inside to the Desktop. The next time TeXShop starts, it will recreate this folder with the latest Templates. You can then move old Templates you have edited from the folder on the desktop to the newly created folder ~/Library/TeXShop/Templates.

4.3 Recommended Preference Changes

Users should consider changing a small number of default preferences provided by TeXShop. Some of these changes depend on characteristics of individual displays and cannot be set until you use the program.

By default, TeXShop opens source and preview windows in the spot they last appeared. This only makes sense when you deal with one document at a time. It is usually better to open source and preview windows in a predetermined optimal position. To make that happen, typeset a simple document. Then arrange the position and size of the source and preview windows. Side by side is best if your screen is large enough. Open TeXShop preferences. In the "Document" tab, select "All Windows Start At Fixed Position" and click "Set With Current Position". Make the same change in the "Preview" tab.

This is a good time to set the magnification in the preview window. Use the Scale button at the top of this window to select the optimal magnification. Under the "Preview" tab of Preferences, type the appropriate magnification and push the "Set" button and then the "OK" button. *However* this preference is only applicable if the

"After Window Resize" preference in the right column is set to "Fixed Magnification". If it is set to "Actual Size" or "Fit to Window", then it doesn't make sense to apply a fixed magnification to the display.

Finally, you may wish to deal with a security issue. By default, TeXShop is configured to permit pdftex and pdflatex to call other programs during typesetting. When pdftex finds a graphic illustration in a format it cannot understand, it will automatically call a conversion program to convert the graphic to a useful format. For example, eps files will be converted to pdf and tif files will be converted to png.

However, the configuration allows pdftex to call ANY program. Conceivable, a user could mail you a tex source file which does unpleasant things to your system during typesetting. If this is a concern, go to the "Engine" tab in Preferences and check the box labeled "Shell Escape Warning". When this button is checked, a dialog will appear the first time a file is typeset during a TeXShop session, allowing you to turn off this feature for that file so pdftex cannot call other programs during typesetting. Thus you can use automatic conversion for your own files, but play it safe with source files from others.

4.4 Converting Graphic Formats Automatically

Support for tiff files has been removed from the latest versions of pdftex and pdflatex. However, it is possible to configure TeXShop so it will automatically convert graphics in tiff format to png format during typesetting. As a bonus, the program will convert graphics in eps format to pdf format during typesetting. In both cases the original graphic file is preserved and a new file with appropriate format is created. Tiff files must have extension ".tif" rather than '.'tiff".

New installations of TeXShop are already configured appropriately. Users who are upgrading should make the changes listed in the next two sections. They should also install the following packages, if not already present:

```
Ghostscript 8
Freetype 2
wmf
GNU gettext libraries
PNG Library
wmf
ImageMagick
```

4.5 Default Preference Items

TeXShop 1.35 and above comes with a new Find panel, which requires system 1.3 or later. Other users will still see the old panel. It is possible to revert to the old panel using TeXShop preferences, but most users will find that the new panel is more powerful; it supports regular expressions. At first this panel may not behave as you expect, but the panel remembers the various choices it presents, and you should be able to configure it to behave as you expect.

The remaining comments are for users with versions before 1.33, who may not have made these changes earlier. In the TeXShop preference panel with the Engine tab, the following changes should be made:

- The pdftex preference should be

  ```
  pdftex --shell-escape
  ```

- The pdflatex preference should be

  ```
  pdflatex --shell-escape
  ```

- The altpdftex or simpdftex preference should be

  ```
  simpdftex tex --maxpfb
  ```

- The altpdflatex or simpdftex preference should be

  ```
  simpdftex latex --maxpfb
  ```

Japanese users of TeXShop 1.26 or earlier should replace the file

```
~/Library/TeXShop/LatexPanel/completion.plist
```

with the English version, because a modification for Japanese(ShiftJIS &yen;) is no longer necessary. To make this happen, remove the completion.plist file from the folder; a new file will be created the next time TeXShop starts.

4.6 Default Latex Template

The file LatexTemplate.tex in ~/Library/TeXShop/Templates should contain the following lines if they are not already present:

```
\usepackage{graphicx}
\usepackage{epstopdf}
\DeclareGraphicsRule{.tif}{png}{.png}{`convert #1 `basename #1.tif`.png}
```

4.7 Hidden Preference Items

TeXShop adds lines which begin with the words \section, \subsection, \subsubsection, or \chapter to the Tag menu. To turn this off, run Terminal and type the following command. Change the word NO to YES to turn the behavior back on.

```
defaults write TeXShop TagSections NO
```

ConTeXt users may want additional words to be recognized and added to the Tag menu. The following preferences adds the words \subsubsubsection, \subsubsub-subsection, \part, \title, \subject, \subsubject, \subsubsubject, \subsubsubsubject, and \subsub-subsubsubject.

```
defaults write TeXShop ConTeXtTags YES
```

When syntax coloring is on, comments are colored red, commands are colored blue, and the symbols $, {, and } are colored dark green. These colors can be changed. A color is determined by the red, green, and blue components of the color; each is a number between 0.00 and 1.00. To change the color of $, {, and } to bright green, issue the following commands in Terminal:

```
defaults write TeXShop markerred 0.0

defaults write TeXShop markergreen 1.0

defaults write TeXShop markerblue 0.0
```

To change the comment color, replace "marker" with "comment" to change the command color, replace "marker" with "command".

The background color of the source window can be changed. For example, to set this background to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal:

```
defaults write TeXShop background_R 0.42

defaults write TeXShop background_G 0.39

defaults write TeXShop background_B 0.77
```

The text color of the source window can be changed. This change requires that syntax coloring be on. For example, to set this foreground color for text to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal:

```
defaults write TeXShop foreground_R 0.42
defaults write TeXShop foreground_G 0.39
defaults write TeXShop foreground_B 0.77
```

The color of the insertion point in the source window can be changed. For example, to set this insertion point color to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal:

```
defaults write TeXShop insertionpoint_R 0.42
defaults write TeXShop insertionpoint_G 0.39
defaults write TeXShop insertionpoint_B 0.77
```

By using the previous three sets of commands in combination, the source window can be made to display white text on a black background or other coloring schemes as desired.

The background color of the preview window can also be changed. For example, to set this background to (r, g, b) = (.42, .39, .77), issue the following commands in Terminal. Change these numbers to 1 to convert back to a white background. It may be easier to view colored TeX documents with a gray background. The background color will not affect printing.

```
defaults write TeXShop Pdfbackground_R 0.42
defaults write TeXShop Pdfbackground_G 0.39
defaults write TeXShop Pdfbackground_B 0.77
```

The transparency of the source, preview, and console windows can be changed. For example,

```
defaults write TeXShop ConsoleWindowAlpha 0.75
defaults write TeXShop SourceWindowAlpha 0.75
defaults write TeXShop PreviewWindowAlpha 0.75
```

Here an alpha value of 0.00 is completely transparent and an alpha value of 1.00 is completely opaque. Using these commands cautiously.

TeXShop can be configured to save a backup file whenever it saves or typesets a source file. To turn this on, run Terminal and type the following command. Change YES to NO to turn it off.

```
defaults write TeXShop KeepBackup YES
```

Occasionally during typesetting, the .aux files, .log files, and other files created automatically by TeX become corrupted and must be removed. TeXShop has a "Trash AUX Files" menu item and associated button in the console window. When either item is selected, TeXShop moves to the trash all files in the folder containing the source file with the same name as the source name, and extensions .aux, .bbl, .blg, .brf, .glo, .idx, .ilg, .ind, .ioa, .log, .log, .lot, .mtc, .mlf, .out, .pdfsync, and .toc. Other extensions can be added to this list, one by one. For instance, to add .dvi files to this list:

```
defaults write TeXShop OtherTrashExtensions -array-add "dvi"
```

To remove all added extensions and return to the original list:

```
defaults write TeXShop OtherTrashExtensions -array
```

Sometimes a more extensive cleanup is desirable. If the option key is held down while choosing “Trash AUX Files”, TeXShop uses the %&SourceDoc and “Set Project Root” mechanisms to find the root file. It then moves all files in the folder of this file and any subfolders of this folder to the trash if they have appropriate extensions, regardless of the names of the files. This behavior can be made the default behavior for “Trash AUX Files” without using the option key; issue the command

```
defaults write TeXShop AggressiveTrashAUX YES
```

TeXShop can be configured to automatically refresh pdf views when the pdf file changes. To do this, once a second it examines the date and time when the pdf was last written to see if this information has changed. The time interval between these checks can be modified. To do this, run Terminal and type the following command. Change 1.00 to the number of seconds desired.

```
defaults write TeXShop RefreshTime 1.00
```

Automatic refresh for pdf views is useful if TeXShop is configured to use an external editor, or if a tex file is opened by “Open For Preview...”. In these cases, a .tex file is opened, but TeXShop only shows the associated pdf file. TeXShop also allows pdf files to be opened directly; this is useful for a brief glance at illustrations before embedding them in a TeX document. If you want pdf files opened for an external editor to be refreshed automatically, but pdf files opened for a brief glance to be left alone, run Terminal and type the following command. The default value of this preference is YES.

```
defaults write TeXShop PdfFileRefresh NO
```

When TeXShop opens a .tex file for an external editor, it checks the dates of the tex and pdf files to make sure that the pdf output is up to date. If this output is not up to date or does not exist at all, TeXShop typesets the .tex file again. To turn off this behavior, run Terminal and type the following command.

```
defaults write TeXShop ExternalEditorTypesetAtStart NO
```

There is a hidden preference to set the default size of the matrix in the Matrix Panel:

```
defaults write TeXShop matrixsize 12
```

There is a new checkbox tool for the Preview window named "ShowSync". It is not part of the default toolkit for this window. When this item is checked, synchronization spots are show in the Preview document. The item is not shown when a preview window first appears, but this changed be changed via:

```
defaults write TeXShop ShowSyncMarks YES
```

When the preview window is updated after typesetting, it comes to the front. This behavior causes trouble for uses with an X11 editor running in Apple's X11 Window Manager. For these users, the behavior can be turned off via:

```
defaults write TeXShop BringPdfFrontOnAutomaticUpdate NO
```

TeXShop used to support a different set of source commands to determine the typesetting engine, file encoding, and root file; examples are %&latex, %&encoding= UTF-8 Unicode, %SourceDoc ../Main.tex. This syntax was a poor choice on my part and has been changed. If you have a lot of old documents, you can temporarily turn this choice back on using the command below. If you do so, the new commands will be recognized, but the old commands will also work. This preference change should only be made in an emergency.

```
defaults write TeXShop UseOldHeadingCommands YES
```

The left and right arrow keys scroll left and right if the preview page is narrower than the total page width, but otherwise they page up and down. A hidden preference changes this behavior so the left and right arrow keys always page.

```
defaults write TeXShop LeftRightArrowsAlwaysPage YES
```

The console text reporting typesetting behavior and errors shows black text. The text can be made to switch to red after the first error with a hidden preference.

```
defaults write TeXShop RedConsoleAfterError YES
```

In the editing window, a soft line break occurs after words. This behavior can be changed to "no line break" = 0, "line break after word" = 1, or "line break after characters" = 2.

```
defaults write TeXShop LineBreakMode 1
```

When the Preview window first appears, its drawer is hidden. A hidden preference changes this behavior so the drawer is visible when the Preview window first appears.

```
defaults write TeXShop PreviewDrawerOpen YES
```

When TeXShop was first released on Tiger, users ran into an annoying bug which caused the program to gradually slow to a crawl after several typesetting actions. This bug was fixed a couple of days after the release. The problem occurred when a new pdf file was loaded into the PdfKitView in the Preview window. According to Apple documentation, this should have released the previous data structure from memory. The release did occur, but it caused the program slowdown. So the bug fix consisted of tricking the system into believing that the data structures were still being used so the system didn't try to release them.

Recent investigation seems to show that this bug is not as severe in Tiger 10.4.4. Consequently a hidden preference has been added to release the data if desired. The values of this preference are

- 0 to release the data on system 10.4.3 or higher
- 1 to never release the data
- 2 to always release the data

The default value is 1, causing the program to behave as previous versions behave.

4.8 .plist Files

Mac OS X makes extensive use of xml files; xml is a structured language closely related to html.

TeXShop uses five xml files for configuration: completion.plist, autocompletion.plist, KeyEquivalents.plist, Macros_Latex.plist, and Macros_Context.plist. These files are used to configure the Latex Panel, Auto Completion, the Keyboard Menu Shortcuts, and the Macros menu. Details are given below. These files are automatically created in subfolders of ~/Library/TeXShop when TeXShop first runs.

Files of type plist are ordinary text files. They can be opened and edited with TeXShop, TextEdit, or other text editors. Each of the files except Macros.plist has a comment at the top explaining the file format. Editing is straightforward, but somewhat tedious. Macros_Latex.plist and Macros_Context.plist will never need to be edited because TeXShop has a Macro Editor built in.

If a plist file contains unicode characters, it needs to be edited and saved in UTF-8 format. Before opening such a file in TeXShop, change the TeXShop encoding preference to UTF-8. Then edit and save the file. Then change the encoding preference back to the original value. The default value is MacOSRoman if you did not reset it earlier.

Users with the Developer distribution installed will discover that double clicking a .plist file opens the file in a program named Property List Editor. Property List Editor is useful for editing plist files, but it is buggy and does not display the comments. So it is better to use TeXShop.

4.9 Modifying the Latex Panel

The Latex panel contains a “Custom” tab; up to sixteen user-defined buttons can be placed on this tab. Other buttons in the Latex panel cannot be changed, but the text inserted when the button is pressed can be changed.

The comments at the start of the file “completions.plist” explain how to make these changes. To do so, edit the file ~/Library/TeXShop/LatexPanel/completions.plist with TeXShop. Be sure to edit and save in UTF-8 format if you use Unicode characters.

4.10 Modifying Auto Completion

The preference panel contains a checkbox to turn auto completion on or off; by default it is off. When auto completion is on, typing certain characters inserts an entire string in the source file. For instance, typing ˆ inserts ˆ{ }, with the cursor positioned inside the brackets.

Auto completion elements can be modified and created by editing the file "autocompletion.plist". The comments at the start of this file explain how to make changes. To do so, edit the file ˜/Library/TeXShop/Keyboard/autocompletions.plist with TeXShop. Be sure to edit and save in UTF-8 format if you use Unicode characters.

4.11 Redefining Keyboard Menu Shortcuts

The keyboard menu shortcuts for TeXShop can be redefined. To do this, read the comments at the top of the file "KeyEquivalents.plist" and make appropriate changes. To do so, edit the file ˜/Library/TeXShop/Menus/KeyEquivalents.plist with TeXShop. Be sure to edit and save in UTF-8 format if you use Unicode characters.

4.12 Modifying the Macros Menu

TeXShop contains a Macro Editor, so it should never be necessary to directly modify the Macros.plist file.

5 Credits

TeXShop is produced under the GPL public license. Coordination by:

- Richard Koch
- Dirk Olmes
- Gerben Wierda

Code by:

- Richard Koch
- Dirk Olmes
- Gerben Wierda
- Mitsuhiro Shishikura
- Seiji Zenitani
- Isao Sonobe
- Martin Kerz
- Michael Witten
- Norman Gall
- Anton Leuski
- Jérôme Laurens
- Piero D'Ancona
- Geoffroy Lenglin
- Jonas Zimmermann
- Sean Luke
- Max Horn
- John Nairn
- Greg Landweber

- Nicolás Ojeda Bär
- Martin Heusse
- Sarah Childers
- Makoto Inoue
- Sven A. Schmidt
- Georg Klein
- David Reitter
- Maarten Sneep
- Kevin Ballard
- Elliott Hughes
- Claus Gerhardt
- Sebastian Siedentopf
- Will Robertson
- Stefan Walsen
- Yu Itoh
- Koichi Inoue

Localization by:

- Maarten Sneep
- Jérôme Laurens
- Hendrik Chaltin
- Keith J. Schultz
- Sascha Beverungen
- Martin Kerz
- Max Horn
- Giuseppe Carlino

- Nicola Vitacolonna
- Seiji Zenitani
- Yoshihisa Okazaki
- Paulo Abreu
- Andrei Teleman
- Roxana Tenea Teleman
- Juan Luis Varona Malumbres

Contact Information:

Richard Koch

Mathematics Department

University of Oregon

Eugene, Oregon 97403

koch@math.uoregon.edu

Dirk Olmes

dirk@xanthippe.ping.de

Gerben Wierda (for teTeX only)

Gerben_Wierda@rna.nl

A Appendix: What's New?

Versions 1.42 and 2.05 add extra features to TeXShop and fix some bugs.

New features in 2.05:

- TeXShop 2.05 is now a Universal Binary, containing code for both the PowerPC and Intel processors. The system automatically runs the correct version of the code.
- The default TeXShop preference for the teTeX binary directory is

  /usr/local/teTeX/bin/powerpc-apple-darwin-current

  But if users have installed Gerben Wierda's latest TeX redistribution, they also have Intel binaries. On Intel machines, the teTeX binary directory should then be

  /usr/local/teTeX/bin/i386-apple-darwin-current

  TeXShop now has code to make this change automatically. When the program first starts, it calls "uname -p" to determine the current processor. If the result is `"i386"`, then TeXShop permanently changes the above preference to

  /usr/local/teTeX/bin/i386-apple-darwin-current

  However this change is not made if the user has manually changed the default

  /usr/local/teTeX/bin/powerpc-apple-darwin-current

  to some other location. Thus if a user has the Fink teTeX distribution or some other distribution and has set the teTeX binary directory preference to point to it, the preference will not be changed.
- When TeXShop was first released on Tiger, users ran into an annoying bug which caused the program to gradually slow to a crawl after several typesetting actions. This bug was fixed a couple of days after the release. The problem

occurred when a new pdf file was loaded into the PdfKitView in the Preview window. According to Apple documentation, this should have released the previous data structure from memory. The release did occur, but it caused the program slowdown. So the bug fix consisted of tricking the system into believing that the data structures were still being used so the system didn't try to release them.

A side effect was that memory gradually filled up and some users learned that they needed to quit TeXShop and restart after each day's work.

Recent investigation seemed to show that this bug is fixed in Tiger 10.4.3. Consequently late versions of TeXShop test which system is running and release the old data structures when the system is at least 10.4.3, but not otherwise.

This behavior is controlled by a new hidden Preference item:

```
defaults write TeXShop ReleaseDocumentClasses 0
```

The value 0 causes the program to behave as just described. If the value is 1, the old data structures are never released and the program behaves exactly as earlier versions of TeXShop 2. If the value is 2, old data structures are always released.

However, still more recent experience suggests that the problem remains, so the default value is currently set to 1. Further investigation is underway.

- Michael Witten, a student at MIT, added multiple wrapping modes to TeXShop. Users can choose "no wrapping" so lines continue right until the user pushes ENTER, or "word wrapping" so text is wrapped at word boundaries (this was the prior behavior), or "character wrapping" so text wraps exactly at the last possible character. Note that these wrappings are "soft"; resizing the window will change the wrapping. The default wrapping is "word wrapping" but a menu command allows the wrapping to be changed. Moreover, a hidden preference allows the default wrapping to be changed:

  ```
  defaults write TeXShop LineBreakMode 1
  ```

  where "None" = 0, "Word Wrap" = 1, and "Character Wrap" = 2. Wrapping is done at the right side of the window unless the ruler is active; if it is, wrapping is done at the "right marker"

- Witten also added the command "Hard Wrap". If a paragraph is selected, this command inserts hard wrapping commands at the right side of this paragraph.

After this step, resizing the window leaves the wraps fixed. This is useful if you send source to a colleague whose editor has fixed width and no wrapping. If the "Hard Wrap" command is chosen but no selection has been made, the hard wrap applies to the entire document. Note that "Hard Wrap" is undoable.

- The TeXShop Application Bundle now contains Norman Gall's TeX-mdimporter Spotlight importer. Files with extensions .tex, .latex, .ltx, .ctx. and .texi will be indexed by Spotlight when saved. In systems 10.4 through 10.4.2, TeXShop files were automatically indexed because they have type TEXT, but Apple changed this procedure in 10.4.3. So the importer is now required to index files. The system automatically recognizes the importer when TeXShop is first installed. It is then used to index .tex, .ltx, etc. files even if TeXShop is not running.

  If Gall later updates the importer and you install the new version in

  ```
  ~/Library/Spotlight
  ```

  or other canonical spots, the updated version will be used rather than the version in the TeXShop bundle because Apple's importer search routines use importers in bundles as a last resort.

  Gall's importer was not written with TeXShop in mind, but is instead designed to be used by all TeX editors and front-ends; the hope is that there will be a universal importer rather than a different one for each front end. For the latest version, see http://www.spookyhill.net/~gall/latex.

- Added ISO Latin 9 encoding

- Now (apple)-[and (apple)-] act differently in the editing window and preview window. In the editing window they are "unindent" and "indent". In the preview window they are "back" and "forward". A disadvantage is that it was not possible to add these commands to the menus, so users need to remember these abbreviations. This change was requested by users with German and other keyboards on which (apple)-{ cycle through windows. Users with English keyboards cycle through windows with (apple)-‘.

- Added a new hidden preference

  ```
  defaults write TeXShop LeftRightArrowsAlwaysPage YES
  ```

  The default value is NO. When set to YES, the left and right arrows scroll by a page even if the horizontal school bar is active.

- Changed the English under the Preview tab of Preferences from "After Window Review" to "Magnification Style" and changed "Preview Window Magnification" to "Preview Window Fixed Magnification" to more carefully explain the function of these preference items. Notice that TeXShop only uses the Preview Magnification value if the Magnification Style has been set to "Fixed Magnification".

- Added a new item

  ```
  %!TEX projectfile =
  ```

  which can be added to the top of TeX source files. This change is primarily for ConTeXt users so they can use the new sync method. When synching from the preview window to the source window, TeXShop needs to know all sources file for the document being previewed so it can open source files not currently open if necessary. It does this by parsing the root document, looking for \include and \input lines. But ConTeXt uses different commands to input files. The new syntax allows ConTeXt users to directly indicate in the root document which additional source files need to be searched. Here are examples:

  ```
  %!TEX projectfile = /Users/koch/MyDoc/chapter1.tex

  %!TEX projectfile = chapter2.tex

  %!TEX projectfile = ../chapter3.tex
  ```

- Added .Rnw as an extension TeXShop can edit. This was a request of Paolo Bosetti, who uses TeX and R and Sweave together.

- When a user tries to open a .dvi file, TeXShop runs a script to convert the dvi file to a pdf file. In previous versions, it ran a different script when the .dvi was in a writeable directory than when its directory was not writeable. But Gerben Wierda has revised the simpdftex script to handle both cases, so now TeXShop always calls simpdftex to do the conversion. Actually it calls the TeX + dvips + distiller script which is set in Preferences; this preference will be simpdftex if the user has a fairly recent TeX distribution.

- Added a new hidden Preference

  ```
  defaults write TeXShop RedConsoleAfterError NO
  ```

  If this default is YES, then after the first error the remaining text in the console will be red. The default value is NO.

- Slightly modified the appearnce of the Console window, particularly in English. More work is needed here. Note that the console window can be resized and relocated, and the system will remember this new size and location when TeXShop is restarted.
- In version 2.05, fixed "undo past a save" using the new Tiger command

  ```
  [textView breakUndoCoalescing]
  ```

  Previous versions of TeXShop allowed users to undo past a save command. But this required "tricky code" and one effect of the trick was that TeXShop sometimes lost track of whether the current state of the document had previously been saved, and so didn't save the document before typesetting, giving strange results in the preview window.
- Versions 2.00 through 2.04 of TeXShop sometimes had trouble remembering new preference settings; it was necessary to set them several times before they "took." This is fixed.

New features in 1.42 and 2.05:

- The commands "altpdftex" and "altpdflatex" in Gerben Wierda's TeX distribution have changed to "simpdftex tex" and "simpdftex latex". These are now the default preference values for new TeXShop installations.

  Moreover, the first time a user tries to typeset in the "tex + ghostscript" mode, TeXShop will check these preference items, and change them if necessary. It does this by determining whether "simpdftex" is in the TeX binary directory. If so, and if the command in the TeX + dvips + distiller "TeX Program" field in the TeXShop Engine Preferences is "altpdftex", then this field is changed to "simpdftex tex". Any additional flags in the preference field are retained. At the same time, the "Latex Program" field is changed. If it is "altpdflatex", it is changed to "simpdftex latex", retaining any additional flags.
- Added ISO Latin 9 encoding
- New German localization and help files.
- New Japanese Help by Yoshihisa Okazaki with help from Seiji Zenitani.
- New Spanish help files and localization.
- Uses version 1.2.4 of OgreKit.

- Conversion of eps, ps, and dvi files to pdf (caused by opening such a file) now works even if the path to the file has folders whose names contain spaces. In all three cases, the file can now be in a folder without write permission.
- Trash AUX files now removes files with more extensions: cos, idv, 4ct, 4tc, lg, xref, ttt, fff, ent, wrm.
- If the user tries to open a file with UTF-8 Unicode encoding, but the file is not a legal utf8 file, a dialog now appears warning of the problem, and the file is opened with MacOSXRoman encoding.
- Bib files are promoted to full class citizens; text can be dragged to them, syntax coloring works, etc.
- A remark: users have reported that they can no longer "find" words in the console window. Actually, they can. This window has two portions. When typesetting ends, the bottom portion is active, so the find panel searches that portion. To activate the top, click on it. Then "find" works.
- Added a preference to control first mouse behavior: "Select on Activate." When this is YES, a click in the source window will also set the insertion point to the click point. If it is NO, a second click is required to change the insertion point.
- Added pdf to the types of files TeXShop can edit, and added a pdf icon. This allows TeXShop to be chosen as the default pdf viewer.
- Added Lilypond, abc, and bst as extensions that can be edited.
- Added code by David Reitter so that selecting the word \int, etc., selects the beginning "\" as well.

Bugs fixed:

- Japanese Image Copy Type Preference failed due to incorrect localization. Now fixed.
- Command Completion's configuration file is now loaded and saved in UTF-8 Unicode.
- Errors fixed in pdfsync.

- When typesetting engines are called, they are now passed the program filename with extension, rather than just the filename.
- In 1.35, it was only possible to switch between the OgreKit Find panel and the Apple Find panel in the English localization. This is fixed.
- The menu item to bring up the statistics panel was only in the English version. Now it is in all versions.
- In all display modes except single page mode, a black border is drawn around each pdf page. Previously this border was slightly inside the page, cutting off a slight border around the page. Now it is just outside the page. Thanks to Scott Ranby for pointing out this error.
- The "%!TEX TS-program" and "%!TEX root =" commands now work when used with an external editor.

Versions 1.37 and 2.00 add extra features to TeXShop and fix some bugs.

New features in 2.00:

- A new synchronization method has been added to TeXShop, using the ability in Tiger to search for strings in pdf files. The new method does not require including a pdfsync.sty file, so it works out of the box on files typeset using any engine: pdftex or pdflatex, TeX + ghoscript or LaTeX + ghostscript, XeTeX, and other engines.
- Click on a word or phrase in the source file. TeXShop will scroll the preview window to the corresponding phrase and circle it in red. Click on a word or phrase in the preview window. TeXShop will open the corresponding source file if it is not already open, scroll the source to the appropriate spot, and highlight the source phrase in yellow.
- A new TeXShop preference item selects the synchronization method to be used: the old pdfsync method, the new search method, or a combination in which the new search is used, but the program falls back on pdfsync if the new search does not succeed.
- TeXShop 2.00 uses Apple's PDFKit to display the pdf preview window. This software is introduced in system 10.4.0 (Tiger), so TeXShop 2.00 requires Tiger.

PDFKit makes the following new features possible:

- TeXShop supports hyperlinks. To activate this feature, add the line

  ```
  \usepackage[colorlinks=true, pdfstartview=FitV
  linkcolor=blue, citecolor=blue, urlcolor=blue]{hyperref}
  ```

  to the heading of the source document. Links to external web sites can then be added to tex documents using commands like

  ```
  \href{http://www.uoregon.edu/~koch/}{Koch homepage}
  ```

  Links to other portions of the tex document can be added using commands like

  ```
  \hyperlink{lemniscate}{Graph of Lemniscate}
  ```

  where the tag "lemniscate" is created using a command like

  ```
  \hypertarget{lemniscate}{}
  ```

  To navigate with these links, choose the new "text" tool and clink on the colored links.

- The preview window toolbar contains "Back" and "Forward" buttons so one can jump to a spot using a link, and then jump back.

- Hyperref.sty automatically adds links to citations, so readers can rapidly jump from a citation to the corresponding bibliography entry.

- The hyperref package also creates a document outline. For example, the main outline of a book is a list of chapters; each chapter entry contains a list of sections, and so forth. To see this outline and navigate through the document with it, use the new "drawer" tool to display the pdf window's drawer.

- The text tool can be used to select a portion of text in the preview window and copy this selection to an editor. This differs from the "pdf selection tool" which copies a portion of the document as a pdf illustration — the text tool copies editable text.

- Searching the pdf preview is supported. Use the search tools in the bottom half of the window's drawer.

  - PDFKit brings additional polish to the display of pdf documents. It supports documents with isolated rotated pages, and correctly prints rotated pages in landscape mode. It supports liveupdate of window resizing if the magnification preference is set to "fit to window." Etc.

New features in 2.00 and 1.37: The remaining new features are available in both new versions.

- Earlier versions of TeXShop allowed users to set the typesetting engine of a file, its encoding, and its root file by adding appropriate comments to the top of the source file. For example, the following commands set the typesetting engine to xelatex, the encoding to UTF-8 Unicode, and the root file to ../Main.tex:

  ```
  %&xelatex

  %&encoding= UTF-8 Unicode

  %SourceDoc ../Main.tex
  ```

  But this syntax was a mistake because the symbols "%&" are reserved for the use of TeX.

- In versions 1.37 and 2.00 of TeXShop, the syntax has been changed to the following:

  ```
  %!TEX TS-program = xelatex

  %!TEX encoding = UTF-8 Unicode

  %!TEX root = ../Main.tex
  ```

  It you used the earlier facility, you need to change your old source files to the new syntax. I'm very sorry to cause this work, but the change is really necessary.

- If you are in the middle of a project and cannot make the change now, you can temporarily set a hidden preference to revert to the old syntax. To do so, open Apple's Terminal program and type

  ```
  defaults write TeXShop UseOldHeadingCommands YES
  ```

  Once this is done, the new commands will be recognized but the old commands will also work. However, I recommend turning this preference off as soon as possible.

- Commands have been added to the default Macros menu which insert the symbols "%!TEX TS-program = " and "%!TEX encoding = " and "%!TEX root = " into the source document; new users will see these entries. Users who are upgrading can easily add these symbols as well. To add the program entry, choose "Open Macro Editor" under the Macro menu. Click the "New Item" button, name the item "Program" and set its content to

  ```
  %!TEX TS-program = #INS#
  ```

  Repeat for the "Encoding" and "Root" items.
- Martin Kerz added a "Check for Updates..." command to TeXShop. He also designed the new TeXShop web page. Thanks!
- OgreKit for searching has been upgraded to the latest 2.0.1 version.
- Additional filetypes can be edited, including files with extensions "abc", "bst", "bib", "lp", and "pdf". The addition of "pdf" allows TeXShop to be chosen as the default pdf viewer. A new pdf icon has been created so the system can use it on pdf files it will display in TeXShop.
- TeXShop has always had a "Revert To Saved" item under the File menu, but it has never worked! Sorry. This is fixed.
- David Reitter modified the selection code so if the user clicks on a control word like \gamma, the initial "\" symbol will also be chosen.
- A new preference item "Select on Activate" was added. When this item is checked, a mouse click on the text window will select this window and also place the cursor at the spot that was clicked. If the item is not checked, the initial mouse click will only activate the window. A separate click is then needed to position the cursor.
- New hidden preferences were added for users who change the default foreground and background colors of the editing window. These preferences set the color of the insertion point (without these preferences, the insertion point could become invisible). To set the insertion point to white, for example,

  ```
  defaults write TeXShop insertionpoint_R 1.0

  defaults write TeXShop insertionpoint_G 1.0

  defaults write TeXShop insertionpoint_B 1.0
  ```
- When a file is typeset, all open changed files with the same root are first saved.

- Improvements were made in the pdfsync code.
- In previous versions, it was possible to add a Macro button to the Preview Window Toolbar in English, but not in other localizations. This is fixed thanks to Juan Luis Varona.
- New Japanese Help by Yoshihisa Okazaki with help from Seiji Zenitani.
- A proxy icon is now added to the title of the preview window. This icon can be dragged to the desktop or other folders to create a copy of the pdf file. The source window has always had a proxy icon. Thanks to Rene Donner for suggesting this feature and explaining how to implement it.

Version 1.36 was never released.

Version 1.35 adds extra features to TeXShop and fixes some bugs.

New features:

- An important recent development is the release of XeTeX and XeLaTeX by Jonathan Kew. See

  http://scripts.sil.org/xetex

  XeTeX is not part of Gerben Wierda's standard installation, but it is available with Wierda's i-Installer as an optional install directly from Jonathan Kew. XeTeX can access Macintosh fonts directly, so TeX documents can be written with Lucida Grande, Zapfino, and any other Mac font. Moreover, XeTeX understands Unicode, so for example users can type Arabic into the source window from right to left, typeset with TeX, and obtain Arabic in the output window. In particular, XeTeX source documents have UTF-8 Unicode encoding.

  TeXShop 1.35 supports XeTeX directly as follows:

  - a) XeTeX and XeLaTeX are now available in the pull-down typesetting menu on the source window
  - b) Using preferences, a user can make XeTeX or XeLaTeX the default typesetting option

- c) If one of the first ten lines of the source has the form

  ```
  %!TEX encoding = UTF-8 Unicode
  ```

  then that file will be loaded and saved with UTF-8 Unicode encoding, regardless of the default encoding chosen for other documents

- d) If the first line of the source has the form

  ```
  %!TEX TS-program = xetex
  ```

  or

  ```
  %!TEX TS-program = xelatex
  ```

  then the appropriate program will be used regardless of the typesetting option chosen.

• These XeTeX features form a special case of a new general method for adding typesetting engines to TeXShop. There is a now a folder in ˜/Library/TeXShop named Engines; the files in this folder are shell scripts which call typesetting programs. When TeXShop first starts, it examines this folder and adds the script names of files it contains to the pull-down typesetting menu. Choosing one of these items and pushing the Typeset button calls the script. Users can write their own scripts and add them to the Engines folder.

  Items in ˜/Library/TeXShop/Engines can be chosen as default typesetting method in TeXShop Preferences.

  The typesetting program can be set in the source code by writing one of the following on any of the first twenty lines of the source:

  ```
  %!TEX TS-program = tex

  %!TEX TS-program = latex

  %!TEX TS-program = pdftex

  %!TEX TS-program = pdflatex

  %!TEX TS-program = personaltex

  %!TEX TS-program = personallatex
  ```

  This new syntax also works for any new typesetting engine added to ˜/Library/TeXShop/Engines. For example, %!TEX TS-program = xelatex chooses XeLaTeX.

The encoding used to open or save a file can be set by writing a line of the form

```
%!TEX encoding = UTF-8 Unicode
```

as one of the first 20 lines of a source document. Any supported encoding is allowed; TeXShop's Help Files list the string which must appear on the right for each of these encodings. To bypass this behavior, hold down the option key while opening a file.

The old SourceDoc syntax has been modified to conform with the above changes. For example, to set the root file to ../Main.tex, write

```
%!TEX root = ../Main.tex
```

(The old syntax is still supported for setting the program, encoding, and SourceDoc, but users are urged to switch to this new syntax.)

- TeXShop has a new Find panel by Isao Sonobe. This panel supports regular expressions. Users can switch between the new panel and the original one in Preferences. The Find panel depends on OgreKit, a Cocoa framework for handling regular expressions by Sonobe. See

  http://www-gauge.scphys.kyoto-u.ac.jp/~sonobe/OgreKit/

  OgreKit is distributed using a slightly modified version of the BSD license. This license can be found in the Documentation included directly in the OgreKit Framework folder in the TeXShop source distribution. OgreKit requires Panther, so the new panel will only appear on machines running Panther.

  There are many nice features in this new Find panel, which users can discover for themselves. OgreKit modifies the "Find" menu submenu of the TeXShop Edit menu, replacing it with a more extensive menu. This might be confusing to Localizers, because the menu in the TeXShop nib file is not the menu they will see when TeXShop is running. The Find menu in the nib file should not be modified because it will be active in system 10.2. Instead the corresponding menu in OgreKit needs to be localized in the TeXShop source. The Find panel presents buttons controlling how it will find words; the settings of the buttons will be remembered from session to session. Adjust them until Find works as expected and then relax.

- TeXShop 1.35 is distributed with the latest pdfsync.sty by Piero d'Ancona and J. Laurens. This fixes typesetting problems caused by the version distributed with TeXShop 1.34. TeXShop did not keep up with the changes by d'Ancona and Laurens for several months; sorry!

In the new version, \include and \input are supported; to use the second, the syntax \input{thisfile} must be used rather than the syntax \input thisfile. The new version supports \pdfsync, \pdfsyncstart, and \pdfsyncstop. Use the first of these commands at any spot where you want to reference a point. If pdfsync breaks your code, enclose the offending section in a \pdfsyncstop, \pdfsyncstart pair.

- Suppose you are typesetting myfile.tex. Pdfsync creates a file named myfile.pdfsync containing synchronization data. Roughly speaking, each data entry describes a synchronization point as follows:

  - the page number of the output where the point occurs
  - the location on this page
  - the name of the source file producing this particular output
  - the line number in this source file for this particular output

  There is a way to get TeXShop to display these synchronization points. The preview window toolbar has a new checkbox item called SyncMarks. By default, this item is not shown; use Customize Toolbar in the Window menu to select it. When the checkbox is checked, synchornization points are shown.

  By default, this item will not be checked when the Preview window first appears. A hidden preference item can change this:

  ```
  defaults write TeXShop ShowSyncMarks YES
  ```

- TeXShop 1.35 has new matrix code by Jonas Zimmermann. The Matrix Panel now makes tables. Examine the panel to find all of the new features. There is a hidden preference to set the default size of the matrix:

  ```
  defaults write TeXShop matrixsize 12
  ```

  A very small number of users may have modified "matrixpanel.plist" in ~/Library/TeXShop/MatrixPanel. This plist has been extended; the new list is called "matrixpanel_1.plist". Please edit this file to add your changes.

- In previous versions of TeXShop, if you clicked elsewhere and then clicked on the edit window to edit, you would need to click twice to correctly position the cursor. This is now changed; the first click in the edit text is recognized and positions the cursor.

When applescript runs under the Macro menu, it starts a small second application embedded in the TeXShop folder to actually run the script. That is because when a command like "latex" runs, and there is an error on the source, the console appears to accept user input, but the TeXShop event loop is not running during the applescript action, so no user input can occur.

Many applescripts do not have this problem. TeXShop now allows users to begin applescript macros with the command

```
-- applescript direct
```

When written this way, the script will be run directly by TeXShop rather than by the second small application.

- Added a menu command "Trash AUX Files" and a button on the console "Trash AUX Files." When involked, these commands move to the trash all files in the current source directory with the same name as the source file and extensions aux, bbl, blg, brf, glo, idx, ilg, ind, ioa, lof, log, lot, mtc, mlf, out, pdfsync, and toc. Thanks to Will Robertson for suggesting this command and producing this list of extensions.

  Additional extensions can be added to this list with a hidden preference. To add "dvi" to the list

  ```
  defaults write TeXShop OtherTrashExtensions -array-add "dvi"
  ```

  Several such extensions can be added in this way, one by one. To remove all additions

  ```
  defaults write TeXShop OtherTrashExtensions -array
  ```

  The original list of extensions above will always remain active.

  Suppose a book project has a main.tex file in a folder, and then chapters in subfolders which are accessed using commands like \include{chapter1/chapter1.tex}. When this book is typeset, main.aux and other files will appear in the primary folder, and chapter1.aux will appear in a subfolder. So the "Trash AUX Files" command does not do a complete cleanup. But if the option key is pressed when the menu item is chosen or the button on the console window is pressed, then

  - a) SourceDoc and Root File information will be used to find the root document
  - b) All files with appropriate extensions listed above will be moved to the trash from this folder and all subfolders, even if the name does not agree with the name of the root file.

  Some users may want to throw caution to the winds and arrange that "Trash AUX Files" always performs this more extensive cleanup. A hidden preference allows this:

  ```
  defaults write TeXShop AggressiveTrashAUX YES
  ```

- Added new templates by Will Robertson. These are heavily commented. It is intended that users will edit them to fit their own requirements. The templates are only installed if TeXShop is running for the first time, or if the Templates folder is completely removed from ~/Library/TeXShop. But Will's templates are in a folder named "More" in the TeXShop distribution; old users can obtain them by moving "More" to ~/Library/TeXShop/Templates.
- Added new macros by Will Robertson to create tables and arrays, to insert a reference, and to open other project files quickly. These macros have been praised on the TeX-On-MacOSX mailing list. The macros are available for new users; older users can obtain them by following simple instructions which come with TeXShop 1.35.
- Added a macro to examine files in the teTeX tree. For example, if "article.sty" is typed in the dialog produced by the Macro, kpsewhich is used to find this file in the tree and open it in TeXShop.
- There are hidden preferences to set the color of the text in the source window

  ```
  defaults write TeXShop foreground_R 0.3

  defaults write TeXShop foreground_G 0.3

  defaults write TeXShop foreground_B 0.3
  ```

  This color will show if syntax coloring is on; otherwise it will be black and then color can be selected in the Font menu.

  When used with existing hidden preferences to set the source window background color, these commands can be used to write source as white on black, or with other color schemes. TeXShop has new macros to set the source window colors, and to reset to default colors.

- There are hidden preferences to make the source, preview, and console windows partly transparent.

  ```
  defaults write TeXShop ConsoleWindowAlpha 0.75

  defaults write TeXShop SourceWindowAlpha 0.75

  defaults write TeXShop PreviewWindowAlpha 0.75
  ```

  Here an alpha value of 0.00 is completely transparent and an alpha value of 1.00 is completely opaque. Use these commands cautiously!

- TeXShop now has a Statistics panel, which lists the number of words, lines, and characters in a document. This is obtained by calling

  ```
  detex myfile | wc
  ```

  When first called, the document on disk is tested. After changes are made to the document, the "update" button saves the document and calls detex again. The detex command removes tex commands, but the word count is still only approximate. Input and include files are counted by this command.

- New German Help by Martin Kerz. Kerz also redesigned the TeXShop Help Window to follow Apple's current guidelines. These changes appear in English and German, but may not appear in other localizations.

- When a file is drag-and-dropped, any alias is now resolved. Thus alias graphic files (and other files) can be used provided they are dragged and dropped to the source. Alias files will not work if their names are typed directly because the tex engine does not understand aliases (it does understand symbolic links).

- Several changes were made in the Japanese portions of the code by Seiji Zenitani, with help from Yu Itoh and Koichi Inoue. There is a new Japanese encoding, Shift JIS X0213, which will become a new standard in Japan. There is now utf.sty support for pTeX. Before this change, Japanese pTeX supported only 6000 Kanji characters, but utf.sty supports more than 20,300 characters. This support is turned on by a preference item in Misc. When on, TeXShop exports non-ptex chracters as utf.sty codes. For example, unicode characters become \UTF(Hex code) and non-unicode characters become \CID(glyph ID).

- Zenitani also added new Japanese default settings. Previously, Japanese ptex distributors provided their own "altpdflatex" scripts, which was confusing for beginning users. This new version of TeXShop bundles "altpdflatex-for-ptex" scripts and installs them in ~/Library/TeXShop/bin. The new Japanese default settings in Preferences automatically set up TeXShop to use these new scripts.

- Added the following Chinese encodings at the request of Adam Si: Mac Chinese Traditional, Mac Chinese Simplified, DOS Chinese Traditional, DOS Chinese Simplified, GBK, GB 2312, and GB 18030.
- Added a new Japanese help system by Yoshihisa Okazaki.
- Added new Spanish localization and help.
- TeXShop can now open and write files with extension .dn and .engine.
- There is a hidden preference

  ```
  defaults write TeXShop BringPdfFrontOnAutomaticUpdate NO
  ```

  which causes the pdf window to remain where it is when it automatically updates and is used with an external editor. This preference was requested by a user with an X11 editor and only seems necessary in this case.
- Users sometimes upgrade Mac OS X via "archive and install". After the installation, TeXShop remains but teTeX is blown away. The first time such users typeset a file, they see an error dialog reporting that "pdflatex cannot be found." This error dialog has been revised to explain more clearly the likely cause, and resolution, of the problem.
- The TeXShop web page now contains a "LaTeX Documentation" section listing recommended books and links to free LaTeX guides on the internet. The web page also makes available a number of short LaTeX example files by Will Robertson.

Bugs fixed:

- When a large number of windows were open, switching from one window to another took a long time and yielded a spinning disk; TeXShop was completely unresponsive during this time. This slowdown was caused by a bug in the Macro code. The problem is now fixed.
- If a dvi file was opened in a directory without write permission, TeXShop could not create and display a corresponding pdf file. Now it will create the pdf file

in a temporary directory and display it.

- If the abort button was pushed in the console and the user later typed and pushed RETURN, the program crashed. Fixed.
- Two minor error dialogs, which should have appeared in a window, instead appeared on the desktop with an inoperable "OK" button. This is fixed.
- In localizations other than English, attempts to get the applescript dictionary crashed TeXShop. Fixed.
- The Bibtex, etc., tools in the source window did not work if a file had a root file. Fixed.

Version 1.34 adds extra features to TeXShop and fixes some bugs.

New features:

- Added a Matrix Panel by Jonas Zimmermann, zimmerleut@gmx.de. Many thanks.
- At the request of Claus Gerhardt, added an extra applescript command, "goto line". For instance

```
tell front document of application"TeXShop"
goto line 15
end tell
```

- Added a first cut at pdfsync. Clicking on a spot in the pdf file while holding down the command key takes the user to the corresponding point in the source file. If the source file has include files, this operation will open the appropriate include file and take the user to a point in that file. Read the help file "General Help: Pdfsync" for important details. This change depends on pdfsync.sty, a file create by Piero D'Ancona with improvements by Jérôme Laurens.
- Also added pdfsync the other way. Clicking on a spot in the source file (including source files with root files) while holding down the command key will select the corresponding page in the pdf file.

Bugs fixed:

- The TeX, Latex, Bibtex, Makeindex, Metapost, Context, and Metafont buttons on the toolbar reset the default typesetting engine. This no longer happens. Thus it is possible to hit the Bibtex button and then hit command-T to typeset again.
- The abort button on the console window did not stay fixed when the window was resized. Thanks to Sean Luke for pointing out these first two errors.
- The command-1 keystroke switches back and forth between the source and preview windows. In previous versions, this did not work when a source window had a root file set by myfile.texshop or %SourceDoc. This is now fixed. Clicking command-1 while in the source brings up the corresponding preview window. Clicking command-1 again brings up the original source file. A given preview window may have several source files. Command-1 will bring up the last source file which was switched to the preview using command-1, or the root source file if there was no previous switch.
- New Spanish help files by Juan Luis Varona Malumbres. Thanks.
- Printing now respects the "scale" setting in Page Setup. It does not respect the "paper size" setting since paper size is set in teTeX. Printing works like this: the dimensions of the printed document are set by tex and encoded in the pdf file; this pdf is resized by the scale factor if this factor is not 100%, but otherwise is placed full size and centered on the printed page; usually the document size and printed page size are the same, but in rare cases when they are not, the edges of the document might be cut off.

Version 1.33 adds extra features to TeXShop and fixes some bugs.

New features:

- The changes in version 1.33 support improved apple scripting, better macro support, and improved interaction with external editors.
- Improved macro support was prompted by Claus Gerhardt, who wrote several useful scripts included with version 1.33. For example, one script calls htlatex to typeset a latex file for the web. The script saves the source, typesets, and opens the resulting html file in Safari. Thus the script behaves exactly like

the Latex typesetting button except that it creates an html rather than a pdf, and displays the html in Safari rather than TeXShop. An advantage of this approach is that users can create their own scripts similarly and thus add features to TeXShop without waiting for new program code.

- TeX typesetting often requires a sequence of operations. To process a file with a bibliography, the source must be run through latex, bibtex must be run, and latex must be run twice more. A script is included to do this automatically. The script saves the source before the first latex run and updates the preview display at the end. Users can easily customize this script for their own workflow.
- Additional scripts convert the tex source file to a file with Windows line feed convention, or a file with Macintosh 9 line feed conventions, or a file with Unix line feed conventions. Mac OS X understands all line feeds without help, but many computers are not so smart; the conversions are useful when sending files to friends. The "flip" binary used to do these conversions was written by Craig Stuart Sapp. See http://ccrma-www.stanford.edu/~craig/utility/flip/ for details.
- A script is included to call pdfselect and extract portions of pdf documents. A user could request one file containing pages 3 through 7 of the tex document, one containing page 29, and one containing pages 31 through 36. The advantage of placing this code in the Macros menu is that a user interface is provided, so users don't need to remember calling conventions for pdfselect and don't need to switch to the Terminal.
- A #DOCUMENTNAME# variable was added to the Macro editor, giving applescript commands the name of the calling document.
- The following applescript commands were added to TeXShop. Consult the TeXShop help files for details about writing your own scripts using these commands.
  - typeset
  - latex
  - tex
  - bibtex
  - context
  - metapost

- makeindex
- typesetinteractive
- latexinteractive
- texinteractive
- bibtexinteractive
- contextinteractive
- metapostinteractive
- makeindexinteractive
- typeset
- refreshpdf
- refreshtext
- taskdone

• Improved support was added for external editors, following prodding by Joachim Kock. Several changes have been made:

  - There is now a preference to turn on continuous updating of the preview window if the user is running in external editor mode. Once each second the program checks to see if the pdf file has been updated. If so, it refreshes the pdf display.
  - The interval between refresh checks is controlled by a hidden preference item named RefreshTime. To reset to another interval in seconds (say every 2.19 seconds)

    ```
    defaults write TeXShop RefreshTime 2.19
    ```

  - Applescript command support has been added to TeXShop so external applications can send commands to it. For external editors, the important script commands are
    * latexinteractive
    * texinteractive

        * bibtexinteractive
        * contextinteractive
        * metapostinteractive
        * makeindexinteractive
        * typesetinteractive
        * refreshpdf
        * taskdone
        * open_for_externaleditor

    The first seven commands call TeXShop's typesetting engine. When one of these commands is called, control immediately returns to the calling program even though the typesetting operation is not complete. The taskdone command returns FALSE while this operation continues and TRUE when it is done, so a calling program wishing to send several commands can send one command and then test that it has been completed before sending another command.

  – Refreshpdf updates the preview display, and can be used instead of continuous updating to control that display. Typesetting commands automatically update the display upon completion.
  – The open_for_externaleditor command opens a .tex file, calling "Open for Preview..."

- Additional features not related to scripting or external editors have also been added:
- "Select All" can be used to select the full page of pdf output in selection mode. There is one restriction; in Multi-Page and Double-Multi-Page mode, select all is only active if the document has at most 20 pages, since otherwise the selected pdf will be enormous and bring the machine to a crawl.
- A preference item now allows users to distill with Apple's pstopdf rather than ghostscript. This only works in Panther because pstopdf is only in Panther. If the preference is chosen but Panther is not running, the old ghostscript code will be used. When the preference is chosen, ghostscript is no longer needed for internal TeXShop scripts, but it may still be required for teTeX style files.

One such case is epstopdf.sty, used to automatically convert eps files to pdf format during typesetting.

- Zenitani provided additional drag and drop support. The new version reports an error if the filename of the dropped file contains a space. The new code also permits customization, in a somewhat strange way. To customize drag and drop code, add a new submenu to the Macros menu titled "Drag & Drop". Inside this folder, insert items for file types and make the text of each item be the code to be produced by drag and drop. For example, one item might be called ".pdf" and the body of this item might be "\includegraphics[#INS#]{%r}" where neither item would include the quotation marks. In these inclusions,
  - %F = full path of an dropped file
  - %f = dropped filename
  - %r = relative path of the dropped file
  - %n = filename without extension
  - %e = extension
- If an extension is not listed in the Drag & Drop menu, or if there is no such menu, then drag and drop behaves as Zenitani proscribed, so most users won't need to customize the code.
- German help has been updated by Martin Kerz, the Italian localization has been updated by Nicola Vitacolonna, the Spanish help was updated by Juan Luis Varona Malumbres, the French localization was updated by Hendrik Chaltin, and a Romanian localization was added by Andrei Teleman. Thanks!
- Macros can now be called when the preview window is active; commands which insert text will be deactivated in this mode. A toolbar item for the Preview window was added so the Preview toolbar can display a Macros button.
- An "Abort" button was added to the console for people who want to stop the typesetting program in midstream. This is a minor change.
- Code by Elliott Hughes was added to clean up some of the code calling a latex, tex, bibtex, etc., task. If the binary file is not found, the program now puts up an error message explaining the error and asking if the preferences bin path is correct. This will be useful for those running fink who forget to change the preference.

- At the request of Joachim Kock, when TeXShop opens a file for Preview, or is in external editor mode and opens a file, the program now compares the dates of the source and preview files. If the preview file is not up to date or does not exist at all, the source file is automatically typeset. There is a hidden preference to turn this behavior off, but it is on by default: To turn it off

```
defaults write TeXShop ExternalEditorTypesetAtStart NO
```

Bugs fixed:

- A few users reported that their printers added a slight yellow background to the page. Only a few printers had that problem. Frank Stengel discovered that it was caused by a NSEraseRect call in the print drawing routine. This call has been removed; now TeXShop prints using only one line, the vanilla Cocoa call [myRep draw].
- In Panther, all Text objects offer word completion. If a portion of a word is typed and option-escape is typed, the system will offer a list of possible completions. This works in TeXShop, TextEdit, and other Cocoa programs. But TeXShop's Command completion was broken in Panther, and made it impossible to use this new feature. This is fixed.
- TeXShop has an applescript command to add text, but this command did not update the undo stack. This was fixed by Stefan Walsen; his patch is in version 1.33.
- If the user printed the source and later printed the typeset document, the document would be lowered on the page. This is fixed.
- Jerry Keough found a strange bug when using TeXShop in Jaguar. If the user's preference setting asked that no empty document appear at startup and if the user opened a document, made the pdf window active, and then reached over and closed the source window, the next menu use would crash the program. This bug did not occur in Panther. It is fixed.

Version 1.32 adds two extra features to TeXShop and fixes some bugs.

New features:

- There is a new menu item, `"New Tag"`, which inserts an empty tag in the source text and positions the cursor so the user can add the name of the tag. As a corollary, there is now a keystroke to add a new tag; the keystroke is command-2.
- In Applescript macros, the terms

  ```
  #LOGPATH#, #AUXPATH#, #INDPATH#, #BBLPATH#, and #HTMLPATH#
  ```

  will now be recognized; this has been added at the request of Claus Gerhardt.
- There is a new option to set the program called by MetaPost. See the TeXShop help file for an explanation.
- The drag and drop code has been improved by Seiji Zenitani. Dropping files of types pdf, jpg, jpeg, tif, tiff, eps, or ps on the source document will produce \includegraphics and a reference to the file. Dropping a file of type cls, sty, or bib will \documentclass, \usepackage, or \bibliographystyle and a file reference. Dropping any other text file will produce \input and a file reference.

Bugs fixed:

- Although the magnification level can increase to 1000 in version 1.31, the arrow keys next to the magnification control only allowed values up to 400. This is fixed.
- If the preview window's toolbar was in text-only mode and the magnification panel was selected, the resulting small dialog window would not go away in 1.31. This is fixed.
- In Preferences, if the user set the tab size or the preview window magnification and then cancelled, the new values would still appear when the Preference panel was again displayed. This is fixed.
- Suppose the preference to place the first page on the right side is active. In 1.31, the left arrow key did now work in double page mode, and the right arrow key stopped one page before the end. In double multipage mode, the end key,

the page down key, and the right arrow key stopped before displaying the last page. All of these problems are fixed.

Version 1.31 adds some extra features to TeXShop and fixes several bugs.

New features:

- Macros items are now specific to the typesetting engine set in the current window. Two default sets are created when TeXShop first starts, one for Latex and one for Context. (The Context set was created by Hans Hagen; thanks!). Suppose a different engine is selected, say TeX. When it starts, it will have the default Latex macros. But if this set of Macros is edited with the Macro editor, the new set will always be associated with TeX, while the old Latex macros will continue to be associated with Latex. Of course the Latex macros can also be changed.
- A new preference item allows the first page in the two double page modes to be either on the left or on the right. Books usually put this page on the right, so that is the default preference.
- The preference dialog has a new pulldown menu to reset preferences to default values. This addition was requested by Seiji Zenitani for users in Japan who must cope with three different versions of pTeX. These users can now rapidly set preferences without consulting documentation on web sites.
- The largest possible magnification in the preview window is now 1000 (previously it was 400).
- Additional French menu translations by Hendrik Chaltin; thanks!

Bugs fixed:

- Three Panther problems are fixed. A few interface changes were made to improve appearance in Panther. This release is essential for Panther.
  - In Panther, older versions of TeXShop refuse to create new files, although they can open existing files. This was fixed by adding two lines to the "displayName" code.

  - In Panther, the small yellow tags which display the current page when scrolling in multipage and double multipage display formats were blank. Also, the small yellow tags which display the size of a selection rectangle to copy a portion of the pdf window were blank. This is fixed.
  - On my system, Panther spellchecking often fails for all programs. This happens at boot time. The computer boots up, but even if the first program used is TextEdit or Mail, it refuses to "spell check as you type" and the "Spelling dialog" refuses to appear. I suspect this is due to a defective third party program on my disk, since other Panther users haven't seen the problem. But I haven't been able to isolate the bug. Once the problem occurs, TextEdit and Mail have minor text input glitches which are cured by turning off continuous spell checking. TeXShop had more serious glitches in this situation. So in 1.31, extra code has been added to turn off continuous spell checking when TeXShop starts if the spell checker is not available.

- Fixed a bug reported by Luis Sequeira: when text was dragged within the source window, it was always copied and pasted. It should have been cut and pasted unless the option key was down. This is fixed.
- Juan Luis Varona Malumbres slightly improved Spanish help (small icon for the list of help files in the drawer).
- In two page mode, typesetting no longer scrolls to the first page.
- The new page and magnification buttons retain firstResponder status (as before 1.29) so users can experiment with several settings without clicking again for each experiment.
- In version 1.30 and before, if the user input an impossible line into the "Go To Line" dialog, the program could crash; this is fixed. Thanks to Eric Seidel for the bug report.
- Kevin Ballard, kevin@sb.org, contributed a new English.lproj folder with revised placement of interface items in preferences, slightly revised menu items, and other changes to improve the look of the interface in Panther. Many thanks!
- Made spacing changes for Panther in the German preference panel. Thanks to Martin Kerz for the suggestions.
- There is a hidden preference item to turn off tag computation: "defaults write

TeXShop TagSections NO" but this preference was disabled somewhere along the line and certainly in 1.30. It is enabled again.

- The Preference window is no longer hidden when the program is deactivated.