TEXShop Tips & Tricks

v0.4.2–2011/04/03

H. Schulz
herbs2@mac.com

Contents

1 Introduction

TeXShop is a "Front End" for a TeX distribution on Mac OS X. As such it allows the user to create and edit TeX source files, interact with the TeX distribution (e.g., typeset the source file) and, finally preview the final pdf file. It also allows the user to go back and forth between preview and source.

Over the years TeXShop has added many features. Some of them are obvious and are meant to help a novice get started. Others are a bit more subtle in their use and the underlying power of these features needs to be coaxed out.

1.1 What Isn't Here

This article is, first of all, *not* about TeX or LaTeX. I don't intend to teach you how to write TeX source. There are many fine books and articles that will teach you how to become a TeXpert or, at least, a TeXpätzer like me.

Although there is some introductory material it is also *not* meant as a complete manual for the use of TeXShop for the total novice. Over time it might evolve into such a document but I've got to start somewhere and this is that start.

1.2 What Is Here

In this article I hope to introduce you to some of the more subtle things you can do to make your life as a TeX source editor easier. These include adding keyboard commands and extending the editing capabilities of TeXShop; helping you make short(er) work of creating documents, etc., with the use of Macros and Command Completion; and, finally, how one can extend the processing capabilities of TeXShop using Engines.

2 Editing, Typesetting and Viewing — the Work Cycle

This is about as close to a beginner's section you will get in this document.

2.1 Editing the Source File

The first thing you've got to do to create that great work is to type it into the source document that will be typeset and viewed later. This involves both putting LaTeX markup as well as your wonderful words into the document.

To get started you can open a new document using File → New (Cmd-N) and then fill in the start of a new document by choosing a template from the Templates popup menu in the Source Window or use—new with TeXShop 2.36—the File → New From Stationery... command and picking appropriate Stationery from the list. Note that the templates and stationery provided are certainly not complete; if you have some that you think are of general use feel free to submit them for inclusion in TeXShop. You can add personal Templates and Stationery to `~/Library/TeXShop/Templates/` and `~/Library/TeXShop/Stationery/` respectively. **Note: `~/Library/` is the `Library` folder in your `HOME` folder.**

2.1.1 LaTeX & Matrix Panels

While I believe that panels with a clickable interface actually hinder learning I'll mention that TeXShop has two panels to help with entering LaTeX code (the LaTeX Panel) and for setting up the basic structure of a matrix or tabular (the Matrix Panel). These are toggled on/off under the Window Menu or with the keyboard shortcuts[1] `Opt-Cmd--` and `Opt-Cmd-=` respectively. Figure (1) shows what the panels look like. It is possible to make a few changes and additions to the LaTeX Panel by editing the `~/Library/TeXShop/LatexPanel/completion.plist` file. **Note: all `plist` files must be edited using UTF-8 Unicode encoding.**

[1] The given shortcuts are for the English localization and may be different with other localizations.

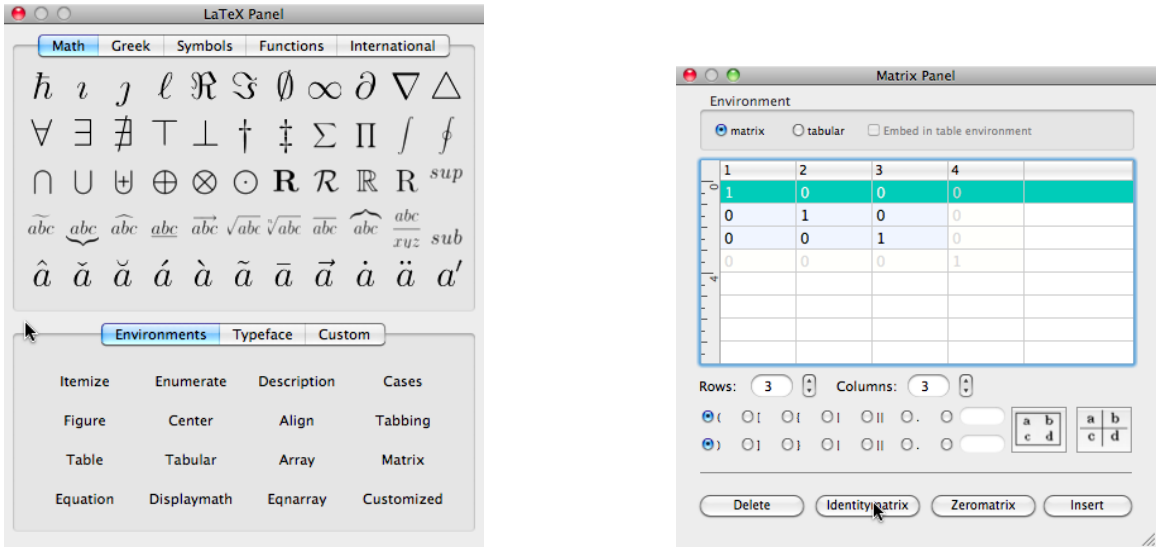

Figure 1: The LaTeX & Matrix Panels.

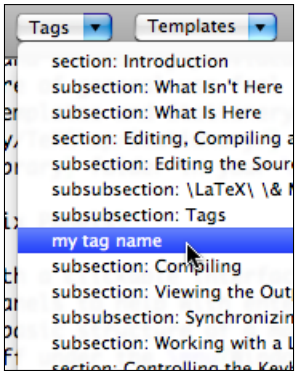

Figure 2: The Tags Popup Menu.

2.1.2 The Tags Popup

The Tags popup menu on the Source Toolbar will automatically list sectioning commands so you can quickly jump to a relevant part of your document source. You can add your own tag to the list at a particular place in the document by placing the line

```
%:my tag name
```

at that position and it will then appear in the popup list so you can jump to that location quickly. See Figure (2). Sorry, tags are not recursively included for files you `\include` or `\input`.

2.2 Typesetting

Once you are ready to take a look at how your document will appear you typeset it with the default engine, `pdflatex` out of the box, by simply using the Typeset → Typeset (Cmd-T) command.

You may wish to use a different engine as your default. You can change the default engine in TeXShop → Preferences → Typesetting.

If you are including many eps graphics files in your document you may wish to typeset using

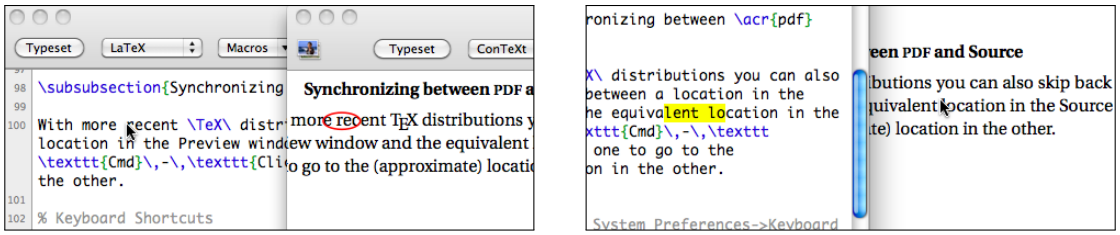

Figure 3: Source → Preview and Preview → Source Synchronization.

`latex` → `dvips` → `ps2pdf` since `pdf(la)tex` does not allow for direct inclusion of eps files[2]. The easiest way to do this is to include the line

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

at the top of your document. Then TeXShop will use the `latex+distiller` typesetting method noted above no matter what the default engine setting. Change `latex` to `pdflatex` to force use of `pdflatex` to typeset your file.

2.3 Viewing the Output pdf File

Assuming the document was successfully typeset the pdf file will automatically open in a separate preview window.

You can control how it's displayed in the Preview Menu. You can change the default settings in TeXShop → Preferences → Preview.

2.3.1 Synchronizing between pdf and Source

With more recent TeX distributions you can also skip back and forth between a location in the Preview Window and the equivalent location in the Source Window by Cmd-Clicking in either one to go to the (approximate) location in the other. See Figure (3) for an example of Source → Preview and Preview → Source synchronization.

2.4 Working with a Large Document

It is often handy to break a large document into more manageable subordinate parts and then create a "root" file which contains the preamble and `\include` commands to bring all the parts together for typesetting.

To have TeXShop "know" which file to typeset when working on a subordinate file put the line

```
% !TEX root = path/to/rootfile.tex
```

at the top of your subordinate file; `path/to/rootfile.tex` is the relative or absolute path to the root file for this document. Once this is done TeXShop will typeset the root file if you press Typeset → Typeset (Cmd-T) even though you are editing a subordinate file and properly synchronize between the Source and pdf. E.g., if the root file is called `mygreatbook.tex` and the chapter files, `chapter1.tex`, etc., are in a `chapters` sub-folder below the root file then place the line

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

at the top of each of the chapter files. The `../` means go up one folder level to find the root file.

[2] The `pdflatex` program in MacTeX-2010 and later will do on-the-fly conversion of eps files.

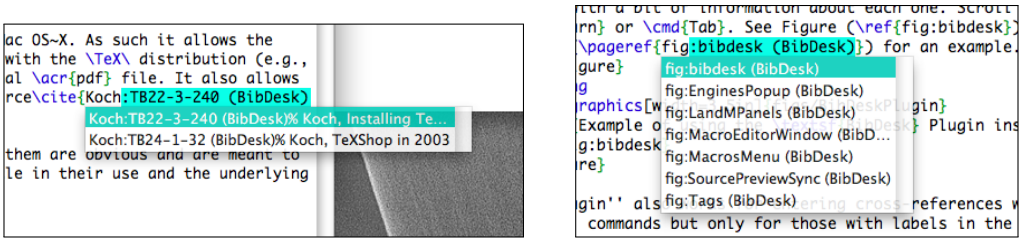

Figure 4: Examples of using the BibDesk Plugin for citations and cross-references inside TEXShop.

2.4.1 Switching between Source Windows

If you have multiple source files open you can switch between just those files by using the Window → Next/Previous Source Window (Cmd-F2/Shft-Cmd-F2) menu commands.

2.5 Working with BibDesk and Citations

TEXShop has a built-in "plugin" that interacts with the BibDesk bibliography application to allow you to complete citation references in the `\cite` command. To enable the use of the "plugin" make sure that TeXShop → Preferences → Source → Editor → BibDesk Completions is checked.

To use it you must first open the required bibliography (`bib`) file(s) in BibDesk. Enter several characters from the reference label within the `\cite` command and press F5 to get a list of matching references from the `bib` file(s) with a bit of information about each one. Scroll to the one you want and press Return or Tab. See Figure (4) on page (5) for an example.

The "plugin" also works for entering cross-references within `\ref` or `\pageref` commands but only for those with labels in the file you are editing.

2.6 Getting Help for Packages

There are many times when having help about a given package can be handy. TEXShop has an interface to `texdoc` which will bring up that documentation. Execute the Help → Show Help for Package... (Opt-Cmd-I) and enter the name of the package.

You can also easily look at a package directly with the Help → Open Style File... command and enter the full package file name *including the proper extension* (e.g., `.sty` for packages or `.cls` for document classes).

3 Controlling the Keyboard

One of the best ways to speed up your entry of text in a source file is to keep your hands on the keyboard as much as possible—only one of the reasons I don't like the "clicky" interface of the LATEX and Matrix Panels. There are many shortcuts associated with the TEXShop menu system but this section is about changing and adding others and other keyboard customizations.

3.1 Menu Shortcuts & System Preferences

Sometimes you'd like to add a shortcut to a menu item that doesn't have one or add one to a command whose shortcut you dislike. Mac OS X 10.4 (Tiger) and later have a method to add shortcuts to specific menu items both globally and in specific programs. This feature has become much more reliable in OS X 10.5 and especially in OS X 10.6.

One example using Mac OS X 10.6 (Snow Leopard): TEXShop 2.36 has added a File → New from Stationery... command, without a shortcut, which can be very helpful once you set up stationery the way you want. To add Opt-Cmd-N as the shortcut to that menu item: open up the System Preferences application to Keyboard → Keyboard Shortcuts and select Application Shortcuts; press the + button to add a shortcut; select TEXShop as the application; enter the exact menu

title [New from Stationery... — note you *must* enter a real ellipsis, '...', (Opt-; with the English keyboard layout)]; and press Opt-Cmd-N as the shortcut.

3.2 More Editing Help

TeXShop is built using Apple's programmers interfaces (called frameworks) and therefore inherits all the properties and functionality of those interfaces. There are many things available through the Text framework that aren't tied to the keyboard by default, e.g., many 'emacs-like' keyboard commands, but Apple has made it possible to add those commands to all applications that use the Text framework; e.g., TextEdit and Mail as well as TeXShop.

This is done by creating a special file, `DefaultKeyBinding.dict`, and placing it in a particular location, `~/Library/KeyBindings/` (you may have to create the `KeyBindings` folder there if it doesn't already exist).

You can get more information about this, as well as a (useful) sample, by downloading the `KeyBindings.zip` file from <http://public.me.com/herbs2>.

3.3 Auto Completion

Besides adding shortcuts to Menu Items you can actually bind keystrokes, within TeXShop, to expand into groups of characters. This is called Auto Completion and is enabled by setting the TeXShop → Preferences → Source → Editor → Auto Complete option. Don't confuse this with Command Completion which is discussed in Section (5) below.

E.g., pressing Opt-, with a US keyboard layout, usually enters ≤ into your document but with Auto Completion enabled `\leq` will be entered. Similarly, with some text selected pressing " will surround the selected text with `‘‘` and `’’`.

You can add your own keybindings and/or remove those you don't need by editing `~/Library/TeXShop/Keyboard/autocompletion.plist`. To add a keybinding to the _ key so that pressing it with a selection will produce the selection surrounded by _{ and } enter the lines

```
<key>_</key>
<string>_{#SEL##INS#}</string>
```

to the list in `autocompetion.plist`.

4 Macros

Macros can be simple text substitutions or Applescript programs that can do all sorts of processing on a file. You can also assign a keyboard shortcut to any macro for direct execution. The ones that are part of TeXShop are found under the Macros Menu.

You can remove or add additional macros to the menu by using the Macro Editor (use the Macros → Open Macro Editor command). The Macro Editor window and extra menu items in the Macros Menu are shown in Figures (5) and (6) respectively.

Besides writing your own macros you can add macros supplied by others to the Macros menu one of two ways: copy and paste the text version of the macro into a New Item in the Macro Editor; or obtain the macro as a `plist` file and use the Add macros from file... command found in the Macros Menu when the Macro Editor is open (again, see Figure (6)).

More information on macros can be found by searching for macros in Help → TeXShop Help Panel....

4.1 Text Macros

Text macros are simple text substitutions. You can also tell TeXShop to insert any selected text using `#SEL#`, place the cursor using `#INS#` and even put in multiple lines in the macro itself. Then you can assign the text macro to a keyboard shortcut.

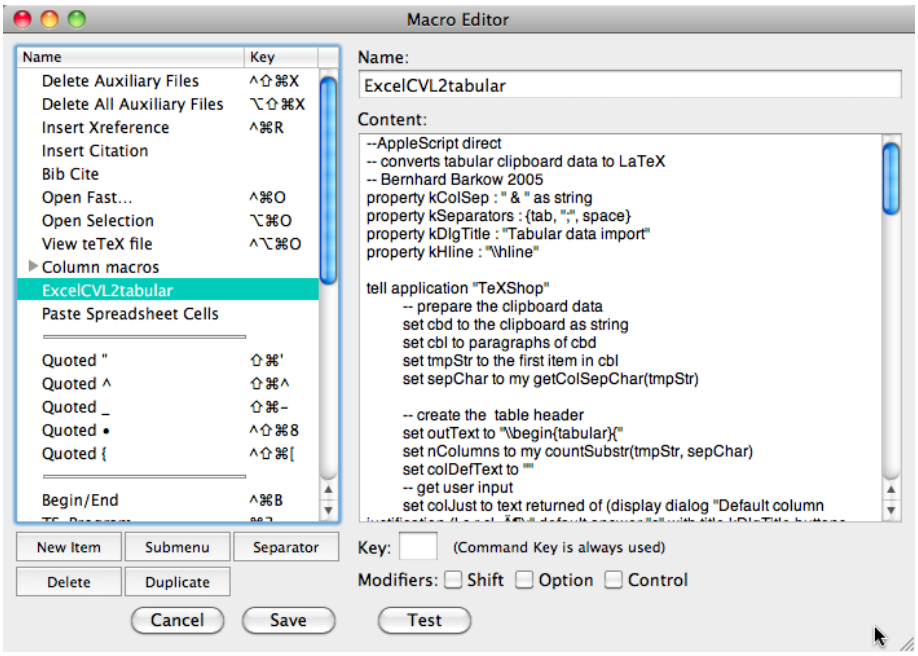

Figure 5: The Macro Editor Window.

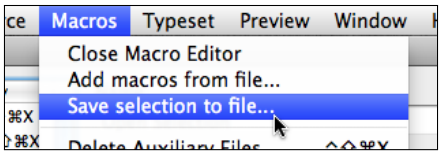

Figure 6: The extra menu items when the Macro Editor is open.

I like to use Cmd-B and Cmd-I to insert `\textbf{...}` and `\emph{...}` into the document where `...` is any possible selected text. Macros to do that are already under the Macros → Text Styles Menu so we need only assign keyboard shortcuts to them. To assign Cmd-I to the emphasize macro: open the Macro Editor where the form of the Macros menu appears in the left hand pane; click the emphasize macro found under Text Styles; click the Key insertion box and simply insert a lower case 'i' (the Cmd key is assumed and additional modifier keys can be checked off).

4.2 Applescript Macros

You cannot distinguish Applescript macros in the Macros Menu from text macros but they can do complicated processing and add/change the source file in TeXShop. One example in the default set is the Program macro that creates a

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

line at the top of a file with your choice of engine substituted for xxxx. You can look at the Applescript code for this macro by clicking on its name in the Macro Editor.

5 Command Completion

LaTeX markup is rather wordy which is nice because it describes what it's supposed to do but a bit painful to write. Command Completion allows you to insert complete environments and

commands with a few keystrokes and the press of a "trigger" key (this is Esc by default but can be changed to Tab in TeXShop → Preferences → Source → Command Completion Triggered By:).

Commands that have arguments usually have a Mark (•) inserted for each argument. You move to the next argument by using the Source → Completion → Marks → Next Mark command (Ctl-Cmd-F [or Opt-Trigger]). This also selects the Mark so typing automatically removes the Mark and substitutes the typed information. See the complete documentation, with lists of commands/abbreviations supplied with TeXShop out of the box, in the `~/Library/TeXShop/CommandCompletion/` folder for much more information.

5.1 Completions

You can complete many commands by starting to type them and pressing the trigger key. Variations on the commands with differing numbers of optional arguments are generated by additional presses of the trigger. One example: typing `\sec` and then the trigger on a new line produces

```
\section{•}
```

while a second press of the trigger gives

```
\section*{•}
```

the *-variant of the command and a final press of the trigger gives

```
\section[•]{•}
```

with the optional argument.

5.2 Substitutions or Abbreviations

Besides completions for partial command insertions there are also many abbreviations. These are short mnemonics for complete substitutions.

All abbreviations for environments start with a 'b'. To generate a complete itemize environment place `\bite` on a line by itself and press the trigger key to get

```
\begin{itemize}
\item
•
\end{itemize}•
```

with an extra Mark at the end so you can easily jump to the end of the environment. Additional items can be generated by typing `\it` and the trigger to get

```
\item
•
```

ready for entry of text.

In addition to the `\section` command lower level sectioning commands have abbreviations. Sub-sections can be generated by typing `\ssec` and the trigger to get

```
\subsection{•}
```

with subsequent presses of the trigger key giving the *-variant and finally the variant with the optional argument.

As a final example `\tt` and the trigger gives the `\texttt{•}` command and a second press of the trigger gives the declaration `\ttfamily` with similar results for other font changing commands.

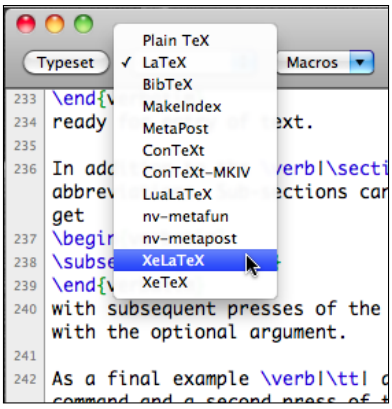

Figure 7: The Engines Popup Menu on the Source Toolbar.

5.3 Hey, it doesn't work!

If these examples don't work you probably need to let TeXShop update the `~/Library/TeXShop/CommandCompletion/` folder; simply delete that folder from `~/Library/TeXShop/` and restart TeXShop.

6 Extending Processing via Engines

TeXShop offers several default "engines" (also referred to as "scripts" which is left over from earlier times) in its Typeset Menu. These include running Plain TeX or LaTeX (either using `pdftex` or TeX+DVI), BibTeX, MakeIndex, MetaPost or ConTeXt. But there are many things you may wish to do that fall outside of this limited set so TeXShop also allows you to create new engines that are stored in `~/Library/TeXShop/Engines/`. These additional engines do not show up in the Typeset menu but only in the popup list on the Source and Preview Toolbar (see Figure (7)).

You can use these engines by choosing from that popup list and then pressing the Typeset button or, a better choice if you use different engines for different documents, by putting a line like

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

at the top of your source file; the example given will run the `xelatex` engine on this file independent of other choices.

TeXShop is shipped with a few engines activated (i.e., directly in the `~/Library/TeXShop/Engines/` folder) but also includes several additional ones in `~/Library/TeXShop/Engines/Inactive/`. As an example let's activate and use the pdflatexmk engine found in `~/Library/TeXShop/Engines/Inactive/Latexmk/`.

6.1 The `pdflatexmk` engine

If your document had cross-references, bibliographies or indexes it takes multiple `pdflatex` runs with intermediate runs of `bibtex` and/or `makeindex` to create the bibliographies, indexes and resolve all cross-references. The `pdflatexmk` engine automates this whole process.

To activate the engine simply move the `pdflatexmk.engine` file from `~/Library/TeXShop/Engines/Inactive/Latexmk/` two folders up, to `~/Library/TeXShop/Engines/`. When you restart TeXShop you can check that pdflatexmk in now in the popup menu.

Now place the line

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

at the top of your source file. From then on when you simply typeset the file (Typeset → Typeset or Cmd-T) TEXShop will use this engine and the complete process for typesetting the document to its final form will be carried out.