The PracTEX Journal, 2006, No. 4
Article revision 2006/09/5

Travels in TEX Land: Using Your Favorite Editor with TEX

David Walden and Yuri Robbers

Abstract In this column in each issue Dave Walden muses on his wanderings around the TEX world. In this column he is joined in his meandering by Yuri Robbers. Walden discusses the benefits of the fact that one can use the editor of one's choice with TEX and the various system built on top of TEX, and Robbers lists a number of the editors that are optionally available. **Walden speaks in the first person in sections 1–4; Robbers speaks in the first person in section 5.**

1 Introduction

Many reasons are given by proponents of TEX as the benefits of using TEX. These typically have to do with excellent typesetting, logical structuring of documents, and so forth. Another reason that is sometimes given has to do with the fact that with TEX one can choose the text editor best suited to one's needs.

More people in the world use a word processor such as MS Word to do their document processing. Using a word processor such as MS Word that has invisible, undocumented, proprietary markup means you have to use its built-in, WYSIWYG editor that knows about that markup. This has several potential disadvantages: (1) you have to learn a new editor for each different word processing system (e.g., Word, Word Perfect, etc.) you decide to use or learn changes to the editor user interface with each new release of the word processor; (2) GUI-based editing often takes a lot more key strokes to do simple things than an editor like WinEdt or Emacs (I provide an example in section 2); (3) an editor like MS Word's does not seem to have a lot of traditional useful features that an editor like WinEdt or Emacs has (I provide examples in sections 3 and 4).

A number of editors oriented to TEX or integrated with a particular distribution of TEX also exist in addition to the WinEdt- or Emacs-like editors I assume in sections 3 and 4; we, primarily Yuri Robbers, mention a number of these in section 5.

Many of the ideas mentioned in sections 2–4 in this column are probably relevant to any typesetting system that has editing capability (such as those I describe below). For all I know, MS Word can do many of these things; but a number of years ago I lost interest in struggling to find stuff buried in all Word's menus, dealing with its ever changing user interface, and its planned-obsolescence-and-forced-upgrades business strategy.

2 Visible markup

Part of my preference for LaTeX over Word comes from the fact that all of the markup is in a file where I can see it and change it rather than it having to be accessed by various menus and being largely unseen (except in its effect) in the document. To take a simple example, suppose I wanted to make the word "brown" in the phrase "quick brown fox" be bold. In Word, many people I know would select "brown" with the mouse, pull down the Format menu, click the Font item on the menu, click Bold in the Font Style column, click OK, and then the text would appear in the document in bold when displayed or printed (alternatively I could type control-B after selecting the word "brown"). To do the same thing in LaTeX, I would change the text "quick brown fox" to "quick `\textbf{brown}` fox" with my text editor, and "brown" would display in bold when my LaTeX file was compiled.

Of course, some slightly more sophisticated users of Word would know that they can select the phrase "quick brown fox" and then type control-b or they may have Word's graphical user interface configured so there is a **B** icon that can be clicked, thus saving a few steps through the menus. However, they are probably less likely to have such short cuts set up for small caps, which is just a simple with TeX: `\textsc{brown}`.

No doubt there are ways in Word to do many if not all of the things I now do with LaTeX, but I find them mostly easier to find and do in LaTeX.

3 Two ways to make a change throughout a document

In my columns in issue 2006-2 and 2006-3, I sketched the benefits of using macros for some sequences of commands (for instance, `\Dash{}` for `---`) that enable the replacement sequence to be changed everywhere in a document by just changing the definition of the macro in one place.

Another option for making a change to a particular sequence of characters throughout a document is to use a text editor's Replace All command. For instance, suppose I hadn't used a macro for em-dashes and instead had closed form instances of `---` throughout my document, e.g., "this is the end—the end of the line." And then suppose I decide to change the style to uses semi-open form em-dashes, e.g., "this is the end — the end of the line." With my editor I can do a Replace All of `---` by `\thinspace---\thinspace{}`. If the document is broken up into separate files for each chapter, it will be good if the text editor has the option for doing the Replace All over all documents open in the editor instead of only in the document where the cursor currently is.

Here is another example of a simple text replacement of the entire document. Suppose I decide (for some reason) to replace all en-dashes by hyphens. Then I can do the following sequence of three steps (the first and third steps are to avoid accidentally changing instances of `---` into `--`):

Replace All `---` with `#X#X#`
Replace All `--` with `-`
Replace All `#X#X#` with `---`

Now suppose I want to add a fourth argument to every instance of the macro call[1]

```
\snfig{  }{  }{  }
```

that is, change the macro call formats to

```
\snfig{  }{  }{  }{  }
```

Of course, one approach is to search for each instance of `\snfig{`, then move the cursor to after the third pair of braces, and then type the fourth pair of braces. However, if your text editor has a capability for dealing with regular expressions, you can make this change more easily (the last book I wrote had a couple of hundred instances of `\snfig`).

1. See the definition and discussion of this in macro my column in issue 2006-2.

While I won't get into the specific format of any particular editor's representation of regular expressions, they would do something like the following in our example case.

Replace All `\snfig{`***(.*)***`}{`***(.*)***`}{`***(.*)***`}` with `\snfig{`***$1***`}{`***$2***`}{`***$3***`}{}`

Everything in the Replace All command that is in a typewriter format is literal characters to be replaced. The characters in italics in the first part of the command are special characters that match any number of characters between balanced braces. For instance, in the command

```
\snfig{Caption title.}{file-name}{.5}
```

the first instance of ***(.*)*** would match `Caption title.` The second instance of ***(.*)*** would match `file-name`, and the third instance would match `.5`. Better than that, the editor stores each matched set of characters in its own place for later reuse, as in the second half of the above command. The part of the command after the word "with" says to replace the `\snfig` command that was matched with all the same literal characters for the command names and braces, but to put the first match text in place of ***$1***, the second match text in place of ***$2***, and the third matched text in place of ***$3***, and to add an extra pair of literal braces at the end of the replacement. Thus,

```
\snfig{Caption title.}{file-name}{.5}
```

is turned into

```
\snfig{Caption title.}{file-name}{.5}{}
```

and the same is done for every other instance of `\snfig` in each case properly maintaining the argument text through the replacement step.

The above example may need some tweaking if the instances of the command being changed sometimes span new line boundaries, but typically this also can be handled, as can be much more complex instances of detecting what should be replaced and what should not be in various instances. In fact, depending on the editor's particular regular expression capability, the earlier example of replacing en-dashes by hyphens perhaps could have been done with one Replace All using a regular expression to search of `--` *not* followed by a third `-`.

In previous columns and I have recommended that anyone not already using LaTeX macros should learn to use them. I recommend the same thing about using regular expressions if your editor supports them. They won't be needed as often as macros, but when they are needed they are a major productivity increaser.

4 Other editor features

All of the serious text editors I have used allowed me to mark the cursor position with a couple of key strokes (e.g., Alt-F11 in WinEdt), move the cursor somewhere else (for instance to select some distant text and cut it, and then jump back to the first cursor position (e.g., Cntl-F11), where I might paste the text cut from elsewhere in the document.[2]

Text editors such as I have in mind also typically have provision to have multiple saved-text buffers rather than just one cut-and-paste clipboard.[3] I gave an example of this in the regular expression example in section 3, where three bits of text were simultaneously saved from the replaced string of characters for placement in the replacement string. With multiple places to save text, it is also possible, for example, to search-for-and-cut one bit of text, search-for-and-cut another bit of text, search-for-and-cut a third bit of text, and then paste together at some other point in the file, saving (in this example) several moves of the cursor in comparison with an editor with a single clipboard.

To give another example of the usefulness of multiple text buffers, I often find something on a web page and want to copy something from the page as a quote in a document I am writing *and* copy the URL as the source of the quote. Without multiple text buffers this requires the following sequence:[4]

1. select text to be copied
2. copy to clipboard,
3. switch window to other document,
4. position cursor, paste contents of clipboard
5. switch window for first document,
6. select URL text

2. I don't claim that none of these features are available in various editors that are packaged with commercial versions of TEX; I hope they are. I am only suggesting that you find an editor that supports such capabilities. What I do know is that with each new release of MS Word I find it harder and harder to find such features, if they exist at all, while they are easy to find in the two text editors I currently use regularly (WinEdt and Emacs).

3. Alex Simonic, the developer of WinEdt, the editor I mostly use, showed me how to write macros to provide multiple text buffers in WinEdt.

4. I know I could have two windows open at the same time and save some of the following steps, but sizing the windows so both can be seen takes too many steps unless I am going to do a lot of copying between two documents.

7. copy to clipboard,
8. switch window to other document
9. position cursor
10. paste contents of clipboard.

With multiple text buffers the same process requires:

1. select text to be copied
2. copy to buffer A
3. select URL text
4. copy to buffer B
5. switch window to other document
6. position cursor
7. paste contents of buffer A
8. reposition cursor
9. paste contents of buffer B

The latter method is not necessarily less key strokes (the macros I have for WinEdt take 12 steps), but it is somehow easier for me not to switch windows and have to refind my place as often.

The fact that LaTeX is not locked to a particular editor also means that each participant in a collaborative project can use the editor with which he or she is most familiar. (Collaboration is also made easier because there is much more compatibility from release to release of TeX and LaTeX, even with multiple providers, than there is from release to release with many non-TeX commercial products. For instance, MS Word seems to go out of its way to enforce inter-release incompatibility in a way apparently aimed at forcing all collaborators to all upgrade to the same release.)

Using a text editor in conjunction with LaTeX with its explicit markup also has advantages. For instance, it is easy to search for an italicized version of a word (i.e., search for `\textit{word}`) as distinct from a non-italicized version of the same word. Similarly, it is possible to search for all headings of a certain level (for instance, all instances of `\subsection`); with a system with implicit markup (e.g., MS Word) one might have to search for the words of each subsection title.

In my column in issue 2006-2 I showed the use of `\include` files. This works because LaTeX has the provision for specifying in one file a list of files to be included as if they were text in that first file. The editor I mostly use, WinEdt, also supports this; it knows enough about LaTeX to search the highest level file for instances of `\include` and gives me a list of visual tabs to the various files to be included; this makes it very easy for me to move among the various files in a longer document.

I could give an unlimited number of examples what text editors can do once one breaks free of the limitations of hidden, proprietary, undocumented markup and those built-in editors whose graphical user interfaces eliminate powerful editing capabilities as part of providing a point-and-click environment to the user. One of the best things is that I can use the same editor with which I have become facile from application to application.

5 Editors optimized for TeX

Dave Walden mostly had WinEdt and Emacs in mind during the discussion of the previous sections. WinEdt comes already knowing a lot about the commands in a TeX document, and Emacs can operate in a mode where it also knows the same sorts of things. In the following paragraphs, Yuri Robbers says a few words about a number of editors that know about TeX.

AucTeX is an (X)Emacs mode for (La)TeX that adds all sorts of extra options for editing TeX code. This includes menus and keyboard shortcuts for creating various environments, commenting or uncommenting regions or even creating bibliography files to use with BibTeX. AucTeX comes complete with preview-LaTeX, which enables automatically typesetting previews of code snippets (such as font changes, equations and PSTricks graphics) within the Emacs buffer that contains the source code. AucTeX can be downloaded from `http://www.gnu.org/software/auctex/` and it runs on any system that runs Emacs (`http://www.gnu.org/software/emacs/`) or XEmacs (`http://www.xemacs.org/`), which is — to my knowledge — just about any OS that TeX itself runs on. AucTeX as well as (X)Emacs are free and open source software.

WinEdt (`http://www.winedt.com/`) is a very powerful TeX editor for the MS Windows platform. It is commercial software, with a discount for members of TUG and Dante. A time-limited trial version is available free of charge. WinEdt is well-organized: it is easy to find all the options and symbols in the plethora of menus and palettes. Apart from powerful editing features it also offers on-the-fly checking for spelling errors. WinEdt also offers optional plugins such as a table designer, and it allows for "regular" line-mode selecting and also for the less usual column-mode selecting which is useful when creating tabular material. A disadvantage is that it does not yet support Unicode (which most other editors mentioned here do support).

TEXmaker (http://www.xm1math.net/texmaker/) is an editor that I have not yet used much, but it definitely looks promising. It is cross-platform (it runs on MacOS X, MS Windows and any brand or flavour of UNIX or Linux, and it may even run on more systems). TEXmaker has a GUI that does not dazzle newcomers while still offering many powerful editing options. The platform-independence is a big bonus for people who deal with multiple platforms, since they can use the same editor on each OS. Also, it is free (in the ice cream as well the freedom sense) software and under active development.

TEXnicCenter (http://www.toolscenter.org/) is also worth mentioning. It is a free program (similar to TEXmaker), but it is available for Windows only. It is comparable to WinEdT, but does not cost any money and it seems to offer more options than TEXmaker. Since it's free and it comes bundled with ProTeXt as a standard editor, it is rapidly gaining a users.

WinShell (http://www.winshell.de/) is similar to TEXnicCenter in many ways. It offers about as many options though not exactly the same options. It is also free software and also available only for MS Windows. The choice between WinShell and TEXnicCenter is a personal choice that depends largely upon the specific options each one has or perhaps on the choice of keyboard shortcuts (there are quite a few differences).

Kile (http://kile.sourceforge.net/) is popular with users of the KDE desktop environment that is commonly used on Linux and UNIX systems. It is a very good program in the sense that it has many interesting and highly advanced options that no other editor I know offers. Kile depends on the KDE desktop, which means that anyone not already running that — such as myself — faces several hundred megabytes worth of installing to do. This means that if KDE and Kile do not come distributed in binary form for one's operating system, one will be up for quite a bit of work compiling and installing everything. This may be a bit much for newbies or the faint of heart, but luckily most Linux and some UNIX distributions include at least KDE and often also Kile. Once one has Kile running, it is a really nice editor, although a bit slow at times on older systems. The interface is both powerful and intuitive and offers many options in well-organized menus.

Kile offers advanced options such as dealing with documents spread over multiple source files, quick preview of parts of a document and a DVI previewer that allows one to click on typeset code and immediately be transported to the source code that generated it. Kile also caters for beginning TEX users by providing a selection of wizards that help the user set all sorts of document options.

Similar to Kile but for the Windows is the freely available **LATEX Editor** (http://www.latexeditor.org/). It seems to be slightly less mature than Kile, lacking some features that Kile already has. It probably does have some features missing from Kile, but since we don't have personal experience with this package we can't be more specific.

PCTEX (http://www.pctex.com/) offers a nice GUI, especially for newbies. It is Windows-only and commercial. PCTEX offers a nice selection of options: not too many advanced options cluttering up the menus, but the ones that are there are easy to find and comfortable to use. PCTEX comes with a package manager that can download additional packages or new versions of installed packages from CTAN and install them. This works like a breeze. The upcoming PCTEX version 6 (already available as a beta) has a range of wizards that help the user set all sorts of package options for various document classes and style files.

VTEX (http://www.micropress-inc.com/) is a TEX-version that is available commercially on MS Windows. There is a free version available for many other operating systems. VTEX/Free (http://www.micropress-inc.com/linux/) does not come with a specific editor, but it does come with an adapted TEX program. A big advantage of VTEX is being able to transparently convert PostScript into PDF. So with VTEX one can, for example, use PSTricks code with pdfLATEX without having to bother with pdftricks, escape sequences or going the TEX → DVI → PS → PDF way (and hence losing all pdfLATEX specific niceties).[5]

The commercial version of VTEX (http://www.micropress-inc.com/) is a different beast altogether from the free version. The freeware version (free as in ice cream, not as in freedom) is an adapted TeX program with various graphics options, but the commercial version is far more than that. It comes with a special editor, spellchecker, LATEX command completion, GUI for graphics creation and a dozen more components.

5. The pst-pdf package provides a ps4pdf script which is another way to use pstricks and pdftex together, albeit not as smoothly as with VTEX (http://www.ctan.org/tex-archive/graphics/pstricks/contrib/pst-pdf/).

For those who want to get as close to the old WYSIWYG programs as is possible without losing the superior typesetting capabilities of TEX there is **LyX** (http://www.lyx.org/). LyX is WYSIWYM — What You See Is What You Mean. It is not everybody's cup of tea, but it might well be yours. LyX offers a LATEX editor with a GUI and an on-the-fly approximation of your typeset document. When you type the LATEX code for a mathematical equation (or select its components from the GUI) the result gets typeset immediately and is displayed in your editor screen. It does not have the quality of the resulting DVI or PDF file but being able to see what one is creating helps the beginning LATEX user a lot. LyX also offers templates for various types of documents. These templates are self-explanatory "fill in the blanks" type documents that make it easy to create LATEX code. LyX is free software available for Linux/Unix, Mac, and Windows.

MS Windows users wishing to have a LyX-like program can buy the commercial **Scientific Word**, **Scientific WorkPlace** or **Scientific Notebook** (http://www.mackichan.com/ or http://www.sciword.demon.co.uk). The latter two programs also include a Computer Algebra System (MuPAD Pro). All three programs offer a WYSIWYG GUI for editing your document and shield almost all LATEX code from the user that prefer to not see it with buttons, palettes and menus. The LATEX code is, by default, depending rather heavily on proprietary class and package files. For those users that already know LATEX there is a "compatibility" option available that can even be made the default. This means it becomes trivial to save or import a regular LATEX document with these programs. If one imports a regular LATEX file, such as the documentation that comes with TEXlive, there do in very rare cases seem to be a few quirks displaying the WYSIWYG version of some LATEX code, even if similar or identical code in an actual Scientific Word file displays correctly. Nevertheless, these quirks are minimal and the package works admirably well in almost all cases. For those that are willing and able to spend the money, these programs have a lot to offer, and especially the smooth integration of an elaborate WYSIWYG editor for LATEX code with a powerful Computer Algebra System will appeal to some users. Free trial versions (fully functional for 30 days) are available from www.sciword.demon.co.uk/demo.htm with license numers available at www.sciword.demon.co.uk/serial.htm.

If all one is looking for is a pleasant editor with powerful functions, and nothing TEX-specific other than perhaps syntax highlighting, there are many options indeed. Actually, such editors often offer their syntax highlighting for other computer languages as well, so for the programmer they may be the best option available. Emacs and XEmacs (see page 7) belong in this category of course, but the AucTEX and preview-LATEX packages gained them a more extensive mention. Other examples of such editors would be **Vi** (`http://www.vim.org/`), **gEdit** (`http://www.gnome.org/projects/gedit/`) or **SciTE** (`http://www.scintilla.org/SciTE.html`). All of these are fine editors for source code with syntax highlighting and no more than a few options specifically for TEX and LATEX. Each one is free software. None of these editors offer nearly as many TEX-specific options as the other programs that have been mentioned, but their advantage is that they are far more generic than TEX-code alone. They might be a good choice for programmers.

6 Editor independence

We already mentioned at the end of section 4 that with TEX one can use different editors for different applications and people collaborating on a TEX document can all use their favorite editors.

With any of the independent editors or editors implicit in a particular distribution, TEX users can, if they desire, edit a TEX document with more than one editor. Thus, for instance, users may prefer their integrated or WYSIWYG editor for most straight forward editing of a TEX document; but when they need a certain capability they already know how to use in another editor, they can switch to use of that editor briefly without having to figure out how to do it in the editor they commonly use.

For example, suppose that we are using our favorite graphically-based editor to write a math paper, and at some point during the writing we decide we need to make an extensive change that can be done most efficiently using regular expressions[6]. But additionally suppose that we have never before used regular expressions in the graphically-based editor and don't want to take time now to figure out how to use regular expressions in our graphical editor or discover if the editor has a regular expression capability.. We could switch to an editor where we already know how to use regular expressions (e.g., Emacs, make the desired changes to our file, and then switch back to our primary editor.

6. Refer back to page 3 for more examples of editing in terms of regular expressions.

Such a temporary switch of editors also provides a way around occasionally needing a capability that is unavailable in one's favorite editor.

The reverse is also possible: one can stick with one's favorite editor while temporarily switching to use a different distribution of TEX. For instance, Dave Walden normally uses the MiKTEX distribution and the WinEdt editor; however on several occasions he has continued to use WinEdt to edit his document while temporarily switching to use of VTEX to compile the TEX document into HTML because he felt VTEX's approach to generating HTML matched his needs better than other options for producing HTML from a TEX file would have.

In summary, one of the considerable advantages of using TEX with its visible markup is that a lot powerful options are available for editing a TEX document.

Acknowledgments

Karl Berry and Will Robertson commented on a draft of this column; their general insight and specific suggestions are greatly appreciated.

Biographical notes

Yuri Robbers holds a degree in Animal Behaviour and is a teacher, a researcher and a published author. He's always had a keen interest in typography, possibly because his father is a professional typographer. Ever since he discovered LATEX in 1995, he's always done his best to avoid using word processors, and he has embarked on a quest to learn as much as possible about TEX and its derivatives and to apply this knowledge whenever possible. Contact him at yuri.robbers@gmail.com.

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. He holds an undergraduate math degree and completed a graduate school sequence of courses in computer science. More history is at www.walden-family.com/dave.