TEX TEXT EDITORS FOR THE IBM PC

A. Hoenig
M. Pfeffer

What's the best way to get material into your computer for use by TEX? The question was moot in the pre-PC era of TEX; you simply made do with whatever editor was available on your system. Editors and word processors have become big business in the PC world, some being strikingly better-designed than others, and the one you use can drastically affect your efficiency. Recently, I looked at a few text editors available for the IBM PC (and compatibles).

What are some of the characteristics a perfect text editor should possess? Remember, the sole purpose of any such editor is to accept your text into a source file that TEX can read and compile. Formatting by the text editor is a minor concern, since TEX handles all that for you.

The editor should produce standard ASCII output. Right away, this disqualifies most of your high-flying, superstar word processors — your WordStars, Multimates, Microsoft WORDs, and so on — which depend upon the presence of nonprintable ASCII characters in the text file to control format characteristics. But the ideal editor should be capable of elementary word- and text-processing capabilities — word wrap (so you don't have to hit the carriage return key at the end of each line), block management (so you can shuffle and reshuffle large chunks of text quickly and easily), and search and replace capabilities.

It should also work *fast*. Surprisingly, most of the superstars aren't, slowly saving files and searching for text strings as if being paid by the hour. I guess the business marketplace puts no premium on speedy execution (where many office workers *are* paid by the hour). This is unacceptable in a TEX session, when you're typically in a frenzy to correct that one last mistake and recompile your source file.

Finally, the editor should *not* be copy-protected. Whatever the relative merits of copy protection for the software developer, it's definitely a drag for the software user.

In addition, there are several other purely idiosyncratic conventions to which I expect my editor to conform. Most controversially, I want to to be able to use a mouse with it. A *mouse* is a palm-sized, electronic gadget, vaguely reminiscent of a real mouse, which you roll around your desk. These rolling motions translate into movement of the on-screen cursor. Pressing a button on the mouse usually affords additional editing control. Controversy arises because many power users find that any productivity gains are offset by the effort and time you waste moving your hand from keyboard to mouse. Admittedly, the mouse is irrelevant during the initial text entry operation, but it's worth its weight in Velveeta during the rewrite process, when keyboard control is less crucial. In my case, the time spent on rewrite dwarfs the time spent on the initial keyboard entry, so my mouse plays a crucial *roll* in productivity enhancement.

Next, I expect to be able to configure my keyboard so that the various shift keys are *sticky*, that is, when you press them they stay in effect for the duration of the next keystroke. The movement towards sticky-shift operations arose in response to the complaints of physically-handicapped users (who may not be able to press, say, the Ctrl and C keys at the same time), but I've found it to be the greatest thing since sliced bread, though I'm blessed with no handicaps (at least no physical ones). With a sticky-shift editor, typing an uppercase "A" (for example) becomes a sequence of 2 strokes, rather than a pair of simultaneous strokes. Your typing rhythm is less disturbed when you keep to a linear succession of single strokes. Nevertheless, some

editors are written to hog the keyboard interrupts, and won't let you "stickify" the keyboard.

In this column, I report on my usage of 3 text editors which distinguish themselves in various ways. All produce ASCII text files, are *not* copy protected, and do their jobs fast enough to be at or close to the limits of human perception.

PC-Write

PC-Write is a nifty little program. Although it was manufactured as a full-featured word processor (which it in fact is), it functions beautifully as a text editor. Like most current-generation word processors, PC-Write works by accepting as commands non-standard key sequences — that is, key sequences involving the function keys, the Escape key, or the Alt or Control shift keys.

The thing I like best about PC-Write is its customizability. Keys can be reprogrammed so that single keystrokes represent dozens of PC-Writecommands, or you can change the command sequences from the PC-Write standard to your own standard. On the fly, you can define a single keyboard "macro," but a PC-Write macro is simply a concatenation of different commands and, as such, possesses a fraction of the sophistication of a TEX macro. Sticky shifts are one of the things you can customize into your version of PC-Write.

There's a fearful symmetry between TEX and PC-Write — both are in the public domain. A disk with the program and reference manual file (it prints out to 180 pages) can be yours for a whopping 10 bucks. Set the program up, run off a copy of the manual, and you're off and running. For a few more dollars, the source listing of the program (a mix of assembly language and Pascal) is yours. The developers, Quicksoft, Inc., are pioneers in *freeware* — they hope its use will spread by word of mouth, and they rely on your sense of fair play to remit an additional contribution to them.

PC-Write comes with support for several of the standard mice. Also, the editor lets you configure it so that shift keys are sticky. (It's a simple matter of changing a special `ruler.def` auxiliary file to which PC-Write is accustomed to check for ruler-related matters and customizations.)

PC-Write, although good, is not perfect. For one thing, it's limited to document files of 64k or less. If you have larger files, it handles them only with a clumsy swapping maneuver — pieces of your file are swapped back and forth from disk to memory as you need them. Furthermore, PC-Write may not be so convenient for programmers. Although you can indent your programs, you have to find out how on your own. (The manual gives little help.) It can't check your matching delimiters for you.

I reviewed version 2.55, the current version. Reportedly, a new version will be available in February or March, presumably for the same price. This new version should handle larger files, hopefully with no degradation in speed, and should come with a spelling checking program, among other enhancements.

The Norton Editor

Peter Norton, no relation to Ed, has won fame and fortune writing as a PC guru. He writes extensively about the guts of the IBM PC (the source of his fame). He's also marketed a clever bunch of programs which extend the PC's operating system in a few much-needed ways (the source of his fortune). Recently, he's lent his prestige to a cute little editor — The Norton Editor — which he bills as *the* programmer's editor.

The Norton Editor makes no pretense about being a word processor. It's incapable of doing many fancy word processing tasks, but does program text editing well. Like PC-Write, it word wraps, manages blocks, and searches and replaces. NE commands are two-key sequences, the first key being one of the function keys, and the second executing a command. Pressing a function key summons a menu of choices to the bottom of your screen, so you don't have to memorize the command sequences. Fortunately, once you know what commands you want, you can tap the key sequences at high speed, and ignore the menus.

This editor is easy to learn. Once you've mastered a short pamphlet (the entire documentation) you know all there is to know about it.

NE provides no sticky-shift provision within the editor, but it's easy enough to get hold of a public domain sticky-shift program (from a local bulletin board) and run it before invoking the Norton Editor. As with PC-Write, there is built-in mouse support.

NE is superior to PC-Write in its ability to work as a programmer's text editor when you've finished using it for TEX. There are a variety of ways of dealing with program indentation and checking of matching delimiters.

The Norton Editor is a new product, and Norton will probably address some missing features in future versions. For example, NE is bereft of any semblance of a macro facility, a shocking omission.

Like PC-Write, this editor is cheap. It lists for $50, but you can do better at some mail-order houses.

Epsilon

Epsilon is a sophisticated, programmer's text editor closely modelled on the EMACS editor. Epsilon will apparently and happily work with document files of any size. As fast as the other two editors are, this one is even faster.

This current version (version 3.x) has been programmed in EEL — Epsilon Extension Language — and anything (within reason) that you don't like about the editor, you can change, by simply rewriting that module. The folks at Lugaru Software — Epsilon's developers — provide EEL source files for Epsilon and an EEL compiler as part of the Epsilon package. EEL is a C-like programming language.

There are a few things which cannot be changed. Epsilon "steals" the keyboard interrupts, so there's no way you can get a mouse to work with it. In the same way, no BBS sticky-shift utility will work, but at least you can reprogram Epsilon to accept sticky (or shifty?) behavior.

On the other hand, Epsilon gives you far greater control over text editing (provided you need neither sticky shift nor mouse). The editor provides an extensive mechanism for creating buffers to store deleted material, additional files, or whatever. Apart from EEL, there's an extensive macro facility, and you can define and use any number of macros on the fly. You may reassign different command strings to Epsilon commands. The Search facility — whereby you search for a particular string in your text file — is *incremental*. Suppose you're looking for the word "Mozart" in your file. Normally, your text editor waits for you to enter the master's name before commencing the search. Epsilon searches as you enter the word. It senses that first "M," and looks for it. When you enter the "o," Epsilon refines its search for the first occurrence of "Mo." When you type "z," it looks for "Moz," and so on. Incremental searches are usually much faster for you. Lugaru has embedded numerous other impressive bells and whistles in their product.

Epsilon is clearly aimed at power users, and nowhere is this more evident than in the manual, which contains almost no examples. Also, their discussion is steeped in the lingo of EMACS, which is a dialect of the usual text editor terminology. The manual takes a bit of getting used to.

All this power has its price. Epsilon weighs in at $195, more than the other two editors, but still a good deal less than much other software on the market. It may be worth checking prices at mail order outfits who cater to hard-core programmers; maybe such places will carry this program at a discount.

And Now for Something Completely Different — Lightning

I hope you use a spell-checking program in conjunction with your editing. There are any number of good programs out there. Just make sure you use one that allows you to add entries to a user dictionary when they're initially flagged as errors. That way, as you use your checking program over a period of time, all special TEX commands enter that dictionary file. After a short while, the program catches not only your prose misspellings, but also your misspellings of TEX commands. I save a lot of time in TEX compilation that way.

Anyway, a new gimmick in the spell check arena is the program Lightning. Not quite a spelling checker, and yet not quite *not* a spelling checker, Lightning works alongside your text editor. It checks your spelling as you enter your text. As soon as you enter a space (or other word delimiter), Lightning checks the word against a dictionary file it has previously loaded into your computer's memory. If the word has no match in this dictionary, you get a gentle beep, to which you may hearken. Note, though, you *cannot* use Lightning in the traditional spell-check manner. It can't search your document file for you in one fell swoop to identify all errors in one pass. You can only catch errors as you make them. You also get to use Lightning in a Thesaurus mode. In this way/fashion/manner, you can spice up your writing/setting forth on paper/authoring/prose/composition/work with synonyms as you write.

The program performs as advertised, is real easy to use, and is cheap. Borland International is a pioneer in the development of good, cheap software. (Remember Turbo Pascal?) The Lightning program is only $99.95. Lightning works well with both PC-Write and the Norton Editor, but not at all with Epsilon, presumably because Epsilon is so greedy of some system interrupts. Borland touts this product as being in the vanguard of a whole series of intelligent software products using the same software engine as does Lightning.

I have some reservations about the utility of the program. So often, what you enter at the keyboard is just the first draft of many. What's the point of getting it letter perfect, when so much of it will be consigned to that great bit-bucket in the sky? However, if you're the type who has done all your

serious revising by hand so the draft you key in is close to final, Lightning could be for you.

I praise the folks at Borland for one aspect of their design. Too often, memory resident programs continue to hog memory long after you're through using them. If you need that memory for some other program, too bad. You can only reclaim this memory by rebooting your system. Not so with Lightning. There is a command which kills the program and knocks it out of the system.

Sources

PC-Write Quicksoft, 219 First N #224, Seattle, WA 98109, (206) 282-0452

The Norton Editor Peter Norton, 2210 Wilshire Boulevard, Santa Monica, CA 90403, (213) 399-3948

Epsilon Lugaru Software, Limited, 5740 Darlington Road, Pittsburgh, PA 15217, (412) 421-5911

Lightning Borland International, Inc., 4585 Scotts Valley Drive, Scotts Valley, CA 95066, (800) 556-2283