TEX for the Word Processing Operator

ROBIN L. KUBEK

ORINCON Corporation
9363 Towne Centre Drive
San Diego, CA 92121

ABSTRACT

The purpose of this paper is to illustrate the need for a training program for word processing operators, secretaries, general management support staff, *et al*, in the use of `plain.tex` without intimidating them with TEX's programming language.

The paper will describe a technique developed by this author to train those familiar with word processing software, such as Microsoft Word, in the basic use of TEX by comparing TEX commands to word processing commands and functions. This type of training concentrates on TEX *operations* alone — the philosophy being that it is possible to learn to drive the car without knowing how to put the engine together.[1] By the use of parallel language, it will be shown that it is possible to have support personnel unfamiliar with TEX producing basic text manuscripts (letters, memos, etc.) and simple math within a few hours' worth of training.

1. Introduction

The subject matter of this paper was born after several years of frustration over the non-availability of TEX training at a local level, in an easily understandable form, *and* geared toward secretarial/management support personnel who don't have degrees in physics, engineering, mathematics, or computer science — *but* — are expected to type and format technical papers as if they did, *and* — by the way — should be fluent in Greek as well![1]

Investigation in my local area has shown that TEX exists on the VAXes and PCs of many high tech R&D firms and at the universities, *but is not supported internally* by the Information Systems Support Division of any company or taught as a class at any school. TEX use seems to be largely confined to the technical professionals with degrees in the fields listed above, who have taught themselves with only the *TEXbook* to guide them (and wouldn't have a clue as to how to teach anyone the program who doesn't speak *at least* three programming languages). For those of us who had to look up the word "algorithm" several times in the dictionary and still wouldn't recognize one even if it asked us to dance, a major dilemma arose when faced with the prospect of learning TEX "by the book". Not only is the *TEXbook* oriented toward programming, but a lot of the commands don't even look like they're in English![2]

Furthermore, TEX really has nothing to do with word processing, *per se*. It is not a word processing software package like WordPerfect or Microsoft Word; it is not a computer language such as FORTRAN or Pascal; and then there are funny words associated with TEX like `\baselineskip`, `\parskip`, `\halign`, `\vbox`, etc. (and let's not forget all those #!$%* curly braces). In other words, TEX doesn't look like anything a secretary or word processing operator has ever seen before in her/his life! Unfortunately, the very fact that TEX *looks* so foreign is the major reason many support personnel pass on the opportunity to learn it; and, coupled with the fact that there is no formal support network where the TEX trainee can turn when she runs into problems (which is often in the beginning), makes TEX a very intimidating program to tackle. It is true that "the best software in the world is useless if

[1] If you made it all the way through this sentence/paragraph, it is advisable to breathe now!

[2] Examine the spelling of `\eqalign` and `\eqalignno`. *Where* is the *u* that's supposed to follow the *q*?

you don't know *how* to use it,"[2] and probably explains why TeX resides inside so many computers but is sadly underutilized. Another misconception is that TeX is only used for math.

So how do we, the TeX community, go about introducing our ferocious looking friend (who is really a pussycat, once you get to know him) to the word processing community at large? If you follow along, I will explain my in-house "TeXnique" of introducing TeX the quick and semi-painless way to the Word Processing Operator.

2. Training Wheels

The first big hurdle in teaching TeX to anyone is convincing them that (a) no matter *what* they've heard, TeX cannot bite,[3] and (b) that while all those `\backslashes` do look a bit bizarre, TeX commands really do make sense. Once you've accomplished that, begin to familiarize your student with the basic TeX commands.

2.1 The Preamble

The preamble (the first several lines at the top of a file) is the best place to start comparing TeX commands to their equivalents in the word processing software. Below is an abbreviated list that I give to my TeX trainees of typical preamble commands and their meanings.

`\magnification=\magstep1`

Magnifies the font 1.2 times. If you are using a 10-point font, this magnification will make the type appear at approximately 12 points, which looks like 10 pitch (pica) on a typewriter. If you do not specify magnification, the default is `\magstep0`. 10 points with no magnification looks like 12 pitch (élite) on a typewriter. **Note:** On Macintosh, the TeX fonts do not magnify well past `\magstep1`. If a larger font is called for, use a PostScript font.

`\font\cm=cmr12 at 12pt`

Font definition for 12 point Computer Modern Roman (TeX font). **Note:** The normal TeX default font is `cmr10`. Any font that is not the default MUST be defined and called out prior to use.

`\footline={\hss\folio\hss}`

Your footer. `\hss` stands for **H**orizontal **S**tretch or **S**hrink and is "infinite glue". By having `\hss` on either side of `\folio` (which is the macro for the page number), your page number will be centered at the bottom of your page. **Note:** This `\footline` is the default setting in TeX and doesn't have to be placed in your preamble unless you are changing the font size, putting the number in either the right or left corner, or adding other information.

`\pageno=1`

Self-explanatory. If your page number is to begin with a lowercase roman numeral (for tables of contents, acknowledgement pages, etc.), you would give `\pageno` a value of -1, -2, etc. — whatever number you wish to start with ($-1 = i$, $-2 = ii$, $-3 = iii$, and so on). **Note:** The `\folio` macro in your `\footline` defines any `\pageno` less than 0 as a lowercase roman numeral.

`\hsize=6.5in`

This is the horizontal size of the page. Unlike ordinary word processing (where you set the *margin* space), in TeXyou set the *text* space. `\hsize=6.5in` is the default horizontal setting in TeX and does not have to placed in your preamble. *However*, until you become comfortable with the differences between TeX and ordinary word processing, it is a wise idea to list the differing commands up front.

`\vsize=8.9in`

This is the vertical size of the page. Again, text space is being set rather than margin space. This `\vsize` is the default setting and does not have to be set in the preamble, but, like `\hsize`, it is a wise idea to keep it up there in the preamble until you become comfortable. **Note:** With an `\hsize` of 6.5in and a `\vsize` of 8.9in, you will end up

[3] Well, maybe nibble a little, but not hard!

with a 1-inch margin on all four sides of your paper (and isn't that what you would have set your margins to anyway?).

`\baselineskip=12pt`

`\baselineskip` is the amount of space between lines, or, more precisely, from the bottom (or base) of a line to the bottom of the next line. In regular typesetting and some desktop publishing software, "leading" is the term used, and means the same as `\baselineskip`. In general, the leading or `\baselineskip` should be 2 points greater than the font for single-spaced text — or 12 points for a 10pt font. For space-and-a-half and double-spacing with a 10pt font, the math is simpler: a 15pt `\baselineskip` on a 10pt font for space-and-a-half, and a 20pt `\baselineskip` for double-spacing.

`\parskip=6pt plus 3pt minus 2pt`

Additional space between paragraphs — but not as much as "two returns". The plus and minus points give TeX additional shrink and stretch "glue" to use at its discretion.

`\parindent=20pt`

This is the paragraph indent. 20 points is approximately the same as 5-space indenting.

`\overfullrule=1pt`

The `\overfullrule` command is a handy little device that has no real equivalent in ordinary word processing. This command, when set for a value greater than 0, will put a black line beside any text that exceeds the `\hsize` on any line. **Note:** You will receive an `Overfull \hbox` message in your log when you send your document to typeset, but if your "overfullness" is less than 5 points, you probably will not be able to see it on the page, and thus not be able to fix it, without the line generated with the `\overfullrule` command.

`\raggedbottom`

No, this is *not* a description of how you will feel after a day of TeX training! This command tells TeX to permit a small amount of variability in the bottom margins on different pages in order to make the other spacing uniform (p. 111 of the *TeXbook*). If you choose not to use `\raggedbottom`, the default is `\normalbottom`, which will make all of your bottom margins the same on every page — even though it means stretching the text space on the page and making it look funny.

Word Processing — First Pass

Once you've established the preamble, it's a good idea to have your student type the commands into a document. Not only does this get her into the feel of typing `\backslash` in front of just about *everything*, it's also a good opportunity to show her that she can use *any* editor/word processing application that suits her fancy. On a VAX, EDT is my editor of choice, but I've also input TeX files in Word-11, TEDI, and on Macintosh, in Microsoft Word.

One word of caution in regards to inputting TeX with a word processing application. You may use all of your "gold" key applications, editor keypad, and user-defined keys (UDKs) as you normally would to move around in your document, cut and paste, etc. However, you may not *format* in that application. No bolding, underlining, tabbing, super- or subscripting, etc. If you are using an application that puts in internal formatting (such as Version 4.0 and above in Word-11), you must strip the file of the formatting *before* sending it to TeX. Your best bet is to simply type in block paragraphs and let the text word wrap. While in a word processing application, you can still type your TeX commands, `\backslash` and all.

On Macintosh I've found that, while typing in Microsoft Word is fine, it takes a lot more steps to get it into *Textures* (the Mac version of `plain.tex`). First, the document must be stripped of formatting by saving as a text-only file, then copied into the clipboard. *Textures* must then be opened and the clipboard pasted in. If the document is too large to move with one cut and paste, *Textures* has to be closed, the text-only Word document re-opened, and the whole cut-and-paste procedure performed again. Consequently, when teaching TeX on a Mac, I wholeheartedly encourage the student to just type in *Textures* — which operates similarly to EDT.[4]

[4] Of course, if she's stubborn, I'll let her type in Word and go through the whole horrid procedure of moving the

2.2 Macros

Once you've gotten past the preamble (and the dreaded first document), macros are a real snap. Macro simply means *definition*. Macros work a lot like UDKs or cut and paste. Instead of storing a string of keystrokes, commands, etc. in memory under a key or in a buffer, you simply define your string (`\def`) and assign it a label (`\cc`). The definition of the keystrokes is then enclosed in curly braces ({}). For example:

`\def\cc{\centerline{}}` This macro defines a `\phantom` (invisible) `\centerline` the depth of 1 `\baselineskip` (or, in plain English, a blank line). The empty set of curly braces next to `\centerline` means the set is empty.

The preamble commands are made up of macros (*i.e.*, `\magnification, \footline, \pageno,` etc.). The *TeXbook* lists all of the macros that come with TeX (hundreds of them). In addition, you can write *your own* macros (just like `\cc` above). Macros can be abbreviations for typing complex commands, or simply for words, phrases, or sentences that are used over and over again. They can contain font changes or formatting such as bolding, italicizing, underlining, and the like. By writing a macro, you can save yourself untold number of keystrokes (or "mouse clicks" or dragging down menus, etc.). You can even "nest" macros within macros. For instance:

`\def\bb{\cc\cc\cc}`

By using the macro `\cc` inside your definition, you've now got a macro for 3 blank lines. Some other macros that I have found helpful are:

`\def\ie{{\it i.e.\/},}` macro for italicized "i.e.," with *italic correction*[5]

`\def\ea{{\it et al\/},}` macro for italicized "et al," with *italic correction*

`\def\eg{{\it e.g.\/},}` macro for italicized "e.g.," with *italic correction*

`\def\underscore#1{$\underline{\smash{\vphantom{y}}{\hbox{#1}}}$}` macro for underlining text so that the underline goes *below* a lowercase descender[6] rather than *through* it.

Regarding `\underscore`, above, I am unaware of *any* word processing application where this feat is possible using the underline key/command. This is also one macro where it is wiser to quote Joe Isuzu[7] rather than to try and explain to a TeX first-timer the meanings of all the commands in the definition.

Word Processing — Second Pass

Once your student has grasped the concept of macros, have her go back to her document (the one with the preamble in it), and create some of her own macros for it. If you want to get her *really* crazy, show her how to write a *whole* document using *nothing but* macros! See Figure 1.

2.3 (The Dreaded) Math

Now that you have your student (a) thoroughly confused, and/or (b) totally in awe of your incredible genius and unique abilities, it is time to drop the "Big One"[8] on her — *Typing Mathematics Equations*!!!!!

Contrary to every word processing operator's belief, math typing *does not* have to be one of those situations where, if given a choice between typing the math and hurling yourself in front of train — you'd choose the train![9] The key to typing math in TeX is to "walk (and talk) yourself through the equation." In other words, if you "say" the equation, you can type it. The hardest part of math typing, TeX style, is remembering all those #!$%* curly braces!

The answer to all of your student's questions, when it comes to math typing is "TeX will take care of that." TeX will:

document. I can guarantee she'll abandon Word entirely for the second document, and I won't even have to nag.

[5] Italic correction squeezes in a liitle bit more space between the last italicized character and the next "normal" character. Otherwise, the slant of the italics could make the letters/characters overlap one another or make them appear too close together.

[6] Fancy terminology for the part of the letter which extends below the baseline: g, j, p, q, and y.

[7] "Trust me".

[8] A little organ music would be nice right here, preferably with very deep bass chords.

[9] That is, if you can type the math in TeX. If I had to type math in ordinary word processing, I too would choose the train!

```
\def\mom{\par  Dear Mom, I love you forever.}
\def\cookies{\par  Please bake me some chocolate chip cookies.}
\def\butter{\par  You're the best Mom in the Whole Wide World!}
\def\junk{\par  Can we go to MacDonald's tonight for dinner?}
\def\ps{\par  By the way, if my teacher calls, I really didn't throw
        the whole can of red paint on her!
        \medskip
        ({\it It was only half a can\/})
       }
\def\snow{\mom \cookies \butter \junk \ps}
\snow
```

Dear Mom, I love you forever.
Please bake me some chocolate chip cookies.
You're the best Mom in the Whole Wide World!
Can we go to MacDonald's tonight for dinner?
By the way, if my teacher calls, I really didn't throw the whole can of red paint on her!

(*It was only half a can*)

Figure 1: A macro to go crazy by ... and the end results!

- center
- proportionalize all the delimeters
- change font size for super- and subscripting
- do Greek and symbols (*all* symbols!)
- put equation numbers where they belong
- put equal space above and below a displayed equation and text
- *not change* your line spacing when typing math in a text sentence
- make your equal signs line up in a complex equation
- change the baby and get dinner started before you get home![10]

Instruct your student that all she needs to do to invoke math mode is type a dollar sign or two; one `$`, and she can type math inside of a text sentence. To leave math mode, she simply types another `$` (a lot easier than having to use code keys, alternate font codes, or leaving the document altogether, going into another program to type, and then having to port the equation back). If she wishes to type a displayed equation, she uses two dollar signs (`$$`) to get in and two `$$` to get back out. Of course, with math, you can instruct until you turn blue, but the best way to teach the concept of math typing is...

Word Processing — Third Pass

The hands-on method is the only way to go when it comes to math typing. Have your student create a document (or use the same one she's been practicing in), and give her some *very simple* math equations to start out with. For example:

$$\frac{1}{2} = \frac{\alpha}{\beta}\,, \quad e = mc^2\,, \quad \text{and } x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Of course, you will have to give her the commands for ± and $\sqrt{\ }$. The next thing is to tell her that whenever she sayes a command (such as "over") that she will use a `\backslash` in front of it. Tell her to think of superscripts as "up" and to use the caret symbol (`^`); subscripts are "sub" and the command is the underline symbol (`_`). Curly braces are also something that need to be walked

[10] No, no — just checking to see if you're still with me on this one.

through. Learning their placement just comes with a little time and practice. It helps if, in talking the student through, you tell her to say "begin" and "end" with each equation and its parts, and to put a { with the "begin" and a } with the "end". Furthermore, tell her to type (in regular English words) everything that she says in the equation (numbers, of course, can be typed in regular English numbers). Thus, the equations above, would look like this:[11]

```
$(begin){1\over 2}(end) = (begin){\alpha\over\beta},(end)$
$e = mc (up)^2,$
$$x = (begin){-b (plus or minus)\pm
                 (square root)\sqrt (begin){b (up)^2-4ac}(end)
              \over 2a}.(end)$$
```

3. Conclusion

...and at the end of the training session, TeX said `''\relax''`*, and it was good.*

By now, you've given your student a smattering of the things that TeX can do. This is by no means a complete smattering. It is, however, enough for an afternoon's worth of training — and enough to get your student's appetite whetted for more TeX. It is enough to get your student thinking about all the things she might be able to accomplish with TeX. And it is enough to make the first few chapters of the *TeXbook* semi-understandable in one reading.

There are quite a number of basic TeX skills that have not been touched upon here. Those include tab fields for tables; itemizing for outlines; commands for underlining, bolding, and italicizing; and commands such as `\leftline`, `\rightline`, and `\centerline` (not to mention how to read error messages, or get back at TeX for some of the snotty comments he's prone to make). But stop and think about what *has* been accomplished here. The training, as outlined above, has taken a program that has generally been perceived as mysterious, forbidding, and unfamiliar, and made it more understandable and *accessible*. It has given your student enough to get started in TeX, and enough to produce a basic text document — and be comfortable doing it. It has shown her that math math typing can be pretty straightforward and much simpler than in any other software application that she may be familiar with. Your student will, of course, have many questions over time, and more than once will require your assistance. She will not know what makes TeX tick, or really *why* TeX works at all. *But*, she will be able to "drive the car."

There will, of course, be other afternoons to show your student (and presumably, co-worker) how to set up tables. You can walk her through `\item` and `\itemitem` over the phone. The same holds true for `\leftline`, etc. But, hopefully, you can see where this technique of in-house training can lead, and you will be able to go about setting up your own training program to meet the needs of your company and its employees.

One word of warning, however; once you have taught a word processing operator the basics of TeX, she may not want to type in WordPerfect, Microsoft Word, Word-11, or any of the myriad assortment of ordinary word processing applications, ever again. She may develop an extreme disdain for ragged right margins and non-proportional spacing. She may even begin to talk funny (offering the opinion that an "overfull `\hbox` is by far a worse problem than an underfull `\vbox`!"). Furthermore, the more she learns about TeX, the more she's going to want to learn. She may no longer be content to merely "drive the car" — she's going to want to know how to put the engine together. You just may be creating a monster![12]

Bibliography

[1] Knuth, Donald E. *The TeXbook*. Reading, Mass.: Addison-Wesley. 1984.

[2] Rees, Clair. "The Training Issue." *WordPerfect, The Magazine* March 1989, p. 4.

[11] The "(begin)", "(end)", "(up)", etc. (all the words enclosed by parentheses) that you see typed in the equation commands are *only* to show what you want your student to *say* while she is typing. In the actual file, these words would not be typed.

[12] Grrrrr!