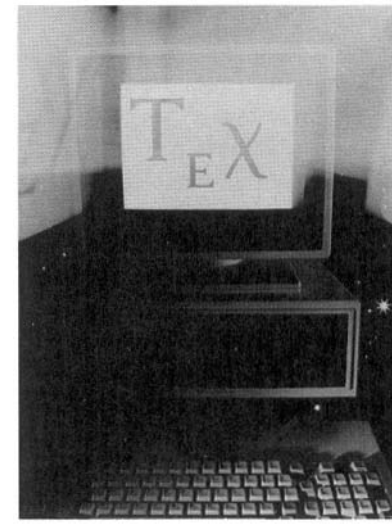
Technical writing and word processing using TEX



By A. E. Siegman

If you write technical papers or prepare manuscripts containing any significant amount of mathematics, you want to learn and use TEX. That's the bottom line of this article.

TEX-nically speaking (and that's how "TEX" is pronounced), TEX is not a conventional word processor or text editor. Rather, it's a typesetting language. As such, it is a superb tool for preparing journal articles, long technical reports, and even entire books (my 1300-page *Lasers* book was done entirely in TEX). It has especially powerful capabilities for typesetting mathematics.

But once you begin working with TEX, you will use it for a great many other things. You can write letters with your own letterhead and logo included in a predefined template; prepare slides and overhead transparencies directly from your manuscripts; compose flyers and announcements; and accomplish any other task that involves putting letters and ink marks on paper.

How does TEX differ from the conventional WYSIWYG ("what you see is what you get") word processors so widely used on personal computers today? With a WYSIWYG system, what you see on the computer screen is indeed more or less what you eventually get (with luck, and discounting glitches) on the printed page. If you select a word in your document and change it to italic (slanted) type, you immediately see that word in italics on the screen. If you select a subheading—for example, Overview of Today's Weather—and change it to boldface or change the type style or the indentation, you immediately see it in the new style on the screen.

Using TEX is a little more complex. With TEX, if you want to write "TEX is a *really superb* system!", you must type into your source file "TEX is a {\sl really superb} system!". The coding indicated by the TEX command {\sl ...} says that you want to use slanted, or italic, type for the words within the curly brackets. Basic rules in TEX are that anything enclosed in curly brackets is to be treated as a group; anything beginning with a backslash is a TEX command or a special symbol. You hardly need know anything more than that to use TEX.

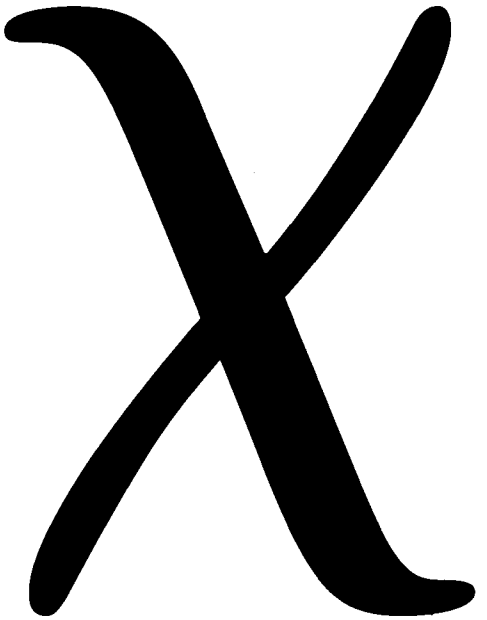
But what about that subheading? To produce this subheading using TEX, I could type into my source file at the appropriate spot:

```
\vskip 15 pt plus 5 pt minus 3 pt
\leftline{\bf Overview of Today's Weather}
\vskip 10 pt
```

What does this jargon mean? The \vskip or "vertical skip"

commands say that I want 15 points of additional space above and 10 points of additional space below the subheading line itself. In fact, in the first line above I've told T_EX it has permission to stretch the vertical spacing above the heading to as much as 15 + 5 = 20 points or or squeeze it down to as little as 15 − 3 = 12 points, if this will make the rest of the page come out better.

The \leftline{. . .} command says I want the bracketed text following the command placed up against the left margin, with no indentation. I could have typed in \noindent instead, and there are also built-in \centerline{. . .} and \rightline{. . .} commands available in TeX. The \bf command in front of the text of the subheading says, set the following words in boldface type, but turn off the boldfacing at the end of the group.



Suppose there are many such subheadings throughout my document and I want them all formatted in exactly the same way. The easy approach in T_EX would be to use a \subhed macro. At each place where a subheading occurs, I could type \subhed{Overview of Today's Weather} or whatever the title of this subheading might be.

T_EX contains many useful built-in macros, and $T_EX$perts have made available packages of standard macros; but I might want to specify for myself what the \subhed macro is to do. To accomplish this, I could insert at the beginning of the source file the following macro definition to produce the same results as above:

```
\def\subhed#1{\vskip 15 pt plus 5 pt minus 3 pt
\leftline{\bf #1}
\vskip 10 pt}
```

In this little ritual, \def is a standard T_EX command that says I'm defining a new personalized T_EX command or macro called \subhed. Each time I use this macro in my source file, it will do the all the things spelled out in this macro definition. That is, each time this macro is invoked, it will do the requested vertical skips. My definition also says that this particular macro will have just one argument, indicated by the #1 immediately following the macro name, and I want the contents of #1 set as a \leftline and in boldface.

If I later decide I want all my subheadings in italics instead of boldface, I don't have to go back through the entire document and change each subheading, one by one. I just replace the \bf by \sl (for "slanted") in the macro definition, and voila!—it's done. I can also change the spacings all at once if I want. That's the beauty of the macro approach.

More than that, if I don't even want to learn this macro stuff, I can prepare my source file using undefined macros—for example, I might start a book chapter by typing:

```
\chapter{Chapter 1. The World's Weather Systems}
\section{Section 1.1 Weather Systems on a Global Scale}
```

and then begin typing the text of Section 1.1. When I'm ready, I can ask some local $T_EX$pert to define the \chapter and \section macros for me to do whatever I want—including building a table of contents with these headings and the associated page numbers automatically included in it.

From coding to typesetting

So, how does a T_EX user, working on a personal computer at home or in the office, convert this "source file" with all its macros and embedded commands and other stuff into a typseset page, once the source file is all complete (including the final T_EX commands "\par \vfill \end" which mean, end this paragraph, fill the rest of the page with empty space, and end processing).

The answer is that you first prepare and save your source file, including text, mathematics (more about that later) and imbedded T_EX commands, as an ordinary text file on your personal computer. Then you tell the T_EX typesetting program itself to go to work on that file. The T_EX program reads your source file and, more or less instantaneously, prepares a printable and viewable typeset output file for you, obeying all your commands and also using its own considerable intelligence about text, mathematics, and typesetting.

I happen to do all my $T_EX$ing on an Apple Macintosh, using a commercially available implementation of T_EX for the Macintosh called Textures. Other similar programs for the Macintosh computer include another commercial program MacT_EX, and a freeware version, OzT_EX. There are also several versions of T_EX available for IBM PCs. I'm not personally familiar with these, but I believe they have comparable capabilities. (See article page 53.)

With Textures, I can prepare my source file and edit it on screen, using a standard Macintosh point-and-click text editor that's built into the Textures program. When my

source file (or part of it) is complete, I pull down a menu selection called "Typeset" and Textures begins typesetting my source file. A logging window opens on the screen to record the progress of the typesetting operation and any errors or complications encountered.

As soon as the first page of the document has been typeset, a real typeset replica of this page—exactly as it will appear on the printed page—appears in a window on the screen (this is called previewing). I can reduce the scale in this on-screen preview window to get an overview of what the entire typeset page looks like, or I can expand the scale to look at fine details of some complex matrix or array of subscripts and superscripts anywhere within the page.

As each page is typeset, it can be brought up on the screen and previewed while Textures continues typesetting the remainder of the document. It takes only a few seconds to typeset each page—a 10-page document might take half a minute or so to be completely typeset. T_EX with an on-screen previewer is not quite WYSIWYG, but it's pretty darn close. The Textures program will run quite happily on a bottom-of-the-line 1 MByte Mac Plus, using only floppy disk drives if necessary, though a small hard disk (20 or 30 MBytes) makes things quite a bit easier.

If the typeset document is right the first time, I can simply send the typeset output to the printer. On the Mac, adequate quality printed output can be obtained, though slowly and with a fair amount of noise, on an inexpensive ImageWriter dot matrix printer. With a LaserWriter or other Postscript™ laser printer, the printed output will be fast, quiet, and beautiful.

Whether the output is previewed on screen or printed on a dot matrix or laser printer, the output pages will be exactly identical, *i.e.*, every character and symbol will be in exactly the same place on the page. The individual characters will just be somewhat coarser on the dot matrix output than on the laser (or ink jet) printer output.

Of course, the typeset output may not be quite right the first time; or the logging window may indicate some errors or other problems in the source file. If so, after looking at the on-screen preview, but without needing to do any printing, I can immediately click back to the source file without leaving Textures; make the necessary corrections using the built-in editor; re-typeset the source file; and preview it again, with only a few seconds turn-around time. It's really quite handy.

T_EX also has an extensive error-handling capability built directly into the typesetting process itself. If an error or an improper command shows up in the source file during the typesetting process, T_EX will pause and ask you (through the logging window) whether you want to correct the error right then and continue typesetting, ignore the error and continue, or quit at that point. Even with a buggy source file, you can usually get useful output on the first pass.

Getting Started

To get started with T_EX, you need to go to any technical bookstore and buy "The $T_EX$book" by Donald E. Knuth (Addison-Wesley, 1984) and then find a convenient system on which you can try out T_EX for yourself (but do this on a personal computer or workstation using a T_EX program with a previewing capability). Further information about T_EX can be obtained from the T_EX Users Group, c/o American Mathematical Society, P. O. Box 6248, Providence, R.I. 02940.

I've already mentioned that Textures from Blue Sky Research, MacT_EX, and OzT_EX are implementations of T_EX for the Macintosh. So far as I know, they are all good programs (OzT_EX is either freeware or shareware). Look in any Mac magazine electronic bulletin board for more details. Commercial versions for the IBM PC include MicroT_EX from ArborText in Ann Arbor, Mich., PC T_EX from Personal T_EX in Mill Valley, Calif., and a program called $T_EX$pert. Public domain versions include emT_EX and sbT_EX. I've used none of these and hence cannot evaluate them, but magazines, electronic bulletin boards, or user groups for IBM PCs should have further information. T_EX is also available on many, if not all, of the more powerful Unix workstations and mainframes.

For more details on getting started, see this month's "Electronic information" column on page 53.

Why bother?

All of the above may, however, still sound like a little too much programming and computing just to write a letter or even a journal article. Why go to this (really pretty minimal) complexity when powerful WYSIWYG editors are available and easy to use? The primary answer is T_EX's handling of mathematics. If your report or manuscript has display equations, Greek letters, math symbols, matrices, or lots of subscripts and superscripts, and you want to be able to type these in easily, modify and edit them easily, and have them come out as beautifully typeset mathematics, then nothing will compete with T_EX.

Suppose, for example, you want a numbered display equation in your output that says:

$$\alpha = \frac{\sqrt{\cos\theta_n}}{\beta^{3/2}} \qquad (3)$$

To accomplish this in T_EX, you type into your source file

```
(—last line of preceding text—)
$$
\alpha = {\sqrt{\cos \theta_n} \over \beta^{3/2}}
\eqno(3)
$$
(—next line of following text—)
```

When T_EX typesets this expression, it gives you back a gorgeous, perfectly proportioned, centered, spaced and numbered book-quality display equation, which you can preview on the screen or put out to paper.

Let's interpret the coding. The double $$ signs indicate the start or end of a display equation (single $ signs would

Fig. 1

$$\begin{pmatrix} B \\ C \end{pmatrix} = \begin{pmatrix} \cos\delta_1 & (i\sin\delta_1)/\eta_1 \\ i\eta_1 \sin\delta_1 & \cos\delta_1 \end{pmatrix} \begin{pmatrix} \cos\delta_2 & (i\sin\delta_2)/\eta_2 \\ i\eta_2 \sin\delta_2 & \cos\delta_2 \end{pmatrix} \begin{pmatrix} \cos\delta_3 & (i\sin\delta_3)/\eta_3 \\ i\eta_3 \sin\delta_3 & \cos\delta_3 \end{pmatrix} \begin{pmatrix} 1 \\ \eta_4 \end{pmatrix}$$

```
\def\layer#1 {\pmatrix
        {\cos\delta_{#1}  &  (i\sin\delta_{#1})/\eta_{#1} \cr
 i\eta_{#1}\sin\delta_{#1}  &  \cos\delta_{#1} \cr}}

$$ \pmatrix{B\cr C} =  \layer{1} \layer{2} \layer{3}
                       \pmatrix{1\cr \eta_4\cr} $$
```

Characteristic matrix used to determine the reflectance of the three-layer filter. (Adapted from "Thin-Film Optical Filters" by H.A. Macleod, Adam Hilger ltd., London, 1969.) This example shows how macro definitions in T_EX can greatly reduce the amount of typing required to generate repetitive equations. This same technique is also useful when generating a series of equations showing a derivation.

say to include the equation as an in-line math expression right in the text). They tell T_EX to break the regular text and insert the display equation.

The \alpha, \theta and \beta terms obviously mean the Greek letters alpha, theta and beta; note that you don't have to remember any arcane ctrl-meta-shift-A, T or B key sequences to type in these Greek letters as you do in many WYSIWYG technical word processors. You just type in the name of the Greek letter, with a backslash proceeding it. (And \delta would mean lower-case delta, \Delta would mean capital delta, \del would mean the del symbol, and so on.)

The \sqrt{. . .} command says: put a square-root sign over all the stuff within the following set of curly brackets, namely the cosine-theta-sub-n term; and the cosine is written as \cos so it will get properly typeset as "cos" in roman characters.

The {. . . \over . . . } command means put whatever stuff comes before the \over as a display fraction over whatever stuff comes after it, within the enclosing curly brackets. The underscore symbol "_n" (read as "sub-n") and the caret symbol "^{3/2}2" (read as "sup-three-halves") mean that whatever follows each symbol is a subscript or superscript, respectively. Finally, the \eqno command will put the equation number in the correct place, to the right of the equation if there's adequate room, below and to the right if the equation itself is wider. The entire expression above could all be typed in-line right in with the main text if you like, though I like to break my equations out on separate lines in the manner above for clarity.

As you can see from the above example, you can generally read a typical T_EX equation out loud, just like ordinary math. As another example, if you type "$ \partial f / \partial x $" in the source file you get $\partial f/\partial x$ in the typeset output.

T_EX can handle enormously complicated mathematical expressions, matrices, tensors, compound subscripts and superscripts, and so on, using coding not much more complex than the example above (see illustrations). And it can typeset them with incredible intelligence in producing book-quality typesetting, fully comparable to the best hand-set typography.

Advantages beyond equations

The discussion so far has been intended not as a short course in T_EX, but as an overview to give the feel and flavor of how T_EX operates and what it does. What are some of the other advantages of using T_EX?

1) T_EX is really very easy to use. You can start with just a few simple T_EX commands and produce excellent output. The elementary commands needed to produce standard math output are very easy to learn. More important, because they're generally mnemonic and match up with ordinary mathematical phrasing, they're easy to remember even when you don't use T_EX for a while. The secretaries in our lab use T_EX routinely with no special training. Students use it—even faculty members seem to be able to cope with it.

2) Yet T_EX is also extremely powerful. At its most sophisticated level, T_EX is a complete programming language (the Game of Life has been programmed in T_EX). You could easily keep a database or a bibliography entirely in T_EX. There seems to be no typesetting or graphic need that cannot be met in T_EX, including all varieties of foreign languages. There certainly is no mathematical construct you're likely to encounter, even in your most baroque calculations, that T_EX will not be able to handle straightforwardly.

3) T_EX is also universal. T_EX is available on nearly every personal computer, workstation, or mainframe computer with nearly every operating system in the world. Yet because of the way in which its author Don Knuth has managed the distribution of T_EX, it's the same on every machine. You don't have to cope with different dialects of T_EX on different machines, because there really is only one T_EX (and only one T_EX manual). The same T_EX source file should produce the same typeset output (except for differences in the shapes of individual characters) on any machine anywhere.

4) The basic core of T_EX is also free. You can get distribution tapes of the full Pascal source code for T_EX for any machine, just for the cost of distribution. There are also well-engineered commercial implementations for specific machines, such as Textures for the Macintosh, which include editors, screen previewers, printer drivers, and even spelling checkers, and which are sold as commercial prod-

Fig. 2

```
\font\bigfont=cmr10 scaled \magstep5
\font\medfont=cmr10 scaled \magstep4

\def\boxit#1 {\vbox{\hrule\hbox{\vrule\kern1pt
              \vbox{\kern.5pt#1\kern.5pt}\kern-.5pt
              \vrule}\hrule}}

\boxit {\vbox {\bigfont\hbox{O} \kern-.40ex
        \hbox{\medfont S\kern-.30em A}}}
```

A T_EX version of the OSA logo. Comparison with the true logo shows differences that arise largely from a difference between the font used in this example and the font used by the artist who created the logo. While this example is assembled from typographical characters and lines, more abstract figures, graphs or drawings can also be generated.

OSA and T_EX

OSA's publications department is pursuing the use of the American Physical Society's REVTEX, or a slightly modified version, for authors of OSA journal articles. Both APS and one of OSA's journal typesetters have offered their enthusiastic support of this developmental activity. The behind-the-scenes work on this project is anticipated to run into 1992.

ucts. (Incidentally, the Textures program for the Mac includes an added capability to embed Postscript™ graphics, Macintosh PICT or MacPaint files, or other standard graphics formats as special blocks within typeset TeX documents. One can combine text, mathematics, and illustrations all in a single source file, which can be edited, typeset, previewed and printed, including illustrations, all in one pass.)

5) The macro capability of T_EX is particularly helpful to the amateur $T_EX$er. An author can prepare a document using a small set of personalized macros (for example \chapter, \section, \subsection, \figure and \footnote) to label those elements in the document that are other than plain text or mathematics. Templates for these macros may already be available or the author can, if necessary, write a few elementary macros like the \subhed example given above to display these elements in draft printouts. A publisher or journal can then take the author's source file verbatim, with no retyping or changes in the main body of the document, and define their own more sophisticated macros to typeset the chapter and section headings and the footnotes and references in whatever style the publisher or journal wants to use. The content of the document will not be changed at all—only the style. Many journals are now defining their own sets of elementary macros, such as the American Mathematical Society's AMS-T_EX package, for use by authors preparing articles for electronic submission to that journal.

6) If you have some complicated mathematical expression that appears repeatedly in your document, you can also define that complete expression as a macro with a short meaningful name, and just type that "\name" instead of the whole expression everywhere the expression appears. And if you want to change the expression everywhere it appears, you just change the macro definition.

7) It's also a big advantage that T_EX source files are pure vanilla ASCII text files (no hidden codes or formatting characters). As a consequence, T_EX source files can be prepared on any computer, with any editor, and they'll look the same to T_EX. Source files can be sent from machine to machine, or author to co-author, by email over a network, or by floppy disk or magnetic tape, since ASCII text files are standard on essentially every computer in the world.

8) Global changes in T_EX documents are easy. If you want to change all the alphas to gammas in the final document, you just change \alpha to \gamma in the source file, using your favorite ASCII editor. And after you print and proofread a typeset copy of your document once and make suitable corrections in the source file, you really never need to proof your printed output and especially all the complex equations again—no laborious retyping and reproofing, over and over.

9) Finally, it's very easy to convert mathematical equations or other material from a T_EX source file directly into enlarged reproductions for overhead transparencies or poster papers. You can just copy the T_EX source code for a couple of complicated equations from a manuscript source file, for example, into a new source file; add suitable headers or text and spacing commands around the equations; and put the T_EX command "\magnification = 2500" at the start of the new file. When you typeset this page, the equations and related text will be typeset with the same quality as always, but magnified by a factor of 2.5 times (or 2 times, or 3 times, or whatever you want), ready to be used as an overhead transparency or an enlarged display for a poster paper. If you have a manuscript you've prepared and printed in 12 point type on standard size paper, and the editor of a conference proceedings wants camera-ready copy in 9 point type and fitted into small blue boxes on some special paper, you can just change a few \font, \hsize, and \vsize parameters at the beginning of your source file, and reset your paper to fit those boxes.

Try it! You'll like it!

A.E. Siegman is the Burton J. and Ann M. McMutry Professor of Engineering at Stanford University, Stanford, Calif.

\par\vfill\end