USING TEX ON THE VAX TO TYPESET DOCUMENTS: A PRIMER

DECEMBER 1990

Joseph E. St Sauver
Assistant Professor/Statistical Programmer
Office of University Computing
235 Computing Center
University Of Oregon
Eugene, OR 97403 U.S.A.

(503) 346-4394 extension 25
(503) 346-4397 (FAX only)

Internet: `joe@oregon.uoregon.edu`
BITNET: `joe@oregon`

PREFACE

System Dependencies

This document contains some system dependent features; to use it at another site, you should obtain a machine readable copy of the TEX source and modify it appropriately for your system. Obtain a copy of this document and any of the files mentioned herein via anonymous FTP from `DECOY.UOREGON.EDU (128.223.32.19)` or contact the author at the address shown on the title page.

Disclaimers

Trademarks

Acknowledgments

To Paul Steinman, who first told me about TEX while we were both working for the University of Alaska-Fairbanks Computer Node many years ago: thanks for taking the time to tell me about TEX's magic!

To Donald Knuth and all the others whose talent, hard work, and generosity have made TEX the world standard it is today: thanks for producing such a great piece of software.

Dedication

To Jamie Claire or Earl Joseph, as eventually proves appropriate: we were thinking about you and waiting for you anxiously, even before you counted for tax purposes! This document is dedicated to you in the hope that you, your children, and your children's children will all be able to enjoy the full portfolio of freedoms secured by the Bill of Rights, American bravery, and the grace of God.

Copyright

CONTENTS

I. INTRODUCTION

What is TEX?

TEX[1] is a computerized typesetting program created by Stanford's Donald Knuth[2].

TEX is *not* the same as what-you-see-is-what-you-get (WYSIWYG) desktop publishing systems such as Aldus PageMaker. TEX is both more and less than those desktop publishing packages. For instance, TEX doesn't require a major investment in an expensive microcomputer loaded with lots of RAM and supported by a fast hard disk and a high-resolution display — you can prepare TEX documents on a simple terminal. On the other hand, TEX doesn't show you the final form of your document as you prepare it — you need to first build your document and then process it with TEX and another conversion utility before you actually get to see the fruits of your typesetting efforts. TEX also doesn't do a great job of handling integration of text and graphics.

Similarly, TEX is also *not* the same as a regular word processing package such as Microsoft Word or Word Perfect, although TEX could be used to produce virtually any document a basic word processor could generate.

The best way to think about TEX is to think of it as it was truly meant to be used: TEX is a computerized printing "press" intended for typesetting scientific and technical manuscripts of virtually any length up to and including entire books.

Because Knuth has allowed TEX to be freely distributed, TEX has been ported to machines ranging from PC's to the largest of IBM mainframes. Here at the University of Oregon Computing Center, we run the Northlake Software[3] VAX/VMS port of TEX on OREGON, the academic/research computing VAX.

Why Should I Bother to Learn to Use TEX?

TEX's biggest attraction is that it can beautifully typeset articles or entire books, including complicated mathematical equations. No longer do you need to prepare manuscript copy, send it to be typed (or typeset) by someone who may not have the slightest understanding of what your equations mean, and then tediously attempt to spot and correct errors and resubmit corrected proofs until you finally get copy which is more-or-less correct. With TEX you can produce camera-ready copy (including *all* of your equations) laid out exactly the

[1] Say "tecchhh" (as in the greek letter chi), rather than "tecks."

[2] Of *Fundamental Algorithms, Seminumerical Algorithms,* and *Sorting and Searching* fame.

[3] Northlake Software, 812 SW Washington, Suite 1100, Portland, OR 97205-3215. Telephone: 503-228-3383. Fax: 503-228-5662. (Formerly Kellerman and Smith.)

way you want it, without many of the costs, delays or headaches associated with traditional copy preparation processes.

If you'd like tangible proof that TeX really *works* for typesetting complex scientific and technical manuscripts, consider the following brief list of some outstanding technical books, all of which were all set in TeX (or a variant thereof):

> *Lisrel 7: A Guide to the Programs and Applications,* 2nd Edition, Karl G. Jöreskog and Dag Sörbom, SPSS Inc, Chicago, Illinois: 1989 (typeset using TeX).
>
> *Mathematica: A System for Doing Mathematics by Computer,* Stephen Wolfram, Addision-Wesley Advanced Book Program, Redwood City, California: 1988 (typeset using TeX and LaTeX).
>
> *Fortran Tools for VAX/VMS and MS-DOS,* R.K. Jones, John Wiley and Sons, New York: 1988 (typeset using LaTeX).
>
> *Numerical Recipes: The Art of Scientific Computing,* William H. Press, Cambridge University Press, Cambridge: 1986 (typeset using TeX).
>
> *The TeXbook,* Donald E. Knuth, Addison Wesley, Reading, Massachusetts, 1986 (typeset in TeX, of course!).

There are many other examples of excellent TeX-created books, but these are just a few examples which happened to be handy on my own bookshelf.

Here at the University of Oregon, TeX will probably be used most extensively by graduate students preparing theses and dissertations, and by scientists, mathematicians, and other technical people preparing manuscripts containing substantial mathematical notation. TeX is also great for typesetting professional looking survey instruments and resumes.

What Can I Expect of the Rest of This Write-Up?

The following sections of this write-up explain how you can create your own TeX documents, and how you can convert your TeX document into a file suitable for previewing or printing on a laser printer.

This write-up *won't* take you deep into the internal anatomy of TeX, nor will it teach you highly efficient but rather obscure tricks for working with TeX. The objective of this write-up is to teach you the "nuts and bolts" of TeX as quickly as possible so you can produce the actual documents *you* need to make with minimal delay.

As a result, in some cases you'll be shown a way to do something without a whole lot of explanation or discussion; that format of presentation has been adopted to avoid obscuring essential concepts with extraneous detail. There may be times when I've gone too far toward the minimalist position, and when that's the case, I urge you to let me know: suggestions for the improvement of future versions of this write-up are always appreciated.

II. ENTERING TEXT (OTHER THAN TABLES AND EQUATIONS) IN TEX

Every TEX document consists of two parts: actual text, and TEX formatting commands.

Most actual text can be entered "as-is," although there are some characters which need to be expressed specially (either because a particular character isn't available on your keyboard, or because TEX has designated those characters for special purposes).

The following section will explain the conventions you need to understand in order to be able to enter your document's text in TEX's most popular font, Computer Modern Roman. Most of the other fonts you can use with TEX should work in about the same way, but you may notice some minor differences. Thus, until you become more familiar with TEX, you should probably stick to using the Computer Modern Roman font described herein.

Text Which Is Entered Normally in TEX

Normal text consists of regular letters, numbers, punctuation, and whitespace, all of which are entered "as-is:"

• Lowercase alphabetic letters:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

• Uppercase alphabetic letters:

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

• The arabic numerals:

```
0 1 2 3 4 5 6 7 8 9
```

• Most punctuation:

```
.  , ?  !  :  ; ( ) [ ] ‘ ’ - * / + = @
```

• "White-space" (spaces, tabs, and carriage-returns).

TEX will freely condense or expand white space to suit its typographical preferences, although there are exceptions to this rule. For instance, you can enter a "control space" (by saying `\␣`), which TEX *will* honor unconditionally as being a "real" space which is not to be adjusted. Insert a tiny space by saying `\thinspace`. Larger spaces can be forced into text with `\enspace` or `\quad` or `\qquad`. Generally, however, simply hit the space bar to put space between two words.

One other note: if you're entering an exceptionally long line and want to break that text into two separate lines but **don't** want TEX to treat the carriage return at the end of the first line as being equivalent to a space, put a `%` sign at the end of the first line. TEX will then concatenate (or "abut") the two lines without any visible break.

Special Characters in TEX

Not all characters can be so easily entered in TEX. Some text must receive special treatment if you want it to come out looking right when typeset in TEX. For example:

• Most typists are accustomed to using a conventional “androgynous” double quote mark (`"`) at both the start and end of a quotation. TEX is a different beast. Like most typesetting programs, TEX uses different marks to open and close quotations: on the left side, use two successive backslant accent marks (` `` `), thereby producing the correct opening quote mark (“); on the right side of the quotation, use two successive single quote marks (`''`), resulting in a proper closing quote mark (”).[4]

• You should also know how to enter the three different types of dashes available in TeX: the hyphen (used to form compound words), the en-dash (used to indicate ranges of items), and the em-dash (used to signal pauses in discourse)

- – —

which are obtained by entering one, two or three dashes in succession:

```
-        --        ---
```

• You should also know how to make an ellipsis properly in TEX. A properly created TEX ellipsis should look like:

The tension was ... palpable.

Obtain such an ellipsis by saying:

```
The tension was $\ldots$ palpable.
```

• Other special characters are the dollar sign, pound sign, percent sign, ampersand sign, underscore, vertical bar, less than sign, greater than sign, and pound sterling sign:

$ # % & _ | < > £

which you can create by saying:

```
\$    \#    \%    \&    \_    $\mid$    $<$    $>$    {\it \$}
```

[4] If you absolutely must have an androgynous double quote mark, enter `{\twltt "}`

• Additional marks which need to be specially constructed include the caret (or “hat”), tilde (or “wave”), left curly brace, right curly brace, and backslash, all of which have special uses in TeX:

ˆ   ˜   {   }   \

Get those characters by saying:

```
\^{ }    \~{ }    $\{$    $\}$    $\backslash$
```

• You can also get many other special symbols, including a dagger, double dagger, copyright symbol, paragraph symbol, section symbol, and “spanish” punctuation symbols:

†   ‡   ©   ¶   §   ¡   ¿

Make them in your document by entering:

```
\dag    \ddag    \copyright    \P    \S    !‘    ?‘
```

• TeX’s support for foreign languages goes far beyond its ability to make upside-down exclamation points and question marks. Unlike many American document processing systems which do a pathetic job of handling the special characters common in foreign languages, TeX handles special foreign characters easily:

å   Å   æ   Æ   ł   Ł   ø   Ø   œ   Œ   ß

Obtain these special foreign characters (in Computer Modern Roman fonts) by saying:

```
\aa   \AA   \ae   \AE   \l   \L   \o   \O   \oe   \OE   \ss
```

• TeX can handle special foreign accents, too. Superior accents include:

ò   ó   ô   ö   õ   ō   ȯ   ŏ   ǒ   ő

Obtain them by writing:

```
\`{o}  \'{o}  \^{o}  \"{o}  \~{o}  \={o}  \.{o}  \u{o}  \v{o}  \H{o}
```

• Inferior and special multi-letter tied accents include:

o̧   ọ   o̱   o͡o

obtained by saying:

```
\c{o}  \d{o}  \b{o}  \t{oo}
```

You now know how to enter all the symbols which appear on a standard computer terminal’s keyboard, as well as selected other text symbols. There are many other symbols available in TeX which you’ll normally need only when setting mathematical equations. We’ll talk about them later in this write-up.

Structuring The Text You Enter; Making Paragraphs

Let's now talk about how you should structure the text you're entering. Generally speaking, you have a lot of leeway in this area.

Most often, people will elect to enter their text "normally" (in paragraph-shaped pieces), with a blank line between paragraphs. Some people, however, prefer to use a "sentence-by-sentence" format, starting each new sentence on a new line (again leaving a blank line between paragraphs).

If you prefer, you can begin paragraphs with the command `\par` instead of signalling the beginning of each new paragraph with a blank line.

You don't need to manually tab or space to indent the first line of each paragraph since TEX will automatically indent the beginning of each paragraph for you.[5]

Comments

One habit you may want to acquire is insertion of comments throughout your document to set off various sections of text or to remind yourself what a particularly obscure TEX command does. To insert a comment in a TEX document, simply type a percent sign and then enter your comment. The percent sign, and anything which comes after it on the same line, will be completely ignored by TEX when it processes your document. For example:

```
% Remember to add disclaimers here before final printing!
```

Comments in TEX can be entered on their own line (thus making it easy for you to spot those comments when you look through your raw TEX file), or you can enter TEX comments at the end of a line containing other TEX commands, whichever you prefer.

Note: if a piece of your document is mysteriously missing when you print it out, look for a "real" percent sign in your text which may have gotten interpreted (incorrectly) as a comment.

[5] If you prefer a block-style format with un-indented paragraphs and an extra space between paragraphs, (i.e., like the style I elected to use in this document), you can replace the blank line between paragraphs with `\bigskip\par\noindent` instead.

You could also set the `\parindent` paragraph indentation parameter to zero, but that can have additional unexpected side effects since the `\parindent` dimension is used in a number of different places for diverse purposes. Therefore, we recommend you manually suppress indentation on a paragraph-by-paragraph basis until you become somewhat proficient in the use of TEX.

Font Size

TEX gives you the ability to work with a variety of sizes and styles of type within a single document. To automatically define a full set of Computer Modern Roman fonts, say:

```
\input fontsize.tex
```

near the top of your document.[6]

You can then declare a default font size family for your document by saying:

Font size declaration	Sample
`\eightpoint`	This is eight point type.
`\ninepoint`	This is nine point type.
`\tenpoint`	This is ten point type.
`\elevenpoint`	This is eleven point type.
`\twelvepoint`	This is twelve point type.
`\fourteenpoint`	This is fourteen point type.
`\seventeenpoint`	This is seventeen point type.

as appropriate. Usually, you'll want to use `\twelvepoint` for most of the copy you'll prepare.

If you're not a typesetter or journalism major, you may not be familiar with using "points" as a unit of measure. There are 72.27 points to an inch, but you are probably better off conceptualizing twelve points as being the "bigness" of the text you're reading right now.

Ten and eleven point fonts are also commonly used, although they may look "small" to a reader who's accustomed to pica or elite typewritten text.

The Computer Modern ("CM") font family consists of:

Generally	12pt	11pt	10pt	Name	Sample
`\rm`	`\twlrm`	`\elvrm`	`\tenrm`	Roman	AaBbCc
`\bf`	`\twlbf`	`\elvbf`	`\tenbf`	Roman Bold Extended	**AaBbCc**
`\it`	`\twlit`	`\elvit`	`\tenit`	Roman Text Italic	*AaBbCc*
`\tt`	`\twltt`	`\elvtt`	`\tentt`	Roman Typewriter Text	`AaBbCc`
`\sl`	`\twlsl`	`\elvsl`	`\tensl`	Roman Slanted Roman	*AaBbCc*
`\mit`	`\twli`	`\elvi`	`\teni`	Roman Math Italic	$AaBbCc$
	`\twlex`	`\elvex`	`\tenex`	Roman Math Extension	`n/a`
`\cal`	`\twlsy`	`\elvsy`	`\tensy`	Roman Math Symbols	$\mathcal{ABC}$

[6] If you'd like to see the contents of that font definition command file, say:

```
$ TYPE/PAGE TEX_INPUTS:FONTSIZE.TEX
```

Note to TEX users on other machines: a copy of this font definition file is available for retrieval via anonymous FTP from DECOY.UOREGON.EDU (128.223.32.19) in pub/tex/samples

The majority of this document has been set in Computer Modern Roman 12 point, with the examples set in CMR Typewriter Text 12 point and emphasized text set in CMR Bold Extended 12 point and CMR Text Italic 12 point. Finally, the titles on the cover page have been set in CMR Bold Extended 17 point (`\stnbf`) .

While the fonts we've just described are probably the only fonts you'll ever need for most documents, there are many other fonts which are also available for your use on the VAX. If you'd like to see all the fonts currently installed on the VAX, use the commands:

```
$ DIR TEX_FONTS:*.*
```

For more information about these fonts, see Appendix F of *The TEXbook*, or see Volume 3 of Knuth's *Computers and Typesetting.*

One more important point about declaring a default font size: when you say `\twelvepoint` (or `\tenpoint` or whatever), by doing so you are choosing a default font size, but you are **also** setting *the default spacing between lines!* If your document's line spacing looks too "loose" or too "tight", check to make sure that the default font size you specified corresponds to the font size of the majority of the copy you're setting!

Font Style

Now that you've got your default font size defined, you're ready to tell TEX what parts of your document should be set in each font. To declare a "default" font style to be used until you specify otherwise, simply enter that font's name near the top of your document, before you begin entering document text. Thus, to use Computer Modern Roman fonts (at the current default size) as your default font, you'd enter:

```
\rm
```

All the text set from that point forward will be typeset in Computer Modern Roman at the default font size you've declared. If you want to set just a few words of text in another font style (such as an italic or bold face font), use curly braces to create a "group" limiting the effect of the font change to the desired section of text:

```
When one is writing, there are times when one needs
{\bf interesting} examples, or at least some which
are {\it moderately relevant}.
```

This should yield output which looks like:

> When one is writing, there are times when one needs **interesting** examples,
> or at least some which are *moderately relevant.*

If you accidentally omit the closing curly brace for a group, you'll probably end up with pages and pages of text set in the wrong font, together with an error message from TEX stating that you have a "`\end at level one`", or something equally non-explanatory. That's your cue to go back and insert the closing curly brace you missed.

Underlining

Underlining is seldom seen in typeset material, because it is a pain to typeset (using mechanical type), because underlining is typically replaced with italics in typeset matter as a matter of convention, and because underlining can interfere with descenders on some letters (see, for example, the letter "g" in the example below).

Nonetheless, we realize that there are some circumstances (such as during the preparation of a dissertation) when you have no choice: when you're writing a dissertation you simply have to underline some things. TEX can handle that requirement:

```
I don't need no stinking \underbar{badge}, Mee--ster!
```

This results in output which looks like:

I don't need no stinking <u>badge</u>, Mee-ster!

Line Spacing (Double-Spacing, Skipping A Single Line, etc.)

To double space text in TEX, issue the command:

```
\baselineskip=2\normalbaselineskip
```

Text set by TEX in double-spaced form looks like:

The use of a double spaced format can do much to make the inconsequential appear weighty, and the weighty, ponderous. It also is convenient when you need to correct those errors introduced by your colleagues.

Return to normal spacing by saying:

```
\baselineskip=\normalbaselineskip
```

To insert just one "extra" blank line at a particular spot in your document, enter:

```
\bigskip
```

To get just an extra *half*-line of space say: `\medskip`

To get just an extra *quarter*-line of extra space say: `\smallskip`

Block Quotations

If you are quoting lengthy passages from a work, you'll normally want to indent left and right and single space the quotation in block format. To do that in TeX you'd say:

```
{\narrower {quotation text goes here} \bigskip\par}
```

Some notes on this process of handling block quotations:

(1) You **must** end the paragraph before you exit the group containing the `\narrower` command; if you fail to do so, the paragraph will revert to its normal width as if you hadn't entered a `\narrower` command at all. (That is why the `\bigskip\par` command is contained within the curly braces.)

(2) The block quotation will be indented by the current value of the `\parindent` parameter. If your default `\parindent` is too large or too small for your requirements, you may need to manually reset it as part of the `\narrower` command. For example, to indent 1" try:

```
{\parindent=1.0in \narrower {quotation text goes here} \bigskip\par}
```

(3) Finally, if you are double spacing the rest of your document, you'll need to explicitly force the block quotation to be single spaced. Try something like:

```
{\parindent=1.0in \narrower \baselineskip=\normalbaselineskip
{quotation text goes here} \bigskip\par}
```

For example, here's a small sample block quotation:

```
{\parindent=0.5in \narrower \baselineskip=\normalbaselineskip
{The terrible thing about our time is precisely the ease with
which theories can be put into practice.  The more perfect, the
more idealistic the theories, the more dreadful is their
realization.  We are at last beginning to rediscover what
perhaps men knew better in very ancient times, in primitive
times before utopias were thought of:  that liberty is bound up
with imperfection, and that limitations, imperfections, errors
are not only unavoidable but salutary.  Merton}\bigskip\par}
```

When processed by TeX, that block quotation would look like:

> The terrible thing about our time is precisely the ease with which theories can be put into practice. The more perfect, the more idealistic the theories, the more dreadful is their realization. We are at last beginning to rediscover what perhaps men knew better in very ancient times, in primitive times before utopias were thought of: that liberty is bound up with imperfection, and that limitations, imperfections, errors are not only unavoidable but salutary. Merton

Under some circumstances, you may only want to indent one side of the block or the other, but not both. When that's the case, use

```
{\leftskip=1.0in
{Here is some text to be indented from the left margin.}
\bigskip\par}
```

or

```
{\rightskip=1.0in
{Here is some text to be indented from the right margin.}
\bigskip\par}
```

You can also indent a single line simply by inserting some horizontal glue:

```
\par\noindent\hglue 1.0in
Here's some text to be indented an inch.
\par\noindent
```

Centering Text (For Headings, etc.)

There are times when you'd like to center some text on the page for headings. To do this in TEX, use the command:

```
\centerline{text to be centered goes here}
```

To set a single line flush left, or a single line flush right, use the commands:

```
\leftline{text to be set flush left goes here}

\rightline{text to be set flush right goes here}
```

Note that you can achieve the same effects using more primitive commands by saying:

```
\par \hfill {text to be centered goes here} \hfill \par

\par {text to be set flush left goes here} \hfill \par

\par \hfill {text to be set flush right goes here} \par
```

This more primitive approach, using `\hfill` commands, will particularly come in handy later when you begin to work with headers and footers, and under other circumstances when the more elegant `\centerline` or `\leftline` or `\rightline` approaches are infeasible.

Footnotes

To create a footnote in TEX, use a command such as the following:

```
Another fascinating condiment is the pink
peppercorn.\footnote{$^{13}$}{See, for example, William Poundstone,
\it{Big Secrets,} Quill, New York, 1983, pages 26-27.\par}
```

Some notes about footnotes:

(1) Footnotes are not automatically numbered as they're created. Rather, you must manually assign a number (or symbol) to each footnote you make. If you rearrange your footnotes, or insert new footnotes, you'll also need to manually adjust the numbers of existing footnotes.[7]

(2) By default, TEX builds its footnotes at the bottom of the page in a rather ugly format. If you'd like to fix the footnote production mechanism so your footnotes come out looking like the ones used in this write-up, instead, put the following commands at the top of your document:

```
\parindent=0pt \global\skip\footins=24pt plus 24pt minus 24pt
\def\footnoterule{\vfil \vglue 12pt \hrule width 2in \vglue 12pt}
```

Note that this footnote fix **does** tinker with the `\parindent` parameter which is used for a whole host of other purposes! Therefore, determine that you truly can't abide the default footnote format before you decide to employ the fix shown above.

(3) If you must have endnotes (rather than footnotes), simply insert your footnote reference number in the right place, but don't actually use the `\footnote` command. Enter the body of your endnote at the **end** of your document, including the corresponding reference number. For example:

```
Another fascinating condiment is the pink peppercorn.$^{13}$

*   *   *

\vfill\eject
\centerline{\bf ENDNOTES}

*   *   *

\bigskip\par\noindent
13.  See, for example, William Poundstone, \it{Big Secrets,}
Quill, New York, 1983, pages 26-27.
\bigskip\par\noindent
```

[7] See the solution to problem 15.12 in Appendix A of Knuth's *The TEXbook* for one way of overcoming the limitations associated with manually numbered footnotes.

Headers and Page Numbers

TEX will automatically number each page (including the first page) at the bottom center of the page. However, you can suppress all page numbers by including the following near the top of your document:

```
\nopagenumbers
```

Many people, however, **do** like to have the pages of their document numbered. Typically, the preferred location for those page numbers is the upper right hand corner of the page.

One way to get page numbers in the upper right hand corner of the page in TEX is to create an appropriate `\headline`. For example, to simply put the page number in the top right corner of each page, enter (near the top of your document):

```
\headline={\rm\hfill\folio\voffset=2\normalbaselineskip}
```

In the preceding command, `\rm` insures that our header will be printed in a normal Roman font, `\folio` is TEX's symbolic name for the current page number, and the `\hfill` will force the page number all the way to the right. By setting `\voffset` to `2\normalbaselineskip`, we insure that a blank line is left between the page number and the page's text.

If you want a running section heading as well as a title, enter something like:

```
\headline={\rm Summary \hfill\folio\voffset=2\normalbaselineskip}
```

Note that whatever heading happens to be defined when TEX comes to the **end** of a page is the heading that page will receive. The critical thing is what's defined when the page is **finished**, not what's defined when the page is **begun**!

A similar policy holds for page numbering. The value of the page number counter (`\count0`) at the **end** of the page determines the page number of that page, **not** the value the page number counter may have started at. To change the page counter, say:

`\count0=`*newpagenumber*

somewhere on the page you want renumbered.

If you need lowercase roman numeral page numbers (as you might for prefatory matter), set `\count0` to a negative value. One way to negate `\count0` is by using the command:

```
\ifnum\count0>0\multiply\count0 by -1\fi
```

Alternatively, assuming you know the exact page number at which you want to begin lowercase roman numeral page numbers, you could simply say:

`\count0=-`*newpagenumber*

Page Size; Margins

To set the size of your TEX document, use commands like:

```
\hsize=7in
\vsize=9in
```

near the top of your document. That will set the size of the text that TEX creates to be seven inches wide and nine inches high, for example.

By default, TEX starts out indenting 1" from the left hand side of the page, and 1" down from the top of the page. If you wanted to increase the left margin by 0.5", and you wanted to decrease the top margin by 0.25", for example, you'd enter:

```
\hoffset=0.5in
\voffset=-0.25in
```

You set the right hand (and bottom) margins implicitly when you set the size of your TEX document and the width of the left and top margins.

Leaving Space for Insertions; Forcing Page Breaks

There will be times when you want to take explicit control over the way TEX spaces paragraphs on the printed page. That is, you might want to pop to the top of a new page before beginning a new section, or you might want to leave a couple of inches of blank space so that you have room to tip in a photograph or other illustration.

To finish the current page and go to top of a new page, enter:

```
\vfill\eject
```

If all you want to do is reserve a couple of inches of blank space for an illustration, try:

```
\vglue 3.0in\par\noindent
```

Be sensitive to the possibility that the three (or however many inches) of space you request in this way may be provided to you partially on one page, and partially on the following page if there isn't enough room on the current page. If you need to insure you'll get a single undivided block of space of the size required, say:

```
\vbox{\vglue 3.0in\par\noindent}
```

Ending Your TEX Document

Signal the end of your TEX document by entering:

```
\vfill\eject\end
```

as the very last line of your TEX command file.

III. TYPESETTING TABULAR MATERIAL

Using Tabs

There may be times when you want to use "tabs" in TEX, just as you might on a typewriter. The first step in using tabs in TEX is to set your "tabstops." While there are many ways to do this, the two most generally useful approaches to doing this are the evenly-spaced tab stop setting procedure, and the specific-spacing tab stop setting procedure.

For example, to create eight evenly spaced tab stops, you'd enter:

```
\settabs 8\columns
```

If you need to create unevenly spaced tab stops, try something like the following:

```
\settabs\+\hglue 1.75in&\hglue 0.4in&\hglue 1.4in&\hglue 1.0in\cr
```

There's a third way of setting tab stops which you may like even more than either of the above two approachs. Essentially, TEX can automatically determine the right width for each column of your copy based on the effective width of the widest element to be put in each column. For example, let's assume you wanted to set up a little summary table describing some common stocks:

```
\settabs\+Ticker Symbol\qquad&Turgekowski Steamship Service\qquad&
999,999\qquad&\$9,999.99\qquad&\$9,999.99\qquad&\$9,999.99\cr
```

Note that we've defined the width of the first column by writing its heading (and a trailing `\qquad` worth of space) because that is the widest element which will be present in that column. For the second column, the longest element will be one particularly long name of an actual firm, together with a little additional space. The remaining columns, which will be used to report stock trading volume and high, low, and closing stock prices, were all created using dummy "worst-case" numeric values (plus a little extra space).

Regardless of the tab-setting approach you elect to employ, once you've created your tab stops, you can easily set type at those tab spots by saying something like:

```
\+Blah&Foo&Phft&&Thdd\cr
\+ABC&DEFGH&IJ&KLMNOP\cr
```

I.E., each tabbed line begins with a `\+` and ends with a `\cr`. Between those delimiters, you indicate you want to go to the next tab stop by inserting an ampersand (`&`). Insert two successive ampersands to skip a column. Note that you do not need to "use" all tabstops on each tabbed line before ending that line — you can end a tabbed line with `\cr` after using only one (or even **none**) of the tabstops which you've defined.

If you ever need to un-define your tabstops, enter:

```
\cleartabs
```

Typesetting Formal Ruled Tables

TEX can also set more complicated "formal" ruled tables. However, before you decide to try doing this, a warning: setting tables in TEX can be rather tricky. Consider this example:

SOME SERVICE PISTOLS			
Manufacturer	**Model**	**Caliber**	**Capacity**
Glock	**17**	9mm Para	19
Sig-Sauer	**P226**	9mm Para	15
Smith&Wesson	**5926**	9mm Para	15
Glock	**21**	.45 ACP	13
Sig-Sauer	**P220**	.45 ACP	7
Smith&Wesson	**4506**	.45 ACP	8

The actual TEX commands used to create that table look like:

```
{\vbox{\offinterlineskip\halign{
\hglue 0.5in\strut\vrule#&
\bf\hfil\quad#\quad\hfil&\vrule#&
\bf\hfil\quad#\quad\hfil&\vrule#&
\hfil\quad#\quad\hfil&\vrule#&
\hfil\quad#\quad\hfil&\vrule#\cr
% template ends; now set the headings
\multispan{9}\hglue 0.5in\vrule\hrulefill\vrule\cr
&&\omit&&\omit&&\omit&&\cr
\multispan{9}\hglue 0.5in\vrule\hfill\bf SOME SERVICE PISTOLS
\hfill\vrule\cr
&&\omit&&\omit&&\omit&&\cr
\multispan{9}\hglue 0.5in\vrule\hrulefill\vrule\cr
&&&&&&&&\cr
&\bf Manufacturer&&\bf Model&&
\bf Caliber&&\bf Capacity&\cr
&&&&&&&&\cr
% headings end; now set the body of the table
\multispan{9}\hglue 0.5in\vrule\hrulefill\vrule\cr
&&&&&&&&\cr
&Glock&&17&&9mm Para&&19&\cr
&Sig-Sauer&&P226&&9mm Para&&15&\cr
&Smith\&Wesson&&5926&&9mm Para&&15&\cr
&&&&&&&&\cr
&Glock&&21&&.45 ACP&&13&\cr
&Sig-Sauer&&P220&&.45 ACP&&7&\cr
&Smith\&Wesson&&4506&&.45 ACP&&8&\cr
&&&&&&&&\cr
\multispan{9}\hglue 0.5in\vrule\hrulefill\vrule\cr
}}}
```

The `\halign` command used to set that table differs from the tabbed environment described in the previous section in three important ways. One difference between the two table construction approaches is that TeX itself determines the required width of each column in the `\halign` approach without the need for you to manually intervene in an effort to guess the widest element present in each column. A second difference, which happens to be a disadvantage of the `\halign` approach, is that `\halign` doesn't yield constant-width columns on a multiple table basis unless you employ some "tricks"; the `\+` tabbing approach is much better than `\halign` when you need to generate a whole series of tables with each table having the same width columns. The third difference between the two methodologies is that tabbed tables are processed line-by-line, and thus don't make TeX work very hard; `\halign` tables, on the other hand, have to be examined in their entirety by TeX before it can decide how to process them. Naturally, this can be quite computationally expensive for large tables! Experience will gradually help you learn to pick the right table-building tool.

Beginning to Decode the Table-Building Commands

But what do all the `\halign` table building commands mean?

Begin by conceptualizing the table as nine columns. Four of the columns are "real" columns containing manufacturer, model, caliber, and capacity information. The other five columns are "columns" in name only: they are actually used solely to hold the the vertical rules (lines) separating the real columns of data.

The first line of TeX commands used to produce our table looks like:

```
{\vbox{\offinterlineskip\halign{
```

In that line of commands:

> The `\vbox` makes sure the entire table is treated as an indivisible organic entity, thereby insuring that the table isn't accidentally split across two pages.
>
> The `\offerinterlineskip` prohibits TeX from automatically vertically skipping `\baselineskip` every time it begins a new line. It is important to keep TeX from doing this when you are setting tables which include "lines" consisting solely of thin horizontal rules, as our table does.
>
> Finally, the `\halign` actually starts our horizontally aligned TeX table.

Table Template

The next five lines of commands, which look like:

```
\hglue 0.5in\strut\vrule#&
\bf\hfil\quad#\quad\hfil&\vrule#&
\bf\hfil\quad#\quad\hfil&\vrule#&
\hfil\quad#\quad\hfil&\vrule#&
\hfil\quad#\quad\hfil&\vrule#\cr
```

define the **default format** for the nine columns comprising each line of the table.

The declared default format for the first column extends from the start of the commands up to the 1st ampersand; the format for the second column goes from the 1st ampersand to the 2nd ampersand; the format for the third column goes from the 2nd to the 3rd; etc.

Specifically, in our case, the leftmost column (and thus the entire table) will be (by default):

> Indented `0.5`" from the left margin (`\hglue 0.5in`), and be of
>
> "Normal" height (`\strut`).

The leftmost column is also defined to contain a thin vertical line (`\vrule`).

The pound sign (`#`) in that line represents the spot where the "contents" of a column should be added when we begin processing the body of the table.[8]

The second column of the table is our first "real" column. The text in that column will be set in twelve point bold face type (`\bf`), and centered (notice the two `\hfil`s), with a minimum of one `\quad` worth of space on each side of the text we'll eventually supply.[9]

The third column of the template consists solely of a `\vrule` and a "dummy" pound sign provided exclusively to placate TEX.

The remaining six column definitions are similar to those already discussed. Note, though, that the last (ninth) column ends with a `\cr`, not another ampersand.

We're now done defining the template TEX needed for the body of our table.

[8] In the case of the columns reserved for vertical rules, we'll never be adding any additional columnular material, but TEX nonetheless demands that we include a pound sign for each column as a matter of form.

[9] Omission of the `\quad`s would result in the lines of the table directly abutting the edges of the widest entry in each column — an aesthetically undesirable occurrence.

Table Headings

The first non-template/"output-producing" table line looks like:

```
\multispan{9}\hglue 0.5in\vrule\hrulefill\vrule\cr
```

It is designed to create the top horizontal line of the table.

The `\multispan{9}` command tells TEX that we want to enter some stuff which will cross (or span) nine columns of the table. The rest of that command says "skip a half inch horizontally, then put in a thin vertical rule, then fill the rest of the line with a thin horizontal rule, putting in another thin vertical rule at the right hand margin. End this line."

The `\multispan` command automatically cancels the commands present in the default table template, including the `\strut` (and all the other good stuff) so that the first "line" of our table will actually be no taller than the height of our thin rules.

The next line of commands:

```
&&\omit&&\omit&&\omit&&\cr
```

is designed to put a little space between the top horizontal line and the title of the table. Although no text is specified for the first two columns of the table, TEX does still obey the format specified for those two columns. Hence, this line of the table is indented a half inch, has a vertical rule, and is the height of a normal line (remember the `\strut`?).

So what do we do to tell TEX **not** to insert any default "stuff" from the template into a particular column? Enter `\omit` We specify `\omit` for our third column of this line so that we don't get an unwanted vertical rule dropping right in the middle of our column-spanning table title.

The rest of this line of table-building commands operates in a similar fashion.

The next two lines, which looks like:

```
\multispan{9}\hglue 0.5in\vrule\hfill\bf SOME SERVICE PISTOLS
\hfill\vrule\cr
```

should be easy for you to decode now that we've explained the earlier table building commands – there really isn't anything new here which you haven't seen before, except that you should notice that the commands to create a single line of table output can be freely continued across multiple lines; TEX knows a line of table matter is finished only if and when it sees a `\cr` command.

In fact, you now know enough to decipher **all** the rest of the table construction commands used to make our sample table.

Table Body

One point which is easy to overlook: while you may bleed, sweat, and cry while setting up a table's template and titles, the actual body of most tables is a *snap* if you set the template up correctly. Thus, because so much of the table is handled automatically for you by the template, the bigger a table is, the easier it is to set it *once you get past* the template and title material. For example, notice how easy it is to set the main part of our sample table:

```
&&&&&&&&&\cr
&Glock&&17&&9mm Para&&19&\cr
&Sig-Sauer&&P226&&9mm Para&&15&\cr
&Smith\&Wesson&&5926&&9mm Para&&15&\cr
&&&&&&&&&\cr
&Glock&&21&&.45 ACP&&13&\cr
&Sig-Sauer&&P220&&.45 ACP&&7&\cr
&Smith\&Wesson&&4506&&.45 ACP&&8&\cr
&&&&&&&&&\cr
```

Pretty straightforward, isn't it? Notice that you can even include actual ampersands as text in your table by entering `\&` for each "real" `&`.

As you become proficient at TeX, you'll find that typesetting tables isn't as difficult as you might have thought it was!

IV. TYPESETTING EQUATIONS

Typesetting Equations is Different From Typesetting Text

Equations generally require special handling to come out looking “right” when printed. For example, mathematical variables are generally set in italics (rather than roman fonts), operators (i.e., symbols such as plus signs, minus signs, etc.) require special spacing, and a whole host of other conventions need to be observed if the user is to end up with normal-looking mathematical copy.

Fortunately, TeX is pre-programmed to automatically “do the right thing” when setting mathematical copy: you don’t need to struggle to get sweet-looking equations in TeX.

The remainder of this section will explain to you how to set the majority of the mathematical copy an average TeX user will need to set.[10]

We’ll now consider the general context of how TeX senses we’re setting mathematical copy, and then we’ll “zoom in” to look at how we can generate various mathematical symbols.

Embedded vs. Displayed Equations

Equations can be constructed two different ways in TeX. Short, simple equations are typically embedded right in the flow of narrative text, while larger, more complex equations are usually indented and set as “displayed” equations, separated from any adjacent text.

To signal TeX that we’re beginning to set an embedded equation, begin the equation with a single dollar sign. To signal the end of the embedded equation, use another dollar sign. For example:

```
Pythagoras' triangular equality $c^2=a^2+b^2$ allows us to $\ldots$
```

When processed by TeX, your output would then look like:

Pythagoras’ triangular equality $c^2 = a^2 + b^2$ allows us to $\ldots$

When you want to set a displayed equation, begin the equation with **two** successive dollar signs (instead of only one), and be sure to also end the equation with two dollar signs:

```
Working the equation through, we arrive at the Sturm-Liouville
expression:
\medskip\par\noindent $$[r(x)y']+[q(x)+\lambda p(x)]y=0$$
Bessel, Legendre, and other equations can all be written in this form.
```

[10] If you are setting particularly esoteric or complicated equations, you’ll probably also want to closely study Chapters 16–19 of *The TeXbook* for more detailed information about setting mathematical copy.

Processed, that looks like:

> Working that equation through, we arrive at the Sturm-Liouville expression:
>
> $$[r(x)y'] + [q(x) + \lambda p(x)]y = 0$$
>
> Bessel, Legendre, and other equations can all be written in this form.

Numbering Equations

If you are preparing a complicated document with many equations, you will almost certainly want to number your equations so that you can easily and unambiguously refer to a particular equation in your narrative discussion.

The way to do that in TEX is to build your displayed equation the way you normally would, but then add:

`\eqno(`*equationnumber*`)`

just before you enter the closing double dollar signs. For example:

```
Working the equation through, we arrive at the
Sturm-Liouville expression:
\medskip\par\noindent
$$[r(x)y']+[q(x)+\lambda p(x)]y=0\eqno(8)$$
Bessel, Legendre, and other equations can all be written in this form.
```

Processed, the numbered version of our sample display equation looks like:

> Working that equation through, we arrive at the Sturm-Liouville expression:
>
> $$[r(x)y'] + [q(x) + \lambda p(x)]y = 0 \tag{8}$$
>
> Bessel, Legendre, and other equations can all be written in this form.

If you'd rather have your equation numbers on the left, instead of on the right, substitute `\leqno` for `\eqno`. Even if you want left hand equation numbers, you should still insert the `\leqno` immediately before the two dollar signs **ending** your displayed equation. Do **not** move the `\leqno` so that it is just after the **opening** double dollar signs, or you'll generate an error.

There's another way to number equations, and that is as part of aligning multiple equations in a single display. Just coincidentally, that's the topic of our next section.

Aligning and Numbering Multiple Equations

When you are preparing a particular mathematical display which contains more than one equation, you will usually want to align those equations by their equal signs (or some other reasonable visual "balance" point).

When you are doing that sort of multiple equation alignment, there are two ways you can number your equations. You can number the block of equations as a unit, or you can number all (or some) of the equations in the block individually.

• If you want to number a block of equations **as a unit**, you'd say something like:

```
Consider the functions:
$$\eqalign{\phi_1={\rm arg}(w+1)&=\arctan (v / [u+1])\cr
           \phi_2={\rm arg}(w-1)&=\arctan (v / [u-1])\cr}
  \eqno(21)$$
\medskip\par\noindent
We wish to solve these equations by $\ldots$
```

Processed, that looks like:

Consider the functions:

$$\begin{aligned}\phi_1 = \arg(w+1) &= \arctan(v/[u+1])\\ \phi_2 = \arg(w-1) &= \arctan(v/[u-1])\end{aligned} \tag{21}$$

We wish to solve these equations by …

• If you wanted to number **every equation** in the display you just built, you'd say, instead:

```
Consider the functions:
$$\eqalignno{\phi_1={\rm arg}(w+1)&=\arctan (v / [u+1])&(45)\cr
             \phi_2={\rm arg}(w-1)&=\arctan (v / [u-1])&(46)\cr}
\medskip\par\noindent
We wish to solve these equations by $\ldots$
```

Processed, the separately numbered `\eqalignno` approach generates output which looks like:

Consider the functions:

$$\phi_1 = \arg(w+1) = \arctan(v/[u+1]) \tag{45}$$
$$\phi_2 = \arg(w-1) = \arctan(v/[u-1]) \tag{46}$$

We wish to solve these equations by …

Notice that this second approach uses `\eqalign`**no** instead of simply `\eqalign`, and notice that our equation numbers are included immediately before the `\cr` which ends each line, separated from the actual equation itself by an ampersand (`&`).

Some Basic Information About Entering Equations

But, you say, it is all fine and good to be able to know the difference between embedded and displayed equations, and to know how to number and align equations, but how do I actually write the damn things in the first place!? Ah! Yes! I guess we should talk about that, eh?

Regardless of whether you are entering an embedded equation, or an equation which will be displayed, there are some basic things you need to know about entering equations in TEX:

• TEX generally ignores spaces in equations. Thus, the following are equivalent to TEX and will come out typeset exactly the same:

```
$$ a + b + c  =  d $$

$$a+b+c=d$$
```

• To force TEX to pay attention to spaces in your equations, use the following:

```
\!       negative thin space (-1/6 \quad)
\,       thin space (+1/6 \quad)
\>       medium space (+2/9 \quad)
\;       thick space (+5/18 \quad)
\quad    one quad -- approximately the size of a capital "M"
\qquad   a double quad
\␣       a control space
```

• In general, math mode text is set in italics. To force TEX to set a few words of text in a normal roman font while in math mode, you can put the text in a `\rm` group:

```
$$22/7 {\rm\ is \ an \ approximation \ for \ } \pi$$
```

which yields output that looks like:

$$22/7 {\rm\ is \ an \ approximation \ for \ } \pi$$

But notice what a pain that is to set!

By far your best move is to avoid setting **displayed** equations which include lots of "regular" text. Use **embedded** equations instead, and just drop into math mode when you need to on a "phrase-by-phrase" basis. For example, look how much cleaner the following re-casting of our example becomes:

```
$22/7$ is an approximation for $\pi$
```

Processed, that looks like:

$22/7$ is an approximation for π

TEX also contains pre-defined roman font text strings for common mathematical "words" such as sin, cos, tan, lim, etc. See below for more information.

Greek Letters

• To enter **lowercase** greek letters in math mode (i.e., between \$ ... \$ or \$\$ ... \$\$), use:

α	β	γ	δ	ϵ	ε
\alpha	\beta	\gamma	\delta	\epsilon	\varepsilon
ζ	η	θ	ϑ	ι	κ
\zeta	\eta	\theta	\vartheta	\iota	\kappa
λ	μ	ν	ξ	o	π
\lambda	\mu	\nu	\xi	o	\pi
ϖ	ρ	ϱ	σ	ς	τ
\varpi	\rho	\varrho	\sigma	\varsigma	\tau
υ	ϕ	φ	χ	ψ	ω
\upsilon	\phi	\varphi	\chi	\psi	\omega

• To enter **uppercase** greek letters in math mode (i.e., between \$ ... \$ or \$\$ ... \$\$), use:

Γ	Δ	Θ	Λ	Ξ	Π
\Gamma	\Delta	\Theta	\Lambda	\Xi	\Pi
Σ	Υ	Φ	Ψ	Ω	
\Sigma	\Upsilon	\Phi	\Psi	\Omega	

Only those uppercase greek letters shown above are available in TEX; those not defined are indistinguishable from their roman counterparts. Thus, for example, `\Alpha` is undefined, and will generate an error if you attempt to include that undefined symbol in your document. Simply substitute {`\rm A`} instead at the point where you need a capital alpha. For italic uppercase greek letters, try: `${\mit \Phi}$`.

Script Letters

Sometimes you'll need to enter a capital "script" letter while in math mode (i.e., , between \$ ... \$ or \$\$ ... \$\$). Use the calligraphic style for that purpose (lowercase isn't available):

$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

{\cal ABCDEFGHIJKLMNOPQRSTUVWXYZ}

Common Mathematical Operators

You also need to be able to set common mathematical symbols. For example, while in math mode (i.e., , between \$ . . . \$ or \$\$ . . . \$\$) some of the symbols you can use include:

$+$	$-$	$=$	$\not=$	$<$	$>$
`+`	`-`	`=`	`\not=`	`<`	`>`
$\le$	$\ge$	$\ll$	$\gg$	$\approx$	$\equiv$
`\le`	`\ge`	`<<`	`>>`	`\approx`	`\equiv`
$($	$)$	$[$	$]$	$\{$	$\}$
`(`	`)`	`[`	`]`	`\{`	`\}`
$\langle$	$\rangle$	$\colon$	$\cdot$	$\cdots$	$\ldots$
`\langle`	`\rangle`	`\colon`	`\cdot`	`\cdots`	`\ldots`
$\vert$	$\Vert$	$\backslash$	$\lceil$	$\lfloor$	∇
`\vert`	`\Vert`	`\backslash`	`\lceil`	`\lfloor`	`\nabla`
∂	∞	$\forall$	$\exists$	$\circ$	$\bullet$
`\partial`	`\infty`	`\forall`	`\exists`	`\circ`	`\bullet`

Symbols for Logic and the Algebra of Sets

$\in$	$\notin$	$\cap$	$\cup$	$\vee$	$\wedge$
`\in`	`\notin`	`\cap`	`\cup`	`\vee`	`\wedge`
$\subset$	$\supset$	$\subseteq$	$\supseteq$	$\oplus$	$\otimes$
`\subset`	`\supset`	`\subseteq`	`\supseteq`	`\oplus`	`\otimes`
$\sim$	$\ast$	$\emptyset$	$\rightarrow$	$\leftarrow$	$\leftrightarrow$
`\sim`	`\ast`	`\emptyset`	`\rightarrow`	`\leftarrow`	`\leftrightarrow`

For more information on available mathematical symbols, see Appendix F of *The TEXbook*.

Subscripts, Superscripts, and Combinations Thereof

While in math mode, make subscripts using an underscore and make superscripts using a caret (or "hat"). For example:

Subscript:	`$X_{14}$`	X_{14}
Superscript:	`$X^{3}$`	X^3
Combined subscript and superscript:	`$X_{6}^{8}$`	X_6^8
Even odder combinations, too:	`$_{4}X^{a^b}`	${}_4X^{a^b}$
	`$^{2}_{3}X_{5}^{1}$`	${}^2_3X_5^1$

Note that you may use (and you may *need to* use) curly braces to indicate exactly what material is to be raised or lowered; if a particularly baroque expression is ambiguous, TEX will complain.

Also note that you can override the automatic reduction in size of subscripts and superscripts. Do so by including a `\textstyle` within the subscript or superscript curly braces:

`$X^{{\textstyle Y}^{\textstyle Z}}$`	$X^{{\textstyle Y}^{\textstyle Z}}$

Math Accents

While in math mode, get accented variables[11] by saying:

x'	x''	$\bar x$	$\hat x$	$\vec x$	$\dot x$
`x'`	`x''`	`\bar x`	`\hat x`	`\vec x`	`\dot x`
$\ddot x$	$\check x$	$\breve x$	$\grave x$	$\acute x$	$\tilde x$
`\ddot x`	`\check x`	`\breve x`	`\grave x`	`\acute x`	`\tilde x`

"Wider" accents are also available:

$\overline{abcde}$	$\underline{abcde}$	$\widehat{abcde}$	$\widetilde{abcde}$
`\overline {abcde}`	`\underline {abcde}`	`\widehat {abcde}`	`\widetilde {abcde}`

[11] Note that mathematical typesetting conventions require you to use a special "dotless" i and "dotless" j when accenting those characters. Get those characters by saying `\imath` to get a dotless $\imath$ or by saying `\jmath` to get a dotless $\jmath$ where needed in math mode.

Roman Font Mathematical "Words"

TeX includes a number of predefined roman font math-mode "words":

• **Trigonometric Functions:**

sin	cos	tan	csc	sec	cot
`\sin`	`\cos`	`\tan`	`\csc`	`\sec`	`\cot`
arcsin	arccos	arctan	sinh	cosh	tanh
`\arcsin`	`\arccos`	`\arctan`	`\sinh`	`\cosh`	`\tanh`
coth					
`\coth`					

Those trigonometric functions not shown in the above list (including the remaining inverse trigonometric functions, the remaining hyperbolic functions, the inverse hyperbolics, exsec, covers, vers, hav, cis, etc.), need to be manually entered as roman text while you're in math mode since TeX considers them too obscure to merit pre-defined symbols.

•**Other Functions:**

The following list of other functions is something of a hodge-podge, consisting of the other roman font mathematical "words" predefined in plain TeX:

min	max	ln	log	lg	exp
`\min`	`\max`	`\ln`	`\log`	`\lg`	`\exp`
gcd	inf	sup	lim	lim inf	lim sup
`\gcd`	`\inf`	`\sup`	`\lim`	`\liminf`	`\limsup`
dim	det	arg	deg	hom	ker
`\dim`	`\det`	`\arg`	`\deg`	`\hom`	`\ker`
Pr	mod	(mod 16)			
`\Pr`	`\bmod`	`\pmod{16}`			

• If the mathematical "word" you find yourself needing isn't listed, you can always extend the above list to meet your circumstances. For example:

```
    \def\lcd{\mathop{\rm lcd}\nolimits}
```

would define a new symbol `\lcd` as a convenience for those who commonly write about least common denominators.

Limits

One example of using a "named" mathematical function is writing a limit:

```
$${\bf u}'(t)=\lim_{\Delta t\rightarrow 0}
    {{{\bf u} (t+\Delta t) - {\bf u}(t)} \over {\Delta t}}$$
```

You'll get output from those commands that looks like:

$$\mathbf{u}'(t)=\lim_{\Delta t\rightarrow 0}{{\mathbf{u}(t+\Delta t)-\mathbf{u}(t)}\over{\Delta t}}$$

Note that the "named" mathematical function can have a subscript just the way any "regular" symbol can, but that TEX is smart enough to center the limit under this particular named function rather than simply dropping the subscript below the base line at the end of the symbol. This decision — to drop the subscript down in place, or to drop the subscript down and then pull it back to center it — is controlled by the `\limits` or `\nolimits` attribute chosen when that symbol was defined. (You may have noticed that little feature in our "`\lcd`" example in the preceding section. Now you know ... the *rest* of the story!)

Radicals: Square Roots, Cube Roots, etc.

• To get a square root as part of a math mode expression, try something like:

`\sqrt{x+y}` to get: $\sqrt{x+y}$

• TEX's `\sqrt` will attempt to be "smart" and match the radical height to the height of the arguments given: `\sqrt` will make a bigger radical for "tall" arguments (such as those containing capital letters) than it would for arguments comprised solely of "short" lower-case letters.

However, there may be times when you *don't want* TEX to tailor each `\sqrt` radical individually; at times it may be cleaner to have all the `\sqrt` radicals be of the same height. When that's the case, include a `\mathstrut` as part of each radical in your equation:

```
\sqrt{\mathstrut x+y} + \sqrt{\mathstrut Z}
```

All of your radicals will then be the same height:

$\sqrt{\mathstrut x+y}+\sqrt{\mathstrut Z}$

• To get any arbitrary root (cube root, quartic root, etc.), while in math mode try:

```
\root 4 \of X
```

You'll then get output which looks like:

$\sqrt[4]{X}$

Making Large Fractions

To generate a “large” fraction while in math mode, use the `\over` command:

```
$${x+y \over a+b}$$
```

That generates output which looks like:

$${x+y \over a+b}$$

• If you need to generate “fractions upon fractions”, you can quickly end up with something which looks like:

```
$${{x+y \over a+b} \over {x+a \over y+b}}$$
```

and, even worse, the output from that “fraction upon fraction” form tends to be hard to read, ambiguous and wasteful of space in the printed page:

$${{x+y \over a+b} \over {x+a \over y+b}}$$

• To disambiguate the “fraction upon fraction” form, you can employ a bold division line:

```
$${{x+y \over a+b} \above 1pt {x+a \over y+b}}$$
```

That yields the following non-ambiguous expression:

$${{x+y \over a+b} \above 1pt {x+a \over y+b}}$$

Even though that “fraction upon fraction” format is now unambiguous, it is still hard to read and still is wasteful of space on the printed page.

• You are far better off writing “fractions upon fractions” horizontally, with an enlarged slash. See the following section on enlarged grouping operators for more information.

Making Large Grouping Operators

When you begin working with large and complicated displayed equations, a number of large grouping operators will become important to you. These are:

- Large parentheses, braces, and brackets,
- Large vertical ("absolute value") bars, and
- Large forward and backward slashes.

Five different sizes of those operators can be requested:

- Normal size
- `\big`, `\bigl` and `\bigr` — a hair bigger than normal
- `\Big`, `\Bigl` and `\Bigr` — 1.5 times the `\big` size operators
- `\bigg`, `\biggl` and `\biggr` — 2.0 times the `\big` size operators
- `\Bigg`, `\Biggl` and `\Biggr` — 2.5 times the `\big` size operators

Thus, for example, you might write something like:

```
$$w=\biggl({a+b \over d+e}\biggr) \bigg/ \biggl({f+g+h \over i+j}\biggr)$$
```

In order to obtain output that looks like:

$$w=\biggl(\frac{a+b}{d+e}\biggr)\bigg/\biggl(\frac{f+g+h}{i+j}\biggr)$$

You'll notice that I used `\biggl(` for the left hand side parentheses, `\biggr)` for the right hand side parentheses, and plain `\bigg` for the center division sign. TEX handles spacing around the big grouping operators differently depending on which "side" the operator is on. Unless you have a compelling reason to the contrary, you should generally use `\biggl` for left hand side operators, `\biggr` for right hand side operators, and `\big` for those operators which cannot really be called either left hand side or right hand side.

To give you an idea of what the various symbols look like at the five available sizes, here's a little sampler:

$$\Biggl(\biggl(\Bigl(\bigl((X)\bigr)\Bigr)\biggr)\Biggr) \qquad \Biggl[\biggl[\Bigl[\bigl[[X]\bigr]\Bigr]\biggr]\Biggr] \qquad \Biggl\{\biggl\{\Bigl\{\bigl\{\{X\}\bigr\}\Bigr\}\biggr\}\Biggr\} \qquad \Biggl\langle\biggl\langle\Bigl\langle\bigl\langle\langle X\rangle\bigr\rangle\Bigr\rangle\biggr\rangle\Biggr\rangle$$

$$\Biggl|\biggl|\Bigl|\bigl||X|\bigr|\Bigr|\biggr|\Biggr| \qquad \Biggl\|\biggl\|\Bigl\|\bigl\|\|X\|\bigr\|\Bigr\|\biggr\|\Biggr\| \qquad \Bigg/\bigg/\Big/\big//X//\big/\Big/\bigg/\Bigg/ \qquad \Bigg\backslash\bigg\backslash\Big\backslash\big\backslash\backslash X\backslash\big\backslash\Big\backslash\bigg\backslash\Bigg\backslash$$

Combination Notation

To denote combinations in TEX, use TEX's `\choose` notation:

`${n \choose r}$` will print as: ${n \choose r}$

Matrices

To typeset a matrix, enter something like:

```
$${\bf X} = \left(\matrix{
                         X_{11}&X_{12}&X_{13}&\ldots&X_{1j}\cr
                         X_{21}&X_{22}&X_{23}&\ldots&X_{2j}\cr
                         \vdots&\vdots&\vdots&\ddots&\vdots\cr
                         X_{i1}&X_{i2}&X_{i3}&\ldots&X_{ij}\cr
          }\right)\eqno(4)$$
```

Processed by TEX, that looks like:

$$\mathbf{X} = \begin{pmatrix} X_{11} & X_{12} & X_{13} & \ldots & X_{1j} \\ X_{21} & X_{22} & X_{23} & \ldots & X_{2j} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ X_{i1} & X_{i2} & X_{i3} & \ldots & X_{ij} \end{pmatrix} \tag{4}$$

Note that you can use symbols other than parentheses to set off your matrix; common alternatives include large square brackets and large bars (for denoting determinants).

Case Structure

Sometimes you need a "case" structure to represent choices and outcomes; for example, maybe you're trying to show quantities and price breaks:

```
    $${\rm your\ price} =
       \cases{\$10.00/{\rm unit}& for 1,000 or more units\cr
             \$12.50/{\rm unit}& for 500 to 999 units\cr
             \$15.00/{\rm unit}& for 1 to 499 units\cr}\eqno(16)$$
```

That produces output which looks like:

$$\text{your price} = \begin{cases} \$10.00/\text{unit} & \text{for 1,000 or more units} \\ \$12.50/\text{unit} & \text{for 500 to 999 units} \\ \$15.00/\text{unit} & \text{for 1 to 499 units} \end{cases} \tag{16}$$

Note that the material to the left of the equal sign, and the material to the left of the ampersand sign, is automatically typeset in math mode. The material to the right of the ampersand sign is automatically typeset in non-math mode.

Summations

To generate large summation displays, use TEX commands like:

```
The Maclaurin power series can be written:
$$ e^{x} = \sum_{m=0}^{\infty} {x^{m} \over m!} =
1+x+{{x^{2}} \over 2!} + {{x^{3}} \over 3!} + \cdots $$
\medskip\par\noindent
```

That yields output which looks like:

The Maclaurin power series can be written:

$$e^{x} = \sum_{m=0}^{\infty} {x^{m} \over m!} = 1+x+{{x^{2}} \over 2!} + {{x^{3}} \over 3!} + \cdots$$

• Summation limits are actually nothing special — in fact, to TEX they are simply a subscript and a superscript. (Did you notice the underscore and caret?)

• Normally, expressions containing summations are large and complicated and thus end up displayed (i.e., set between double dollar signs). If you find yourself with summations in embedded (single dollar sign) equations, you need to be aware that the capital sigma will be smaller than in displayed form, and the summation limits will be moved so they follow the summation rather than appearing above and below the summation. For example, by changing our double dollar signs to single dollar signs, our Maclaurin example becomes transformed into:

The Maclaurin power series can be written: $e^{x} = \sum_{m=0}^{\infty} {x^{m} \over m!} = 1+x+{{x^{2}} \over 2!} + {{x^{3}} \over 3!} + \cdots$

Make a determined effort to set summations in displayed form only.

Integrals

Integrals are quite similar to summations. Consider the following:

```
$$\int_{a}^{b}f(x)\,dx=F(b)-F(a)$$
```

That comes out looking like:

$$\int_{a}^{b}f(x)\,dx=F(b)-F(a)$$

Just as in summations, integration limits are simply sub and superscripts; the `\,` which follows the `f(x)` simply adds a little space between the function and its differential.

Definitions

By now you may have formed the impression that using TEX means using impossibly verbose commands to accomplish even the simplest of tasks.

For example, consider getting double spaced copy. Right now you get it by saying:

```
\baselineskip=2\normalbaselineskip
```

Thirty-four characters is a lot to type every time you want to begin double spacing.

• Fortunately, TEX's author allows you to extend or customize TEX by defining your own shorthand commands for frequently employed (but excessively lengthy) "native" TEX instructions. Thus, for instance, we can create a new TEX command called `\DS` which we'll use as a convenient way of requesting double spaced copy in TEX:

```
\def\DS{\baselineskip=2\normalbaselineskip}
```

Clearly, it is much easier to type `\DS` when we want double spaced copy than it is to type `\baselineskip=2\normalbaselineskip`, don't you think?

Notice that I elected to use a shorthand name for my new command consisting solely of upper case letters; by doing so, I've practically insured that my new command won't conflict or interfere with any existing TEX commands[12].

• TEX macros can also take substitutable parameters (or "macro variables"). For example, maybe you'd like to have a more general line space setting macro. In that case, you could define a new command by saying something like:

```
\def\SP#1{\baselineskip=#1\normalbaselineskip}
```

You could then get triple-spaced copy by saying `\SP 3` or quad-spaced copy by saying `\SP 4`.

• If you end up creating a lot of these definitions, collect them all into a file called `macro.tex` You can then automatically include those definitions in each of your documents by saying:

```
\input macro.tex
```

near the top of each of your TEX documents.

• There is a *lot* more you can do with macros in TEX, and a lot more you should *know* about TEX macros before you attempt to get too tricky using them. Aspiring TEX hackers should thoroughly study Chapter 20 of *The TEXbook* for more information before jumping right into defining a whole pile of macros.

[12] Virtually all commands automatically defined in plain TEX are lowercase or mixed case only, and since TEX distinguishes between upper and lowercase letters in parsing command names, our uppercase-only command names should generally not cause any trouble.

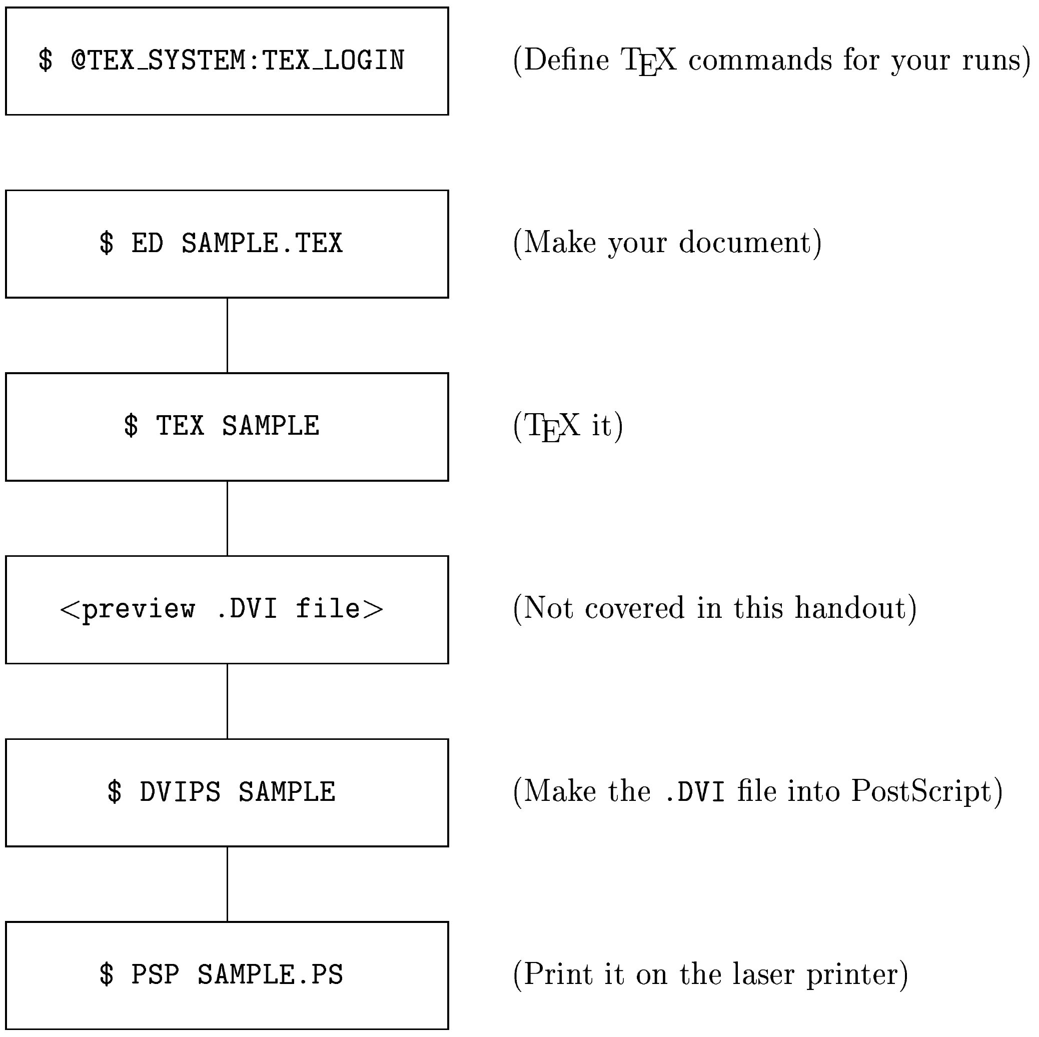
V. TEX ON THE OREGON VAX 8800

Up to now, we've been talking about TEX in general — we haven't really tied ourselves to any particular machine. What we've told you should work just as well on a PC running TEX as on a NeXT running TEX as on a VAX running TEX.

Now, though, it is time to get down to the peculiarities of TEXing on the OREGON VAX.

The TEX Execution Cycle

The general process you'll be using is as follows:



```
$ @TEX_SYSTEM:TEX_LOGIN      (Define TEX commands for your runs)
$ ED SAMPLE.TEX              (Make your document)
$ TEX SAMPLE                 (TEX it)
<preview .DVI file>          (Not covered in this handout)
$ DVIPS SAMPLE               (Make the .DVI file into PostScript)
$ PSP SAMPLE.PS              (Print it on the laser printer)
```

We'll now look at each of those steps, in order.[13].

[13] . . . except for previewing! For information on previewing your document on screen, please see the separate write-up *Previewing Your TEX* `.DVI` *File On-Screen*

Building Your TEX Document Using an Editor

• Your first task is to actually build your TEX document. You can use any of the regular VAX text editors to do this; EDT, EVE and EMACS will all work fine, for instance. As an example:

```
$ ED SAMPLE.TEX
*C
\twelvepoint
\centerline{\bf Introduction}
\bigskip
It is well established that the primary principle of all Architecture
is balance.  Without balance, <blah blah blah>
<CTRL-Z>
*EX
```

• Some users may prefer to build their TEX documents using a microcomputer word processor or text editor, uploading their file to the VAX once it has been prepared. That's fine, too; just remember to hit return at the end of each line of text as you enter it (or save your TEX document as "text with line breaks").

Upload your TEX document just as you would any other text file; raw TEX documents consist solely of regular printable characters so there is no need to do a "binary" file transfer when uploading a TEX document prepared on a microcomputer.

Defining TEX

Because a relatively small percentage of all VAX users will initially be using TEX, the commands needed to run TEX are not automatically defined at login time. Instead, in order to be able to use TEX, you need to enter the command:

```
$ @TEX_SYSTEM:TEX_LOGIN
```

at the dollar sign. The TEX commands obtained by executing that command will remain available for the duration of your current session, but will be lost when you log off.

Therefore, if you plan to do a lot of work with TEX, you'll want to add the above DCL command to your account's `LOGIN.COM` file so you don't need to manually re-request TEX's commands every time you login to do some computerized typesetting. (Be sure to actually type in the dollar sign at the start of this (or any) command you add to your `LOGIN.COM` file!)

Running TEX

Once you've got your TEX document entered and TEX's commands defined, you're then ready to process your document into a device independent (`.DVI`) file by saying:

```
$ TEX filename
```

For example, to run TEX on the raw TEX document SAMPLE.TEX, you'd enter:

```
$ TEX SAMPLE
```

Note that you don't have to bother giving the file extension of your TEX document file (`SAMPLE.TEX`) because TEX will assume the file name ends in `.TEX` if you omit the file extension from the file name. To take advantage of that feature, and to help you keep your files organized, you are strongly encouraged to have all your TEX document filenames end in `.TEX` unless you have a particularly good rationale for doing otherwise.

Anyhow, when you run your document through TEX your screen should look something like:

```
$ TEX SAMPLE
This is TeX, NLS VMS Version 2.991a (preloaded format=plain 89.10.26)
(DISK$USER8:[JSMITH]SAMPLE.TEX;3 (TEX_TEXROOT:[INPUTS]FONTSIZE.TEX;2) [1]
[2] [3] [4] [5]
Output written on DISK$USER8:[JSMITH]SAMPLE.DVI;3 (5 pages, 9648 bytes).
Transcript written on DISK$USER8:[JSMITH]SAMPLE.LIS;3.
```

TEX is telling you that you are running Version 2.991a of the Northlake Software port of TEX for VMS, using the normal default TEX format ("`plain`") prepared in October of 1989.

TEX then announces that it is beginning to read and process a document called `SAMPLE.TEX`. While processing that file, it reads in another file called `FONTSIZE.TEX` (which we know contains TEX commands defining the various fontsizes we might want to use).

TEX then begins to build and output pages one at a time: as each page is output, TEX writes the number of that page inside square brackets on your terminal.

When TEX finishes processing your document, it provides a summary explaining how many pages it has created, the size of the device independent output file (`SAMPLE.DVI`), and the name of the transcript file (`SAMPLE.LIS`) which contains details about any problems TEX had while processing your document.

If you have encountered no errors, you're now ready to convert the output `.DVI` file into PostScript for printing on a laser printer.

In many cases, however, you'll have at least a couple of TEX run time errors to resolve. When you encounter errors, see the next section for error deciphering assistance.

Decoding TEX Errors

When TEX has a problem with your document, it will report an error message which will probably look roughly like the following:

```
$ TEX MYPAPER
This is TeX, NLS VMS Version 2.991a (preloaded format=plain 89.10.26)
(DISK$USER8:[JSMITH]MYPAPER.TEX;2
!  Undefined control sequence.
l.18 ...small file.  Here is some text which is {\bg
                                                     emphasized.}
?
```

This error message isn't actually too bad, as error messages go. But what does it mean?

• First of all, TEX is telling you that you are trying to use a TEX command which doesn't exist (`Undefined control sequence`).

• Next, TEX tells you the line on which the problem exists (`l.18`), and even shows you the point at which the problem was detected (TEX marks the point where a problem was detected by dropping down to a new line without doing a carriage return).

In our case, the user entered `\bg` where we can assume they probably meant to enter `\bf` for "boldface." Fine. We now know what caused our error. But what do we do when we're looking at the question mark TEX has issued?

Your best move as a beginner is to type an `X` and hit return to exit TEX. You can then edit your file and fix the line with the problem, and rerun TEX on your corrected file.[14] [Remember that under VMS you can readily roll back to earlier commands by hitting one or more <CTRL>-B's at the dollar sign — this command "history" feature is very convenient when you repeatedly edit-and-rerun-TEX on a document during the document development process.

Now we'll explain the most common TEX errors you're likely to encounter.

[14] If you are feeling more adventurous, you can type a question mark in response to TEX's question mark; TEX will then outline your other options for dealing with the reported error.

The Most Common TeX Errors

In addition to the **Undefined control sequence error** we just discussed, there are a number of other TeX errors which are also commonly encountered. (At least the errors listed below are the sort of errors *I* typically see — *you* may tend to generate an entirely different set of errors depending upon your mental processes, TeX habits, and keyboarding expertise). Anyhow, the errors *I* see most often are:

• **Overfull \hbox** This error message is saying that TeX is wrestling with a line of text which is too wide for the document, and which it cannot break into acceptable width lines (either by splitting the line at whitespace, or by hyphenating a word near the end of the line and breaking the line there).

If you get this error, begin by inspecting the text that's making TeX gag. Do you have a particularly long and hard-to-hyphenate word near the end of the line? TeX may be encountering trouble breaking that word.[15] Have you incorrectly used un-breakable ties (`~`) instead of regular spaces for all or most regular interword spaces? Did you explicitly create a too-large unbreakable `\hbox`? Did you make a huge table that TeX just can't fit on the page? Maybe you've made a monster equation? Can you force a break with `\break` somewhere near the end of the line to resolve this problem?

If inspection of the raw TeX command file fails to give you insight into what makes the line appear too long, ignore the `Overfull \hbox` message and proceed to `DVIPS` and `PSP` the document, error and all. TeX will mark the overfull `\hbox` with a solid right bar near the right hand side of the page. If after looking at the printed output you still can't see what the problem is, come see me and I'll try to help you resolve the problem.

• **Underfull \vbox** This error message usually arises when TeX has too few lines to fill out a page. Most commonly, this is caused by issuing a `\eject` without a preceding `\vfill` to pad out the rest of the page.

• **\end occurred inside a group at level <X>** You missed one or more closing braces somewhere. Don't be surprised if pages and pages of your document (instead of just the couple of words you may have intended) are typeset in italics, or are boldfaced, or are underlined. Go back and insert the missing curly braces to eliminate this error.

• **Too many }'s** Now you've got **too many** right hand curly braces! The number of right hand curly braces should match the number of left hand curly braces.

You may just need to remove a redundant right hand curly brace; on the other hand, you may need to snoop around and make sure you aren't short a necessary or important left hand curly brace!

---

[15] You may need to give TeX a hand by explaining where it can safely hyphenate unusual words. Add a `\-` to tell TeX where it can break a tough word if it needs to do so. Don't bother providing this sort of advice unless TeX demonstrates a need for it — TeX normally does a fantastic job of breaking words at an appropriate point on its own.

• **Missing $ inserted** Usually this means that you've: (a) attempted to use a math-mode-only symbol (such as `\backslash`) while you weren't in math mode, (b) you missed a dollar sign at the beginning or end of an equation, or (c) you entered `$` instead of `\$` when you actually wanted a dollar sign to appear in your text.

If you want to use a math-mode-only symbol such as `\backslash` while entering regular text, remember to encapsulate the code for that symbol within an opening and a closing dollar sign.

• **Missing number, treated as zero** TeX expected a number (most often as part of a dimension), which you didn't provide. Insert an appropriate numeric value to eliminate this error.

• **Misplaced alignment tab character &** TeX has detected an ampersand, which is normally used by TeX as a tab character or as an alignment point within a stack of equations, in a context where that didn't make sense. Were you working on setting up a table or aligning equations?

If not, you probably just wanted to have an ampersand actually appear in your text. Change the `&` into a `\&` to make the ampersand actually show up in your document.

• **Double subscript** You have a "subscript upon a subscript" (such as `a_b_c`) which is ambiguous. TeX needs to know if you meant:

```
a_{b_c}
```

or

```
{a_b}_c
```

since TeX treats those two cases somewhat differently when typesetting them. Insert left and right curly braces to clearly show just what is getting subscripted in your expression.

• **Ambiguous; you need another { and }.** This is another error you'll see if you tend to be parsimonious with curly braces. Most often you'll see this error when you're creating a compound fraction and you've entered an ambiguous expression like:

```
$ a \over b \over c $
```

Enter curly braces to clarify "what you want to have above what". For example:

```
${a \over b} \over c $
```

Remember, however, that "fractions upon fractions" are generally a bad idea, and you should rewrite the expression to eliminate them if at all possible.

Converting Your .DVI File Into PostScript

After TEX processes your `.TEX` file without error, you'll have a *device independent* (`.DVI`) intermediate file which you can then process into a final *device specific* file for printing on a laser printer.

`.DVI` files can be processed into device specific output files suitable for printing on any of a number of different printers. For example, there are `.DVI` converters, or "filters," for PostScript laser printers, HP LaserJet printers, certain bit-mapped CRT's, etc. In this write-up we're only going to explain how you can write output for PostScript printers (such as the Computing Center's Xerox 4045/160) using the `DVIPS` (DVI-to-PostScript) driver written by Tomas Rokicki.[16]

Essentially, once TEX finally processes your document without complaint, the procedure for converting your `.DVI` file into PostScript is simple. All you have to do is say:

```
      $ DVIPS filename
```

For example, if your original TEX document file was named `SAMPLE.TEX`, and TEX processed that file into `SAMPLE.DVI`, you can then convert it into PostScript by saying:

```
      $ DVIPS SAMPLE
```

After executing that command you should see something which looks like:

```
      $ DVIPS SAMPLE
      This is dvips, version 5.392 (C) 1986-90 Radical Eye Software
      ' TeX output 1990.11.15:1443' -> SAMPLE.PS
      [tex.pro].  [1] [2] [3] [4] [5]
      $
```

`SAMPLE.PS` is the output file containing your document in final PostScript form, ready to print on a PostScript laser printer.

• There may be times when you only want to print certain pages of your document. For instance, you may not have touched the first twenty-four pages of your document, but you need a copy of all pages from there on. To get a copy of all pages from 25 through the end of your document, you'd say:

```
      $ DVIPS -p25 SAMPLE
```

[16] If you're using a machine other than OREGON, you can obtain a copy of `DVIPS` via anonymous FTP from `NEON.STANFORD.EDU`; look in directory `pub`. Users at sites without network access can write Tomas Rokicki, Radical Eye Software, Box 2081, Stanford, CA 94039 for information on obtaining a commercial distribution of DVIPS.

To get only pages 17 through 28, inclusive, try:

```
$ DVIPS -p17 -l28 SAMPLE
```

(The second command line argument in the above example begins with the letter "el," not the number "one"!)

To get only the first 8 pages of SAMPLE.DVI, use:

```
$ DVIPS -n8 SAMPLE
```

• NOTE: If you are familar with DVIPS from another system, you may be surprised to notice that DVIPS on the VAX automatically lowercases any command line arguments you provide. This is a feature (or a bug) caused by compiling DVIPS using VAX/VMS C, which happily trashes the case of all the command line arguments it touches. If you need to preserve the case of uppercase (or mixed case) command line argument strings, you'll need to enclose those command line argument strings inside double quotation marks.

DVIPS Features

• DVIPS (**NOT** TEX) can generate special PostScript effects, such as overprint screens. For example, each of the appendices to this guide was set with a large overprinted Helvetica-Bold letter, denoting the current appendix. For an example of how this was done, look at the TEX source file for the appendices.

Note that these special effects are strictly a function of `DVIPS`; it is virtually certain that almost any other DVI-to-<whatever> translator won't be able to handle generation of these special effects! Think twice before you decide you need to have this sort of special effect included in your document, since their use will limit its portability.

• Another `DVIPS`-specific feature is use of PostScript fonts instead of, or in addition to, the normal Computer Modern Roman fonts. For an example of a document which does this, see Appendix F. Note that like the use of special effects (such as overprinting), use of PostScript fonts may limit your ability to use `DVI` translators other than `DVIPS`.

• A final `DVIPS`-specific feature is inclusion of PostScript graphics directly as part of your TEX document. See Appendix G for an example of this. Be aware that PostScript graphic files tend to be quite large, and use of this feature may limit the portability of your document to other DVI translators. You may also need to hack on the actual PostScript file you want to include to get it into "includable" form.

Printing PostScript Output on the VAX's Xerox 4045/160

• After you've run your document through `TEX` and `DVIPS`, you'll have a `.PS` (PostScript) file which can be printed on a PostScript laser printer.

To print the file `SAMPLE.PS` on the the Xerox 4045/160 PostScript-compatible printer connected to the OREGON VAX, enter the command:

```
$ PRINT/QUE=SYS$LASER/SETUP=POSTSCRIPT  SAMPLE.PS
```

Your PostScript file will then be queued to print.

• Because the above print command is an easy one to fumble, the Computing Center has defined a special "PostScript Print" command that's a little easier to type. Specifically, instead of having to enter the complicated `PRINT` command shown above to print the file SAMPLE.PS, you can simply say:

```
$ PSP  SAMPLE.PS
```

Why has the Computing Center gone to the trouble of creating a special command just to prevent possible user problems in entering that `PRINT` command? Well, if a user omits either the required `/QUE` or the required `/SETUP` string, the raw PostScript commands in the user's file will be printed as page upon page of seeming gibberish instead of being properly interpreted by the printer. This results in a frustrating waste of printer resources we'd like to help you avoid if possible.

One small caution about using the PSP command: note that the `PSP` command won't automatically pick up any special `/NOTE` qualifiers you may have added to your regular PRINT command (such as those which you may have added to route your printer output to a special output box). If you can't find your PostScript output in your normal printer output box when you're using the `PSP` command, be sure to check out front in the general printer output pick up area, too.

• One more point you should be aware of: be **sure** to explicitly include the `.PS` file extension when you name the PostScript file you want to print! If you accidentally omit the `.PS` file extension, VMS will (incorrectly) assume that you actually want to print **`filename.LIS`** instead of **`filename.PS`**,and you won't get any output whatsoever (except the print job header page).

VI. CONCLUSION

Where from Here?

You now know enough about TEX to be able to handle most typesetting projects. You should be able to prepare typical text-oriented documents, some basic tables, and most commonly-seen types of equations. You have become, in effect, a journeyman TEXnician.

However, this doesn't mean that you are done learning about TEX. On the contrary, now that you understand the basics of using TEX from reading this write-up, you should feel ready to dig into Knuth's *TEXbook* in earnest. There are a lot of subtle TEX topics which have gotten cursory coverage (at best) in this write-up, but which *The TEXbook* covers in much greater detail and from a properly grounded context. You *really* want to obtain a copy of the *TEXbook* and read it in detail, even going so far as to work each of the little exercises provided.

In addition, you may enjoy reading *TEX For The Impatient*[17]. Like this write-up, *TEX For the Impatient* offers a less technical discussion of TEX than Knuth's *TEXBook*.

Also think about picking up a copy of the original write-up that comes with Tom Rokicki's `DVIPS` program: it has some interesting details about DVIPS which we haven't covered in this write-up. Copies are available at a nominal cost from the Computing Center Documents Room.

What If I Get Stuck?

Feel free to come visit if you need help with a TEX problem. I'll be glad to try to help, and your visit will help me see what topics need expanded coverage in the next revision of this write-up.

You may also want to begin following `comp.text.tex` in `NEWS`. That newsgroup gives you access to some of the best TEX experts in the country for those times when you are *really* stuck and no one can help you locally. However, **PLEASE** exhaust local resources **FIRST** before you post a question to the newsgroup — you really don't want to bother thousands of people nationwide with a question you could easily get answered locally by asking me, or by looking in this handout, or by looking in *The TEXbook*.

Another general resource you should be aware of is the TEX Users Group, P.O. Box 9506, Providence, Rhode Island 02940; phone 401-751-7760; FAX 401-751-1071. When you join TUG for $35.00 ($25.00 for students) you get a whole host of benefits, including a subscription to TUGboat (the TEX User Group's newsletter), information on TEX training, TEX meetings, a directory of other TUG members, access to software, etc.

[17] Paul W. Abrahams, et. al., *TEX For The Impatient*, Addison-Wesley, Reading, MA: 1990, 357 pps.