TEX Made Easy

Using TEX With The Plain Macro Package

Daniel M. Zirin

Version: 1.2
Published by Zar Limited, P.O.Box 13107, Denver, CO 80201, U.S.A..

The TEX macros presented in this book have been included for their instructional value. They have been tested with care but are not guaranteed for any particular purpose. The author does not offer any warranties or representations, nor does it accept any liabilities with respect to the macros.

Preface

When I was asked to give a full-fledged TEX course in early 1987, I had to decide what my objectives were. After teaching a one day TEX seminar in 1985, I realized that to achieve anything, I needed several days of one to two hour classes.

My first thought was to use Donald Knuth's "The TEXbook" to teach the course. But after using TEX since 1981 and tutoring users for several years, I'd received numerous requests for a small "TEX User's Guide" to quickly get started on the typesetting road. The users I dealt with were usually graduate students or secretaries in Chemistry interested in spending as little time as possible to start typesetting sophisticated math formulae as well as continue to generate memos and letters using existing templates. For this reason, I decided it was time for a new approach.

Once committed to writing my own text, I had to make some objectives. My audience would be mostly secretaries with a few students (undergraduate and graduate) and faculty members. Knowing that most of the people had experience with some type of computer, I assumed prior knowledge of how to use an editor but made no assumptions about their knowledge of word-processors (not just ignorance of TEX). I also felt it was necessary to make sure they lost no standard word-processing functions needed to make simple letters, memos, and tables while exposing them to the real reason to use TEX: math equations. Finally, to pique the interest of those that absorbed all I could give them, I finished the text with some "neat" macros to get them interested in further pursuing the joys of TEX.

The outline I used for the first course used five days for a ten hour class (two hours a day). The first day covered basic TEX and how to get started and was unchanged for the final out-line. The second and third days of the original course mixed standard text processing, math, and display modes. This became confusing so the final outline uses the second day to discuss text processing and the third day to discuss math and display modes. The forth day stayed more or less the same covering alignment (tables). The fifth day was initially used to cover questions and typesetting needs not covered by the previous days text. For this reason, you may find the material covered in the last day's text to be random in nature without much flow or direction. However, the questions asked by users after the fifth day were far fewer than after the first course (and a great deal more obscure).

The font used to typeset this document, Computer Modern Roman ten point lettering magnified 44% (magstep2), was used to generate glossies for an overhead viewer. If you should happen to be appointed head TEX guru and find it necessary to give your own class, feel free to use "TEX Made Easy" as your course notes as a supplement to Donald E. Knuth's "The TEXbook". Whenever the user has a question about one of the topics in "TEX Made Easy", refer to the bottom of the page where you will find a refernce to a page in "The TEXbook" for further detailed information. Together, "TEX Made Easy" and "The TEXbook" make a beautiful pair!

The Great Zar

Acknowledgements

I wish to thank my loving mother, Dr. Mary F. Zirin, for her endless proof-reading and grammar help. My sister, Dana, should be thanked for helping me find a new name for this book. Thanks should be given to my wife Diane for her patience and love. I'd like to thank Barbara Beeton and Michael Eck for their assistance with technical errors. And finally, I wish to thank Robert Messer of Albion College for suggestions, comments, and help in detecting technical errors.

If the reader is interested in other more professional works by Zirin's, may I direct you to '*Astrophysics Of The Sun*' by Harold Zirin published by the Cambridge University Press in 1988 (ISBNs 0–521–30268–4 and 0–521–31607–3). For those not interested in seeing the light, I suggest '*The Cavalry Maiden*' by Nadezhda Durova translated by Mary Zirin published by the Indiana University Press in 1988 (ISBN 0–253–31372–4). Dare I also mention my wife's thesis on '*Calculation of* $e^- - H$ *Scattering Processes using Hyperspherical Coordinates*' by Diane Hood at the California Institute of Technology in 1986. Perhaps my sister will get into the act someday!

TEX Made Easy

Using TEX With The Plain Macro Package

Day 1

Daniel M. Zirin Instructor

What Is TEX?

TEX[1] is a computerized typesetting program which was developed by Donald E. Knuth of Stanford University. TEX's capabilities include:

- Standard text formatting such as left and right justification, automatic page breaks, paragraph indentation, proportional spacing, automatic word hyphenation, etc.
- Ligature replacements ('fifty' is replaced with 'fifty' and 'fluffier' is replaced with 'fluffier').
- Generating math expressions.
- Generating tables (also known as alignment).
- Permitting the use of macros (also known as definitions).

TEX reads a standard ASCII text file as input, formats the text with TEX commands, and generates an output file with printer instructions (known as a DVI file). TEX is a post-processor program, not a WYSIWYG program (What You See Is What You Get). There are, in addition to TEX, programs which will read a DVI file and display your formatted text on a variety of terminal screens and printers.

The Campus Computing Organization has purchased site-licenses for MicroTEX[2] (a PC-DOS version of TEX), a few DVI programs, one of which is called DVIPS[3], to generate output for the Apple Laserwriter laser printer), and PREVIEW[2] (a PC-DOS graphics screen preview program). Other versions of TEX exist for hundreds of different computers and operating systems.

---

[1] TEX is a registered trademark of the American Mathematical Society.

[2] Available from Addison-Wesley Publishing Company, Inc.

[3] Available from ArborText, Inc. (a.k.a. Textset, Inc.)

How TEX Is Used

The typical TEX user will follow the steps below when generating a paper:

Editing

1) Use an editor which generates ASCII output to create or modify a file containing the text of the paper with embedded TEX commands.

TEX

2) Leave the text editor and start processing the file with the TEX program. The TEX program will create a DVI (DeVice Independent) output file.

DVI

3a) With the DVI file and a graphics terminal screen, first process the DVI file with a *previewer* (if one is available) to check for obvious errors before generating a printed hardcopy of your formatted text. This will save you time and paper.

3b) Process the DVI file with a DVI-to-printer program. These programs typically process a DVI file and generate an additional output file your printer will understand. I will call this output file 'PINPUT'.

Print

4) Print the 'PINPUT' file created in step 3b on the desired printer.

Editing Your Input File

To start generating a TEX input file, you should reacquaint yourself with your keyboard. If you are still using a lower case 'l' to type the number '1', change today.

- **Spacing** • When TEX encounters multiple spaces, all but one space is ignored. TEX will automatically add a little extra space when a sentence ends (a lowercase letter followed by a period followed by a space signifies the end of a sentence). If a fixed number of spaces are needed, use the TEX command *backslash space* ('\ ') to generate a forced space. It is also important to know that TEX considers the end of a line (<RETURN>) to be a space.
- **Indentation** • If TEX encounters a blank line, a new indented paragraph is started automatically and the blank line removed. TEX will treat a series of blank lines like a single blank line.
- **Quotation** • When you want to quote a phrase, TEX uses and understands all three quote marks on your keyboard (left & right single quote and double quote). To generate the quoted phrase, "I couldn't understand.", you should use two left single quotes before and two right single quotes after the text of the phrase. If you need a right single quote followed by a right double quote, you should type:

  ```
  ... to leave us.'\thinspace''
  ```

  The double-quote key produces the same result as two single right quotes.
- **Hyphenation** • Hyphens, dashes, and minus signs are visibly different. To generate a **hyphen** (for compound words), type one minus key. To generate an **en-dash** (used for number ranges), type two minus keys. To generate an **em-dash** (used for sentence punctuation), type three minus keys. And finally, to generate the **minus sign** used in math formulas, type one minus key preceded and followed by a dollar sign.

Editing Your Input File (Continued)

- **The Tie** • Finally, when entering text, you should never split words across lines since TEX does this for you. TEX determines how to split lines by first trying to break a line where a space is encountered. If the result creates large gaps between words, TEX starts trying to hyphenate words. This algorithm means that TEX could split a line in the middle of a name. To keep TEX from doing this, use the tilde ('~') instead of a space where you want to discourage TEX from breaking lines. For example,

  Prof.~David A.~Miller

  will keep TEX from breaking a line between 'Prof.' and 'David' as well as between 'A.' and 'Miller'. The tilde (called a *tie*) can also be used to keep TEX from placing extra space after the period in 'Prof.' because TEX considers it the end of a sentence.

A Sample Input File

```
 1:Once upon a time, in a distant
 2:galaxy, there lived a man known as Yoda.
 3:
 4:This man
 5:was short and     ugly and had  huge
 6:warts on his
 7:nose.
 8:
 9:
10:Not a   happy   man   at   all.
```

The Result

Once upon a time, in a distant galaxy, there lived a man known as Yoda.

This man was short and ugly and had huge warts on his nose.

Not a happy man at all.

Commands

The first thing you need to learn is how to speak to TEX. TEX treats everything except TEX commands as text. A command always starts with a prefix character (a backslash) followed by a series of letters (a word) or a single non-letter.

- Word commands cannot use numbers. A word command is a backslash followed by one or more letters (a–z and A–Z).
- Word commands are case sensitive. `\Input` is not the same as `\input`.
- TEX ignores spaces after word commands. If you need a space to follow a word command, add a backslash-space (‘\ ’) or an open-curly-bracket close-curly-bracket (‘{}’).

Some examples of what TEX word commands might look like:

Command	`Word`	**Command**	`Word`
`\input`	`‘input’`	`\Bigl`	`‘Bigl’`
`\TeX`	`‘TeX’`	`\alpha123`	`‘alpha’`
`\pi^2`	`‘pi’`	`\bullet a`	`‘bullet’`
`\TeX\ it`	`‘TeX’ ‘ ’`	`\gamma{}Ef`	`‘gamma’`

Some examples of what TEX character commands might look like:

Command	`Character`	**Command**	`Character`
`\&`	`‘&’`	`\$`	`‘$’`
`\*`	`‘*’`	`\,`	`‘,’`

(See page 7 of the TEXbook[1] for more information.)

[1] Written by Donald E. Knuth and published by Addison-Wesley Publishing Co. & the American Mathematical Society.

Special Characters

As you may have already guessed, TEX assigns some characters (such as a backslash) special meanings. The following table lists all special characters and tells how to generate the character desired with TEX commands:

Sym	Special TEX Meaning	Replace With
\	Command prefix.	`$\backslash$`
{	Start of a group.	`$\{$`
}	End of a group.	`$\}$`
$	Start math or display mode.	`\$`
&	Separate columns in a table.	`\&`
#	Parameter for definitions or tables.	`\#`
^	Superscript in math mode.	`\^{}`
_	Subscript in math mode (underscore).	`\_`
%	Used for adding comments in TEX.	`\%`
~	Put a space in this location and don't break a line here.	`\~{}`
<	w/out the dollar signs, TEX makes '¡'.	`$<$`
>	w/out the dollar signs, TEX makes '¿'.	`$>$`

The symbols backslash ('\'), right & left curly bracket ('{', '}'), hat ('^'), underscore ('_'), and tilde ('~') are rarely used separately. Most of the other symbols are preceded by a backslash.

The percent sign will stop the rest of the text line from being processed. For example, typing

```
... was 75% of market value.
```

in your text file will produce '... was 75'. In this case, you should type

```
... was 75\% of market value.
```

to produce the desired result.

(See page 37 of the TEXbook for more information.)

Accents

To generate an accent above or below a character, you should precede the letter with an appropriate TEX accent command. For example, if I wanted the word

Cesàro

to have the grave accent, I would type

```
Ces\`aro
```

If I want to create

garçon

I must use the command '\c'. Since TEX cannot distinguish between a single letter command and a word command, I would have to add a space between the TEX command and the letter to be accented or add a null curly bracket, as in:

```
gar\c con
```

or

```
gar\c{}con
```

or

```
gar\c{c}on
```

If I attempted '`gar\ccon`', TEX would try to determine what the command '`\ccon`' means and cry bloody murder.

You may also come across another tricky situation involving accenting 'i' or 'j'. To accent these letters, you should remove the dot from the letter. This can be accomplished by using the TEX commands '\i' and '\j' which produce 'ı' and 'ȷ'.

(See page 52 of the TEXbook for more information.)

Symbol And Accent Commands

Accents

type	*to get*	
\‘o	ò	(grave accent)
\’o	ó	(acute accent)
\^o	ô	(circumflex or ‘hat’)
\"o	ö	(umlaut or dieresis)
\~o	õ	(tilde, ‘squiggle’, or ‘snook’)
\=o	ō	(macron or ‘bar’)
\.o	ȯ	(dot accent)
\u o	ŏ	(breve accent)
\v o	ǒ	(háček or ‘check’)
\H o	ő	(long Hungarian umlaut)
\t oo	o͡o	(tie-after accent)
\c o	o̧	(cedilla accent)
\d o	ọ	(dot-under accent)
\b o	o̲	(bar-under accent)

Special Letters

type	*to get*	
\oe,\OE	œ,Œ	(French ligature OE)
\ae,\AE	æ,Æ	(Latin and Scandinavian ligature AE)
\aa,\AA	å,Å	(Scandinavian A-with-circle)
\o,\O	ø,Ø	(Scandinavian O-with-slash)
\l,\L	ł,Ł	(Polish suppressed-L)
\ss	ß	(German ‘es-zet’ or sharp S)
\i,\j	ı,ȷ	(undotted ‘i’ and ‘j’)
!‘,?‘	¡,¿	(upsidedown bang & question mark)
\dag	†	(dagger or obelisk)
\ddag	‡	(double dagger or diesis)
\S	§	(section number sign)
\P	¶	(paragraph sign or pilcrow)
{\it\$}	*£*	(English pounds sign)

Simple Font Changes

TEX starts generating letters of your text in Computer Modern Roman ten point or CMR10 (similar to Times Roman font). Ten-point font means the letters will be limited to ten points in size from top to bottom (this is a bit more than one-seventh inch). TEX has commands to change this font to a different style:

- \rm • This resets the font back to the default setting: Roman 10 point (CMR10).
- **\bf • This sets the current font to Boldface eXtended roman 10 point (CMBX10).**
- *\it • This sets the current font to Text Italics roman 10 point (CMTI10).*
- *\sl • This sets the current font to SLanted roman 10 point (CMSL10).*
- `\tt • This sets the current font to try and imitate your typewriter at 10 point (CMTT10).`

With the '\it' and '\sl' fonts there is an extra command '\/' (called an italic correction) which adds a tiny space when leaving italics or slant and entering a non-slanted font. For example:

This changes the flavor of things.

The above line switched from slant to roman between the letter 'f' and 'l' in the word flavor and as a result, the letters slightly overlap. To fix, type the line

```
{\sl This changes the f\/}lavor of things.
```

to produce:

*This changes the f*lavor of things.

(See page 13 of the TEXbook for more information.)

More About Fonts

Loading Different Fonts

If you want to make use of more fonts in your documents, use the '\font' command as follows:

`\font\`*name*`=`*font*

The 'name' field can be any series of letters you want to use to identify the new font. The 'font' field must be a valid font name of a character font that comes with your TEX software package. When I discussed the five basic fonts, I hinted at a few valid font names ('\rm' is called CMR10, '\bf' is called CMBX10, '\it' is called CMTI10, '\sl' is called CMSL10, and '\tt' is called CMTT10). In the MicroTEX software package, you will find a list of all valid font names in the directory '\TEX\FONTS'.

Once you have typed a '\font' command, you may use the font by typing a backslash followed by the 'name' identification. For example:

```
\font\smallfive=cmr5
{\smallfive This will be tiny.}
```

Magnification

If you find the final hardcopy output from TEX too small to read, use the '\magnification' command at line one in your text file. The format is:

`\magnification=`*value*

The 'value' field is equal to '\magstep0' by default. If you want larger text, increase '\magstep0'. For example:

```
\magnification=\magstep2
```

Valid magsteps are 0, 1, 2, 3, 4, 5, and `\magstephalf` but using anything other than '\magstep0' requires special font files that may not exist in your TEX software package.

If you use '\magnification', in addition to your text, all dimensions in your document not specified as 'true' will be magnified.

More About Fonts (Continued)

Loading Different Sized Fonts

There is another way to load fonts with the '\font' command:

`\font\`*name*`=`*font* `scaled` *mag*

This form of the '\font' command allows you to load a magnified font without magnifying your entire file. For example:

```
\font\bigbold=cmbx10 scaled \magstep2
{\bigbold This will be very large.}
```

Again, if you load magnified fonts larger than '\magstep0', you may not have the needed font files in your TEX software package.

If you use '\magnification' at the top of your file and use a '\font' with 'scaled', the new font will be loaded at the combined magnifications. For example, if you magnify your entire file at magstep1 and load a font scaled magstep2, the new font will be loaded at magstep3 (1+2).

Magnification Sizes Using 10 Point Lettering

This is typeset using magstep0.

This is typeset using magstephalf.

This is typeset using magstep1.

This is typeset using magstep2.

This is typeset using magstep3.

This is typeset using magstep4.

This is typeset using magstep5.

(See pages 16 and 153 of the TEXbook for more information.)

Dimensions

Dimensions in TEX can be specified in a variety of measuring units. The below table should help explain each possible TEX dimension:

Unit	Meaning	Unit/Inch
pt	point	72.27
in	inch	1
cm	centimeter	2.54
mm	millimeter	25.4
pc	pica	6.023
dd	didôt point	67.54
bp	big point	72
sp	scaled point	4736086.72
em	≈width of 'M' in current font	(varies)
ex	≈height of 'x' in current font	(varies)
mu	math unit	(≈18em)
	mu can only be used in math mode.	

You may precede a dimension with an optional plus ('+') or minus ('−') sign. Dimensions do not have to use decimal points. You can also precede a dimension name with the word 'true' (for example '10.125truebp'). Use of 'true' dimensions overrides the effect of using '\magnification' at the top of your file. The maximum size of a dimension is approximately 18.9 feet, so you shouldn't have to worry about making a page too big for TEX.

(See page 57 of the TEXbook for more information.)

Page Boundaries And Margins

When TEX starts reading your text file, it makes some assumptions about generating page boundaries. TEX has four page-boundary commands to change the default settings:

- **\hoffset** • This command defines the left margin of your formatted text. The default setting is 0 (no left margin).
- **\voffset** • This command defines the top margin of your formatted text. The default setting is 0 (no top margin).
- **\hsize** • This command defines the printed width of your formatted text excluding the left margin. The default setting is 6.5 inches.
- **\vsize** • This command defines the printed length of your formatted text excluding the top margin. The default setting is 8.9 inches.

Unless these commands appear before text at the top of your file, a page-boundary change will go into effect after the current page has been formatted. Normally, you should never reset these commands. Margin settings should be changed instead (explained on page 31).

To change the default settings for any of these commands, type the following at the top of your file

name=*dimension* (preferred)

or

name *dimension*

where *name* is the command you wish to change and *dimension* is the setting you wish to apply to *name*. For example:

```
\hoffset=7.0in
\vsize 10.5truein
```

(See page 251 of the TEXbook for more information.)

Parameters And Grouping

Some TEX commands require text parameters (not dimensions). If a TEX command requires a text parameter, TEX takes the first character following the command unless the character is a 'start group' character (a left curly bracket), in which case TEX assigns the text between grouping characters to the text parameter. An example, please...

```
\centerline A letter.
```

The TEX command '\centerline' takes the required text parameter and centers it on the current line. In this example, TEX only centers the letter 'A' on the current line and starts a new line with an indented paragraph beginning 'letter.' (the space after '\centerline' isn't centered because TEX ignores spaces after commands). To center the entire phrase 'A letter.' on the current line, use the grouping characters to form the text parameter for '\centerline' as follows:

```
\centerline{A letter.}
```

Grouping characters have other uses as well. In some previous examples we used a null group ('{}') as a method for separating TEX commands and the text following a command. The example

```
\bf{}Boldface
```

separates the command '\bf' (change into the boldface font) from the text 'Boldface'. Another example of grouping would be:

```
\sl This is slanted.\/  {\rm But this isn't.} But this is.
```

Here we start in the slanted font, and set up a group for the roman text "But this isn't." When we exit the group, TEX automatically restores the previous settings and the slanted font typesets "But this is." Here is the result:

This is slanted. But this isn't. *But this is.*

(Also, note the proper use of an italic correction after the word 'slanted.')

(See page 19 of the TEXbook for more information.)

Let's Start TEXing

During this course, I will describe how to use TEX on an IBM PC with an Apple LaserWriter. We will use the sample file on page 6 as our test file and name it 'sample.tex'. In the example below, all typewriter text will be what the PC displays on your screen and all slanted text will be what you type and send to the computer.

```
C:\USER> tex sample
MicroTeX Version...
Copyright (c) 1986 by Addison-Wesley Publishing...

This is TeX, PC-DOS...(preloaded format=plain...)
(c:\user\sample.tex)
*\end
[1]
Output written on c:\user\sample.dvi (1 page...)
Transcript written on c:\user\sample.log.
237Kbytes remaining

c:\USER>
```

On the first and last line, 'C:\USER>' is the prompt from the PC. On the first line we typed 'tex sample' to start the MicroTEX program and process the file 'sample.tex' (we don't need to indicate the full name of the file if the file-name suffix is '.tex'). On line six, TEX tells us that the file 'c:\user\sample.tex' was completely processed and finally, TEX prompted us with an asterisk ('*'). This is because the file 'sample.tex' didn't have the TEX command '\end' signifying the end of all processing. Therefore, we enter the '\end' command now, and TEX finishes processing and returns us to the PC prompt. On line nine, TEX tells us the name of the output DVI file 'sample.dvi' which will be used by a *previewer* or DVI-to-printer program later.

Let's Start TEXing (Continued)

In addition to telling us about our new DVI file, TEX also tells us about a 'transcript' file called 'sample.log'. The transcript file normally contains a list of errors TEX found and any action you took to correct those errors. In our example, the transcript file would contain lines five through nine (including the text we entered at the asterisk prompt).

There is one more line in the transcript file which doesn't appear in the output you see on your screen. That extra line appears between line one and line two on the previous page and contains '**sample'. The double asterisk is a prompt from TEX for an input file, and 'sample' is our response. Let's suppose that we typed 'TEX' at the PC prompt. TEX wouldn't know what file to process so TEX would prompt with '**' for an input file. We could then type 'sample' or 'sample.tex' and TEX would proceed normally. You may encounter this in the future if you make a typo to start TEX. For example, typing 'TEX SAMPEL' at the PC prompt would cause TEX to look for a file that doesn't exist. TEX would then display a message indicating a file 'SAMPEL.TEX' wasn't found and prompt you for a new or different file-name.

When we entered the TEX command '\end' at the TEX prompt '*', we could have typed any TEX command or text. Until TEX sees the command '\end', TEX will continue processing text by prompting with '*'. This can be useful if you have separated your paper into separate files containing different sections. You could start TEX with a command like 'TEX FILE1'; then, after TEX finishes processing FILE1 and prompts with '*', you proceed to issue multiple '\input' commands for each separate file you wish to process in addition to FILE1. When you finished, you would stop TEX with the '\end' command.

Let's Start TEXing (Continued)

It is also possible to generate a separate file containing '\input' commands as in the following:

```
1:\hoffset=1.5in
2:\hsize=5.5in
3:\voffset=1.0in
4:\vsize=9.0in
5:\input toc
6:\input chap1
7:%\input chap2
8:\input epilog
9:\end
```

With a file like this, we have a simple method for generating a large paper or manuscript. In line 7, we are temporarily not processing the file 'chap2' by using the TEX comment character. When we finish typing chapter 2, the percent sign will be removed.

Finally, let us suppose that in the above file, line two reads '\hszie=5.5in'. When TEX reads this typo, you would see the following:

```
! Undefined control sequence.
1.2 \hszie
          =5.5in
?
```

The first line tells us what type of error occurred. This error message means there is no such TEX command '\hszie'. The second line of the message tells us the line number of this error in our file. The third line, shifted to the right, contains the rest of the line that TEX has not processed. The question mark at the end of the error is the error prompt from TEX.

Let Start TEXing (Continued)

Typing a question mark in response to the TEX question mark prompt following an error displays a list of acceptable replies to the question mark prompt. Here is that display:

```
?
Type <return> to proceed, S to scroll future error
                                                messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
H for help, X to quit.
?
```

For starters, lets reply with an 'H' for help with our error message:

```
?H
The control sequence at the end of the top line
of your error message was never \def'ed. If you have
misspelled it (e.g., `\hobx'), type `I' and the correct
spelling (e.g., `I\hbox'). Otherwise just continue,
and I'll forget about whatever was undefined.
?
```

TEX is making this easy for us to correct. To fix our error concerning '\hszie', we simply type 'I\hsize' at the question mark prompt and TEX will be happy. **However**, correcting the error using the 'I' command does **not** correct our file. We will have to note the line number and re-edit our text file to fix the error permanently in our input file.

If we chose not to correct this error message and continue by typing the return key, TEX would remove the text '\hszie', and continue with the rest of our file. The result would cause our output not to have the correct '\hsize' and, in addition, the output would show the characters '=5.5in' at the top of our file as the start of the first indented paragraph.

(See page 23 of the TEXbook for more information.)

Generating Output

After a DVI output file is created with the TEX program, we can display the DVI file on our screen with the PC command 'PREVIEW' as follows:

C:\USER> *PREVIEW SAMPLE*

Once the PREVIEW program starts, you can send commands to the program simply by pressing a single key (no need to press the return key following the command). Some commands are:

- ? • Display help. Use the space bar to scan through the help display. After three pages, the help display automatically returns you to your formatted text.
- m • Magnify display and center display at the location containing the plus or hash marks.
- s • Shrink display.
- (arrows) • The arrow keys on your keyboard move the plus or hash mark to allow you to magnify the display at different locations.
- l • Move display to the left.
- r • Move display to the right.
- u • Move display up.
- d • Move display down.
- q • Quit.

Generating Output (Continued)

If you are happy with your output and want a hardcopy version, use the 'DVIPS' PC command to read your DVI file and produce a PostScript output file (a PS file). The PS file can then be sent to the local LaserWriter[1] laser printer with the PC command 'SPR'. Here is an example:

```
c:\USER> DVIPS SAMPLE
DVILASER/PS IBM-PC Version...
Copyright (c) 1985 by Textset, Inc...
TeX output...
DVILASER> <Return>
 Doing [1]. (end)

DVILASER Job Summary:

  Pages processed:  1
  Maximum paper width required:  5.72in
  Maximum paper height required:  10.24in
  No character overruns.
  Output written on sample.ps, 957 bytes.

c:\USER> SPR SAMPLE.PS
```

On line one, we start the DVIPS program. On line five, we were asked to type the return key. And finally, on the last line, we ask the PC to print the PS file.

[1] LaserWriter is a registered trademark of Apple Computer, Inc.

TEX Made Easy

Using TEX With The Plain Macro Package

Day 2

Daniel M. Zirin Instructor

Movement

TEX provides the commands '\hskip', to move across a line, and '\vskip', to move down the page. To use them, type:

`\hskip` *dimension*
`\vskip` *dimension*

For example, typing

```
\hskip 2truein This starts two inches to the
right.\par
\vskip 40truept
This line is 40 points below the last line.\par
```

would generate:

This starts two inches to the right.

This line is 40 points below the last line.

In the first line, the '\hskip' is actually two inches plus the automatic paragraph indentation of 20 points. To skip exactly two inches across the line at the start of a paragraph, precede '\hskip' with '\noindent'.

TEX also has three more commands for vertical movement called '\smallskip' (≈3 points), '\medskip' (≈6 points), and '\bigskip' (≈12 points). These values are approximately $\frac{1}{3}$, $\frac{1}{2}$, and 1 line height respectively.

(See page 70 of the TEXbook for more information.)

Paragraphs

At the beginning of the course, I discussed how to generate paragraphs by using blank lines. There is another space-saving way to signify the end of a paragraph by using the TEX command '\par'. For example:

```
This is a paragraph.\par This is another.\par
```

This example makes two paragraphs on one text line in your file.

When TEX first starts reading your input file, '\vskip' commands are ignored until a printable character is generated. This poses a problem if you want a chapter or title page generated with a few blank lines at the top of the page. To get around this problem, you can generate a printable space at the top of the page before any '\vskip' commands by using the '\null' command. The '\null' command, however, starts a blank paragraph and requires a '\par' to follow to indicate the end of a paragraph. Here is an example of a title page:

```
\null\par
\vskip 2in
\centerline{\bf The TeX Talk}
```

The example places an empty paragraph at the top of the page, moves down two inches and displays our title centered in boldface print.

Once TEX starts a new paragraph, the paragraph **must** end before a vertical move command (like '\vskip') can be used. That is why the above example contained a '\par' after '\null'. We could have also placed a blank line after the '\null' instead of '\par'. When TEX generates a paragraph, TEX considers itself in *horizontal mode*.

Paragraphs (Continued)

If you want to skip an extra line or two, before starting a new paragraph, use the '\parskip' command as follows:

`\parskip=`*dimension*

By default, '\parskip' is set to 0pt. Note that it works in addition to any line spacing. For example

```
\parskip=12pt
```

will skip 12 points of space down the page, in addition to the spacing inserted after the previous line, before starting a new paragraph.

Items Of Interest

When you want to make several points in a paper, you may wish to indent your separate descriptions and precede the beginning of each description with a number or letter sequence. For example:

---

1. This is the first of several items that are long and boring to the typical human.
   a) Ear wax.
   b) Taxes.
2. These next few cases are excluded for your protection.

The above was made with the '\item' and '\itemitem' commands. These two commands take a required text argument to be placed before the start of the paragraph. The above was made with the following:

```
\item{1.} This is ... human.\par
\itemitem{a)} Ear wax.\par
\itemitem{b)} Taxes.\par
\item{2.} These next few ... protection.\par
```

The text argument can be anything you want. These two commands also use another command '\parindent' to determine the amount of space to indent or double indent each paragraph. If we typed '\parindent=1in' before the above lines, the result would be:

---

1. This is the first of several items that are long and boring to the typical human.
   a) Ear wax.
   b) Taxes.
2. These next few cases are excluded for your protection.

Items Of Interest (Continued)

TEX uses commands '\hangindent' and '\hangafter' to generate the '\item' command formatting process. The dimension '\hangindent' specifies the amount to indent the next paragraph and the variable '\hangafter' determines when '\hangindent' takes effect in the current paragraph. If '\hangafter' is a positive integer, the paragraph lines will start indenting on line number '\hangafter'. Otherwise, the paragraph indentation will end on the line number equal to the absolute value of '\hangafter'. When each paragraph ends, the values of '\hangindent' and '\hangafter' are automatically reset to '0pt' and '1' respectively.

(See page 102 of the TEXbook for more information.)

Margins And Referencing

Suppose you want to quote a long passage. TEX supplies the command '\narrower'. Using '\narrower' causes the left and right margins to be increased by the size of '\parindent'. For example:

This text is using the '\narrower' command. It helps the TEXpert generate beautiful passages in the middle of a manuscript.

The above example was made with the following:

```
{\narrower This text is ...  a manuscript.\par}
```

Changing '\parindent=0.5in' before '\narrower' gives us narrower width and more paragraph indentation...

This text is using the '\narrower' command. It helps the TEXpert generate beautiful passages in the middle of a manuscript.

We can use a large '\parindent' and place a name in the text argument of '\item' to generate references, but this method gives us a staggered left column of names:

Smith, Maile E., Modern Poetry In Physical Chemistry, Cambridge Press, 1985.

Thompson, Butler A., Semi-sweet Coffee From The West Indies, New York Tribune, 1987.

To make references start in the same column, we use '\narrower' and '\parindent' in the following manner:

```
{\narrower\narrower\parindent=-40pt
$\phantom{0}$1) Smith, Maile E., Modern...\par}
```

And the result is...

Margins And Referencing (Continued)

---

1) Smith, Maile E., Modern Poetry In Physical Chemistry, Cambridge Press, 1985.
12) Thompson, Butler A., Semi-sweet Coffee From The West Indies, New York Tribune, 1987.

The command '`\phantom{0}`' was used to leave a space (equal to the width of the character '`0`') to line up with reference numbers greater than nine (another method would be to use the tie character '`~`'). This isn't quite what we want because we increased the right margin also. Instead of using '\narrower', we will use '\leftskip' to change only the left margin as follows:

```
{\leftskip=40pt\parindent=-40pt
$\phantom{0}$1) Smith, Maile E., Modern...\par}
```

And we have reached our objective:

---

1) Smith, Maile E., Modern Poetry In Physical Chemistry, Cambridge Press, 1985.
12) Thompson, Butler A., Semi-sweet Coffee From The West Indies, New York Tribune, 1987.

In the same fashion, you may also use '\rightskip' when you want to increase the right margin.

(See page 100 of the TEXbook for more information.)

Footnotes

To generate a footnote in TEX, use the '\footnote' command. The '\footnote' command requires two text arguments. For example:

```
\footnote{ text1 }{ text2 }
```

The first text argument is the symbol to place at the current location in the current paragraph and at the beginning of the footnote at the bottom of the page. The second text argument is the text to be placed at the bottom of the page.* You will see the result of the footnote generated before the start of this sentence at the bottom of the page. It was generated with the following:

```
page.\footnote{*}{This is ...  command.}
```

(See page 116 of the TEXbook for more information.)

* This is a sample footnote for those interested in the TEX '\footnote' command.

Filler And Boxes

When TEX processes your document, all characters, words, lines, paragraphs, and pages are considered to be different size boxes. Where there are no boxes (blank spaces), TEX considers there to be glue or rubber. Take a look at this page for example. The page including margins is a box. Within that box, the top, bottom, left, and right margins are four more boxes containing glue. The inner box surrounded by margin boxes is another box. Each line containing text is a long rectangle box. In between these line boxes are glue. Inside of the line boxes are lots of tiny boxes containing word boxes. In between each pair of word boxes is more glue. And finally, inside the word boxes are the smallest boxes containing a letter.

Its important to know this to explain how the TEX filler commands '\vfill' and '\hfill' work. For example, when TEX read the first line of this page, the '\centerline' command told TEX the following:

```
\line{\hfill\bf Filler And Boxes\hfill}
```

The '\line' tells TEX to generate a line (a long rectangle box) containing the first required text argument. Inside the text argument the phrase 'Filler And Boxes' was preceded and followed by '\hfill' commands. The properties of the command '\hfill' are:

1) To push any text on either side of the '\hfill' in the opposite direction;
2) To put blank space in the area left open by pushing the neighboring text.

Therefore, the first '\hfill' pushes 'Filler And Boxes' to the right and the second pushes this phrase to the left with equal force. This caused our text to be centered. Finally, the '\hfill' commands put blank space in the area to the left and right of our centered text.

Filler And Boxes (Continued)

In addition to '\hfill', there is another command called '\vfill' which has the following attributes:

1) It cannot be used in horizontal mode.
2) It pushes lines above and below in opposite directions.
3) It puts blank space in the vacancy made by moving neighboring lines.

This command **must** be used at the end of your document to fill the last page with blank space and make TEX happy.

If you want to force TEX to make a new page, you will also need a '\vfill' followed by an '\eject' command. For example:

```
This is on page one.\par\vfill\eject
This isn't.\par\vfill\end
```

The text "This is on page one." will be on page one, the text "This isn't." will be on page two and the document will end after the word "isn't."

You can't use '\hfill' and '\vfill' just anywhere. These commands can only be used when they are restricted inside of a box. That is to say, '\hfill' can only be used in a line box, and '\vfill' can only be used in a column box or at the end of a page where you will use '\eject' or '\end'.

Both '\hfill' and '\vfill' have close relatives. The '\hfil' and '\vfil' commands do the same thing with infinitely less strength (if you precede a word with '\hfil' and follow the word with '\hfill', '\hfill' will overpower '\hfil' and the result will push the word to the left margin). There is also '\hfilll' to push with infinitely more strength of '\hfill' (there is no '\vfilll').

The commands '\hss' and '\vss' shrink or pull on neighboring text. Normally this means they remove blank space, but if you precede and follow a word by '\hss', the result will center the word and add blank space in the vacancies to the left and right.

(See page 71 of the TEXbook for more information.)

Page Numbering

When TEX generates a page, it leaves some room at the top and bottom of a page for a page number. By default, TEX makes page numbers centered at the bottom of the page. To change this format, use the following commands:

\headline This is the command that reserves the top line of each page intended for a possible page number or other heading. By default, this is empty.

\footline This is the command that reserves the bottom line of each page intended for a possible page number or other footing. By default, '\footline' is set equal to '\hss\folio\hss'.

\folio This is the command that generates a text version of the current page number.

\pageno This command can be used to change a page number within your document.

\nopagenumbers This command removes page numbering from the bottom of the page.

The location of a page number should be changed before the first character for the current page is seen by TEX (usually after an '\eject' or at the top of your text file).

To change '\headline', type '`\headline={ whatever }`'. If you define '\headline' to be '\hfill\folio', the page number will appear at the top right corner of each page. If you move the '\hfill' to the righthand side of '\folio', the page number will be displayed at the top left corner of each page. Typing

```
\headline={\hfill}
```

removes the page number from the top of the page altogether (this is the default). If you change '\headline', you should change '\footline' as well, or the page number will appear at the top and bottom of each page. If you don't precede '\folio' with '\rm', the page number will be printed in the font in use at the bottom of the page.

Page Numbering (Continued)

To change '\footline', use:

```
\footline={ whatever }
```

The same rules apply to '\footline' as '\headline'. The default setting for '\footline' is to center the page number (exact text is shown on the previous page). Setting '\footline' to '\hfill' removes the page number from the bottom of the page.

The '\nopagenumbers' command sets the '\footline' command equal to '\hfill', thus removing the page number from the bottom of the page.

To change the page number of the next page, use the command:

\pageno=*number*

If the number is negative, roman numerals will be used to number the pages. Here are two examples:

```
\pageno=12
\pageno=-6
```

In the first example, the page number will contain '12', the following will be '13', and so on. For the second example, the page number will be 'vi', the following page will be 'vii', and so on.

The following text was typed to generate the page numbering format of the page you are reading now:

```
1: \multiply\pageno by -1
2: \headline={\hss(\rm\folio)\hss}
3: \footline={\hfil- \rm\folio\  -\hfill}
```

The text on line one could also be '`\pageno=-\pageno`' to accomplish the same thing.

(See page 252 of the TEXbook for more information.)

Making A Table Of Contents

Now that you know about filler commands, you can make a simple TOC (Table Of Contents). To make our sample TOC, we will use the commands '\centerline', '\line', '\dotfill' and '\hrulefill'. Let me explain these separately:

- \centerline • This centers a line of text. The text to be centered is the required argument for this command.
- \line • This creates a line box. The required text argument is forced into a box that must be contained on a single line.
- \dotfill • This does the same thing as '\hfill' except it fills the area left blank with a row of periods.
- \hrulefill • This does the same thing as '\hfill' except it fills the area left blank with a solid line across the baseline.

Here is our text file:

```
1:\centerline{\bf Table Of Contents}
2:\null\par
3:\noindent Day 1\par
4:\line{\indent Using Fonts \dotfill\  1}
5:\line{\indent Using Math \dotfill\  2}
6:\line{\indent Paragraphs \hrulefill\  3}
```

The '\null\par' is used to move down the page one line. Note that we could not use either fill command without the use of '\line'. And now, the result:

Table Of Contents

Day 1

Using Fonts 1
Using Math 2
Paragraphs _____ 3

(See page 223 of the TEXbook for more information.)

Line Spacing

When TEX generates characters or letters, the characters are placed on a baseline to make words of a line appear to sit on a horizontal string. Let's take the letter 'M' for example. The legs of this letter begin on the baseline and extend up the length of the character height boundary. If we look at the letter 'g', we have a different situation. This character hangs a bit below the baseline. The top bowl of the letter 'g' extends above the baseline while the bottom bowl of the letter 'g' starts at the baseline and extends below the baseline. The length from the baseline. The distance from the baseline to the lowest point of the letter 'g' is called the depth of the character 'g'.

Each pair of baselines in TEX are separated by the value of '\baselineskip'. If the interline spacing is less than the value of '\lineskiplimit', then the value of '\lineskip' is used to separate the maximum depth of the previous line from the maximum height of the current line.

In short, the value of '\baselineskip' determines the spacing between lines and '\lineskip' is a backup spacing in case you generate a peculiarly large character. The default setting for '\baselineskip' is '12pt', the default setting for '\lineskiplimit' is '0pt', and the default setting for '\lineskip' is '1pt'.

To change these values for any of these three commands, use:

`\baselineskip=`*dimension*
`\lineskiplimit=`*dimension*
`\lineskip=`*dimension*

A good double-space setting for these commands would be to change '\baselineskip' as follows:

```
\baselineskip=18pt
```

(See page 78 of the TEXbook for more information.)

Definitions

I have been using the term 'commands' to mean all TEX control words and control letters. This is not exactly true. TEX has a small set of primitive commands. All other control words and control letters are definitions (also called macros). For example, the '\par' control word is a TEX primitive command, while '\centerline' is a definition that combines control words to perform the task of centering a line of text (see page 33). In fact, many of the control words in the '\centerline' definition reference other control words which are also definitions.

The simplest form of a definition is:

`\def\`*name*`{ whatever }`

Let's take, for example, the '\nopagenumbers' definition:

```
\def\nopagenumbers{\footline={\hfill}}
```

The text following '\def' is the definition name and can be any series of letters or a single non-letter. A definition name must conform to the rules for valid TEX commands explained on page 7. Because of this, you cannot mix letters and numbers in a definition name (this is a common error). The text enclosed in the grouping characters after the definition name can be anything you please. For example, if a person's name appears several times in your document and the name is long or has many accents, you can make your own definition to generate the person's name as follows:

```
\def\zar{Mr.~D\'ani\u el M.~Z\H\i rin}
```

Once I have typed the above line in my text file, every future occurrence of '\zar' will generate:

Mr. Dániěl M. Zűrin

You must be careful to type '\zar\ ' or '\zar{} ' if you want a space to follow the text generated by your new control word. Definitions are extremely useful and should be utilized as much as possible.

(See page 199 of the TEXbook for more information.)

Making Definitions With Arguments

To make a definition (macro) using the '\def' command, use the following format:

```
\def\name#1#2#3...#9{ whatever }
```

I will make a simple example to explain how this form of '\def' works:

```
\def\zar#1#2{This is #1 number #2.}
```

The name of this definition (command) is '\zar'. The text '#1#2' tells TEX there are two required text arguments for this definition. The left curly bracket starts the definition and text argument number 1 is placed where '#1' is located in the definition. The second text argument is placed where '#2' is located and the right curly bracket ends the definition.

If I were to type '\zar{reference}{12.3}', TEX would generate 'This is reference number 12.3.'.

(See page 200 of the TEXbook for more information.)

Word Hyphenation

If a line within a paragraph is too long or short because a word or characters at the end of the line cannot be hyphenated, TEX will complain about an underfull hbox (indicating the line is too short and has some extra large spacing) or an overfull hbox (indicating the line is too long and ends in the right margin). To fix this, try one of the following:

1) Use a discretionary hyphen in the middle of the word at the end of the overfull line or at the beginning of the line following the underfull line. The discretionary hyphen command ('\-') is placed in the middle of the word between two letters where you would allow hyphenation. For example, typing 'dif\-fi\-cult' would allow TEX to hyphenate the word 'difficult' between the two 'f's and between the letters 'ic'. [1] If a line break is not needed in the middle of 'difficult', no hyphens will appear.
2) Remove one or more words prior to the problem line.
3) Rephrase part of the sentence prior to the problem line.

If you want to make TEX hyphenate a word differently, use the '\hyphenation' command as follows:

```
\hyphenation{ di-f-fi-cult table }
```

You may place any number of words inside the '\hyphenation' command text argument as long as each word is separated by a space. If you don't put hyphens in a word, TEX will never hyphenate that word.

If you don't want TEX to hyphenate any words, type:

```
\pretolerance=10000
```

This may cause some large spacing between words in lines of your document. By default, '\pretolerance=100'.

(See pages 95, 96, and 452 of the TEXbook for more info.)

[1] A discretionary hyphen will suppress possible ligaturing.

The Dreaded Black Box And Spacing

When TEX can't break a line and has to format text or math in the right column, an ugly black box will appear in your output at the end of the line (in the right margin). There are various ways to fix this problem:

1) If the line is strictly text in a paragraph, a discretionary hyphen can be used to allow TEX to hyphenate a word in a place it normally wouldn't.
2) TEX tries to force lines to line up within 0.1 point. You can change this strict setting with the '\hfuzz' command (as in '\hfuzz=1pt' for example).
3) If the line is a long math equation, split the equation into two lines.
4) Reword or rearrange the words in the sentence containing the black box.
5) Remove the ugly black box altogether with the '\overfullrule' command. Typing '\overfullrule=0pt' makes the box 0 points wide (non-existent). By default, the box is 5 points wide (this method will not cure the problem line).

If you want french style spacing in your document, you can use the '\frenchspacing' command ('\nonfrenchspacing' is the default). If you use '\frenchspacing', no extra space is added after punctuation.

If you want a staggered or ragged right margin, use the '\raggedright' command.

(See pages 30 and 307 of the TEXbook for more information.)

TEX Made Easy

Using TEX With The Plain Macro Package

Day 3

Daniel M. Zirin Instructor

Math Mode And Greek Lettering

When TEX sees a dollar sign (the *math-mode* shift character), TEX enters *math mode*. In *math mode*, all roman and lowercase greek letters are italicized while other characters are not. In addition, all spaces (not just spaces after control words) are ignored. When a second dollar sign is encountered, TEX exits *math mode* and returns to *horizontal mode*. For example, typing the next two lines

```
$x  +y/  (x-  y  )$
 $\delta t  =t1-t2$
```

would generate the following two lines:

$$x + y/(x - y)$$
$$\delta t = t1 - t2$$

Note, however, the space following '\delta' was needed to separate the control word from the following letter 't' (the space was ignored otherwise). If you want to force extra space in a math formula, use the backslash space ('\ ') control sequence.

As you may have guessed from the previous paragraph, you generate greek letters by entering *math mode* and typing a backslash followed by the word for the greek letter desired. For some letters, capitalizing the first letter of the greek word will generate uppercase greek letters.

If you want a roman letter unitalicized, use the '\rm' command. For example

```
$ This\ is\ italics.\ {\rm This\ isn't.}$
```

would produce:

This is italics. This isn′t.

Notice that the right quote mark used in "isn't" becomes a prime in *math mode*. The left quote would not be changed. If you want a capital greek letter to be italicized (by default, it isn't), use the '\mit' (math italics) command in the same manner.

(See page 127 of the TEXbook for more information.)

Greek Letter Commands

Here is the full list of greek letter commands.

Lowercase Greek Letters

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

Uppercase Greek Letters

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

(Other capital greek letters are found in the standard roman alphabet.)

For a large list of math mode symbols, see Appendix A.

Making Simple Displayed Equations

In addition to math mode, there is something called *display mode*. It works exactly the same as math mode except that the equation is separated from the text and centered on its own line. To enter *display mode*, type two successive dollar signs, enter your equation, then type two successive dollar signs to leave *display mode*. For example, if I type '$ x - y = z $' right here, I get '$x - y = z$'. If I type '$$ x - y = z $$', I will see '

$$x - y = z$$

'. Notice that the paragraph never stops and my single quotes before and after the equation are left dangling by themselves (I do this only to illustrate how *display mode* works).

There are a few special symbols you may use in math or *display mode*. To superscript, use the hat symbol ('^'). To subscript, use the underscore character ('_'). If you need to use both superscripts and subscripts, TEX doesn't care which comes first. For example, typing '$ x^{22}_5 $' and '$ x_{5}^{22} $' produce the same result x_5^{22}. In the first example, I didn't enclose the number '5' in a group because I was only subscripting a single character. It's a good idea to get used to using groups in order to avoid trouble.

If you use large operators, such as integrals and summations, you will see a difference in their appearance between math and *display mode*. For example, if I type '$ \sum_{n=1}^m and \int_{-\infty}^{\infty} $', TEX produces '$\sum_{n=1}^{m} and \int_{-\infty}^{\infty}$'. If I use *display mode*,

$$\sum_{n=1}^{m} and \int_{-\infty}^{\infty}.$$

(See page 139 of the TEXbook for more information.)

Table Of Contents

More About Math

If you need to make '...' (ellipsis), typing three periods in a row would give you '...' (closer together). Instead, you should enter math mode and type '\ldots'. Math mode also has a '\cdots' for centered dots (as in '$\cdots$') and '\cdot' for a single centered dot.

When generating parentheses, braces, and other variably sized symbols, TEX automatically increases or decreases the symbol size for you. Here is an examples:

```
$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}$$
```

generates

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}$$

In the next example, I will be using '\left(' for left parentheses and '\right)' for right parentheses. The rest of the commands needed should be in your TEX vocabulary.

$$\left(\sum_{\sum_{n=1}^{m}}^{\sum_{n=1}^{m}}\left((1+x)\int_{-\infty}^{\infty}\right)\right)$$

The parentheses could also have been replaced with other delimiters such as brackets ('[' and ']').

Refer to appendix A for a list of available symbols.

Making Fractions

When you want to make a fraction in TEX, use the control sequence '\over' by typing '${1\over 2}+x$' to produce '$\frac{1}{2}+x$'.

If you don't want a line between the top and bottom entities, use the '\atop' or '\choose' commands. For example, typing '$ {1 \atop 2} + x $' would give you '$\binom{1}{2}+x$'. '\choose' removes the dividing line and encloses the fraction in parentheses.

If you want to specify the thickness of the line between the top and bottom entities, use the '\above' command as in '`${1+x\above 3pt 2}+y$`' to generate '$\frac{1+x}{2}+y$'

When TEX typesets a subscript, the characters subscripted are generated in a different font style. There are four math styles: *display style* which generates most characters identical to text style except for large operators; *text style* ($x+y$); *script style* used to typeset subscripts ($_{x+y}$); and *scriptscript style* used to typeset sub-subscripts ($_{_{x+y}}$). For example, if I want a summation to appear in my paragraph with limits above and below it, I could type '$\displaystyle\sum_{n=1}^{m} $' to produce '$\displaystyle\sum_{n=1}^{m}$'. The style you specify will stay in effect until you or TEX changes it, or you finish the equation. Do you know why two '\displaystyle' commands are needed in '$$\displaystyle{a\over b}\above1pt \displaystyle{c\over d}$$' to produce:

$$\frac{\frac{a}{b}}{\frac{c}{d}}?$$

This is because the command '\above' changes the styles. The '\over' commands changed the styles from '\displaystyle' to '\textstyle' inside the groups.

(See page 139 of the TEXbook for more information.)

Math Punctuation

In addition to using '\ldots' in math mode to achieve '...', there are a few other punctuation rules in math mode which you should know about.

When you use math mode, the comma and semi-colon are treated as punctuation in formulas, so TEX adds some extra space after these characters. The period, however, is not treated as punctuation, so no extra space is added after or before this character.

If you want to use a comma as punctuation in math mode, enclose the comma in a separate group as in '$12{,}345x$'.

If you use the colon in math mode, TEX treats it like a relation ('$x := y$' for example). If you want a colon to act as punctuation in math mode, use the '\colon' command.

If you want to underline text, math mode provides a '\underline' command. You may use it as follows:

`$\underline{\rm` *text*`}$`

Underlining is strongly discouraged outside of math mode for four good reasons:

- Since underlining is done in math mode, words must be separated with a backslash space.
- If you underline words in a phrase separately, the line below words will vary in depth below each word depending on the maximum character depth of each word.
- If you underline multiple words and separate each word with a backslash space, TEX will have trouble breaking a line when encountering an underlined phrase because TEX treats everything underlined as one word that can't be hyphenated.
- You have several other methods for enhancing a word or phrase with different fonts (such as boldface or italics).

(See page 161 of the TEXbook for more information.)

Math Accent Commands

type	*to get*
$\hat o$	$\hat{o}$
$\check o$	$\check{o}$
$\tilde o$	$\tilde{o}$
$\acute o$	$\acute{o}$
$\grave o$	$\grave{o}$
$\dot o$	$\dot{o}$
$\ddot o$	$\ddot{o}$
$\breve o$	$\breve{o}$
$\bar o$	$\bar{o}$
$\vec o$	$\vec{o}$
$\imath$	$\imath$
$\jmath$	$\jmath$

(See page 135 of the TEXbook for more information.)

How To Strut Your Stuff

If you type the equation

```
$$a_0+{1\over a_1+
{1\over a_2+
{1\over a_3+
{1\over a_4}}}}$$,
```

the result will be

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}.$$

This is undesirable because the fractions get too small to read. If you try to add '\displaystyle' commands after all but the last '\over', TEX will produce

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}.$$

Again, this is undesirable because the '1's are almost touching the bars above them. For this equation, we need a '\strut' command. When you type '\strut', TEX generates a vertical line with a height of 8.5pt, a depth of 3.5, and a width of 0pt. The effect is to tell TEX to use the maximum possible line height and depth for ten point lettering.

How To Strut Your Stuff (Continued)

The correct way to type our equation is:

```
$$a_0+{1\over\displaystyle a_1+
{\strut 1\over\displaystyle a_2+
{\strut 1\over\displaystyle a_3+
{\strut 1\over a_4}}}}$$
```

The '\strut' could also have been placed after the '1's but definitely before the '\over' commands. Our final product will look like:

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}$$

Use of '\strut' is not limited to math or display mode. In fact, '\strut' is used more often in tables than anywhere else.

In short, '\strut' tells TEX to make the current line as tall and deep as possible.

The '\strut' command is almost equivalent to the command:

```
\vrule height8.5pt depth3.5pt width0pt
```

If you ever need a larger or smaller '\strut' command, use the above line to create your own (this may be useful if you use different sized fonts in your document). Use of '\magnification' at the top of your file will also magnify the size of '\strut'.

(See pages 142 and 246 of the TEXbook for more information.)

Math Limits And Equation Numbers

When TEX typesets integral and summation signs in display mode with subscripts and superscripts, it places the limits of the summation above and below the sigma, but the limits of the integral are placed to the right. For example,

$$\sum_{n=1}^{m} \quad \text{and} \quad \int_{-\infty}^{\infty}.$$

To change TEX's conventions for limits, use '\limits' or '\nolimits' immediately after the math mode command. For example,

`$$\int\limits_{0}^{\pi\over 2}$$ yields` $\int\limits_{0}^{\frac{\pi}{2}}$;

`$$\sum\nolimits_{n=1}^{m}$$ yields` $\sum\nolimits_{n=1}^{m}$.

If you need an equation number in display mode, TEX provides '\eqno' and '\leqno' to place the equation number to the right of the equation and left of the equation respectively. For example, typing '$$x^2-y^2 = (x+y)(x-y).\eqno(15)$$' will produce

$$x^2 - y^2 = (x + y)(x - y). \eqno(15)$$

Replacing '\eqno' with '\leqno' will generate

$$(15) \qquad x^2 - y^2 = (x + y)(x - y).$$

You can type anything for the equation number. The commands '\eqno' and '\leqno' simply separate the equation from the equation number.

(See page 144 of the TEXbook for more information.)

Math Functions

Mathematical functions like 'log' and 'cos' are never italicized in formulas or equations. Instead of changing fonts everytime you need to use a function in math or display mode, use one of the following commands to generate functions in roman type:

```
\arccos   \csc    \ker    \min
\arcsin   \deg    \lg     \Pr
\arctan   \det    \lim    \sec
\arg      \dim    \liminf \sin
\cos      \exp    \limsup \sinh
\cosh     \gcd    \ln     \sup
\cot      \hom    \log    \tan
\coth     \inf    \max    \tanh
```

Note that '\Pr' is capitalized. You need not worry about adding italic corrections after using one of the above functions (TEX will do it automatically, if needed).

If you use subscripts and superscripts with the functions '\det', '\gcd', '\inf', '\lim', '\liminf', '\limsup', '\max', '\min', '\Pr', and '\sup' in display mode, TEX will use limits (that is, place the subscripts and superscripts above and below the function).

If you use a function not listed in the above table, define a new function command at the top of your file. For example, if you prefer to use 'acos' instead of 'arccos', generate a definition similiar to

```
\def\acos{\mathop{\rm acos}}.
```

(See page 162 of the TEXbook for more information.)

Fine Points Of Mathematics Typing

- **Punctuation** • To use punctuation in a formula, put the period, comma, semicolon, colon, question mark, bang, etc., after the $ in math mode. If the formula is typeset in display mode, place the punctuation inside the $$. For example,

```
If $x<0$, we will prove that $$y=f(x).$$
...for $x = a$, $b$, or $c$.
```

  Typing '$x = a, b$ or c.' would disallow a line break between the comma and the letter 'b'. The space after the comma would be removed by math mode as well. If the formula includes a comma, for example

```
$x = f(a, b)$,
```

  put the comma between the dollar signs. The difference is in its use; in the first example, the comma is used for sentence punctuation, while the comma in the second example is actually part of the formula.
- **Fonts** • If you need to use a non-italicized letter or word in a formula (other than functions), change fonts while in math or display mode with '\rm' (or any of our other font commands). By default, all math is typeset in the font '\mit' (math italics) which differs slightly from regular italics. TEX restores the original text font after leaving math mode.
- **Spacing Between Formulas** • If you typeset more than one formula in a single display, you need special spacing between formulas. For example,

$$F_n = F_{n-1} + F_{n-2}, \qquad n \ge 2.$$

  To separate the condition 'n\ge2' from the rest of the formula, use either '\ ' (backslash space ' '), '\quad' (' '), or '\qquad' (' '). A 'quad' is a printer's term for a space with the width of a capital 'M'. Both '\quad' and '\qquad' may be used in horizontal mode when you need extra space in a text paragraph.

Fine Points Of Mathematics Typing (Continued)

- **Spacing Inside Formulas** • In the equation

$$F_n = F_{n-1} + F_{n-2},\ n \geq 2,$$

TEX inserts a thick space before and after the equal sign and greater-than-or-equal sign and a medium space before and after the plus sign. If you need to fine tune spacing within a formula, you can use one of the following commands:

\, Thin space ($\frac{1}{6}$ of a '\quad')
\> Medium space ($\frac{2}{9}$ of a '\quad')
\; Thick space ($\frac{5}{18}$ of a '\quad')
\! Negative thin space ($-\frac{1}{6}$ of a '\quad')

If you use these commands in a subscript or superscript group, they will come out smaller to compensate for the change in size from the current font.

- **Line Breaking** • If a line break is required in the middle of math mode, TEX will automatically break a formula after a relation sign ($=$, $\geq$, $\rightarrow$, etc.) or after a binary operation symbol ($+$, $-$, $\times$, etc.) if it isn't in a group (between curly brackets). If you want to discourage TEX from breaking a line, enclose the sensitive part of the formula in a separate group. If you want to allow TEX to break a line in the middle of an equation, use '*' (discretionary multiplication sign). Use of '*' will cause a line break and insertion of a $\times$ sign at the end of the line. In the example,

```
$f(x,y) = {x^2-y^2} = (x+y)\*(x-y)$,
```

TEX will allow a line break after the two equal signs, after the plus sign, after the second minus sign, and at the discretionary multiplication sign.

(See pages 161 and 173 of the TEXbook for more information.)

All Delimiters Great And Small

`(`	Left parenthesis: (
`)`	Right parenthesis:)
`[ or \lbrack`	Left bracket: [
`] or \rbrack`	Right bracket:]
`\{ or \lbrace`	Left curly brace: {
`\} or \rbrace`	Right curly brace: }
`\lfloor`	Left floor bracket: ⌊
`\rfloor`	Right floor bracket: ⌋
`\lceil`	Left Ceiling bracket: ⌈
`\rceil`	Right Ceiling bracket: ⌉
`\langle`	Left angle bracket: ⟨
`\rangle`	Right angle bracket: ⟩
`/`	Slash: /
`\backslash`	Backslash or reverse slash: \
`| or \vert`	Vertical bar: |
`\\| or \Vert`	Double vertical bar: ‖
`\uparrow`	Upward arrow: ↑
`\Uparrow`	Double upward arrow: ⇑
`\downarrow`	Downward arrow: ↓
`\Downarrow`	Double downward arrow: ⇓
`\updownarrow`	Up-and-down arrow: ↕
`\Updownarrow`	Double up-and-down arrow: ⇕

All of the above delimiters can be enlarged with '\big' ('$\big[\big]$'), '\Big' ('$\Big[\Big]$'), '\bigg' ('$\bigg[\bigg]$'), and '\Bigg' ('$\Bigg[\Bigg]$') prefixes.

If the delimiter is an opening or closing delimiter, you should use 'bigl', 'bigr', 'Bigl', 'Bigr', etc.. If the delimiter is a relation, like a greater-than sign, use 'bigm', 'Bigm', etc.. Using '\big' without a suffix makes the delimiter spacing equivalent to that of a normal variable. Proper use of '\big' commands ensures proper math spacing.

All Delimiters Great And Small (Continued)

TEX also has commands '\left' and '\right' to adjust automatically the size of a delimiter for you. For example,

```
$$1 + \left(1\over1 - x^2\right)^3$$
```

$$1 + \left(\frac{1}{1 - x^2}\right)^3.$$

The left and right parentheses were automatically made just big enough to cover the enclosed material. You must use '\left' and '\right' in pairs (you can't use a '\left' without having a '\right' in the same equation). In addition, '\left' and '\right' work exactly like grouping characters ('{}'). This means, you can use a '\over' command between '\left' and '\right' commands (as in the above example).

(See pages 145–150 of the TEXbook for more information.)

Multiple Line Displayed Equations

If an equation is too long to fit on one line, use the '\displaylines' command. Here is an example:

```
$$\displaylines{ equation 1 \cr
    equation 2 \cr
    equation 3 \cr
        ⋮
    equation n \cr}$$
```

The equations will be centered on separate lines (when we get to tables, I'll explain how to align equations). For example, to generate the displayed equations

---

$$x \equiv x;$$
$$\text{if} \quad x \equiv y \quad \text{then} \quad y \equiv x;$$
$$\text{if} \quad x \equiv y \quad \text{and} \quad y \equiv z \quad \text{then} \quad x \equiv z.$$

I could type:

```
$$\displaylines{ x\equiv x;\cr
{\rm if}\quad x\equiv y\quad{\rm then}\quad y\equiv
        x;\cr
{\rm if}\quad x\equiv y\quad{\rm and}\quad y\equiv z
        \quad{\rm then}\quad x\equiv z.\cr}$$
```

All equations must be enclosed in a huge text argument. Each equation is separated by a '\cr' command. The last '\cr' is required to finish the last line of the '\displaylines' equation.

The '\displaylines' uses '\hfil' commands before and after each line of the equation to center each equation. If you want to change this, use '\hfill' to override the default action.

(See page 194 of the TEXbook for more information.)

Math Mode Fonts

To change the fonts used by math mode, you will need to reload four fonts for three sizes (total of twelve '\font' commands). The four basic fonts used by math mode are Roman, Math Italic, SYmbols, and EXtended math (CMR, CMMI, CMSY, and CMEX). Here is how TEX assigns these fonts prior to processing your document:

```
 1:\font\tenrm=cmr10\textfont0=\tenrm
 2:\font\teni=cmmi10\textfont1=\teni
 3:\font\tensy=cmsy10\textfont2=\tensy
 4:\font\tenex=cmex10\textfont3=\tenex
 5:\font\sevenrm=cmr7\scriptfont0=\sevenrm
 6:\font\seveni=cmmi7\scriptfont1=\seveni
 7:\font\sevensy=cmsy7\scriptfont2=\sevensy
 8:\font\sevenex=cmex7\scriptfont3=\sevenex
 9:\font\fiverm=cmr5\scriptscriptfont0=\fiverm
10:\font\fivei=cmmi5\scriptscriptfont1=\fivei
11:\font\fivesy=cmsy5\scriptscriptfont2=\fivesy
12:\font\fiveex=cmex5\scriptscriptfont3=\fiveex
```

The '\font' commands actually load or make the font available for use, and the '\textfont', '\scriptfont' and '\scriptscriptfont' commands actually tell math mode what font should be used.

Lines 1–4 load fonts at ten point size for textstyle and displaystyle characters. Lines 5–8 load fonts at seven point size for scriptstyle characters (subscripts and superscripts). And, lines 9–12 load fonts at five point size for scriptscriptstyle characters (sub-subscripts and super-superscripts). Lines 5–12 may produce errors depending on the font set supplied by your TEX software package. If errors occur, omit lines 5–12 and scriptstyle and scriptscriptstyle characters will remain seven points and five points in size respectively.

You should normally not change the font type for any of the above math mode fonts since TEX assumes that certain characters exist in specific locations within the above fonts. You may, however, reload any of the above fonts at a magnified size.

(See page 153 of the TEXbook for more information.)

More Math Fonts And Setup Instructions

Some people hate math italics with a passion. If you are one of these people, TEX has the commands '\everymath' and '\everydisplay'. Typing

```
\everymath={\rm}
```

at the top of your file will cause the math shift character ('$') to change mode to math mode and then '\rm' (change font to roman) before any math expressions are read. For example,

```
\everymath={\bf}
$f(x) = \displaystyle{1\over 2}y$
```

would produce

$$\mathbf{f(x) = \frac{1}{2}y.}$$

(I used '\displaystyle' because my TEX software package didn't have the font I needed to typeset the fraction without it) You may also use '\everydisplay' to change the font when you enter display mode ('$$').

(See pages 153 and 179 of the TEXbook for more information.)

TEX Made Easy

Using TEX With The Plain Macro Package

Day 4

Daniel M. Zirin Instructor

Using Tabs To Make A Table

The simplest way to make a table is to use tabs to generate columns. TEX provides a command called '\settabs' to split the page into columns. For example,

```
1:\settabs 3\columns
```

will divide the printed width of your page into three equally wide columns. You can then fill the columns with information by preceding each line with a '\+' command, ending each line with a '\cr' command sequence, and separating each column with an ampersand ('&'). It is not necessary to fill all columns and you can always finish a line with '\cr' without filling all columns. After typing the above example, you can enter the following information:

```
2:\+Column 1&Column 2&Column 3\cr
3:\+&This starts in column 2\cr
4:\+&&Column 3 info\cr
5:\+\cr
6:\+\sl Slanted&&Regular type\cr
7:\+This is a very long column&Ooops.&&Whoa!\cr
```

Here is what we will see when TEX formats the above seven lines:

Column 1	Column 2	Column 3
	This starts in column 2	
		Column 3 info
Slanted		Regular type
This is a very long column.	Ooops.	Whoa!

Notice that on line six we typed '\sl' to slant the text in that column. The font was removed when the column ended. On line seven, placing information in column four when we only asked for three columns causes TEX to continue making equally sized columns in the right margin.

Using Tabs To Make A Table (Continued)

Once a '\settabs' command has been typed, tabbed lines may appear anywhere in your file (you can even separate tabbed lines with paragraphs of text). For example:

```
\settabs 5\columns
\+This&is&a&table&of info.\cr
This is a new paragraph.\par
\+&&This is&another&table\cr
```

The above four commands would generate the following output:

This	is	a	table	of info.
This is a new paragraph.				
		This is	another	table

To change the number of columns, type another '\settabs' command.

When entering information for a column in a table, you can use '\hfill' (and its variants) to center, left justify, or right justify text. For example, typing

```
\+This\hfill&\hfill is a\hfill&\hfill table.&\cr
```

will cause 'This' to be flush left in column one, 'is a' to be centered in column two, and 'table.' to be flush right in column three.[1]

If you need variably sized columns, there is another form of the '\settabs' command which uses a sample line to determine the width of each column and to determine the number of columns. For example,

```
\settabs\+I.D.\quad&Horacio L.~Fernandez\quad&\cr
```

will generate a table with the width of column one equal to 'I.D.' plus a space, the width of column two equal to 'Horacio L.~Fernandez' plus a space, and the third column can occupy the rest of the line.

(See page 231 of the TEXbook for more information.)

[1] The last column of a '\settabs' table is special and will ignore filler commands.

Making Tables With Alignment

In addition to making tables with '\settabs', you can also use '\halign'. Use of '\halign' is a bit more complicated and takes more memory to generate a table, but, as we will see in the following pages, '\halign' is more versatile.

To use '\halign' type

\halign{ *preamble* **\cr**
row 1 **\cr**
row 2 **\cr**
⋮
last row **\cr**}

Let's first explain the easy part. To enter information into the various rows of your table, use the same format as tabbed lines excluding the '\+' command. Separate each column with an ampersand ('&'), and finish each line with a '\cr' command. After entering the last row of your table, you need an end group character ('}') after the final '\cr' command.

The first line contains a *preamble* which is similar to the sample line format of '\settabs'. Enter TEX commands to format each column of the table in the *preamble* using a number sign ('#') to represent the information being placed in each column, using an ampersand ('&') to separate each column's format instructions, and using a '\cr' to end the *preamble*.

Once the preamble is set up, TEX reads the rest of the table (until the end group character) and automatically sizes each column for you by the information in the preamble and the maximum width of the columns. Because TEX reads the entire table before formatting the table, you cannot insert a paragraph in the middle of a table generated with '\halign'. In addition, using '\halign' for a large table can cause the TEX program to exceed the computer's memory and crash.

Making Tables With Alignment (Continued)

To generate the last table we made with '\settabs', type the following to use horizontal alignment:

```
\halign{#\quad&#\quad&#\cr
13&Horacio L. Fernandez&(818) 555-1213\cr}
```

The first line told TEX to start the table ('{'), place the first column entries at the start ('#'), separate the first column from the second column by the longest entry for column one plus a quad of space ('\quad'), start column two ('&'), separate the second column from the third column by the width of the longest entry in column two plus a quad of space ('\quad'), start column three ('&'), enter column-three entries next ('#'), and finally end the line ('\cr').

The second line is our only row of the table. The number '13' is placed in column one, 'Horacio L. Fernandez' is placed in column two, the phone number '(818) 555-1213' is in column three, the current row in the table finishes ('\cr'), and the table ends ('}').

If you use '\halign' to format a table, you cannot tell TEX there are more columns than you specify in the preamble. You can, however, leave columns blank as long as all column separators ('&') are present. If you don't have entries for the last three columns in a row, you may also type the '\cr' command prematurely, if desired.

Each column is enclosed by implied grouping characters ('{}'), so you can enter a font command for a column entry and the previous font will be restored when the column ends. In addition, you could also use a font command in the preamble and all entries for the column containing the font command will be formatted in the new font.

Making Tables With Alignment (Continued)

As an example, let's try to make the following table (the lines aren't part of the table):

I.D.	**Restaurant**	(Costs)	Income	*Profit*
1	**McRonalds**	($12,500)	$1,325,000	*$1,312,500*
2	**Oh Henri's**	($2,350)	$4,735	*$2,385*
12	**Jack's Diner**	($725)	$24	*−$701*
23	**Taco Roundup**	($1,475)	$2,460,500	*$2,459,025*

Here's how we set up the preamble:

```
\halign{\hfil#\quad&\hfil\bf#\hfil\quad&\hfil
    (#)\quad&\hfil#\quad&\sl\hfil#\cr
```

The '\halign' command starts the first column at the left margin, a quad of space separates column one from column two, column two is centered and displayed in boldface, column two and three are separated by another quad of space, column three is flush right and enclosed in parentheses, column three and four are separated by a quad of space again, column four is flush right and separated from column five by a quad of space, and column five is flush right using the slant font.

To add a line of information in the middle of the table, use '\noalign' as in the following two examples:

```
\noalign{(Jack's Diner is new)}
\noalign{\vskip 6pt}
```

In the first example, we insert the phrase "(Jack's Diner is new)" without placing the information in any columns, and without indentation. The second example separates the previous row from the next by an extra six points of space.

The '\noalign' command can only appear immediately after a '\cr' command in a table alignment.

Making Tables With Alignment (Continued)

In addition to '\noalign' you can use a command '\span' to allow information to span the width of two columns. '\span' replaces the ampersand ('&') in an alignment table.

If you need to enter information that spans several columns, use the '\multispan' command as follows:

```
\multispan{4}\hfil This is a comment.&\cr
```

The above example places the text 'This is a comment.' flush right across the first four columns in a five column table (the ampersand before the '\cr' command is used to separate columns one through four from column five). Column five is left empty.

If you don't want to enter information for a column and you also don't want the information in the preamble to cause TEX to generate printable characters, use the '\omit' command between ampersands ('&') instead of leaving the column blank. Here's an example:

```
1:\halign{\indent\hfil#lb.~bag\quad&\hfil#\cr
2:10&\$3.50\cr 5&\$2.00\cr
3:\omit&or \$2.05\cr
4:1&\$1.25\cr}
```

The '\omit' command in line three will remove the text 'lb.~bag' from the preamble for that column as well as leaving the column entry blank. The result will look like the following:

10lb. bag	$3.50
5lb. bag	$2.00
	or $2.05
1lb. bag	$1.25

(See page 235 of the TEXbook for more information.)

Making Tables With Line Borders

To generate a table with borders requires that you learn about '\offinterlineskip' and '\vrule'.

If you type '\offinterlineskip', TEX removes the spaces between lines (ignoring '\baselineskip' and '\lineskip'). Make sure that you start a new group before typing '\offinterlineskip', so line spacing will be restored when you finish the table and the group.

The '\vrule' command makes a vertical line (rule) 0.4 points wide and the height & depth of the current line ('|'). The actual format for typing the '\vrule' command is:

`\vrule width`*dim* `height`*dim* `depth`*dim*

where the width, height, and depth are all optional. For example, the 'dreaded black box' that TEX makes at the end of lines which stick out into the right margin was generated with

```
\vrule width5pt
```

In addition to '\vrule', there is also '\hrule' that works exactly the same way generating a horizontal line. By default, '\hrule' makes a line the width of the page, 0.4 points high, and 0 points deep.

Making Tables With Line Borders (Continued)

To make a table with vertical lines, a column in the '\halign' preamble must be reserved for each vertical line across the table. Here is an example:

```
 1:{\offinterlineskip
 2:\halign{ \vrule # & \strut\  # \  & \vrule # &
 3:         \  # \  & \vrule # \cr
 4:\multispan{5}\hrulefill\cr
 5:height2pt & \omit & & & \cr
 6:& Country & & Score & \cr
 7:height2pt & \omit & & & \cr
 8:\multispan{5}\hrulefill\cr
 9:height2pt & \omit & & & \cr
10:& U.S.A. & & 210 & \cr
11:& Germany & & 140 & \cr
12:& Japan & & 50 & \cr
13:height2pt & \omit & & & \cr
14:\multispan{5}\hrulefill\cr}}
```

Don't worry about the '\halign' preamble right now. Instead, I will try to explain what happens for each row of the table.

On line one, a new group is started and interline spacing is turned off. When the table ends, a close-group character will be needed and line spacing will return to its original settings.

On line two and three, the '\halign' starts the table and defines the preamble. Notice that when '\halign' is typed a second group begins.

On line four, '\multispan' overrides the preamble for all five columns and inserts a horizontal line.

On line five, column one will contain '\vrule height2pt'. This is needed because no line space was added between the horizontal line above and the next line with table entries. The '\omit' destroys the preamble for the second column containing an undesirable '\strut' (which would force the current line to a height greater than two points).

Making Tables With Line Borders (Continued)

On line six, the first column is skipped (the '\vrule' will expand to the maximum height and depth of the current row of our table). The second column uses '\strut' to make the height and depth of the current row the maximum and the title 'Country' will be displayed.

Here is the result of typing the previous example:

Country	Score
U.S.A.	210
Germany	140
Japan	50

Now that you know how to type everything needed to make a beautiful table, I'll let you in on a secret. If we typed the following two lines,

```
\def\smalline{height2pt&\omit&&&\cr}
\def\hrline{\multispan{5}\hrulefill\cr}
```

at the top of the file (between lines one and two of the example on the previous page), lines 4, 8, and 14 could have been replaced with '\hrline' and lines 5, 7, 9, and 13 could have been replaced with '\smalline'. Here is the shorter version:

```
 1:{\offinterlineskip
 2:\def\smalline{height2pt&\omit&&&\cr}
 3:\def\hrline{\multispan{5}\hrulefill\cr}
 4:\halign{ \vrule # & \strut\  # \  & \vrule # &
 5:         \  # \  & \vrule # \cr
 6:\hrline\smalline
 7:& Country & & Score & \cr
 8:\smalline\hrline\smalline
 9:& U.S.A. & & 210 & \cr
10:& Germany & & 140 & \cr
11:& Japan & & 50 & \cr
12:\smalline\hrline}}
```

(See page 246 of the TEXbook for more information.)

Making Matrices In Displayed Equations

• \matrix •

Here is an example of what '\matrix' can generate:

$$R = \begin{pmatrix} e_{11} & e_{12} & \ldots & e_{1c} \\ e_{21} & e_{22} & \ldots & e_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ e_{r1} & e_{r2} & \ldots & e_{rc} \end{pmatrix}$$

This command works exactly like '\halign'. Here is how the above equation was made:

```
$$R=\left(\matrix{e_{11}&e_{12}&\ldots&e_{1c}\cr
e_{21}&e_{22}&\ldots&e_{2c}\cr
\vdots&\vdots&\ddots&\vdots\cr
e_{r1}&e_{r2}&\ldots&e_{rc}\cr}\right)$$
```

The above example uses '\ldots', '\vdots', and '\ddots' for low, vertical, and diagonal dots respectively.

See page 176 of the TEXbook for more information.

• \pmatrix • \bordermatrix •

The command '\pmatrix' is a variant of '\matrix' that automatically encloses a matrix in parentheses. The '\bordermatrix' command inserts parentheses and generates column and row labeling of a matrix.

You can read about these on pages 176 and 177 of the TEXbook.

Making Matrices In Displayed Equations (Continued)

• **\cases** •

If you need to typset the equation

$$f(x) = \begin{cases} 1/3 & \text{if } x \ge 10; \\ 1/2 & \text{if } 5 \le x < 10; \\ 1 & \text{if } x < 5. \end{cases}$$

TEX offers a command '\cases'. It works a bit like '\halign'. Any number of cases can be specified, and TEX will generate and enlarge the left curly bracket in the displayed formula. Here is the text needed to display the above formula:

```
$$f(x) = \cases{ 1/3 & if $x\ge 10$;\cr
1/2 & if $5\le x<10$;\cr
1 & if $x<5$.\cr}$$
```

The ampersands ('&') are used to separate two columns; the formula to the left of the ampersand and the conditional to the right of the ampersand. The '\cr' commands tell TEX where each line of '\cases' ends. Notice that the formula to the left of the ampersand is typeset in math mode while the information to the right is typeset in horizontal mode (the mode used to generate standard text). The '\cases' starts a group and finishes the group after the last '\cr'.

See page 175 of the TEXbook for more information.

Aligning Displayed Equations

To make the below equation,

$$\begin{aligned} T(n) \le T(2^{\lceil \lg n \rceil}) &\le (3^{\lceil \lg n \rceil} - 2^{\lceil \lg n \rceil}) \\ &< 3c \cdot 3^{\lg n} \\ &= 3cn^{\lg 3} \end{aligned}$$

use the '\eqalign' command. Use of '\eqalign' is very similiar to a two-column '\halign' table with the left and right columns pushed to the center. Here is the text used to make the above example:

```
$$\eqalign{T(n)\le T(2^{[\lg n]}) &
    \le (3^{[\lg n]}-2^{[\lg n]}) \cr
& < 3c\cdot 3^{\lg n} \cr
& = 3c n^{\lg 3} \cr}$$
```

On line one and two, the first line of the equation is generated with the alignment of subsequent lines under the second less-than or equal sign.

If a '\eqno' was used after the end of '\eqalign' (the end group '}') and before the end of the displayed equation ('$$'), the equation number would be flush right vertically centered over all lines of the equation. Here is an example:

$$\begin{aligned} T(n) \le T(2^{\lceil \lg n \rceil}) &\le (3^{\lceil \lg n \rceil} - 2^{\lceil \lg n \rceil}) \\ &= 3cn^{\lg 3}. \end{aligned} \tag{18$'$}$$

If you don't like the spacing between lines that '\eqalign' automatically makes, insert a '\noalign{\vskip *dimension*}' immediately after a '\cr' command.

Aligning Displayed Equations (Continued)

If you need equation numbers for each line of a '\eqalign', use either '\eqalignno' for equation numbers to the right or '\leqalignno' for equation numbers displayed to the left of each line. These two commands work like a three-column table. The first column is for the left side of the formulas, the second column is for the right side of the formulas and the third column is reserved for the equation number (if any). Here is an example of '\leqalignno':

$$\begin{aligned} (x+y)(x-y) &= x^2-xy+yx-y^2 && (1)\\ &= x^2-y^2; &&\\ (x+y)^2 &= x^2+2xy+y^2. && (2) \end{aligned}$$

The above displayed equation was made with:

```
$$\leqalignno{(x+y)(x-y) & =x^2-xy+yx-y^2 & (1) \cr
& =x^2-y^2; & \cr
(x+y)^2 & =x^2+2xy+y^2.  & (2) \cr}$$
```

(See page 190 of the TEXbook for more information.)

TEX Made Easy

Using TEX With The Plain Macro Package

Day 5

Daniel M. Zirin Instructor

Centering An Aligned Table

To cause a table generated with '\halign' to be centered, precede the TEX commands needed to produce the table with[1]

```
\centerline{\vbox{
```

and terminate the table with two extra end-group characters ('}}'). Here's an example:

```
\centerline{\vbox{
\offinterlineskip
\def\skipit{\cr\noalign{\vskip3pt}}
\halign{\hfil #\quad&#\cr
Column 1&Column 2\skipit
\multispan{2}\hrulefill\skipit
Frank&210lb.\skipit
Joe&325lb.\cr}}}
```

The result of the above TEX commands follows:

Column 1	Column 2
Frank	210lb.
Joe	325lb.

[1] See chapter 11 and page 77 of the TEXbook for more information about the '\hbox' and '\vbox' commands.

Leaving Room To Insert Figures

If a figure needs to be placed in a page of your document, use '\midinsert' to save room for the figure when TEX formats your document. The command '\midinsert' checks to see if the text or blank space can be placed at the current line without causing a page eject. If there is enough room, the text or blank space will be formatted on the current page otherwise it will be placed at the top of the following page.

To use this command, type '\goodbreak\midinsert', followed by text and/or '\vskip' commands, and terminated with '\endinsert'. For example, typing

```
\goodbreak\midinsert
\hrule\vskip 2in\hrule
\endinsert
```

will cause a horizontal line to appear before and after two inches of vertical blank space. You could replace '`\hrule \vskip 2in \hrule`' with a text paragraph, table, equation, or any combination. Using '\midinsert' and '\endinsert' implies a separate group, so changes made after '\midinsert' will go away when '\endinsert' is typed.

(See page 115 of the TEXbook for more information.)

Making Your Own Filler Commands

If you don't like using '\dotfill' or '\hrulefill' for making a table of contents, you can make your own specialized filler command by defining a character-repeat sequence. To make your own filler, use the '\leaders' command as follows:[1]

```
\def\myfill{\leaders\hbox to 5pt{\hss:\hss}\hfill}
```

Once the above is typed, our new filler can be used in the following examples:

```
\line{\myfill This is cute.\myfill}
\line{Chapter 1\myfill 14}
```

to produce the following:

: This is cute. :
Chapter 1 : 14

If we increase the size of the box containing our repeat sequence to ten points ('{\hss:\hss}' is our repeat sequence and the box is currently five points wide), the above two '\line' commands would generate:

: : : : : : : : : : : This is cute. : : : : : : : : : :
Chapter 1 : 14

Changing the colon in our repeat sequence to '\%' would make:

% % % % % % % % % % % This is cute. % % % % % % % % % % %
Chapter 1 % 14

(See page 223 of the TEXbook for more information.)

[1] See chapter 11 and page 77 of the TEXbook for more information about the '\hbox' command.

Character Overstriking

To print one character on top of another, use one of the following commands:

\rlap '\rlap' takes a text argument to overstrike the characters immediately following '\rlap'. For example, '\rlap{/}c' would print '¢'.

\llap '\llap' takes a text argument to overstrike the characters immediately preceding '\llap'. For example, '/\llap{c}' would generate '¢'.

To cause TEX to make the next character baseline higher or lower than the current baseline use the '\raise' and '\lower' commands. For example, to place a tilde under the greek letter tau, define a new command as follows:[1]

```
\def\tildetau{$\tau$\llap{\lower7.5pt\hbox{\~{}}}}
```

Once the above definition is made, typing '\tildetau' would produce 'τ'.

(See pages 66 and 82 of the TEXbook for more information.)

[1] See chapter 11 and page 77 of the TEXbook for more information about the '\hbox' command.

Character Duties

Here is the table of all character duties and category codes:

\ (Category 0) The command character.
{ (Category 1) The start group character.
} (Category 2) The end group character.
$ (Category 3) The math shift character.
& (Category 4) The alignment tab character.
<CR> (Category 5) The end of line character.
(Category 6) The parameter character.
^ (Category 7) The superscript character.
_ (Category 8) The subscript character.
<NULL> (Category 9) This character is ignored.
<SPACE> (Category 10) The space character.
letters (Category 11) Letters A...Z and a...z.
others (Category 12) All characters not found in other categories.
~ (Category 13) The 'active' character.
% (Category 14) The comment character.
<DEL> (Category 15) Invalid character.

Suppose you don't have a hat key on your keyboard (used for superscripting in math or display mode). TEX has a method of using another key for superscripting. To use the double quote key for superscripting, type:

```
\catcode`\"=7
```

This makes the double-quote key belong to category seven (superscripting). This would not change the category of the hat key. If you have a hat key, but you want to use the double quote for superscripting, change the hat key to category twelve so it can be used in your document as follows:

```
\catcode`\^=12
```

(See pages 37 and 39 of the TEXbook for more information.)

Boxes

If you want to enclose a word, phrase, or anything at all inside a box, type the following at the top of your file to define the command '\boxit':

```
\def\boxit#1{\leavevmode\hbox{\vrule\vtop
   {\vbox{\hrule\kern1pt\hbox{\kern1pt
   \strut#1\kern1pt}}\kern1pt\hrule}\vrule}}
```

And you can box just about anything by typing:

```
\boxit{ whatever }
```

This sentence is using the '\boxit' command to box the word 'sentence'. Using '\boxit' will not allow TEX to hyphenate a word or break a line in the middle of a box containing multiple words.

Automatic Numbering

The following page contains a few commands and definitions to generate automatic numbering for sections, references, as well as other uses.

The package defines two new commands: '\sectno' for the counter we will use and automatically increment, and '\section' to display the contents of '\sectno'.

'\sectno' is merely a variable (like the '\pageno' counter). It will be initially set to the value one, but may be changed by typing '\sectno=*value*'. Here are the necessary commands:

```
1:\newcount\sectno \sectno=1
2:\def\section{\the\sectno
3:\global\advance\sectno by 1}
```

And here is an example of its use:

```
1:\noindent Section \section\par
2:\noindent Section \section\par
3:\sectno=13
4:\noindent Section \section\par
```

Automatic Table Of Contents Generator

The following pages contain a few commands and definitions to automatically generate a table of contents.

The package defines four new commands: '\tocref' to make entries in the table of contents, '\tocline' to put special lines in the table of contents, '\tocgen' to force the last table of contents page to be printed, and '\tocpageno' used as the independent page number counter for our TOC.

'\tocref' takes two text arguments: the text to be itemized in the table of contents (this text will also be placed in your file at the same location where '\tocref' is found) and the optional indentation for the table of contents line containing this entry. For example, if you type:

```
\centerline{\tocref{\bf Chapter 1}{}}
```

the boldface text 'Chapter 1' will be both centered on the current line and flush left in the table of contents. If you now type:

```
\noindent\tocref{Section 1}{\indent}
```

the text 'Section 1' will be located at the point where '\tocref' was typed, as well as indented in the table of contents. When using '\tocref', the text arguments will be followed by dots and a page number in the table of contents.

The command '\tocline{ *text* }' will place '*text*' on the next line in the table of contents (without dots and a page number).

The '\tocgen' command will save the current page number, create a table of contents page with independent page numbering, restore the old page number, and remove all '\tocref' entries from TEX's memory. This command should always be used before '\end' to flush the last few TOC entries out of TEX's memory.

The '\tocpageno' command is a counter variable that keeps track of the current TOC page number independently of the document's page number. By default, TOC pages are numbered with roman numerals. To change the current TOC page number, simply type '\tocpageno=*value*'.

Automatic TOC Generator (Continued)

First, we select a non-letter to temporarily become a member of the TEX letter family. Later, we will create variables and macros that use this non-letter making it nearly impossible for the end-user to accidentally redefine or reuse a macro or variable essential to the operation of this TOC macro.

```
1:\catcode`\@=11
```

Next, we declare some new variables for our TOC macros. Variable 't@cnew' will determine if this is a new TOC (1), a new page of the TOC (3), or in the midst of generating a TOC (2). Variable 't@copage' will be used to save the current document's page number when a TOC page gets printed. Variable 'tocpageno' will be our TOC page number (as opposed to the current document's page number). Since 'tocpageno' doesn't use our special at-sign (@) letter in the variable name, the end-user is allowed to alter this variable if desired. Variable 't@cbox' is the box that contains the lines of the current page of the TOC. Variables 't@cboxtest' and 't@ctestsize' are used to determine when the TOC page, currently being generated, becomes full (about to exceed `\vsize`).

```
2:\newcount\t@cnew\t@cnew=1
3:\newcount\t@copage\newbox\t@cbox
4:\newcount\tocpageno\tocpageno=-1
5:\newbox\t@cboxtest\newdimen\t@ctestsize
```

Now we generate a 't@cstrut' definition to give our TOC pages a uniform line separation. The definition assumes you are using 10 point lettering and expands if you are using '\magnification' at the top of your file. The following setting makes the baselineskip 8.5 points in height and 3.5 points in depth (allowing for 2 points of line spacing).

```
6:\def\t@cstrut{{\vrule height8.5pt depth3.5pt
7:    width0pt}}
```

Automatic TOC Generator (Continued)

Since our TOC pages will be generated simultaneously with normal document pages, we need special macros for numbering our TOC pages separately from the rest of our document. The next macro, 't@cfolio' is exactly the same as the normal definition of 'folio' (explained earlier) except that 't@cfolio' will use our own page number counter 'tocpageno'.

```
 8:\def\t@cfolio{\t@cstrut\ifnum\tocpageno<0
 9:     \romannumeral-\tocpageno\else
10:     \number\tocpageno\fi}
```

Now that we have our own TOC page numbering, we must provide a method for incrementing 'tocpageno' the same way 'pageno' is incremented.

```
11:\def\t@cadvpageno{\ifnum\tocpageno<0
12:     \global\advance\tocpageno by -1\else
13:     \global\advance\tocpageno by 1\fi}
```

Our definition of 'tocref' first writes text argument number one at the current location in the document being prepared (not in the TOC). Next, 'tocref' calls 't@ctest' to check that adding the next line in or TOC will not exceed a page worth of text. If 't@ctest' determines that the current TOC page is full, the TOC page is output and emptied. The argument for macro 't@ctest' is the content of the next line to be added to the current TOC page (since 't@cstrut' is known to be larger than 'dotfill' and 'folio', we exclude the latter two from the 't@ctest' argument). The next step is to call the definition 't@cbanner' in case 't@ctest' just emptied the current TOC page and a TOC banner is required. Finally, 'tocref' globally changes 't@cbox' to be equal to 't@cbox' plus the next line of the TOC.

```
14:\def\tocref#1#2{#1\t@ctest{\t@cstrut#2#1}
15:     \t@cbanner\global\setbox\t@cbox=\vbox{
16:     \box\t@cbox\vbox{
17:     \line{\rm\t@cstrut{#2#1}~\dotfill~\folio}}}}
```

Automatic TOC Generator (Continued)

To define 'tocline', we do everything we did for the definition of 'tocref' except double printing text argument one and formatting the new TOC line differently.

```
\def\tocline#1{\t@ctest{\t@cstrut#1}
    \t@cbanner\global\setbox\t@cbox=\vbox{
    \box\t@cbox\vbox{
    \line{\rm\t@cstrut{#1}}}}}
```

The definition for 't@cbanner' checks the value of 't@cnew' to determine if a page one TOC banner is needed or a continuation banner is needed. If so, 't@cbanner' calls 't@cstart' or 't@cont' to generate the desired TOC banner.

```
\def\t@cbanner{\ifnum\t@cnew=1\t@cstart\fi
    \ifnum\t@cnew=3\t@cont\fi}
```

To make the 'tocstart' and 'tocont' definitions, we simply change the contents of 'tocbox' to be equal to the new banner and reset 'tocnew' to two (2).

```
\def\t@cstart{\global\setbox\t@cbox=\vbox{
    \centerline{\t@cstrut\bf Table Of Contents}
    \line{\t@cstrut\hfil} % blank line
    }\global\t@cnew=2}
\def\t@cont{\global\setbox\t@cbox=\vbox{
    \centerline{\t@cstrut
            Table Of Contents (Continued)}
    }\global\t@cnew=2}
```

The 't@ctest' macro is used by 'tocref' and 'tocline' to check if a new line of TOC text will cause the current TOC page to become overfull. It does this by adding the height and depth of the current 't@cbox' page, the height and depth of the new TOC line of text, and the additional height and depth of our 't@cstrut' definition (used for the TOC page numbering) and compares the total with 'vsize'. If the current TOC page is full, call definition 'tocgen' to output the page and set counter 't@cnew' to equal three (used by 't@cbanner').

Automatic TOC Generator (Continued)

```
32:\def\t@ctest#1{\setbox\t@cboxtest=\vbox{
33:    \line{#1\hss}}\t@ctestsize=\ht\t@cbox
34:    \advance\t@ctestsize by\dp\t@cbox
35:    \advance\t@ctestsize by\ht\t@cboxtest
36:    \advance\t@ctestsize by\dp\t@cboxtest
37:    \setbox\t@cboxtest=\vbox{\line{\t@cfolio\hss}}
38:    \advance\t@ctestsize by\ht\t@cboxtest
39:    \advance\t@ctestsize by\dp\t@cboxtest
40:    \ifdim\t@ctestsize>\vsize
41:    \tocgen\global\t@cnew=3\fi}
```

And last but not least, we must define 'tocgen'. The definition of 'tocgen' checks that a TOC exists, saves the current page number, resets the page number to our TOC page number, adds the TOC page number to the end of the current TOC page, ships out the current TOC page, resets the current page number to the old saved setting, and changes 't@cnew' to equal one (1).

```
42:\def\tocgen{\ifnum\t@cnew=1
43:    \message{No TOC entries found.}\else
44:    \t@copage=\pageno\pageno=\tocpageno
45:    \global\setbox\t@cbox=\vbox to\vsize{
46:    \box\t@cbox\vfil\vbox{
47:    \centerline{\t@cfolio}}}
48:    \message{(TOC}\shipout\box\t@cbox\message{)}
49:    \t@cadvpageno\pageno=\tocopage
50:    \global\tocnew=1\fi}
```

The next line restores the original category code of the at-sign changed back on line one.

```
51:\catcode`\@=12
```

Automatic TOC Generator (Continued)

The following lines show how the previous TOC macro may be used:

```
 1:\centerline{\bf\tocref{Chapter 1}{}}
 2:This is the first paragraph of Chapter 1.\par
 3:\noindent\tocref{\bf Section 1}{\indent}\par
 4:This is the first paragraph of
 5:Section 1 in Chapter 1.\par
 6:\tocline{\hfil (Cont.)}
 7:\noindent\tocref{Section 2}{\indent\bf}\par
 8:This is the first paragraph of
 9:Section 2 in Chapter 1.\par
10:\tocgen\vfil\end
```

Rather than show the output produced by the above ten lines, I suggest you try it yourself.

TEX Made Easy

Using TEX With The Plain Macro Package

Appendix A

Daniel M. Zirin Instructor

Math Mode Symbols

Greek Letters

For a list of Greek letters, see page 46. To generate slanted uppercase Greek letters, use the "\mit" font command in the following manner:

"`$\mit\Omega$`" creates "$\mit\Omega$"

Calligraphic Capitals

To create any uppercase calligraphic letter, simply precede a capital letter with the command "`\cal`" as in:

"`$\cal D\cal E\cal K$`" would generate "$\cal D\cal E\cal K$"

Large Operators

$\bigcap$	`\bigcap`	$\bigcup$	`\bigcup`	$\bigodot$	`\bigodot`
$\bigoplus$	`\bigoplus`	$\bigotimes$	`\bigotimes`	$\bigsqcup$	`\bigsqcup`
$\biguplus$	`\biguplus`	$\bigvee$	`\bigvee`	$\bigwedge$	`\bigwedge`
$\coprod$	`\coprod`	$\int$	`\int`	$\oint$	`\oint`
$\prod$	`\prod`	$\sum$	`\sum`		

The `\sum`, `\prod`, `\coprod`, and `\int` seen above in `\textstyle` format have different spacing than the TEX commands `\Sigma`, `\Pi`, `\amalg`, and `\smallint`, respectively.

Miscellaneous Symbols

$\aleph$	`\aleph`	$\angle$	`\angle`	$\backslash$	`\backslash`
$\bot$	`\bot`	$\clubsuit$	`\clubsuit`	$\diamondsuit$	`\diamondsuit`
ℓ	`\ell`	$\emptyset$	`\emptyset`	$\exists$	`\exists`
$\flat$	`\flat`	$\forall$	`\forall`	$\hbar$	`\hbar`
$\heartsuit$	`\heartsuit`	$\Im$	`\Im`	$\imath$	`\imath`
∞	`\infty`	$\jmath$	`\jmath`	∇	`\nabla`
$\natural$	`\natural`	$\neg$	`\neg`	∂	`\partial`
$\prime$	`\prime`	$\Re$	`\Re`	$\sharp$	`\sharp`
$\spadesuit$	`\spadesuit`	$\surd$	`\surd`	$\top$	`\top`
$\triangle$	`\triangle`	$\wp$	`\wp`	$\|$	`|`

Math Mode Symbols

Relations

$<$	`<`	$>$	`>`	$=$	`=`
$\approx$	`\approx`	$\asymp$	`\asymp`	$\cong$	`\cong`
$\equiv$	`\equiv`	$\geq$	`\geq`	$\leq$	`\leq`
$\prec$	`\prec`	$\preceq$	`\preceq`	$\sim$	`\sim`
$\simeq$	`\simeq`	$\sqsubseteq$	`\sqsubseteq`	$\sqsupseteq$	`\sqsupseteq`
$\subset$	`\subset`	$\subseteq$	`\subseteq`	$\succ$	`\succ`
$\succeq$	`\succeq`	$\supset$	`\supset`	$\supseteq$	`\supseteq`
$\bowtie$	`\bowtie`	$\dashv$	`\dashv`	$\doteq$	`\doteq`
$\frown$	`\frown`	$\gg$	`\gg`	$\in$	`\in`
$\ll$	`\ll`	$\mid$	`\mid`	$\models$	`\models`
$\ni$	`\ni`	$\parallel$	`\parallel`	$\perp$	`\perp`
$\propto$	`\propto`	$\smile$	`\smile`	$\vdash$	`\vdash`

The first 21 relation commands above can be prefixed with the "`\not`" command to create a negative relation. For example, "`\not\succeq\not>`" would produce "$\not\succeq\not>$".

Binary Operations

$+$	`+`	$-$	`-`	$\amalg$	`\amalg`
$*$	`\ast`	$\bigcirc$	`\bigcirc`	$\bigtriangledown$	`\bigtriangledown`
$\bigtriangleup$	`\bigtriangleup`	$\bullet$	`\bullet`	$\cap$	`\cap`
$\cdot$	`\cdot`	$\circ$	`\circ`	$\cup$	`\cup`
$\dagger$	`\dagger`	$\ddagger$	`\ddagger`	$\diamond$	`\diamond`
$\div$	`\div`	$\mp$	`\mp`	$\odot$	`\odot`
$\ominus$	`\ominus`	$\oplus$	`\oplus`	$\oslash$	`\oslash`
$\otimes$	`\otimes`	$\pm$	`\pm`	$\setminus$	`\setminus`
$\sqcap$	`\sqcap`	$\sqcup$	`\sqcup`	$\star$	`\star`
$\times$	`\times`	$\triangleleft$	`\triangleleft`	$\triangleright$	`\triangleright`
$\uplus$	`\uplus`	$\vee$	`\vee`	$\wedge$	`\wedge`
		$\wr$	`\wr`		

The difference between commands `\backslash` and `\setminus` is the extra spacing used for the later.

Math Mode Symbols

Openings
For a list of delimiters and openings, see page 58.

Arrows

$\downarrow$	`\downarrow`	$\Downarrow$	`\Downarrow`
$\hookleftarrow$	`\hookleftarrow`	$\hookrightarrow$	`\hookrightarrow`
$\leftarrow$	`\leftarrow`	$\Leftarrow$	`\Leftarrow`
$\leftharpoondown$	`\leftharpoondown`	$\leftharpoonup$	`\leftharpoonup`
$\leftrightarrow$	`\leftrightarrow`	$\Leftrightarrow$	`\Leftrightarrow`
$\longleftarrow$	`\longleftarrow`	$\Longleftarrow$	`\Longleftarrow`
$\longleftrightarrow$	`\longleftrightarrow`	$\Longleftrightarrow$	`\Longleftrightarrow`
$\longmapsto$	`\longmapsto`	$\mapsto$	`\mapsto`
$\longrightarrow$	`\longrightarrow`	$\Longrightarrow$	`\Longrightarrow`
$\nearrow$	`\nearrow`	$\nwarrow$	`\nwarrow`
$\rightarrow$	`\rightarrow`	$\Rightarrow$	`\Rightarrow`
$\rightharpoondown$	`\rightharpoondown`	$\rightharpoonup$	`\rightharpoonup`
$\rightleftharpoons$	`\rightleftharpoons`		
$\searrow$	`\searrow`	$\swarrow$	`\swarrow`
$\uparrow$	`\uparrow`	$\Uparrow$	`\Uparrow`
$\updownarrow$	`\updownarrow`	$\Updownarrow$	`\Updownarrow`

To place symbols above a relation or arrow, use the "`\buildrel`" command as follows:

```
${\buildrel dt\over\longrightarrow}$
```

In the above example, the text "dt" is placed on top of a long right arrow and the result would look like:

$$\overset{dt}{\longrightarrow}$$

Warning: `\over` is not used to define a fraction when used with the `\buildrel` command.

Math Mode Symbols

Aliases
The following table shows some commands that have aliases.

$\neq$	`\ne`, `\neq`, and `\not=`
$\leq$	`\le` and `\leq`
$\geq$	`\ge` and `\geq`
$\{$	`\{` and `\lbrace`
$\}$	`\}` and `\rbrace`
$\rightarrow$	`\rightarrow` and `\to`
$\ni$	`\ni` and `\owns`
$\wedge$	`\land` and `\wedge`
$\vee$	`\lor` and `\vee`
$\neg$	`\lnot` and `\neg`
$\vert$	`\vert` and `|`
$\Vert$	`\Vert` and `\\|`
$\Longleftrightarrow$	`\iff` and `\Longleftrightarrow`

Non-Math Symbols

Oldstyle Numbers
In addition to the standard six font commands (`\rm`, `\sl`, `\bf`, `\it`, `\tt`, and `\mit`) you may use the `\oldstyle` font command for numbers only to generate "0123456789".

Miscellaneous Symbols

§	`\S`	¶	`\P`
†	`\dag`	‡	`\ddag`

TEX Made Easy

Using TEX With The Plain Macro Package

Appendix B

Daniel M. Zirin Instructor

TEX Command Reference Table

The following list of commands and a brief description of their meaning have all be introduced in this document. The page number following the description is the page where the command is first referenced. For a list of other useful math commands not referenced in this book, see appendix A.

`\ `
: (Backslash-space) Generate a white-space character [5].

`\{`
: Generate a left curly bracket character [8] {Math}.

`\}`
: Generate a right curly bracket character [8] {Math}.

`\$`
: Generate a dollar sign character [8].

`\&`
: Generate an ampersand character [8].

`\#`
: Generate a number sign character [8].

`\^{}`
: Generate a hat character for use without accenting [8].

`\_`
: Generate an underscore character [8].

`\%`
: Generate a percent sign character [8].

`\~{}`
: Generate a snook character for use without accenting [8].

`\‘`
: Generate a grave accent above the following character [9].

`\’`
: Generate an acute accent above the following character [10].

`\^`
: Generate a circumflex accent above the following character [10].

`\"`

Generate an umlaut accent above the following character [10].

`\~`

Generate a tilde accent above the following character [10].

`\=`

Generate a macron accent above the following character [10].

`\.`

Generate a dot accent above the following character [10].

`\/`

Generate a little extra white-space for separating a slanted font from an unslanted font [11].

`\-`

Allows the temporary automatic hyphenation for automatic line breaking [41].

`\*`

Allows the automatic line breakage of a math formula by inserting a 'times' sign where one was implied [57] {Math}.

`\,`

Generates $\frac{1}{6}$ of a `\quad` of horizontal white-space [57] {Math}.

`\>`

Generates $\frac{2}{9}$ of a `\quad` of horizontal white-space [57] {Math}.

`\;`

Generates $\frac{5}{18}$ of a `\quad` of horizontal white-space [57] {Math}.

`\!`

Generates $-\frac{1}{6}$ of a `\quad` of horizontal white-space [57] {Math}.

`\+`

Defines the beginning of a tabular line [65].

`\aa`

Generate the lowercase Scandinavian A-with-circle character [10].

`\AA`
: Generate the uppercase Scandinavian A-with-circle character [10].

`\above`
: Generates a fraction with an arbitrary thickness for the dividing line [49] {Math}.

`\acute`
: Generate an acute accent above the following character [51] {Math}.

`\advance`
: Advances the following internal TEX variable by a specified integer amount [87].

`\ae`
: Generate the 'æ' lowercase Latin ligature [10].

`\AE`
: Generate the 'Æ' uppercase Latin ligature [10].

`\alpha`
: Generates the lowercase Greek alpha character [46] {Math}.

`\arccos`
: Generates the letters 'arccos' in roman font for use as a function name [55] {Math}.

`\arcsin`
: Generates the letters 'arcsin' in roman font for use as a function name [55] {Math}.

`\arctan`
: Generates the letters 'arctan' in roman font for use as a function name [55] {Math}.

`\arg`
: Generates the letters 'arg' in roman font for use as a function name [55] {Math}.

`\atop`
: Generates a fraction excluding the dividing line [49] {Math}.

`\b`
: Generate a bar-under accent below the following character [10].

`\backslash`
: Generate a backslash character [8] {Math}.

`\bar`
: Generate a bar accent above the following character [51] {Math}.

`\baselineskip`
: Defines the uniform vertical line separation [38].

`\beta`
: Generates the lowercase Greek beta character [46] {Math}.

`\bf`
: Change the current font to boldface extended [11].

`\big`
: Generates a larger version of the following math character [58] {Math}.

`\Big`
: Generates a larger version of the following math character [58] {Math}.

`\bigg`
: Generates a larger version of the following math character [58] {Math}.

`\Bigg`
: Generates a larger version of the following math character [58] {Math}.

`\biggl`
: Generates a larger version of the following opening math delimiter symbol [58] {Math}.

`\Biggl`
: Generates a larger version of the following opening math delimiter symbol [58] {Math}.

`\biggm`
: Generates a larger version of the following math character [58] {Math}.

`\Biggm`
: Generates a larger version of the following math character [58] {Math}.

`\biggr`
: Generates a larger version of the following closing math delimiter symbol [58] {Math}.

`\Biggr`
: Generates a larger version of the following closing math delimiter symbol [58] {Math}.

`\bigl`
: Generates a larger version of the following opening math delimiter symbol [58] {Math}.

`\Bigl`
: Generates a larger version of the following opening math delimiter symbol [58] {Math}.

`\bigm`
: Generates a larger version of the following math character [58] {Math}.

`\Bigm`
: Generates a larger version of the following math character [58] {Math}.

`\bigr`
: Generates a larger version of the following closing math delimiter symbol [58] {Math}.

`\Bigr`
: Generates a larger version of the following closing math delimiter symbol [58] {Math}.

`\bigskip`
: Generates 12 points of vertical white-space [25].

`\bordermatrix`
: Generates a matrix for use in a displayed equation enclosed with variably sized parentheses and an extra column and row for border labels [74] {Math}.

`\box`
: Identifies an internal box variable [90].

`\boxit`
: Macro to generate a square box around arbitrary text [86].

`\breve`
: Generate a breve accent above the following character [51] {Math}.

`\c`
: Generate a cedilla accent under the following character [9].

`\cases`
: Generates a multi-lined case statement for use in a displayed equation with a variably sized open curly bracket [75] {Math}.

`\catcode`
: Defines or redefines a character to a TEX category or function [85].

`\cdot`
: Generates a single vertically centered dot character [48] {Math}.

`\cdots`
: Generates a vertically centered ellipsis [48] {Math}.

`\centerline`
: Generates a line of centered text [16].

`\check`
: Generate a check accent above the following character [51] {Math}.

`\chi`
: Generates the lowercase Greek chi character [46] {Math}.

`\choose`
: Generates a choose between function [49] {Math}.

`\colon`
: Generates a colon for punctuation [50] {Math}.

`\columns`
: Generates a number of equally wide columns for the tabular environment defined by `\settabs` [65].

`\cos`
: Generates the letters 'cos' in roman font for use as a function name [55] {Math}.

`\cosh`
: Generates the letters 'cosh' in roman font for use as a function name [55] {Math}.

`\cotcot`
: Generates the letters '' in roman font for use as a function name [55] {Math}.

`\coth`
: Generates the letters 'coth' in roman font for use as a function name [55] {Math}.

`\cr`
: Terminates the end of a table line or the end of one line in a multi-lined displayed equation [60].

`\csc`
: Generates the letters 'csc' in roman font for use as a function name [55] {Math}.

`\d`
: Generate a dot-under accent below the following character [10].

`\dag`
: Generate a single crossed dagger character [10].

`\ddag`
: Generate a double crossed dagger character [10].

`\ddot`
: Generate a double dot accent above the following character [51] {Math}.

`\ddots`
: Generates a diagonal top-to-bottom ellipsis [74] {Math}.

`\def`
: Generates a new macro or replaces an existing macro [39].

`\deg`
: Generates the letters 'deg' in roman font for use as a function name [55] {Math}.

`\delta`
: Generates the lowercase Greek delta character [46] {Math}.

`\Delta`
: Generates the uppercase Greek delta character [46] {Math}.

`\det`
: Generates the letters 'det' in roman font for use as a function name [55] {Math}.

`\dim`
: Generates the letters 'dim' in roman font for use as a function name [55] {Math}.

`\displaylines`
: Generates a multi-lined displayed equation [60] {Math}.

`\displaystyle`
: Changes the current font style size to displaystyle format [49] {Math}.

`\dot`
: Generate a dot accent above the following character [51] {Math}.

`\dotfill`
: Generates infinite horizontal dot filler [37].

`\downarrow`
: Generates a down arrow symbol [58] {Math}.

`\Downarrow`
: Generates a double down arrow symbol [58] {Math}.

`\dp`
: Retrieves the depth dimension of the following internal box variable [92].

`\eject`
: Stops generating material for the current page and starts a new page [34].

`\else`
: When the first conditional evaluation is not true, `\else` begins the alternate conditional block of commands [90].

`\end`
: Terminates processing of a TEX input document [17].

`\endinsert`
: Ends a floating insertion [82].

`\epsilon`
: Generates the lowercase Greek epsilon character [46] {Math}.

`\eqalign`
: Generates an aligned multi-lined displayed equations [76] {Math}.

`\eqalignno`
: Generates an aligned multi-lined displayed equations with optional right-flush equation numbers [77] {Math}.

`\eqno`
: Generates an equation number vertically centered to the right [54] {Math}.

`\equiv`
: Generates an equivalence symbol [60] {Math}.

`\eta`
: Generates the lowercase Greek eta character [46] {Math}.

`\everydisplay`
: Defines a command or series of commands to automatically execute upon entering display mode [62] {Math}.

`\everymath`
: Defines a command or series of commands to automatically execute upon entering math mode [62] {Math}.

`\exp`
: Generates the letters 'exp' in roman font for use as a function name [55] {Math}.

`\fi`
: Ends a conditional block of commands [90].

`\folio`
: Generates a page number for the current page [35].

`\font`
: Request to use a new font and define a command for using the new font [12].

`\footline`
: Sets the running line of material to appear at the bottom of every page [35].

`\footnote`
: Generates a footnote to appear at the bottom of the current page [32].

`\frenchspacing`
: Defines the spacing algorithm to be 'frenchspacing' [42].

`\gamma`
: Generates the lowercase Greek gamma character [46] {Math}.

`\Gamma`
: Generates the uppercase Greek gamma character [46] {Math}.

`\gcd`
: Generates the letters 'gcd' in roman font for use as a function name [55] {Math}.

`\ge`
: Generates a greater-than-or-equal sign symbol [75] {Math}.

`\global`
: Causes the following expression to take affect globally throughout the entire document regardless of the current group level [87].

`\goodbreak`
: Temporarily tricks TEX into thinking the current page is longer to cause an early page eject [82].

`\grave`
: Generate a grave accent above the following character [51] {Math}.

`\H`
: Generate the long Hungarian umlaut accent above the following character [10].

`\halign`
: Generates a table [67].

`\hangafter`
: Sets the line of the next paragraph to start hanging paragraph indentation [29].

`\hangindent`
: Sets the amount of horizontal paragraph hanging indentation for the next paragraph [29].

`\hat`
: Generate a hat accent above the following character [51] {Math}.

`\hbox`
: Generates an arbitrary horizontal box of material with undefined depth and height [83].

`\headline`
: Sets the running line of material to appear at the top of every page [35].

`\hfil`
: Generates infinite horizontal filler [34].

`\hfill`
: Generates infinite horizontal filler [33].

`\hfilll`
: Generates infinite horizontal filler [34].

`\hfuzz`
: Defines the amount of tolerance for line justification [42].

`\hoffset`
: Defines the document left margin [15].

`\hom`
: Generates the letters 'hom' in roman font for use as a function name [55] {Math}.

`\hrule`
: Generates a horizontal line or rule [71].

`\hrulefill`
: Generates infinite horizontal rule filler (solid horizontal line) [37].

`\hsize`
: Defines the document printed width [15].

`\hskip`
: Moves the point of reference for generating the next character horizontally across the page [25].

`\hss`
: Generates infinite horizontal filler [34].

`\ht`
: Retrieves the height dimension of the following

`\hyphenation`
: Defines a new automatic hyphenation algorithm for one or a series of words [41].

`\i`
: Generate a dotless lowercase I [9].

`\if`
: Begins a conditional block of commands [90].

`\ifdim`
: Begins a conditional block of commands limiting the evaluation to comparing dimensions or dimension variables [92].

`\ifnum`
: Begins a conditional block of commands limiting the evaluation to comparing numbers or counter variables [92].

`\imath`
: Generate a dotless lowercase I [51] {Math}.

`\inf`
: Generates the letters 'inf' in roman font for use as a function name [55] {Math}.

`\infty`
: Generates an infinity character [47] {Math}.

`\input`
: Suspends processing the current file and starts processing an external file [18].

`\int`
: Generates an integral sign character [47] {Math}.

`\iota`
: Generates the lowercase Greek iota character [46] {Math}.

`\it`
: Change the current font to italics [11].

`\it\$`
: Generate an English pounds sign character [10].

`\item`
: Starts generating a first-level enumerated paragraph [28].

`\itemitem`
: Starts generating a second-level enumerated paragraph [28].

`\j`
: Generate a dotless lowercase J [9].

`\jmath`
: Generate a dotless lowercase J [51] {Math}.

`\kappa`
: Generates the lowercase Greek kappa character [46] {Math}.

`\ker`
: Generates the letters 'ker' in roman font for use as a function name [55] {Math}.

`\l`
: Generate the lowercase Polish suppressed L character [10].

`\L`
: Generate the uppercase Polish suppressed L character [10].

`\lambda`
: Generates the lowercase Greek lambda character [46] {Math}.

`\Lambda`
: Generates the uppercase Greek lambda character [46] {Math}.

`\langle`
: Generates a tall left less-than symbol [58] {Math}.

`\lbrace`
: Generates a normal sized left curly bracket symbol [58] {Math}.

`\lbrack`
: Generates a normal sized left square bracket symbol [58] {Math}.

`\lceil`
: Generates the upper half of a double height left square bracket symbol [58] {Math}.

`\ldots`
: Generates an ellipsis [48] {Math}.

`\le`
: Generates a less-than-or-equal sign symbol [75] {Math}.

`\leaders`
: Generates a repeat sequence for a specified horizontal width [83].

`\left`
: Generates a larger version of the following opening math delimiter symbol and starts an implied group [59] {Math}.

`\leftskip`
: Defines the extra amount of left margin white-space [31].

`\leqalignno`
: Generates an aligned multi-lined displayed equations with optional left-flush equation numbers [77] {Math}.

`\leqno`
: Generates an equation number vertically centered to the left [54] {Math}.

`\lfloor`
: Generates the lower half of a double height left square bracket symbol [58] {Math}.

`\lg`
: Generates the letters 'lg' in roman font for use as a function name [55] {Math}.

`\lim`
: Generates the letters 'lim' in roman font for use as a function name [55] {Math}.

`\liminf`
: Generates the letters 'liminf' in roman font for use as a function name [55] {Math}.

`\limits`
: Forces superscripts and subscripts to appear above and below the preceding symbol [54] {Math}.

`\limsup`
: Generates the letters 'limsup' in roman font for use as a function name [55] {Math}.

`\line`
: Generates material restricted to a line box of undefined height and depth [33].

`\lineskip`

Defines the amount of extra vertical line spacing used when lines are separated by `\lineskiplimit` vertical space or less [38].

`\lineskiplimit`

Defines when backup vertical line spacing is used [38].

`\llap`

Overlaps the specified text on top of the preceding equally wide text [84].

`\ln`

Generates the letters 'ln' in roman font for use as a function name [55] {Math}.

`\log`

Generates the letters 'log' in roman font for use as a function name [55] {Math}.

`\lower`

Temporarily adjusts the current baseline lower on the current page [84].

`\magnification`

Cause the entire document to magnify or enlarge (must be used at line one of the document) [12].

`\magstep`

Defines the amount to magnify an object [12].

`\magstephalf`

Defines the magnification of an object to be 9.5% [12].

`\matrix`

Generates a matrix for use in a displayed equation [74] {Math}.

`\max`

Generates the letters 'max' in roman font for use as a function name [55] {Math}.

`\medskip`

Generates 6 points of vertical white-space [25].

`\message`
: Generates a interactive message displayed at the users terminal or logfile while processing the document with the TEX program [92].

`\midinsert`
: Begins a floating insertion to be placed at the current location if possible or move to the top of the next page [82].

`\min`
: Generates the letters 'min' in roman font for use as a function name [55] {Math}.

`\mit`
: Change the current font to math italics [45].

`\mu`
: Generates the lowercase Greek mu character [46] {Math}.

`\multiply`
: Multiplies a times b and stores result in a [36].

`\multispan`
: Temporarily merges one or more columns of an aligned table into a single column for the current row and removes the automatic formatting for all merged columns [70].

`\narrower`
: Increases the left and right margins for a quoted passage [30].

`\newbox`
: Generates a new internal TEX box variable [89].

`\newcount`
: Generates a new internal TEX counter variable [87].

`\newdimen`
: Generates a new internal TEX dimension variable [89].

`\noalign`
: Generates material that will be unaffected by the environment of a table or an aligned displayed equation [69].

`\noindent`
: Supresses automatic paragraph indentation [25].

`\nolimits`
: Forces superscripts and subscripts to appear offset to the right on the preceding symbol [54] {Math}.

`\nonfrenchspacing`
: Defines the spacing algorithm not to be 'frenchspacing' [42].

`\nopagenumbers`
: Sets the running line of material at the bottom of the page to be blank [35].

`\nu`
: Generates the lowercase Greek nu character [46] {Math}.

`\null`
: Generates a zero-width white-space character [26].

`\number`
: Generates a visual representation of the value of an internal counter variable [90].

`\o`
: Generate the lowercase Scandinavian O-with-slash character [10].

`\O`
: Generate the uppercase Scandinavian O-with-slash character [10].

`\oe`
: Generate the 'œ' lowercase French ligature [10].

`\OE`
: Generate the 'Œ' uppercase French ligature [10].

`\offinterlineskip`
: Disables uniform line spacing possibly allowing vertically adjacent lines to connect [71].

`\omega`
: Generates the lowercase Greek omega character [46] {Math}.

`\Omega`
: Generates the uppercase Greek omega character [46] {Math}.

`\omit`
: Temporarily removes the automatic formatting for the current column in the current row of an aligned table [70].

`\over`
: Generates a standard fraction [49] {Math}.

`\overfullrule`
: Defines the width of the square box placed on lines the protrude into the right margin [42].

`\P`
: Generate a paragraph sign character [10].

`\pageno`
: The variable containing the current page number [35].

`\par`
: Terminates a paragraph [26].

`\parindent`
: Sets the amount of automatic horizontal paragraph indentation [28].

`\parskip`
: Sets the amount of vertical white-space to separate paragraphs [27].

`\phantom`
: Generates white-space equal in horizontal width to a specified character [31] {Math}.

`\phi`
: Generates the lowercase Greek phi character [46] {Math}.

`\Phi`
: Generates the uppercase Greek phi character [46] {Math}.

`\pi`
: Generates the lowercase Greek pi character [46] {Math}.

`\Pi`
: Generates the uppercase Greek pi character [46] {Math}.

`\pmatrix`
: Generates a matrix for use in a displayed equation enclosed with variably sized parentheses [74] {Math}.

`\Pr`
: Generates the letters 'Pr' in roman font for use as a function name [55] {Math}.

`\pretolerance`
: Defines the tolerance for automatic word hyphenation [41].

`\psi`
: Generates the lowercase Greek psi character [46] {Math}.

`\Psi`
: Generates the uppercase Greek psi character [46] {Math}.

`\qquad`
: Generates a horizontal white-space the width of two `\quad` commands [56].

`\quad`
: Generates a horizontal white-space the width of a capital 'M' when using the CMR font (differs by font) [56].

`\raggedright`
: Disables automatic line justification [42].

`\raise`
: Temporarily adjusts the current baseline higher on the current page [84].

`\rangle`
: Generates a tall right greater-than symbol [58] {Math}.

`\rbrace`
: Generates a normal sized right curly bracket symbol [58] {Math}.

`\rbrack`
: Generates a normal sized right square bracket symbol [58] {Math}.

`\rceil`
: Generates the upper half of a double height right square bracket symbol [58] {Math}.

`\rfloor`
: Generates the lower half of a double height right square bracket symbol [58] {Math}.

`\rho`
: Generates the lowercase Greek rho character [46] {Math}.

`\right`

Generates a larger version of the following closing math delimiter symbol and ends an implied group [59] {Math}.

`\rightskip`

Defines the extra amount of right margin white-space [31].

`\rlap`

Overlaps the specified text on top of the following equally wide text [84].

`\rm`

Change the current font to roman (the default) [11].

`\romannumeral`

Generates a visual representation of the value of an internal counter variable in roman numeral form [90].

`\S`

Generate a section number sign character [10].

`\scriptfont`

Defines the fonts to use for script style symbols and lettering for math and display modes [61] {Math}.

`\scriptscriptfont`

Defines the fonts to use for scriptscript style symbols and lettering for math and display modes [61] {Math}.

`\scriptstyle`

Changes the current font style size to scriptstyle format [49] {Math}.

`\scriptscriptstyle`

Changes the current font style size to scriptscriptstyle format [49] {Math}.

`\sec`

Generates the letters 'sec' in roman font for use as a function name [55] {Math}.

`\setbox`

Causes the following internal box variable to be replaced with the new specified box material [90].

`\settabs`

Defines the number of equally wide columns to be used in a tabular environment [65].

`\shipout`

Causes the following box material to be ejected as a new page [92].

`\sigma`

Generates the lowercase Greek sigma character [46] {Math}.

`\Sigma`

Generates the uppercase Greek sigma character [46] {Math}.

`\sin`

Generates the letters 'sin' in roman font for use as a function name [55] {Math}.

`\sinh`

Generates the letters 'sinh' in roman font for use as a function name [55] {Math}.

`\sl`

Change the current font to slanted roman [11].

`\smallskip`

Generates 3 points of vertical white-space [25].

`\span`

Temporarily merges two columns of an aligned table into a single column for the current row [70].

`\sqrt`

Generates a square root [48] {Math}.

`\ss`

Generate the German 'es-zet' character [10].

`\strut`

Generates a zero-width character 8.5 points high and 3.5 points deep (used for line spacing and in math equations) [52] {Math}.

`\sum`

Generates a summation sign character [47] {Math}.

`\sup`
: Generates the letters 'sup' in roman font for use as a function name [55] {Math}.

`\t`
: Generate a tie accent above the following two characters [10].

`\tan`
: Generates the letters 'tan' in roman font for use as a function name [55] {Math}.

`\tanh`
: Generates the letters 'tanh' in roman font for use as a function name [55] {Math}.

`\tau`
: Generates the lowercase Greek tau character [46] {Math}.

`\textfont`
: Defines the fonts to use for text and display style symbols and lettering for math and display modes [61] {Math}.

`\textstyle`
: Changes the current font style size to textstyle format [49] {Math}.

`\theta`
: Generates the lowercase Greek theta character [46] {Math}.

`\Theta`
: Generates the uppercase Greek theta character [46] {Math}.

`\thinspace`
: Generate a small amount of white-space [5].

`\tilde`
: Generate a tilde accent above the following character [51] {Math}.

`\topinsert`
: Begins a floating insertion to be placed at the top of the current page if possible or move to the top of the following page [82].

`\tt`
: Change the current font to typewriter roman [11].

`\u`
: Generate a breve accent above the following character [10].

`\underline`
: Generates underlined material [50] {Math}.

`\uparrow`
: Generates an up arrow symbol [58] {Math}.

`\Uparrow`
: Generates a double up arrow symbol [58] {Math}.

`\updownarrow`
: Generates an up and down arrow symbol [58] {Math}.

`\Updownarrow`
: Generates a double up and down arrow symbol [58] {Math}.

`\upsilon`
: Generates the lowercase Greek upsilon character [46] {Math}.

`\Upsilon`
: Generates the uppercase Greek upsilon character [46] {Math}.

`\v`
: Generate a check accent above the following character [10].

`\varepsilon`
: Generates a variant of the lowercase Greek epsilon character [46] {Math}.

`\varphi`
: Generates a variant of the lowercase Greek phi character [46] {Math}.

`\varpi`
: Generates a variant of the lowercase Greek pi character [46] {Math}.

`\varrho`
: Generates a variant of the lowercase Greek rho character [46] {Math}.

`\varsigma`
: Generates a variant of the lowercase Greek sigma character [46] {Math}.

`\vartheta`
: Generates a variant of the lowercase Greek theta character [46] {Math}.

`\vbox`
: Generates an arbitrary vertical box of material with undefined width [81].

`\vec`
: Generate a vector symbol above the following character [51] {Math}.

`\vdots`
: Generates a horizontally centered ellipsis [74] {Math}.

`\vert`
: Generates a normal sized single vertical bar symbol [58] {Math}.

`\Vert`
: Generates a normal sized double vertical bar symbol [58] {Math}.

`\vfil`
: Generates infinite vertical filler [34].

`\vfill`
: Generates infinite vertical filler [33].

`\voffset`
: Defines the document top margin [15].

`\vrule`
: Generates a vertical line or rule [53] {Math}.

`\vsize`
: Defines the document printed depth [15].

`\vskip`
: Moves the point of reference for generating the next character vertically down the page [25].

`\vss`
: Generates infinite vertical filler [34].

`\xi`
: Generates the lowercase Greek xi character [46] {Math}.

`\Xi`

Generates the uppercase Greek xi character [46] {Math}.

`\zeta`

Generates the lowercase Greek zeta character [46] {Math}.

Table Of Contents (Continued)