November 1983

Report No. STAN-CS-83-985

Stanford Department of Computer Science

(Version 1)

First Grade TEX

A Beginner's TEX Manual

by
Arthur L. Samuel

This manual is based on the publications of Donald E. Knuth who originated the TEX system and on the recent work of Professor Knuth and his many students and collaborators who have helped bring the TEX82 system to its present advanced state of development. The TEX logo that is used in this manual is a trademark of The American Mathematical Society.
The preparation of this report was supported in part by National Science Foundation grant IST-820/926 and by the System Development Foundation.

Table of Contents

Acknowledgements

It is impractical to mention all of the people who have proofread this manual and who have contributed many valuable suggestions. The detailed contributions made by Don Knuth, Mrs. Knuth, Dave Fuchs, Arthur Keller, Ron Bracewell, and Howard Trickey were particularly helpful. Needlessly to say, much of the basic material in this manual came directly or indirectly from The TEXbook by Don Knuth.

A Beginner's Preface

The beginner can easily be confused by the different versions of TEX82 that seem to exist on different computers and by the conflicting claims for different Macro Packages that are supposed to making TEX 'user friendly'. This preface attempts to clarify the situation.

In the first place, there is only one official version of TEX82 and users are specifically cautioned not to make any changes to the basic program itself. A 'change file' mechanism is provided to allow the program to be compiled differently as dictated by the hardware and system software constraints that exist at any particular computer installation, but these changes should not modify TEX's formatting capabilities.

It is customary, and indeed desirable, to supply TEX with a fairly large amount of additional information, either in the form of a separate input file or as information that is preloaded with TEX so that it becomes available automatically when the TEX program is used. Such files are usually called macro packages and you, as a user, may have to concern yourself with the number and kinds of macro packages that are available at your installation.

Macro packages are used to supply TEX with several quite different kinds of information, much of it being fairly standard and of no immediate concern. Macro packages often assign values to a fair number of the approximately eighty TEX primitives that can be so preassigned and you may wish to assign different values to these. Of major concern are the fairly large number of control sequences that are predefined. These control sequences attach names to certain logical combinations of selected primitives (and of other control sequences) that together perform frequently-used formatting function. The availability of these predefined sequences can make your task very much easier when you wish to do the specific things for which they were designed.

This manual is written on the assumption that you will be using the `PLAIN .TEX` macro package. This package simplifies many formatting tasks without interfering with the separate use of any of TEX's primitives. Having mastered plain TEX to the extent that it is explained in this manual, you may later want to define some additional control sequences that will help you in specifying the formatting conventions that you employ.

Some macro packages are written from quite a different point of view from that used for plain TEX, where the aim is to make certain global formatting conventions very easy to use at the expense (and sometimes with the deliberate intent) of making it more difficult for other formatting conventions to be specified. Some of these packages actually redefine terms that TEX uses for primitives so that these primitives can no longer be called directly. If one of these packages completely meets your needs then you should use it, but beware.

One final word-there is a certain amount of circularity in the language used in this manual. If you do not understand everything the first time through, plow straight ahead until you bog down and then start all over again at the beginning. You will be surprised how much clearer everything will be on the second reading.

First Grade TEX

A Beginner's TEX Manual

Introduction

This is an introductory ready-reference TEX82 manual for the beginner who would like to do First Grade TEX work. Only the most basic features of the TEX system are discussed in detail. Other features are summarized in an appendix and references are given to the more complete documentation available elsewhere.

TEX is a computerized typesetting system. As TEX is normally used, the original text is typed (into an input file) very much as it would be typed for submission to an old-fashioned printer except that this input file must now contain all of the instructions that are needed to describe the desired format for the printed output. Given such a description, still in rather general terms, the TEX compiler is able to specify in precise detail the font (i.e., the size and kind of typeface) and the location for each character that is to be printed. Your final output can meet the very best publishing standards.

Defining book quality text is not an easy task. While the TEX system takes care of many of the tedious details, the wealth of facilities that TEX provides can be very confusing to the beginner and sometimes even to the experienced user. These facilities include the handling of such matters as: 1) ligature replacements (for example, fi for fi), 2) kerning (different spacings between certain letter pairs), 3) automatic hyphenation, 4) line justification, 5) centering, 6) flushing right or left, 7) tabular aligning, 8) the formatting of complicated mathematical expressions, 9) section and page numbering, 10) the introduction of running heads, 11) the numbering and placing of footnotes, and 12) the preparation of a table of contents and an index, to name but a few.

It will be assumed that the reader is already acquainted with the use of a computer and with at least one text editor that can be used while typing the TEX input file. One minor warning at this point: Do not use an editor that requires or leaves its own special formatting marks (line numbers, word processing commands, etc.) in the file that it produces, unless your version of TEX has been specially tailored to tolerate them. Such marks will be assumed to be a part of the text by TEX. The text in the input file should be broken up into reasonably short lines. TEX will ignore 1) the way you break your paragraphs into lines, 2) extra spaces between words, and 3) extra blank lines between paragraphs, although it does accept one or more blank lines as indicating a paragraph break.

TEX is usually preloaded with one or more special files that define many very useful commands. This manual assumes that you will be using a basic file called `PLAIN .TEX`, and TEX when so loaded will be refered to as plain TEX. We will not discuss the details as to how one actually types the input information, how this is saved as a file on the system, how one evokes the TEX compiler, and how one instructs the computer system to print the final document, as these features are highly system dependent.

Toward Book Quality

Book quality text differs from ordinary typing in a number of simple but far from trivial ways. In the first place, a distinction is made between the upper case 0 and the number 0 and between the lower case 1 and the number 1, then there is a distinction between hyphens, dashes, and the minus sign. Also, your terminal should contain two varieties of 'single' quotes and when you want a "double" quote you simply type two of the appropriate single quotes and TEX will space these correctly, as shown below.

Name	Hyphen	En-dash	Em-dash	Minus sign	Double quotes
To print	-	–	—	−	"text"
You type	-	--	---	$ 4	"t e x t "

A word about hyphenation—TEX can usually be depended upon to use hyphens correctly (when this is necessary to achieve right justification) and to avoid excessive hyphenation (by increasing or slightly decreasing the inter-word spacings within preassigned limits). There are exceptions, and there are words like pres-ent and present where the positioning of a hyphen depends upon how the word is used. When TEX either refuses to hyphenate or makes a mistake, you can gain hyphenation control over the word in question by inserting one or more discretionary hyphens at acceptable locations, using the control sequence '\-', as an example in the word re\-cord or rec\-ord. These will be ignored when not needed and they will print as normal hyphens when needed.

TEX provides for the automatic substitution of ligatures when available, replacing ff by ff, fi by fi, and fl by fl, unless you specifically disallow it by typing either {f{}f} to get ff or f\/f to get ff (slightly more space between).

A nice feature of plain TEX, is the ease with which special accents and a few special letters may be produced.

Name	grave	acute	"hat"	umlaut	tilde	"bar"	dot
To print	à	é	ô	ii	ñ	ȳ	ṗ
You type	\ 'a	\'e	\^o	\"u	\~n	\=y	\.p

Name	check	breve	long	tie-after	cedilla	bar-under	dot-under
To print	ǐ	š	ʃ̋	i͡u	ç	k̲	ḥ
You type	\v\i	\u s	\H\j	\t\i u	\c c	\b k	\d h

To print	œ,Œ	æ,Æ	å,Å	ø,Ø	ł,Ł	¿	i	ß
You type	\oe,\OE	\ae,\AE	\aa,\AA	\o,\O	\l,\L	?'	!'	\ss

Note that \i and \j give dotless i and j for use with accents. All of the above special symbols and characters work equally well in roman, italic, and bold type fonts. A few do not work in the fixed-width typewriter font. Many other symbols, mainly mathematical, are listed in the Appendix.

Special Symbols

Plain TEX assigns special meanings to ten infrequently used but normally available typewriter symbols. These symbols are used to simplify the task of issuing commands to TEX. Should you wish these symbols to be printed in your final document they must appear in your input file as shown below.

Symbol	To print, type	Special TEX meaning when used directly
\	$\backslash$	Used to indicate the start of a TEX control sequence, usually refered to as a backslash
(and)	$\{$ and $\}$	Grouping symbols, to indicate range of action of a control sequence to the enclosed text
$	\$	Used to initiate and terminate portion of the text that is to be in Mathematics Mode
&	\&	Alignment tab, used to delineate fields within a table
#	\#	Parameter, used to signify a field within a table or in a control sequence
	\^{}	Superscript (also used as an accent, see previous page)
	_	Subscript
%	\%	Comment symbol, the rest of the line is ignored by TEX
	\~{}	Used to introduce a single space in locations where a line break is not to be allowed.

Issuing Commands to TEX

Commands to TEX, as distinguished from the text itself, consist of the backslash, '\', then the name of the command (without an intervening space) and then, in some cases, the parameters that are associated with the command. The command name can be of two types, either 1) a single non-alphabetic character, or 2) one or more alphabetic characters that must be terminated by a space if the name is to be followed by other alphabetic characters. These alphabetic characters are often in lower case, but please note that the correct case, as specified, must always be used. For example, 'Ω' produces 'Ω', while ω' produces 'ω'. Also note the basic distinction between non-alphabetic-character command names which are not to be followed by a space (unless a space is actually wanted in the output) and alphabetic-character command names which require a terminating space if they are followed by other alphabetic characters.

Commands are of two types. Over 300 commands, called primitives, are a part of TEX's built-in vocabulary . Then there are ***control*** sequences, sometimes called macros, that are constructed from these primitives (or from other control ***sequences).***

Many of the primitives are normally used only to define control ***sequences*** and need not concern the beginner, but others specify basic characteristics of the TEX's output, such

as, for example, the page size ('`\hsize=6.5in \vsize=8.9in`'), the desired margins, and the spacing between paragraphs ('`\parskip=10pt` plus `1pt`'). All of these latter are preassigned values in plain TEX although they can be changed by the user.

Control sequences are used to simplify the typing of commands to TEX, since they are shorter and much easier to remember than the sequence of ***primitives*** that they specify. Many very useful ***control sequences*** are to be found in the file PLAIN. TEX and these ***control sequences*** are, in effect, added to the basic vocabulary that TEX understands when this file is preloaded with TEX. In addition, there exist a number of special collections of ***macros*** that TEX experts have found useful for special purposes such as writing business letters, preparing internal memoranda, and preparing manuscripts for publication. The very first line appearing in your main input file may well be the control sequence `\input` followed by the name of the file that contains the desired macro collection. Incidentally, TEX interprets a file name that is so used without a file name extension to mean a file with the extension of .TEX, for example, `\input MYMACS` will cause the file `MYMACS.` TEX to be loaded.

Plain TEX understands well over 900 ***control sequences*** but many of these are self-evident from their names and others fall into a relatively few catagories, so that they can be easily learned when needed. Some of the more important of these are listed in the Appendix.

Fonts

TEX provides for the use of as many as 256 different fonts, each containing as many as 256 different characters. Plain TEX, as loaded, has 16 fonts that are assigned special control-sequence names. The default font, called roman, is requested by the use of the ***control sequence*** '`\rm`'. If you wished the entire text to be typed in boldface you would type '`\bf`' before your text. If, on the other hand, you want a single word or a phrase to appear in a different font you should use the grouping symbols '`{`' and '`}`' to delimit the range.

For example, typing '`to be {\bf bold}is to {\sl emphasize}something`' will produce 'to be **bold** is to ***emphasize*** something'. Incident ally, an ***italic correction*** command '`\/`' introduces a little extra space when used after letters in the the \it and `\sl` fonts to compensate for a change in slope of the letters. Typing '`\it italicized\/ text`' produces '***italicized*** text' as compared to '*italicized* text' when the correction is not used.

The following ***control sequences*** are for the type fonts as shown:

`\rm`	`\sl`	\it	`\tt`	`\bf`
Roman	***Slanted***	***Italic***	Typewriter	**Boldface**

Type fonts also come in different sizes. To demonstrate, if you type:
`{\tenrm` smaller `and}{\ninerm` smaller `and}{\eightrm` smaller and)
`{\sevenrm` smaller `and}{\sixrm` smaller), you get

smaller and `smaller` and smaller and **`smaller and smaller`**, all lined `up` to a common ***baseline.***

This manual uses *10 point* type, as called for by PLAIN .TEX, and defined to be \tenrm, except that the author has had everything magnified by 1.2 (for beginners), by having the command '\magnification=1200' appear ahead of any text in the input file. So the actual sizes, in the above example, are all multiplied by 1.2. You may have also noticed that this manual uses *'slanted type'* for emphasis and 'typewriter type' for things that you are told to type and for the messages that the computer displays.

The characters within a font, including those not on your keyboard, can be gotten by typing '\char<number>' where <number> is the decimal number of the character position, or by typing '\char '<octal number>' where <octal number> is an octal number. For example, typing '\char'35' produces Æ. Incidentally, the normal alphanumeric characters are at their *ascii* positions regardless of the code that your computer may use externally.

Another way to get symbols that are not on your keyboard is to define *control sequences* for them. For example, PLAIN. TEX defines '$\ne$', '$\le$', and $\ge$' for '≠', '≤' and '≥', respectively. Over 300 such PLAIN. TEX definitions are listed in the Appendix.

Dimensions and Keywords

TEX understands a variety of dimensional units as *keywords* that are used without the '\'. These convey fixed meanings to TEX when they are used in the proper context. These units and their meanings are:

Unit	Meaning	Per inch	Unit	Meaning	Per inch
pt	point	72.27	mm	millimeter	25.4
pc	pica	6.023	dd	Didot point	67.54
in	inch	1	bp	big point	72
cm	centimeter	2.54	sp	scaled point	4736286.72

Sometimes it is convenient to use dimensions that are relative to the character sizes in the font being used at the time. Two such units are the em and the ex. The em, used for measuring horizontal distances, is, by tradition, the width of the upper case 'M' but is an arbitrarily assigned value in TEX that is the width of a 'quad'. The ex is used for measuring vertical distances and is approximately the height of a lower case 'x'.

To Use TEX

Having written a simple input file, say, MYFILE.TEX, which contains the desired text together with a few *control sequences* that define your wishes, you may now call on TEX to operate on your file. On some computers you do this by typing: 'run tex; MYFILE'. Since the correct form can vary from computer to computer, the best thing to do is to ask someone who knows what to do on your particular computer. Usually, TEX will begin by prompting you with a double asterisk '**', to indicate that you can then type the name of your input file. If, on the other hand, the prompt is a single asterisk '*', (or if you are

calling for another file within a file) then you must precede the file name by the '**\input**' command thus: '**\input** MYFILE. The control control ***sequence*** **\input** must usually be typed in lower case and typing MYFILE implies MYFILE.TEX.

The input file itself may begin by referencing one or more additional files (this time using the '**\input**' command). These files may define some special fonts that are to be used and they may contain the definitions of some specialized, frequently-used, constructs (control sequences or macros) that are used to simplify the typing of the desired instructions. Once defined, these constructs can then be called by name as needed in the text file itself. Caution: Do not load up your working space with macros that you never use.

The TEX program produces a DVI (device independent) file, that specifies each character that is to be printed, together with its font and its exact location in the printed document. The DVI file may require some further translation before it is acceptable as input to the available printer but on most systems this further action is usually automatic, requiring, at most, a carriage return confirmation. The DVI file is usually saved and it can then be reprinted at a later date, perhaps on a different printer, if this should be desired.

It should be noted that TEX does not need to know the exact shape of each character, but it does need to know the overall size and the details as to how the individual characters may interact with adjacent characters (resulting in ligature replacements or in kerning). This information is usually contained in what are known as TFM files. The printer, on the other hand, does need to have access to the detailed information as to the exact configuration of each character that is to be printed and this information is usually contained in what are known as PXL files. The two sets of files must agree as to the fonts that they describe, or chaos will result.

Boxing with Glue

There are a few facts about the inner working of TEX that you will need to know. TEX approaches the task of formatting a page of text much as a mason might build a brick wall, which is to be laid in courses to a fixed width using bricks that vary in size (and that have to be used in a fixed order), with limits as to the minimum and maximum amount of mortar that can be used between bricks.

TEX happens to use different terms, which you should learn to recognize. Instead of bricks, TEX talks about 'hboxes'(initially, the individual words in your text), which it assembles out of simple 'boxes'(the individual letters). No mortar is used between letters so words always look the same. There may be some ***'kerning',*** e.g., there will be less space between an 'o' and an 'x', as in box, than there will be between two o's as in book. TEX then puts these hboxes together to form larger hboxes (this time, a collection of words), using ***'glue',*** in place of mortar for the inter-word and inter-sentence spacings. The glue has properties of being ***stretchable*** and, to a lesser extent, ***shrinkable.*** This list of boxes and ***glue*** can then be expanded or compressed to meet the '*hsize*' dimension that has been preset by PLAIN .TEX to 6.5 in, or that you have changed by typing \hsize **4.25in**, for example.

Plain TEX takes care of such details as putting slightly more space after 'commas and after periods and by allowing these globs of glue to stretch more and shrink less than that allowed for the normal glue between words. While this is desirable in most cases, there are times when it is not desired and then you should use the backslash-space control sequence, '\ ', (that is a backslash followed by a space) to guarantee a normal space, or perhaps use the tie symbol, '~', in place of the space after an abbreviation such as in 'Fig. 23' or *Mr.* Smith, where it is also desirable to prevent the sentence from being broken between the abbreviation and the following number or word.

Actually, TEX treats an entire paragraph as a unit and tries to distribute the text into as many lines as are required with the individual lines meeting the *hsize* requirement without excessive stretching or shrinking. If this cannot be done, TEX then tries to hyphenate. TEX will usually find several ways that the paragraph can be broken, and it will then pick the way that it thinks is best. The paragraph will then be broken up into a sequence of *hboxes*, each of which should meet the specified line width.

TEX will complain with an 'overfull hbox' message, if it is unable to meet the tolerances that have been specified. Your task then is to inspect the offending line or lines to see if there is some simple way to overcome the difficulty; perhaps a discretionary hyphen is needed. Lacking such a remedy, you can force a line break at an earlier place in the offending line or earlier in the paragraph, but this is seldom desirable. It is usually better simply to reconstruct the sentence or paragraph so as to avoid the trouble. Actually, there are a number of ways that you can anticipate trouble and provide for it in advance, as you will learn through experience.

An Example

Perhaps, it is time to stop and show you an example, and what better example than the title page of this manual itself, which was typed using plain TEX. This example may seem a bit too complicated for a beginner but it does provide a convenient vehicle to illustrate a number of TEX's idiosyncrasies that you need to know about and that are hard to explain in abstract terms. So do go through it line by line with the explanation that follows.

My input file is divided into pages, although this is not required and some installations may not permit this nicety. TEX, of course, ignores page breaks in the input, just as it ignores line breaks.

The first part of the input file defines some control sequences and contains:

```
\font\ninerm=cmr9  \font\eightrm=cmr8  \font\sixrm=cmr6  \font\csc=cmcsc
\font\seal=stan70 % For use to produce the Stanford seal.
\def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125em X}

\magnification=1200  % This magnifies everything by 1.2.
\parskip 10pt plus 1pt  % This puts some empty space between paragraphs
\parindent 0pt  % Paragraphs are not to be indented
```

The next part of the input, file contains:

```
\nopagenumbers % The title page is not to be numbered.
\null\vskip-46pt % Put the first line 46 points higher than normal
\hbox to 6.5truein (November 1983 \hfil Report No. STAN-CS-83-985)
   % A convenient way to mske a box of specific size.
\vskip .1in % Skip down 0.1 inch
\line (Stanford Department of Computer Science\hfil (Version 1)}
\vfill % This and similar commands later, will divide the space evenly.
\centerline{\bf First-Grade \TeX}
\vskip .1in
\centerline{\bf A Beginner's \TeX\ Manual)
\vskip .25in
\centerline{by}
\centerline{Arthur L. Samuel)
\vfill
\centerline{\seal S}
\vfill
This manual is based on the publications of Donald E. Knuth who originated
the \TEX\ system and on the recent work of Professor Knuth and his many
students and collaborators who have helped bring the TEX82 system to its
present advanced state of development. The \TEX\ logo that is used in this
manual is a trademark of The American Mathematical Society. The preparation
of this report was supported in part by National Science Foundation grant
 IST-820/926 and by the System Development Foundation.
\eject\end % \eject, only, would be used if other pages were to follow.
```

So let us consider this example in detail.

We first note that the *control sequence* '\font' is used to assign names to several fonts that PLAIN. TEX had loaded but did not name. Actually, only one of these fonts was used on the cover page but it is good practice to start your input file by naming all of the other fonts that you intend to use beside those in plain TEX's standard set. Note that this same ***control sequence*** would have instructed TEX to copy the font metrics (the dimensional information that TEX actually needs) from the named file, if this ***font*** information had not been preloaded, as it had been by PLAIN .TEX.

Next comes a definition for the TEX logo. You will probably not need to cause letters to be artificially displaced from their normal positions, but if you do, here is an example. It also serves as a model for other ***control sequence*** definitions that you may find occasion to use. The'**\magnfication**' ***control sequence*** has already been explained (under **Fonts).**

The ***controlsequence*** '\parskip=lOpt plus lpt'isanexample of a general class of ***control sequences*** that take dimensions. I could have written it as '**parskip10pt** plus lpt minus **1pt**' had I been willing for the space to be shrunk by as much as lpt on some occasions and stretched on others. Incidentally, plain TEX sets '**\parindent=20pt**', but this has

been redefined for this manual by '**\parindent** Opt'. Note that the use of the '**=**' sign is optional but that a unit of measure must be specified even when the value is zero. So much for the first part.

After telling TEX that the title page is not to be numbered, the information for the title page starts with the control sequence '**\null**', which, as its name implies, normally does nothing. So why use it? One of TEX's idiosyncrasies is that it gobbles up extra spaces that you leave between words, extra space between paragraphs, and any extra space that might be left over when it has just introduced a page break. So, without the '**\null**', TEX will assume that the negative space called for by the '\vskip-46pt' control sequence was left, over from a previous page and simply gobble it up. The '**\null**' control sequence starts the new page by putting almost nothing (actually an *hbox* that contained nothing) on the first line. Now since the page has been started, the negative-space control sequence is honored. The \vskip primitive is an example of a control sequence that takes a dimension. Looking down the page, you will see other examples.

The next ***control sequence,*** '\hbox to 6.5 **truein**', also takes a dimension, but in this case using the extra word 'to'. The dimension in this ***case*** is in ***true*** inches since I did not want the value to be subject to the **\magnification** command. Two lines below this is another ***control sequence,*** '**\line**', that does essentially the same thing, in the context of this example, since ***\line*** is defined in PLAIN. TEX to produce an ***\hbox*** to the current dimension of ***\hsize***, which happens to be 6.5 true inches. Were I later to alter the page width by changing ***\hsize***, I would have to find and alter all ***control sequences*** of the first variety if I wanted the lines that they produced to still line up with the rest of the text, so the **\line** form is better in this case.

The text that is to be put in each of these two *hboxes* is enclosed in ***grouping*** symbols and contains still another ***control sequence,*** to wit, \hfil. This tells TEX where to put the extra space needed to fill out the text to exactly 6.5 inches, with the text before the \hfil ***flush*** left and the text to the right of it ***flush right.*** Without this \hfil ***control sequence,*** TEX complains (Underfull box (badness 10000)), and then it would print the first of these two lines as:

November 1983 Report No. STAN-CS-83-985

But to continue with the example: **\centerline** is another ***control sequence*** that requires the use of ***grouping*** symbols to enclose the text to be centered, that is, if you want more than the single next character to be centered. Then there is an embedded ***control sequence*** within the ***grouping symbols*** which specifies that the centered text is **to** be in **bold face type**.

It can be a bit confusing ***to*** observe that some ***control sequences*** take arguments within ***grouping*** symbols while others are embedded within the ***grouping*** symbols along with the text that they affect. There is always a logical reason for this difference in treatment. When TEX has interpreted the centering command, it must look ahead to determine how much text there is to be centered before taking any action, and it expects to find this

information within a pair of grouping symbols. By way of contrast, TEX can start putting text in a different font, without needing to know the range over which this command is to be effective. So, if you want the range of action to be limited, you put the font designation within the grouping symbols and the former font choice is reinstated when the closing symbol is reached. For a more detailed explanation, see "The TEXbook" by Don Knuth.

Next note the use of three **\vfill** ***control sequences*** to divide the excess vertical space that remains available on the page into three equal portions. The fact that I used one 'l' in \hfil and two 'l' in \vf ill was deliberate, to call attention to the fact that both forms work for both horizontal or vertical fills. The two-l variety simply is more stretchable than the one-l variety and will do all of the stretching if both forms are used together.

Finally, there is some normal text at the bottom of the page and then the commands \e j ect **\end** that cause TEX to finish off the page, to close out the DVI file and to terminate the session. Had this page been a normal text page, I would have used the ***control sequence*** **\bye**, the recommended way to stop, defined as \par\vf**ill\supere** j ect\end. This closes the paragraph, checks to see that there is no text yet to be processed and causes the last page to be filled out with blank space, if necessary, instead of spreading out the text to occupy the entire page, as we wanted in this case.

It will help your understanding if you make a file containing a title page for some paper that you have written or intend to write by copying this sample and modifying it as needed. Then try to run TEX on this file. Unless you are extremely lucky, you will be apt to get an error message that you will not understand-so read on. If you have been lucky you might try changing the page width by putting '**\hsize=4truein**' at the top of your input file, or better yet start TEX so that you get the '**' prompt, and then type '**\hsize=4truein**', a space, and then **\input** and the name of your file.

Understanding Error Messages

Most of the errors that the beginner is apt to make, other than simple typos, will have to do with 1) missing or misplaced ***grouping*** symbols, 2) attempts ***to*** use ***control sequences*** in the wrong context, and 3) a failure to understand some of the principles that TEX uses in deciding how to break the text into lines and these lines into pages with the result that you ask it to do some quite impossible things.

TEX will report that "**\end** occurred inside group at level 1" if a single '**}**' is missing and "too many **}'s**" if the contrary condition exists. Usually, it will have reported all sorts of strange things due to a mismatch of ***grouping*** symbols. So find and fix such troubles before checking further. A somewhat similar problem may arise if you do not having matching $ signs. You will usually get the message 'Missing $ inserted' but only after some otherwise normal text has been improperly handled.

TEX may get confused if you tell it to do something out of context, for example, to do something that relates to the formatting of words into sentences when it is busy putting sentences onto pages. At any given time, TEX will be operating in any one of six different

modes, as will be explained below. If you type `'h'` in response to an error message that you do not understand, TEX will try to help you by explaining what it thought that it was doing. Often the help message will suggest a way to recover from the error.

Finally, you need to know that TEX decides how to format the text by considering the amount of 'badness' that is charged against each possible arrangement. Penalties are assigned to each possible line break point, usually automatically, although you can specify a penalty if you wish. These penalties measure the undesirability of a break occurring at each particular place. Demerits are assigned, 1) to each line for different features as to its departure from the ideal, 2) for the presence of adjacent lines that differ from each other too much in their inter-word spacing or that are too similiar in certain obnoxious ways, for example, in both ending with a hyphen, and 3) for poor paragraphing, such as leaving a ***widow*** word to appear at the top of a new page. If TEX is unable to find a solution that will meet certain tolerance limits, it will complain, usually about the 'badness', and expect you to do something to fix the difficulty. Should you want TEX to be more tolerant, you can increase the value of *\tolerance* from 200 to, say, 1000.

The Six Modes

When TEX is processing your text,, it is constantly switching between six different modes of operation. The details need concern you only when TEX reports an error on your part or when you are doing something special like making a table or displaying an equation.

The two most obvious ***modes*** are 1) the horizontal ***mode*** when TEX is putting letters together to form words, and when it is putting spaces or glue between words, in preparation for making hboxes after the line breaks are chosen, and 2) the vertical mode when it is stacking these hboxes (or already prepared vboxes) vertically to go on a page. TEX switches from horizontal mode to vertical ***mode*** when it encounters something that clearly indicates that it should be in the vertical mode, such, for example, as 1) an empty line or the control sequence **\par**, both of which tells it to start a new paragraph, or 2) a v-type control sequence, such as `\vf ill` or `\vskip .1in`, both of which were used in the example on page 8. TEX switches from vertical ***mode*** to horizontal ***mode*** when it encounters an ordinary character or any one of several horizontal ***mode*** control sequences such, for example, as **\indent** or **\noindent** with obvious meanings.

The six ***modes*** are:

Mode	Used when building
Vertical	Main vertical list for a page
Horizontal	Horizontal list for a paragraph
Internal vertical	Vertical list for a vbox
Restricted horizontal	Horizontal list for an hbox
Math	Mathematical formula for a horizontal list
Displayed math	Math formula on a separate line.

You signal the entering and leaving of the math ***mode*** by the use of a single '$' sign as in

'**$<math formula>$**', and the entering and leaving of the display math *mode* by the use of a pair of $ signs as in '**$$<formula to be displayed>$$**'. The display math mode is also useful for tables that may not actually involve mathematical material, because it centers material on the page and introduces a space above and below the material.

A typical use of restricted horizontal mode occurs when you center some material on the page by writing '**\centerline{\bf** First Grade \TeX **}**'. TEX will be in the vertical mode on encountering the '**\centerline**' control sequence; it, will shift to the restricted horizontal mode while processing the material delineated by the braces and then back to the vertical mode to add this to the material that is to form the page. Restricted horizontal mode differs from regular horizontal mode because **\par** and $$ do not end a paragraph or start a display; a single line is always produced.

Making Tables

TEX provides two different, methods for typesetting tables. The first method apes the typewriter's tab-setting ability, and is preferred for very large tables that may extend over several pages. Tabs, set by the \settabs command, are preserved when you introduce some ordinary text (or even with a grouping-symbol-delineated insertion of a table with a different set of tab settings). This makes it possible to have several different tables, on the same page or even on different pages, that are similiar in their columnar alignments. The disadvantage is that you must specify where these tab settings are to be, either by giving the number of columns, assuming that they are to be evenly spaced, or by supplying a set of sample entries made up of the largest entry for each individual column. This can be a bit difficult to do if the table contains characters of different widths.

The more general method, which we will consider first, gives TEX the task of setting the tab positions for each column to meet the possibly varying maximum widths of the material that goes into these different columns. This **\halign** method allows you to achieve an optimum design for each table, but with a possible lack of uniformity between different, tables. Most of the tables in this manual were made using this method. Having made these tables, were I to find it necessary to replace an entry or even to add another column, I could depend upon TEX to make the necessary adjustments. This method should not be used for large tables that span many pages, as TEX must read the entire table in order to determine the column widths.

The '\halign' Alignment Method

The table near the bottom of page 4 will be used as an example. This was typed as:

```
$$\vbox {\tt \halign {\hfil #\hfil && \quad \hfil #\hfil \cr
\\rm& \\sl& \\it& \\tt& \\bf\cr
\rm Roman& \sl Slanted& \it Italic& \tt Typewriter& \bf Boldface\cr
}}$$
```

Note that the first line and the last line are concerned with setting up the table, while the

material that actually goes into the table appears in the second and third lines only. Had there been more lines to the table, this portion would be have been expanded accordingly.

The '**$$**' at the start and end of this input-text sample is used here to center the table and to introduce some space above and below it. As explained earlier, the use of the double-dollar-sign construct normally causes TEX to enter the displayed math mode. By following the $$ with the \vbox control sequence, the other math-mode effects are temporarily suspended so that the table will appear in normal non-math characters. The \vbox has the additional effect of preventing TEX from splitting the table between pages. The '**{**', that follows, and the very last, '**}**' on the last line are the grouping symbols that define the material that is to be put in the vbox. Since much of the material is to be in 'typewriter' type, we start with the '**\tt**' control sequence, and then comes the '**\halign**' control sequence that requires a second set of matching braces to define the scope of the table. After the opening brace there follows a series of statements, each ending with the control sequence '**\cr**'.

The rest of the first line contains the first statement, sometimes called the preamble. This is a template that specifies how the different columns of the table are to be treated. It does this by referring to the material that is to go into any specific column by the parameter symbol '**#**', while the different columns are separated by the symbol '**&**'.

After the **\halign{** and before the first **&** we find \hf**il**#\hf il, which tells TEX that the material in the first, column is to be centered (by putting the same amount of stretchable glue on each side). Normally, this would be followed by a single **&** symbol, then another column specification and so on for as many columns as desired, each of which could differ in some way from the other columns, finally terminating with a **\cr**. In this case, all of the subsequent columns are to be treated the same and TEX provides a shortcut, this being the use of two adjacent **&** symbols to tell TEX to use as many columns as are later supplied, using the same pattern that appears between the **&&** and the terminating **\cr**. A **\quad** of extra space is put before the second and subsequent columns, to keep the columns separate. Incidentally, the **&** symbol has the useful property of ignoring any extra spaces that immediately follow it, so I have added spaces after these symbols to make the example easier to read.

After the template there follows the two lines of data (each ending with a **\cr**) for the two lines of the resulting table. There is nothing unusual about the first text line except for the use of backslashes in pairs to tell TEX that the backslashes are, in fact, to be printed. The second line is a bit unusual since each column is to be printed in a different font. You will note that we have not had to use braces to delimit the range of action of the font-specifying control sequences. This is beause the **\halign** control sequence automatically limits the range of action of control sequences to the individual table entries where they are used. If no font, specification appears for any specific entry, then the default font is used. This table has but two lines, but for longer tables one simply adds as many lines as desired, always ending each line with a **\cr**.

A slightly more complicated situation is illustrated by the table near the bottom of page 2.

Here, the amount of material to go on each line is such that it is difficult to decide how much space to leave between the different columns. So we leave this decision to TEX by removing the **\quad** command from the preamble and by specifying a value for the stretchable glue that is to be put between columns (and at the left and right) as \tabskip lem plus 2em minus .5em. The table specification then becomes:

```
$$\vbox{\tabskip lem plus 2em minus .5em
\halign to \hsize{\hfil #\hfil &&\hfil #\hfil \cr
```

The specification **\halign** to \hsize tells TEX to use the stretchability and shrinkability of \tabskip so that the width of the line is the value of \hsize.

We will not be concerned with the details of typing the text material itself but one additional feature is worth noting. Extra vertical space has been inserted between the different sections of the table by extra lines in the source file that read '**\noalign {\bigskip}**', where '\bigskip' is defined below. In general any desired extra (vertical mode) material may be so introduced, including one or more lines of text. PLAIN. TEX defines three convenient vertical skip instructions and it is usually better to use these, for reasons of uniformity, than to specify any specific \vskip. These control sequences are:

Macro	Equivalent to
\smallskip	\vskip 3pt plus lpt minus lpt
\medskip	\vskip 6pt plus 2pt minus 2pt
\bigskip	\vskip 12pt plus 4pt minus 4pt

The table on page 3 is an example of a still more complicated situation in which the material to go into one column is too long to be contained in one line. This requires several new features in the preamble which reads:

```
$$\vbox{\halign{\tt \hfil #\hfil\quad &\tt \hfil #\hfil\quad &
\vtop{\hsize=26em\strut #\strut}\cr
```

The first thing to note is the use of a \vbox, this time a special one called \vtop, to contain the text for the last column. The \vtop differs from a \vbox in that it is aligned at the baseline of its top line. The width of this box is defined by **\hsize=26em**, a figure that was arrived by cut and try. The **\struts** were introduced to make sure that the space allowed for the \vtop would not be adversely affected if the last line of text in the box happened not to have any descenders, or if the first line had no tall letters, since \vboxes are normally made only high enough to enclose the material that they contain. A **\strut** produces a zero-width invisible box of the correct height and depth for the font, being used. Incidentally, a character can be assigned a zero width and still not be invisible. It would still print but then it would be over-printed by the next character.

If you want to make tables that are significantly more complicated than those used so far, you need to be an expert so you will have to study The TEXbook. The following example illustrates still other features. This example is taken from The TEXbook and attributed to

Machael Lesk as published in the Bell Laboratories Computing Science Technical Report 49 (1976).

AT&T Common Stock		
Year	Price	Dividend
1971	41-54	$2.60
2	41-54	2.70
3	4655	2.87
4	40-53	3.24
5	45-52	3.40
6	5 l-59	.95*

* (first quarter only)

This table was produced by typing:

```
$$\vbox{\tabskip=0pt \offinterlineskip
\halign to150pt{\strut#& \vrule#\tabskip=1em plus2em& \hfil#& \vrule#&
\hfil#\hfil& \vrule#& \hfil#& \vrule#\tabskip=0pt\cr \noalign{\hrule}
& & \multispan5\hfil AT\&T Common Stock\hfil& \cr \noalign{\hrule}
& & \omit\hidewidth Year\hidewidth& & \omit\hidewidth Price\hidewidth& &
\omit\hidewidth Dividend\hidewidth& \cr \noalign{\hrule}
& & 1971& & 41--54& & \$2.60& \cr \noalign{\hrule}
& & 2& & 41--54& & 2.70& \cr \noalign{\hrule}
& & 3& & 46--55& & 2.87& \cr \noalign{\hrule}
& & 4& & 40--53& & 3.24& \cr \noalign{\hrule}
& & 5& & 45--52& & 3.40& \cr \noalign{\hrule}
& & 6& & 51--59& & .95\rlap*& \cr \noalign{\hrule}
\noalign{\smallskip}
& \multispan7* (first quarter only)\hfil\cr}}$$
```

The most obvious difference, between this table and the ones discussed so far, is the use of so-called ***'rules'***, to enclose and separate the entries. You will also note that one entry extends over several columns within the area defined by the outside ***rules*** and the last entry also extends over several columns. Finally, while the numerical items in the Year column and in the Dividend column seem to be '***flush right***', the words 'Year' and 'Dividend' and entry '.95*' are certainly not.

Referring to the listing, we first reset some parameters. Setting \tabskip=Opt may not be needed since ***\tabskip*** is usually set to zero, but this is a precaution. TEX normally puts some *tabskip* glue before the first column, between columns, and after the last column in each line of an alignment (if ***\tabskip*** is not zero), and we will want to take advantage of this feature later to help center the entries.

The \of f interlineskip ***control sequence*** is used to set the usual interline spacing to zero. This prevents TEX from interposing glue between the the individual **\vrule** segments, glue that would prevent them from abutting each other properly. Having done this, we must then specify the vertical space assigned to these **\vrules** (and to the text as well) by using

a \strut. As explained earlier, this produces a zero-width invisible box of the correct, height for the font being used and the \vrules are, of course, produced to this height.

The \halign to <dimen> comes next,, then the preamble. The first thing to note about the preamble is that eight columns are specified, not just the three that contain the data. A template is specified for each column. The first template contains # \strut, but no values are specified to match the # sign so only the \strut is ever put in this column, but this does fix the vertical space allowed for the *vrule* segments and for the text. The initial & in the listings of the row data that follow the preamble is all that is necessary to cause the \strut to apply to the entries in each row. The four vrule segments in each row after the first are similiarly called into action by the second, fourth, and sixth & symbol in the row specification and by the final \cr.

The first row of data containing 'AT&T Common Stock' is an exception to the general rule, in that this caption spans five columns as signalled by the control sequence \multispan 5. The next row also illustrates how you ***omit*** the application of the formatting rules specified in the preamble by using \omit ***control sequence*** and how you prevent, the widths of the entries from being used to determine column width by the \hidewidth ***control sequences.*** Finally, the \rlap command is used to overlap the * symbol in one entry so that its width is not considered in making the alignment. One final detail, note the \noalign{\hrule} statements that define the horizontal ***rules.***

The Fixed-Column- Width '\settab' Method

The fixed-column-width \settab method can be used in simple cases when the fixed width restriction is of no consequence. It also functions best where the entries can all be left aligned. The table on page 4 does not meet these restrictions but it will reveal most of the complication that you are apt to encounter. So you type the following, noting the use of the '\+' ***control sequence*** to start each row and the '\cr' ***control sequence to*** end it:

```
$$\vbox{\tt \settabs 5 \columns
\+\\rm& \\sl& \\it& \\tt&z \\bf\cr
\+\rm Roman& \sl Slanted& \it Italic& \tt Typewriter& \bf Boldface\cr
}$$
```

You get five columns, each flush left:

\rm	\sl	\it	\tt	\bf
Roman	*Slanted*	***Italic***	Typewriter	**Boldface**

You can make the columns narrower by calling for an extra unused column. You can center the text in the columns by introducing '\hf ill' ***control sequences*** both before and after each entry (using '\hf il' ***control sequences*** will not work) and by adding an extra '&' symbol after the last entries (to force TEX to handle the last, column like all of the rest). All of this gets to be more trouble than to use the '\halign' method but you can type:

```
$$\vbox{\tt \settabs 6 \columns
\+\hfill\\rm\hfill&\hfill\\sl\hfill&
\hfill\\it\hfill&\hfill\\tt\hfill&\hfill\\bf\hfill&\cr
\+\rm\hfill Roman\hfill&\sl\hfill Slanted\hfill&\it\hfill
Italic\hfill&\tt\hfill Typewriter\hfill&\bf\hfill Boldface\hfill&\cr}$$
```

The Variable-Column- Width '\settab' Method

The variable-column-width method line does a somewhat better job. This method requires you to supply a sample row for the table, which you type in place of the '5 \columns' on the first line (this sample is used for dimensions only and is not printed). You use the largest entry taken from each column and you add some desired amount, of space between entries. In this case the sample line will have all of its entries taken from the second row of the table which is a bit tricky, since these entries are all in different typefaces.

Here you type:

```
$$\vbox{\tt\settabs\+\rm Roman&\quad\sl Slanted&
\quad\it Italic&\quad\tt Typewriter&\quad\bf Boldface&\cr
\+\hfill\\rm\hfill&\hfill\\sl\hfill&
\hfill\\it\hfill&\hfill\\tt\hfill&\hfill\\bf\hfill&\cr
\+\rm\hfill Roman\hfill&\sl\hfill Slanted\hfill&\it\hfill
Italic\hfill&\tt\hfill Typewriter\hfill&\bf\hfill Boldface\hfill&\cr}$$
```

to get a table similiar to the one on page 4 except for a slight difference in centering.

Typing Mathematical Formulas

Math formulas are, by tradition, printed using different conventions from those used for printing ordinary text. Some of these differences may seem quite trivial but they contribute greatly to the legibility and yes even to the beauty of well printed mathematical texts. Fortunately, TEX knows about these math conventions and can usually be trusted to follow them.

Some of these conventions are: 1) Alphabetic characters are printed in ***math italics*** rather than in a more normal font and ***math italics*** differs in minor detail from normal ***italics.*** 2) Many special symbols are used that are not available in the normal text mode, such as Greek letters, conventional mathematical symbols, such as $\geq, \leq, \neq, \approx$, etc. 3) The spacing conventions are different from those used in ordinary text. In fact,, TEX completely ignores the spaces that you use in typing formulas and applies its own spacing rules (e.g., $x + y$ instead of x + y or of $x+y$). 4) The alignment rules are also quite special as you will observe in later examples, and many symbols are frequently printed in enlarged forms. 5) Superscripts and subscripts are used and these are normally automatically reduced in size, such as in x^{a^b} (typed as `$x^{a^b}$`) or as in x_{a_b} (typed as `$x_{a_b}$`). 6) It is also customary to use a somewhat different style for printing formulas that are displayed on separate lines from the style used for formulas within text.

A math mode formula is enclosed within either single dollar signs '**$**...**$**', if the formula is to appear in a line with ordinary text, or within double dollar signs if the formula is to be typed on a separate line, in so-called display math mode, thus '**$$**...**$$**'.

Eight different styles of math typesetting are used, four regular styles and four "cramped" variations. The letters and numbers are typeset in three different sizes, these being text size ($x + y - z$), script, size ($x+y-z$), and scriptscript size ($x+y-z$). If you want to enforce a size that TEX might not otherwise use, as was just done, you use the control sequences **\scriptstyle** and **\scriptscriptstyle**. The display style and the text style use the same size letters and numbers but differ in the sizes used for large operators, in the positioning of exponents and in the way they handle fractions. Should you want to enforce the use of either the text style or the display style, you can use the control sequences **\textstyle** or **\displaystyle**. A discussion of how these styles are used in fractions will be deferred until later.

Symbols classed as large operators (including the 'summation' and 'integration' signs $\sum$ and $\int$) are printed in a larger size when in display math mode from the way they appear in normal math mode. If you type **$\sum_{n=1}^m$** you get $\sum_{n=1}^{m}$ and if you type **$\int_{-\infty}^{+\infty}$** you get $\int_{-\infty}^{+\infty}$. On the other hand, these same expressions typed as display formulas yield:

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

The control sequence **\nolimits** will cause the summation sign to have subscripts and superscripts just as in the text mode, and the control sequence **\limits**, if typed after the \int will cause the integration limits to appear above and below the intergal sign thus:

$$\sum\nolimits_{n=1}^{m} \quad \text{and} \quad \int\limits_{-\infty}^{+\infty}$$

The following example illustrates the way in which TEX will increase the size of certain symbols and will make the horizontal lines long enough to extend over the subformula to which they apply and high enough or low enough not to bump into it. Typing: **$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}}$$** produces:

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

The control sequence '**\over**' is used to specify fractions, and the fraction line is made as long as needed. TEX assumes that everything within the same sub-formula grouping before the '**\over**' control sequence is to go above the line and everything that follows, again within the same sub-formula grouping, is to go below the line that '**\over**' generates. A variant,, called '**\above**', allows you to specify the weight of the line, thus:

`$${a\over 1+b}+c$$` yields $\frac{a}{1+b}+c$ and `$${a\above2pt1+b}$$` yields $\frac{a}{1+b}$

As an aside, there are four related control sequences:

`$${a\atop 1+b}, {n+1\choose k}, {m\brack n}, {m\brace n-1}$$` that yield:

$$\begin{matrix} a \\ 1+b \end{matrix} \qquad \binom{n+1}{k} \qquad \begin{bmatrix} m \\ n \end{bmatrix} \qquad \begin{Bmatrix} m \\ n-1 \end{Bmatrix}$$

TEX provides a somewhat similiar extendability with respect, to parentheses and other delimiters, this time to extend them vertically as required. By using the control sequences '`\left`' before a left delimiter and '`\right`' before the corresponding right delimiter, you can let TEX decide as to the correct size of delimiter to use. The '`\left`' and '`\right`' control sequences have an auxiliary function in that they also act as grouping symbols so it is not necessary to use the '`{`' and '`}`' symbols with them (unless, of course, these are the delimiters to be printed, and then they must be typed as `\{` and `\}`). It is necessary to use the `\left` and `\right` *control* sequences in matching pairs (although the delimiters themselves need not match).

The following example illustrates these features:

You type `$$1+\left(1\over 1-x^2\right)^3$$` to get $1+\left(\frac{1}{1-x^2}\right)^3$

This example introduces yet another feature, namely the handling of exponents, which was alluded to briefly once before. The '`^`', is used to indicate a superscript and its companion the '`_`', is used to indicate a subscript. Both normally apply only to the next character so you must use grouping symbols for those cases where a multi-character superscript or subscript is to be shown. Thus if you type:

`$$1+\left(1\over 1-x^{21}_3\right)^{32}$$` you get $1+\left(\frac{1}{1-x_3^{21}}\right)^{32}$

A series of control sequences for specifying delimiter sizes are available for use in situations where TEX may not make the desired choice. For opening delimiters these are, in order of increasing size: `\bigl`, `\Bigl`, `\biggl`, and `\Biggl` and for closing delimiters they are `\bigr`, `\Bigr`, `\biggr`, and `\Biggr`. It should be noted that the `\bigl` and `\bigr` delimiters are larger than ordinary ones so that the difference can be perceived, yet small enough to be used within the text of a paragraph. Even larger delimiters than those named can be made by using the `\left` and the `\right` conventions and by adding an empty `\vbox` of the appropriate size within the field when necessary. For example, you can type:

```
$$\left(\vbox to 27pt{}\left(\vbox to 24pt{}\left(\vbox to 21pt{}
\Biggl(\biggl(\Bigl(\bigl(({\scriptstyle({\scriptscriptstyle
(\hskip3pt)})})\bigr)\Bigr)\biggr)\Biggr)\right)\right)\right)
    (With sets of three similar lines added for brackets and for braces)$$
```

to get

$$\Biggl(\biggl(\Bigl(\bigl((\,(\,(\,(\,)\,)\,)\,)\bigr)\Bigr)\biggr)\Biggr)\Biggl[\biggl[\Bigl[\bigl[[\,[\,[\,]\,]\,]\bigr]\Bigr]\biggr]\Biggr]\Biggl\{\biggl\{\Bigl\{\bigl\{\{\,\{\,\{\,\}\,\}\,\}\bigr\}\Bigr\}\biggr\}\Biggr\}$$

TEX recognizes 16 other basic delimiters that can be obtained in different sizes, either by using the **\left** and **\right** technique or by specifying their size as illustrated above for parentheses, brackets, and braces. There are restrictions as to the sizes available for **\langle**, **\rangle**, **/**, and **\backslash**, but not for the other extendable delimiters.

Additional extendable delimiters for use in Display Math mode:

For	You type	For	You type	For	You type	For	You type
$\lfloor$	**\lfloor**	$\langle$	**\langle**	$\vert$	**\vert**	$\downarrow$	**\downarrow**
$\rfloor$	**\rfloor**	$\rangle$	**\rangle**	$\Vert$	**\Vert**	$\Downarrow$	**\Downarrow**
$\lceil$	**\lceil**	$/$	**/**	$\uparrow$	**\uparrow**	$\updownarrow$	**\updownarrow**
$\rceil$	**\rceil**	$\backslash$	**\backslash**	$\Uparrow$	**\Uparrow**	$\Updownarrow$	**\Updownarrow**

Before considering matrices and other uses of large delimiters, let us dispose of the matter of fractions. It is usually wise to avoid the use of more than one **\over** in an equation and remember that there cannot be more than one **\over** in any one subexpression. In fact, it is often preferable to use the "slashed" fraction form, particularly for mathematical expressions that are not printed as separate displays.

When you do use more than one **\over**, you probably should introduce our old friends **\strut** and **\displaystyle** to keep the spacings and character sizes from shrinking alarmingly. Certainly,

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

typed as

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

looks better than

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

which is what you get without **\strut** and **\displaystyle**.

As an exercise, you might reproduce this example and make several copies, one without using **\strut** and another without \displaystyle, to reveal the exact effect of these refinements.

Perhaps the most common use of large delimiters is in printing matrices, where the **\matrix** macro can be used. This macro is similiar in many respects to the **\halign** macro that was discussed starting on page 12 except that \matrix only works in math mode, and no template need be supplied. To get:

$$A=\begin{pmatrix} x-x & 1 & 0 \\ 0 & x-x & 1 \\ 0 & 0 & x-x \end{pmatrix}$$

you type

```
$$A=\left(\matrix{x-\lambda&1&0\cr
0&x-\lambda&l\cr
0&0&x-\lambda\cr)\right)$$
```

It helps in typing a matrix such as this if you line up the columns, which you can do since TEX pays no attention to the spaces that you leave. So you might type:

```
$$A=\left|\matrix{
x-\mu & 1      & 0      \cr
0     & x-\mu & 1      \cr
0     & 0     & x-\mu \cr
}\right|$$
```

This, of course, produces:

$$A = \begin{vmatrix} x-p & 1 & 0 \\ 0 & x-\mu & 1 \\ 0 & 0 & x-\mu \end{vmatrix}$$

since I used vertical bars instead of parentheses, and I substituted \mu for **\lambda** just to be different. If you are wondering how I was able to preserve the spacings in showing you what I typed, I used plain TEX's macros, \parskip=Opt **\obeylines** \obeyspaces.

Ellipses (i.e., dots) are used in many places but they appear to good advantage in matrices where you might type:

```
$$A=\pmatrix{
a_{11}&a_{12}&\ldots&a_{1n}\cr
a_{21}&a_{22}&\ldots&a_{2n}\cr
\vdots&\vdots&\ddots&\vdots\cr
a_{m1}&a_{m2}&\ldots&a_{mn}\cr}$$
```

to get

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix}$$

Here \pmatrix is like **\matrix**, but it puts parentheses in for you.

The **\ldots** ***control sequence*** (as in the above matrix) is used between letters and commas while another ***control sequence,*** '**\cdots**', is used between signs and similiar operators as in a + *b* + -- + *z*. Incidentally, a single centered dot is a \cdot. Still another ***control sequence,*** '**\dots**', is used for ellipses in non-math text . . . , when needed.

Single braces are often used in refering to different cases and plain TEX provides a convenient, macro, you guessed it, called \cas es. This bears a remarkable resemblance to \pmatrix except that, 1) only one delimiter, a brace, is printed, 2) the entries are adjusted flush left in each of two columns, and *3)* the material that appears after the '&'

symbol is not in math mode unless specifically surrounded with '$' symbols. For example, to get:

$$|x| = \begin{cases} x, & \text{if } x \geq 0; \\ -x, & \text{otherwise.} \end{cases}$$

you type

```
$$|x|=\cases {x,&if $x\geO$;\cr -x,&otherwise.\cr}$$
```

If a left delimiter without a matching right delimiter (or the reverse) is needed elsewhere, TEX allows you to type '`\right.`' (or '`\left.`'), where the period acts as a null delimiter, terminating the extent of the grouping effect (or initiating it) without anything being printed.

Roman characters and words may be printed within mathematical formulas using three quite different mechanisms. For only an occasional letter or word, you can switch to roman and type:

`$\exp(x+{\rm constant})$` to get exp(x + constant)

This will still work for several words but, since spaces are ignored, you have to use the control sequence '`\ `' to preserve the spacings between words.

The second way is to use a \hbox, thus to get:

x^3 + lower order terms you type `$x^3+\hbox{lower order terms}$`

This scheme was used to type this very example and it has been used extensively throughout this manual. It has two obvious disadvantages, 1) the current (text) font will be used, and it may not be the font, you want (but this can be fixed) and 2) the content of the box will always be in the same size unless extra precautions are taken when the words are wanted in a different size, perhaps for use in a superscript or subscript.

Finally, if you plan to use a word or a fixed sequence of roman typed words frequently in different mathematical formulas, for example the sequence just used, then you can assign a name to it as one of your own macros thus:

```
\def\loterms{\hbox{\rm low order terms)).
```

Thereafter when you want the sequence 'low order terms' you simply type `\loterms`. The above example would then be typed as '`$x^3+\loterms$`'.

Since the names of the common mathematical function are always set in roman type, the following control sequences have been predefined in plain TEX:

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

These control sequences lead to roman type with appropriate spacing. Certain of these are treated as large operators (just like \sum), to wit: \det, \gcd, \inf, \lim, \liminf, \limsup, \max, \min, \Pr, and \sup. The following examples are taken from The TEXbook but rearranged to use the \cases control sequence, to demonstrate that \cases will work for a larger array than previously shown.

To get	You type (while in math mode)
$\sin 2\theta = 2\sin\theta\cos\theta$	`\sin2\theta=2\sin\theta\cos\theta`
$O(n \log n \log\log n)$	`O(n\log n\log\log n)`
$\Pr(X > x) = \exp(-x/\mu)$	`\Pr(X>x)=\exp(-x/\mu)`
$\max_{1\le n\le m} \log_2 P_n$	`\displaystyle\max_{1\le n\le m}\log_2P_n`
$\lim_{x\to 0} \frac{\sin x}{x} = 1$	`\displaystyle\lim_{x\to0}{\sin x\over x}=1`

Incidentally, I wanted these examples to be spaced a little farther apart than they would be with the spacing set by \cases. So I inserted **\noalign{\vskip2pt}** after every **\cr** (except the last one).

Punctuation should be used with caution. When a formula is followed by punctuation, such as a period or a comma, put the punctuation after the terminating $ sign, when the formula is in text (even in the extreme case when you type something like '$x=a$, b, or **c**.'; but put the punctuation before the terminating **$$**, when the formula is displayed. Note: Punctuation symbols have been omitted from the display equations used so far in this manual (so as not to confuse you), but I will start using them now.

Numbering isolated display formulas poses no special problem as plain TEX provides two control sequences, **\eqno** for numbers that are to go on the right, and **\leqno** for numbers that are to go on the left. In both cases, these control sequences go after the formula to be numbered. They act much like our old friend **&** in that they separate two fields; everything to the left is part of the formula and everything to the right up to the terminating $$ will appear as the equation number.

Thus, `$$(x+y)(x-y)=x^2-y^2.\eqno(21)$$` will produce:

$$(x+y)(x-y) = x^2 - y^2. \eqno(21)$$

If you type, **`$$(x+y)(x-y)=x^2-y^2.\leqno[21a]$$`** you will get:

$$[21a] \qquad (x+y)(x-y) = x^2 - y^2.$$

The formulas are centered in both cases (without regard for the presence of the formula numbers), and the formula numbers are in math style unless you specify otherwise.

Several display formulas, appearing together, can be aligned at any desired location (on an equal sign perhaps) by using the control sequence **\eqalign**, which works with **&** and

\cr in a manner somewhat similiar to the use of these markers in \matrix and \cases. For example, if you type:

```
$$\eqalign{ax^2+bx+c&=0\cr x&={-b\pm\sqrt{b^2-4ac}\over2a}.\cr}$$
```

you get

$$ax^2 + bx + c = 0$$
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

If you want the equations individually numbered, you use \eqalignno and add a second & with the number added as usual for each equation that is to be numbered. Typing:

```
$$\eqalignno{ax^2+bx+c&=0&(1)\cr
x&={-b\pm\sqrt{b^2-4ac}\over2a}.&(2)\cr}$$
```

produces

$$ax^2 + bx + c = 0 \tag{1}$$
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{2}$$

Using \eqalign (not \eqalignno) and adding \eqno(3), for example, will cause the set of formulas to be numbered as a group, with the number centered vertically with respect to the group, producing:

$$ax^2 + bx + c = 0$$
$$x = \frac{-\ b\ \&\ d\ -}{2a}. \tag{3}$$

It is also possible to introduce extra text lines between the different formula lines without disturbing the alignment, for example by including \noalign{\hbox{This is the text to be introduced)). Finally, using \leqno and \leqalignno, with an initial letter 'l' and with no other changes, will cause the formula numbers to be placed at the left.

Some fundamental differences between \eqalign and \eqalignno should be noted.

\eqalign makes a single vertically-centered vbox that is no wider than necessary, that cannot be broken between pages, and that can only take a single vertically-centered equation number. More than one \eqalign can be put on a line (if space permits).

\eqalignno generates full-width lines. These lines can be broken between pages and the individual lines can take individual line numbers. \eqno cannot be used with \eqalignno to assign a group equation number. Lines of normal text can, however, be placed between the \eqalignno equation lines by using the \noalign{\hbox{ . . . }} construction.

Long equations pose a difficult problem, particularly if they must still be aligned in some way with other equations in a group, and you may have to help TEX to do a satisfactory job. One solution is illustrated below.

$$\begin{aligned} x_n u_1 + \cdots + x_{n+t-1}u_t &= x_n u_1 + (ax_n + c)u_2 + \cdots \\ &\qquad + \bigl(a^{t-1}x_n + c(a^{t-2} + \cdots + 1)\bigr)u_t \\ &= (u_1 + au_2 + \cdots + a^{t-1}u_t)x_n + h(u_1,\ldots,u_t). \qquad (47) \end{aligned}$$

The first and third lines were aligned on the = signs as usual but then we wanted the second line to be aligned with the + sign as shown, and the \eqalignno control sequence makes no provision for secondary alignments. The solution is to use the control sequence **\phantom**{...} which causes TEX to leave the same amount of space that would be taken by the indicated text without printing it. So we put a **** in front of the second line and align it with the other two lines on the = sign. This does not work perfectly because the **\phantom** command creates an empty box of the size necessary to contain the indicated characters without regard for their surroundings and the space allowed before the = and + signs depends on the context. The net, result is that we have to add a bit of space defined by '**\;**'. So what we type is:

```
\eqalignno{x_nu_1+\cdots+x_{n+t-1}u_t
&=x_nu_1+(ax_n+c)u_2+\cdots\cr
&\phantom{=x_nu_1}\;+\bigl(a^{t-1}x_n+c(a^{t-2}+\cdots+1)\bigr)u_t\cr
&=(u_1+au_2+\cdots+a^{t-1}u_t)x_n+h(u_1,\ldots,u_t)\quad&(47)\cr}
```

Normally, you can depend on TEX to space things correctly, but when TEX needs your help while in math mode, you can use the following:

You type	\!	\,	\>	\;
To get	$-1/6$ quad	$1/6$ quad	$2/9$ quad	$5/18$ quad

Some other math features, not yet covered, are the use of **\prime** to produce prime superscripts and subscripts where you type:

$ y_1^\prime+y_2^{\prime\prime}+y_3^{\prime\prime\prime}$ to get $y_1' + y_2'' + y_3'''$,

the use of **\root** where, $\root 3 \of {x^2+y^2}$ produces $\sqrt[3]{x^2+y^2}$,

and the use of \mathstrut to inforce uniformity in the positioning, say, of the square root signs by typing $\sqrt{\mathstrut a}+\sqrt{\mathstrut d}+\sqrt{\mathstrut y}$ to get $\sqrt{a}+\sqrt{d}+\sqrt{y}$ instead of $\sqrt{a}+\sqrt{d}+\sqrt{y}$.

There are, also, two variations on **\phantom** that you will find useful, **\vphantom** with zero width and **\hphantom** with zero height, and depth, and there is **\smash** which tells TEX to print a subexpression but to assume that it has zero height and depth.

Some Odds and Ends

You are about ready to undertake ordinary typing on your own but you still do not know how to produce footnotes,* how to allow for insertions, and how to change the page format if you do not want the pages numbered at the bottom.

If you should like your footnotes to be numbered automatically, even this can be done by defining a new control sequence which might be called \note.[1] Before the first footnote[2] to be so numbered you write (as I have done):

```
\newcount\notenumber
\def\clearnotenumber{\notenumber=0}
\def\note{\advance\notenumber by1 \footnote{$^{\the\notenumber}$}}
\clearnotenumber
```

There is an art to inserting illustrations or other independently derived material into a text. A number of people are working on supplements to TEX to allow for the direct introduction of computer derived graphics into TEX output, but I will assume that you will be content to add photograph and graphical material manually.

Plain TEX provides for three basic type of insertions, \topinsert, \midinsert, and \pageinsert. These can only be given between paragraphs and not inside of boxes or other insertions. The general form for these is: \topinsert <vertical mode material that can have embedded paragraphs> **\endinsert**.

TEX tries to put a **\topinsert** at the top of the current, page, if there is still room when the \topinsert is encountered, otherwise it will be put at the top of the next page. If several **\topinsert** commands are given close together, some may be carried to still later pages. The \midinsert is put on the page in the position where it appears, if this is possible, otherwise it is handled as a **\topinsert**. A \pageinsert is automatically expanded to fill an entire page and put on the next page. Complications requiring human intervention may arise if TEX is asked to put pageinserts and footnotes on the same page and if both topinserts and footnote extensions are carried over to a following page.

I will now type a **\topinsert**, which, as you can see, appears at the top of the next page. Perhaps you should read it now, if you have not already done so.

---

* Like this, which was produced by typing '**\footnote{*}{Like** this, which was produced by typing . . . **}**' right along following the macro itself. TEX will usually do the right thing like putting the indicated mark (which is typed in the first set of braces) in the text where the macro **\footnote** appears and putting the footnote itself (which was typed in a second pair of braces) at the bottom of the same page and even dividing an extra long footnote between pages.

[1] This should be footnote number 1.

[2] You can also number equations automatically (but that's a different story).

This is a \topinsert.
This text is printed in this shape both to set it apart from the rest of the text, as befitting a \topinsert, and to demonstrate that it is possible to specify an essentially arbitrary paragraph shape by saying \parshape=n, where n is the number of lines, and by following this with the n sets of dimensions for these lines, each specified as one number for the indentation and a second number for the length of the line. For a more detailed explanation, see The TEXbook, chap 14, page 101. In this case, these lines, so specified, were put into a box and this box was used as a \topinsert.

Fig. E. This is a \topinsert

This example may be a bit too complicated to explain in detail but in essence it involved the creation of a box by the command **\setbox0=\vtop{ . . . }**, where the dots stand for the command \parshape 10 followed by the specification of the line indentations and lengths and then by the text itself. Having defined such a box it was then only necessary to give the commands:

```
\topinsert
\box0
\vskip10pt
\centerline{\bf Fig.  E. This is a \\topinsert}
\endinsert
```

to tell TEX to generate the \topinsert.

If there is any danger of an insertion that does not get made before the end of the appropriate section, you can force its printing by typing \vfill\supereject.

You should also know how to produce so-called leaders, such as these, which are often used in Contents tables.

Introduction 1
Toward Book Quality 2

The examples shown here were typed as:

```
\def\leaderfill{\leaders\hbox to 1em{\hss.\hss}\hfill}
\line{\qquad\qquad Introduction\leaderfill 1\qquad\qquad}
\line{\qquad\qquad Toward Book Quality\leaderfill 2\qquad\qquad}
```

Still another feature, with \parindent set, to some non-zero amount (it is set to 20pt in plain TEX and to 35pt for this example), the control sequence **\narrower** followed by **{** the desired contents of the paragraph ending with a **\smallskip** (also within the enclosing braces) **}**, will cause a paragraph, such as this one, to be narrowed by the \parindent amount.

Output Routines

Defining variant forms of output routines is properly in the domain of the TEXpert, but you may like to have a choice of at least, one other format which I am now using, starting with this page. This change was introduced by typing:

```
\nopagenumbers % suppress footlines
\headline={\ifodd\pageno\rightheadline \else\leftheadline\fi}
\def\rightheadline{\tenrm\hfil A Beginner's \TeX\ Manual\hfil\folio}
\def\leftheadline{\tenrm\folio\hfil First Grade \TeX \hfil}
```

Of course, if you want to get fancy, you can have TEX automatically use your current chapter or section headings as running heads, but this may be too much for the beginner.

Defining Macros

Several macros have been defined and used in this manual, a typical one being:

```
\def\loterms{\hbox{\rm low order terms)).
```

Such macros all begin with the control sequence '**\def**' followed by the new name and then the meaning to be assigned to this new name (enclosed in braces that are not a part of the definition). If braces are wanted as a part of the defintion they must be added.

As useful as these simple macros are, you will come across numerous situations where you will want to define a macro that can take parameters that are to be defined at the time the macro is used. You have already been introduced to control sequences of this sort that were defined in PLAIN. TEX, and that you have learned to use.

One such definition is: **\def\centerline#1{\line{\hss#1\hss}}**.

This says that if TEX encounters the command **\centerline**, it is next to look for a parameter that is here designated as Xl which you should have typed in braces following the **\centerline** command. TEX is then to apply the control sequence **\line** to the braced expression \hss#l\hss, where **\line** is itself a derived control sequence defined as **\def\line{\hbox to\hsize}**. In other **words-Center It on a Line!**

Two things are to be noted: 1. It is possible to use other control sequences within definitions and 2. The symbol # 1 is used to designate a parameter that is given to TEX at the time that the control sequence is invoked. Actually, up to 9 parameters can be so specified. The definition might then take the form **\def\zzz#1#2#3{a** complex expression in which #1, #2, and #3 can be used in any order and repeatedly if needed). The parameters, as listed following the name of the control sequence, must, however, be in serial order.

It is also possible to write macros that are conditional. A simple example of a conditional was used to define the current page format, as explained earlier on this page, when

we typed: **\headline={\ifodd\pageno\rightheadline\else\leftheadline\fi}**. Note particularly the use of \f i (if spelled backward) to end the conditional.

To round out our discussion of **\def**, we will note that there exists a **\let** primitive that is somewhat analogous to \def but differs from it in the timing of its execution. The difference can most easily be explained by noting that the expression \let\b=\a sets the value of \b to the value of \a at the time when this expression is read by TEX, while **\def\b{\a}** does not set the value of \b to the value of \a until this macro is executed.

Before naming a new macro, it is always wise to make sure that the proposed name is not already in use either by TEX itself or by the macro package that you intend to use. You can do this by interrogating TEX directly. Simply run TEX82 without specifying a file name and in response to the asterisk prompt type **\show** followed by the proposed name. If your macro package is not preloaded, you will want to load it before typing the **\show**.

For	A Macro	A Primitive	An undefined name
Typing	**\show\centerline**	\show\hbox	\show\zzzz
Shows	**>** \centerline=macro:	**>** \hbox=\hbox.	**> \zzzz=undefined**.
	#1->\line{\hss#1\hss}.	**<*>** \show\hbox	**<*>** \show\zzzz
	<*> \show\centerline	?	?
	?		

The line beginning with **#1->** for the **\centerline** case, shows that the macro expects one parameter and this parameter is then used in the expression that follows. The next to the last line is to show you what TEX has read and the last line is asking you what you want to do about it. A carriage return will restore the asterisk prompt, and you can then type another **\show**, or give a **\input** command.

By way of summary, there are several ways to assign meaning to a control sequence:

Typing	\f **ont\cs=\<f** ont name>	makes **\cs**	a font identifier
	\chardef\cs=\<num.>		a character code
	\countdef\cs=<num.>		a \count register
	\def\cs...**{...}**		a macro.
Typing	**\let\cs=<token>**	gives **\cs**	the token's current meaning.

You will need to know quite a bit more before you can be an efficient macro designer. The whole story appears in Chapter 20 of The TEXbook.

Two special macro packages are soon to be available. LaTEX by Leslie Lamport probably will appeal to the computer programmer, while AMS-TEX by Michael Spivak probably will appeal to the mathematician. Both packages allow the user to specify one of several formatting styles by name and in so doing they greatly simplify the task of using TEX. Facil TEX by Max Diaz is also a contender but no date is available for its conversion to TEX82.

Appendix

Characters that are reserved for special purposes: \ { } $ & # % ^ _ ~

To print - – — − “text” ¿ ¡ $ # & % å Æ œ Œ à
You type - -- --- $-$ “text” ?‘ !‘ \$ \# \& \% \ae \AE \oe \OE \aa

Å ß ø Ø à é ô ü ȳ ñ ṗ ǐ š ǰ ů̇ ḵ
\AA \ss \o \O \'a \'e \^o \¨u \=y \~n \.p \v\ı \v s \H\j \t\i u \b k

ç ḥ ł Ł † ‡ § ¶ ¢ £ & © …
\c c \d h \l \L \dag \ddag \S \P \rlap/c \it\$ \it\& \copyright \dots

Line break controls: \break \allowbreak \nobreak \hbox{unbreakable}
dis\-cre\-tion\-ary hy\-phens virgule/breakpoint: \slash

Breakable horizontal spaces:
\␣ normal interword space
\enskip this much
\quad this much
\qquad this much
\hskip <arbitrary dimen>

Unbreakable horizontal spaces:
~ normal interword space
\enspace this much
\thinspace this much
\negthinspace thismuch
\kern <arbitrary dimen>

Vertical spaces: \smallskip \medskip \bigskip

Page break controls: \eject \supereject \nobreak \goodbreak \filbreak
Vertical spaces and good breakpoints: \smallbreak \medbreak \bigbreak

\line{\downbracefill} \hrulefill \dotfill \line{\upbracefill}

\rm Roman \sl *Slant* \it ***Italic*** \tt Typewriter \bf **Boldface**

Typical font table (\rm for amr10). If necessary, one can type \char'10 to get Φ, etc.

Γ Δ Θ Λ Ξ Π Σ Υ Φ Ψ Ω ff fi fl ffi ffl ı ȷ ` ´ ¸ ß æ œ ø Æ Œ Ø
ˊ ¡ ” # $ % & ’ ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; ¡ = ¿ ?
@ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [“] ˆ ˙
‘ a b c d e f g h i j k l m n o p q r s t u v w x y z – — ˝ ˜ ¨

Greek Letters, in Math mode (**γ**, or Γ for upper case where shown)

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

Miscellaneous special symbols, available in Math mode

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

Binary operators (in addition to $+$, $-$, and $*$) available in Math mode

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

Relations (in addition to $<$, $>$, and $=$), available in Math mode

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

Negated relations (the \not symbol is considered to have zero width)

$\not<$ \not<	$\not>$ \not>	$\not=$ \not=
$\not\leq$ \not\leq	$\not\geq$ \not\geq	$\not\equiv$ \not\equiv
$\not\prec$ \not\prec	$\not\succ$ \not\succ	$\not\sim$ \not\sim
$\not\preceq$ \not\preceq	$\not\succeq$ \not\succeq	$\not\simeq$ \not\simeq
$\not\subset$ \not\subset	$\not\supset$ \not\supset	$\not\approx$ \not\approx
$\not\subseteq$ \not\subseteq	$\not\supseteq$ \not\supseteq	$\not\cong$ \not\cong
$\not\sqsubseteq$ \not\sqsubseteq	$\not\sqsupseteq$ \not\sqsupseteq	$\not\asymp$ \not\asymp

Arrows for use in Math mode

← \leftarrow	⟵ \longleftarrow	↑ \uparrow
⇐ \Leftarrow	⟸ \Longleftarrow	⇑ \Uparrow
→ \rightarrow	⟶ \longrightarrow	↓ \downarrow
⇒ \Rightarrow	⟹ \Longrightarrow	⇓ \Downarrow
↔ \leftrightarrow	⟷ \longleftrightarrow	↕ \updownarrow
⇔ \Leftrightarrow	⟺ \Longleftrightarrow	⇕ \Updownarrow
↦ \mapsto	⟼ \longmapsto	↗ \nearrow
↘ \searrow	↙ \swarrow	↖ \nwarrow
↩ \hookleftarrow	↪ \hookrightarrow	

Some alternate names used in Math mode

≠	\ne	{	\{	A \land	→	\to	I \vest
≤	\le	}	\}	v \lor	←	\gets	‖ \Vert
≥	\ge	3	\owns	¬ \lnot			

Large operators as used in Math ($. . . $) and in Math Display ($$ . . . $$) modes

Σ	$\sum$	\sum	∩	$\bigcap$	\bigcap	⊙	$\bigodot$	\bigodot
∏	$\prod$	\prod	∪	$\bigcup$	\bigcup	⊗	$\bigotimes$	\bigotimes
∐	$\coprod$	\coprod	⊔	$\bigsqcup$	\bigsqcup	⊕	$\bigoplus$	\bigoplus
∫	$\int$	\int	∨	$\bigvee$	\bigvee	⊎	$\biguplus$	\biguplus
∮	$\oint$	\oint	∧	$\bigwedge$	\bigwedge			

Extendable delimiters, in addition to ( ) { } [and], used in Display Math mode

⌊	\lfloor	⟨	\langle	|	\vert	↓	\downarrow
⌋	\rfloor	⟩	\rangle	‖	\Vert	⇓	\Downarrow
⌈	\lceil	/	/	↑	\uparrow	↕	\updownarrow
⌉	\rceil	\	\backslash	⇑	\Uparrow	⇕	\Updownarrow

Function names that print in roman type, for use when in Math mode

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

Dimensions, preset by PLAIN .TEX, that you may want to change

\hsize=6.5in	\parindent=20pt	\abovedisplayskip= 12pt plus 3pt minus 9pt
\vsize=8.9in	\parskip=Opt plus lpt	\belowdisplayskip= 12pt plus 3pt minus 9pt

Macros for setting ordinary text

\frenchspacing	\break	\bigbreak	\hidewidth	\ttraggedright
\nonfrenchspacing	\nobreak	\line	\multispan# 1	\leavevmode
\normalbaselines	\allowbreak	\leftline# 1	\cleartabs	\hrulefill
\null	\filbreak	\rightline# 1	\hang	\dotfill
\obeyspaces	\goodbreak	\centerline# 1	\textindent#l	\downbracefill
\loop#l\repeat	\eject	\rlap#1	\item	\upbracefill
\iterate	\supereject	\llap# 1	\itemitem	
\nointerlineskip	\smallbreak	\underbar# 1	\narrower	
\offinterlineskip	\medbreak	\strut	\raggedright	

Macros for setting math

\joinrel	\vphantom	\openup	\cases#1	\ldots
\relbar	\hphantom	\eqalign# 1	\matrix# 1	\cdots
\Relbar	\phantom	\displaylines# 1	\pmatrix# 1	\vdots
\bowtie	\mat hst rut	\eqalignno# 1	\ bordermat rix# 1	\ddots
\models	\smash	\leqalignno# 1		

Parameters preset by PLAIN. TEX that may be reset, with caution

\vbadness= 1000	\tracinglostchars= 1	\delimitershortfall=5pt
\linepenalty= 10	\uchyph= 1	\nulldelimiterspace= 1.2pt
\hyphenpenalty=50	\newlinechar=-1	\scriptspace=0.5pt
\exhyphenpenalty=50	\delimiterfactor=901	\parindent=20pt
\binoppenalty=700	\showboxbreadth=5	\parskip=Opt plus lpt
\relpenalty=500	\showboxdepth=3	\parfillskip=Opt plus lfil
\hbadness= 1000	\adjdemerits=10000	\topskip= 10pt
\clubpenalty=150	\hfuzz=O. lpt	\maxdepth=4pt
\widowpenalty=150	\vfuzz=0.1pt	\normalbaselineskip= 12pt
\displaywidowpenalty=50	\overfullrule=5pt	\normallineskip= lpt
\brokenpenalty=100	\hsize=6.5in	\jot=3pt
\predisplaypenalty= 10000	\vsize=8.9in	\tolerance=200

\doublehyphendemerits= 10000	\belowdisplayskip= 12pt plus 3pt minus 9pt
\finalhyphendemerits=5000	\belowdisplayshortskip=7pt plus 3pt minus 4pt
\thinmuskip=3mu	\smallskipamount=3pt plus lpt minus lpt
\medmuskip=4mu plus 2mu minus 4mu	\medskipamount=6pt plus 2pt minus 2pt
\thickmuskip=5mu plus 5mu	\bigskipamount= 12pt plus 4pt minus 4pt
\interdisplaylinepenalty=100	\abovedisplayskip= 12pt plus 3pt minus 9pt
\interfootnotelinepenalty= 100	\abovedisplayshortskip=0pt plus 3pt

Unassigned parameters, set to zero automatically

\pausing	\tracingparagraphs	\tracingrestores	\spaceskip
\tracingonline	\tracingpages	\leftskip	\hoffset
\tracingmacros	\tracingoutput	\rightskip	\voffset
\tracingstats	\tracingcommands	\tabskip	

Fonts preloaded and named by PLAIN. TEX.
Can be magnified by $\sqrt{1.2}$, 1.2, $(1.2)^2$, $(1.2)^3$, $(1.2)^4$, and $(1.2)^5$.

roman text	\tenrm amr10	\sevenrm amr7	\fiverm amr5
boldface extended	\tenbf ambx10	\sevenbf ambx7	\fivebf ambx5
math italic	\teni ammi10	\seveni ammi7	\fivei ammi5
math symbols	\tensy amsy10	\sevensy amsy7	\fivesy amsy5
math extension	\tenex amex10		
typewriter	\tentt amtt10		
slanted roman	\tensl amsl10		
text italic	\tenit amti10		

Fonts preloaded by PLAIN. TEX but un-named.
Can be magnified by $\sqrt{1.2}$, 1.2, and $(1.2)^2$ (\magstephalf, \magstep1, and \magstep2).

roman text	amr9 amr8 amr6	sans serif	amss 10 amssq8
math italic	ammi9 ammi8 ammi6	sans serif italic	amssi 10 amssqi8
math symbols	amsy9 amsy8 amsy6	slanted roman	amsl9 ams18
bold face	ambx9 ambx8 ambx6	typewriter	amtt9 amtt8
text italic	amti9 amti8 amti7	slanted typewriter	amslt t 10

unslanted text italic	amu10	Dunhill style	amdunh10
bold math italic	ambi 10	sans serif bold extended	amssbx10
bold math symbols	ambsy 10	caps and small caps	amcsc 10

Preloaded in magnified form for titles but un-named.

amr7 scaled \magstep4 amtt 10 scaled \magstep2 amssbx10 scaled \magstep2