LaTeX

The macro package for TeX

by
Leslie Lamport.

Edition 1.1

July 1993

This is edition 1.1 of the LaTeX documentation, and is for the Texinfo that is distributed as part of Version 18 of GNU Emacs. It uses version 2.90 of the 'texinfo.tex' input file.

This is translated from LATEX.HLP v1.0a in the VMS Help Library. This pre-translation version was written by George D. Greenwade of Sam Houston State University. It has been edited to this form by Paul Nothard of Edinburgh University.

The original (latex.texi and latex2.texi) was distributed by Stephen Gilmore, stg.ed.ac.uk, August 26th 1993.

Version 1.1 was made by Piet van Oostrum <piet.ruu.nl> on Dec 14, 1993 by merging and cleaning up latex.texi and latex2.texi.

Licensing Information

The programs currently being distributed that relate to Texinfo include two portions of GNU Emacs, plus two other separate programs (`texindex` and `texinfo.tex`). These programs are *free*; this means that everyone is free to use them and free to redistribute them on a free basis. The Texinfo related programs are not in the public domain; they are copyrighted and there are restrictions on their distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of these programs that they might get from you.

Specifically, we want to make sure that you have the right to give away copies of the programs that relate to Texinfo, that you receive source code or else can get it if you want it, that you can change these programs or use pieces of them in new free programs, and that you know you can do these things.

To make sure that everyone has such rights, we have to forbid you to deprive anyone else of these rights. For example, if you distribute copies of the Texinfo related programs, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must tell them their rights.

Also, for our own protection, we must make certain that everyone finds out that there is no warranty for the programs that relate to Texinfo. If these programs are modified by someone else and passed on, we want their recipients to know that what they have is not what we distributed, so that any problems introduced by others will not reflect on our reputation.

The precise conditions of the licensees for the programs currently being distributed that relate to Texinfo are found in the General Public Licensees that accompany them. The programs that are part of GNU Emacs are covered by the GNU Emacs copying terms (see section “License” in *The GNU Emacs Manual*), and other programs are covered by licensees that are contained in their source files.

1 Overview of LaTeX and Local Guide

The LaTeX command typesets a file of text using the TeX program and the LaTeX Macro package for TeX. To be more specific, it processes an input file containing the text of a document with interspersed commands that describe how the text should be formatted. It produces at least three files as output:

1. A "Device Independent", or '`.dvi`' file. This contains commands that can be translated into commands for a variety of output devices. You can view the output of LaTeX by using a program such as `xdvi`, which actually uses the '`.dvi`' file.
2. A "transcript" or '`.log`' file that contains summary information and diagnostic messages for any errors discovered in the input file.
3. An "auxiliary" or '`.aux`' file. This is used by LaTeX itself, for things such as sectioning.

For a description of what goes on inside TeX, you should consult *The TeXbook* by Donald E. Knuth, ISBN 0-201-13448-9, published jointly by the American Mathematical Society and Addison-Wesley Publishing Company.

For a description of LaTeX , you should consult:

LaTeX*: A Document Preparation System*, by Leslie Lamport, ISBN 0-201-15790-X, published jointly by the American Mathematical Society and Addison-Wesley Publishing Company, 1985.

LaTeX *for Engineers & Scientists*, by David J. Buerger, ISBN 0-07-008845-4, McGraw-Hill, 1990.

The LaTeX *Cookbook*, by F. Teagle, Informatics Department, Rutherford Appelton Laboratory, Chilton, Didcot, Oxon, OX11 0QX, 1991.

LaTeX *for Everyone*, by Jane Hahn, available from PCTeXin California, and from the TeX Users Group, 1991.

LaTeX *Line by Line: Tips and Techniques for Document Processing* by Antoni Diller, Chichester: John Wiley and Sons, 1992.

A Guide to LaTeX , by Helmut Kopka and Patrick Daly, Addison-Wesley, 1992.

2 Commands

A LaTeX command begins with the command name, which consists of a \ followed by either (a) a string of letters or (b) a single non-letter. Arguments contained in square brackets, `[]`, are optional while arguments contained in braces, `{}`, are required.

NOTE: LaTeX is case sensitive. Enter all commands in lower case unless explicitly directed to do otherwise.

2.1 Counters

Everything LaTeX numbers for you has a counter associated with it. The name of the counter is the same as the name of the environment or command that produces the number, except with no `\`. (enumi - enumiv are used for the nested enumerate environment.) Below is a list of the counters used in LaTeX's standard document styles to control numbering.

```
part            paragraph       figure          enumi
chapter         subparagraph    table           enumii
section         page            footnote        enumiii
subsection      equation        mpfootnote      enumiv
subsubsection
```

2.1.1 \addtocounter

```
\addtocounter{counter}{value}
```

The `\addtocounter` command increments the `counter` by the amount specified by the `value` argument. The `value` argument can be negative.

2.1.2 \alph

```
\alph{counter}
```

This command causes the value of the `counter` to be printed in alphabetic characters. The `\alph` command causes lower case alphabetic alphabetic characters, i.e., `a, b, c...` while the `\Alph` command causes upper case alphabetic characters, i.e., `A, B, C...`.

2.1.3 \arabic

`\arabic{counter}`

The `\arabic` command causes the value of the `counter` to be printed in arabic numbers, i.e., `3`.

2.1.4 \fnsymbol

`\fnsymbol{counter}`

The `\fnsymbol` command causes the value of the `counter` to be printed in a specific sequence of nine symbols that can be used for numbering footnotes.

eg. From 1-9: $* \dagger \ddagger \S \P \| ** \dagger\dagger \ddagger\ddagger$

NB. `counter` must have a value between 1 and 9 inclusive.

2.1.5 \newcounter

`\newcounter{foo}[counter]`

The `\newcounter` command defines a new counter named `foo`. The optional argument `[counter]` causes the counter `foo` to be reset whenever the counter named in the optional argument is incremented.

2.1.6 \roman

`\roman{counter}`

This command causes the value of the `counter` to be printed in roman numerals. The `\roman` command causes lower case roman numerals, i.e., `i, ii, iii`..., while the `\Roman` command causes upper case roman numerals, i.e., `I, II, III`....

2.1.7 \setcounter

```
\setcounter{counter}{value}
```

The `\setcounter` command sets the value of the `counter` to that specified by the `value` argument.

2.1.8 \usecounter

```
\usecounter{counter}
```

The `\usecounter` command is used in the second argument of the `list` environment to allow the counter specified to be used to number the list items.

2.1.9 \value

```
\value{counter}
```

The `\value` command produces the value of the `counter` named in the mandatory argument. It can be used where LaTeX expects an integer or number, such as the second argument of a `\setcounter` or `\addtocounter` command, or in:

```
\hspace{\value{foo}\parindent}
```

It is useful for doing arithmetic with counters.

2.2 Cross References

One reason for numbering things like figures and equations is to refer the reader to them, as in "See Figure 3 for more details."

2.2.1 \label

```
\label{key}
```

A `\label` command appearing in ordinary text assigns to the `key` the number of the current sectional unit; one appearing inside a numbered environment assigns that number to the `key`.

A `key` can consist of any sequence of letters, digits, or punctuation characters. Upper and lowercase letters are different.

2.2.2 \pageref

```
\pageref{key}
```

The `\pageref` command produces the page number of the place in the text where the corresponding `\label` command appears. ie. where `\label{key}` appears.

2.2.3 \ref

```
\ref{key}
```

The `\ref` command produces the number of the sectional unit, equation number, ... of the corresponding `\label` command.

2.3 Definitions

2.3.1 \newcommand

```
\newcommand{cmd}[args]{def}
\renewcommand{cmd}[args]{def}
```

These commands define (or redefine) a command.

`cmd` A command name beginning with a `\`. For `\newcommand` it must not be already defined and must not begin with `\end`; for `\renewcommand` it must already be defined.

`args` An integer from 1 to 9 denoting the number of arguments of the command being defined. The default is for the command to have no arguments.

`def` The text to be substituted for every occurrence of `cmd`; a parameter of the form `#n` in `cmd` is replaced by the text of the nth argument when this substitution takes place.

2.3.2 \newenvironment

```
\newenvironment{nam}[args]{begdef}{enddef}
\renewenvironment{nam}[args]{begdef}{enddef}
```

These commands define or redefine an environment.

`nam` The name of the environment. For `\newenvironment` there must be no currently defined environment by that name, and the command `\nam` must be undefined. For `\renewenvironment` the environment must already be defined.

`args` An integer from 1 to 9 denoting the number of arguments of the newly-defined environment. The default is no arguments.

`begdef` The text substituted for every occurrence of `\begin{nam}`; a parameter of the form `#n` in `cmd` is replaced by the text of the nth argument when this substitution takes place.

`enddef` The text substituted for every occurrence of `\end{nam}`. It may not contain any argument parameters.

2.3.3 \newtheorem

```
\newtheorem{env_name}{caption}[within]
\newtheorem{env_name}[numbered_like]{caption}
```

This command defines a theorem-like environment.

`env_name` The name of the environment to be defined. A string of letters. It must not be the name of an existing environment or counter.

`caption` The text printed at the beginning of the environment, right before the number. This may simply say "Theorem", for example.

`within` The name of an already defined counter, usually of a sectional unit. Provides a means of resetting the new theorem counter **within** the sectional unit.

`numbered_like`
The name of an already defined theorem-like environment.

The `\newtheorem` command may have at most one optional argument.

2.3.4 \newfont

```
\newfont{cmd}{font_name}
```

Defines the command name `cmd`, which must not be currently defined, to be a declaration that selects the font named `font_name` to be the current font.

2.4 Document Styles

Valid LaTeX document styles include:

- article
- report
- letter
- book

Other document styles are often available. See ⟨undefined⟩ [Overview], page ⟨undefined⟩, for details. They are selected with the following command:

```
\documentstyle [options] {style}
```

The options for the different styles are:

1. `article` - 11pt, 12pt, twoside, twocolumn, draft, fleqn, leqno, acm
2. `report` - 11pt, 12pt, twoside, twocolumn, draft, fleqn, leqno, acm
3. `letter` - 11pt, 12pt, fleqn, leqno, acm
4. `book` - 11pt, 12pt, twoside, twocolumn, draft, fleqn, leqno

If you specify more than one option, they must be separated by a comma.

2.4.1 \flushbottom

The `\flushbottom` declaration makes all text pages the same height, adding extra vertical space when necessary to fill out the page.

This is the standard for the book style.

2.4.2 \onecolumn

The `\onecolumn` declaration starts a new page and produces single-column output.

2.4.3 \raggedbottom

The `\raggedbottom` declaration makes all pages the height of the text on that page. No extra vertical space is added.

2.4.4 \twocolumn

The `\twocolumn` declaration starts a new page and produces two-column output.

2.5 Environments

LaTeX provides a number of different paragraph-making environments. Each environment begins and ends in the same manner.

```
\begin{environment-name}
.
.
.
\end{environment-name}
```

2.5.1 array

```
\begin{array}{col1col2...coln}
column 1 entry & column 2 entry ... & column n entry \\
 .
 .
 .
\end{array}
```

Math arrays are produced with the array environment. It has a single mandatory argument describing the number of columns and the alignment within them. Each column, `coln`, is specified by a single letter that tells how items in that row should be formatted.

- `c` - for centred
- `l` - for flushleft
- `r` - for flushright

Column entries must be separated by an `&`. Column entries may include other LaTeX commands. Each row of the array must be terminated with the string `\\`.

2.5.2 center

```
\begin{center}
Text on line 1 \\
Text on line 2 \\
.
.
.
```

```
\end{center}
```

The `center` environment allows you to create a paragraph consisting of lines that are centred within the left and right margins on the current page. Each line must be terminated with the string `\\`.

2.5.2.1 \centering

This declaration corresponds to the `center` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`. The text of a figure or table can be centered on the page by putting a `\centering` command at the beginning of the figure or table environment.

Unlike the `center` environment, the `\centering` command does not start a new paragraph; it simply changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like quote) that ends the paragraph unit.

2.5.3 description

```
\begin{description}
\item [label] First item
\item [label] Second item
.
.
.
\end{description}
```

The `description` environment is used to make labelled lists. The `label` is bold face and flushed right.

2.5.4 enumerate

```
\begin{enumerate}
\item First item
\item Second item
.
.
.
```

```
\end{enumerate}
```

The `enumerate` environment produces a numbered list. Enumerations can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments.

Each item of an enumerated list begins with an `\item` command. There must be at least one `\item` command within the environment.

2.5.5 eqnarray

```
\begin{eqnarray}
math formula 1 \\
math formula 2 \\
.
.
.
\end{eqnarray}
```

The `eqnarray` environment is used to display a sequence of equations or inequalities. It is very much like a three-column `array` environment, with consecutive rows separated by `\\` and consecutive items within a row separated by an `&`. An equation number is placed on every line unless that line has a `\nonumber` command.

2.5.6 equation

```
\begin{equation}
 math formula
\end{equation}
```

The `equation` environment centres your equation on the page and places the equation number in the right margin.

2.5.7 figure

```
\begin{figure}[placement]
```

```
 body of the figure

\caption{figure title}
\end{figure}
```

Figures are objects that are not part of the normal text, and are usually "floated" to a convenient place, like the top of a page. Figures will not be split between two pages.

The optional argument `[placement]` determines where LaTeX will try to place your figure. There are four places where LaTeX can possibly put a float:

1. `h` (Here) - at the position in the text where the figure environment appears.
2. `t` (Top) - at the top of a text page.
3. `b` (Bottom) - at the bottom of a text page.
4. `p` (Page of floats) - on a separate float page, which is a page containing no text, only floats.

The standard report and article styles use the default placement `tbp`.

The body of the figure is made up of whatever text, LaTeX commands, etc. you wish. The `\caption` command allows you to title your figure.

2.5.8 flushleft

```
\begin{flushleft}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{flushleft}
```

The `flushleft` environment allows you to create a paragraph consisting of lines that are flushed left, to the left-hand margin. Each line must be terminated with the string `\\`.

2.5.8.1 \raggedright

This declaration corresponds to the `flushleft` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushleft` environment, the `\raggedright` command does not start a new paragraph; it simply changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like quote) that ends the paragraph unit.

2.5.9 flushright

```
\begin{flushright}
Text on line 1 \\
Text on line 2 \\
.
.
.
\end{flushright}
```

The `flushright` environment allows you to create a paragraph consisting of lines that are flushed right, to the right-hand margin. Each line must be terminated with the string `\\`.

2.5.9.1 \raggedleft

This declaration corresponds to the `flushright` environment. This declaration can be used inside an environment such as `quote` or in a `parbox`.

Unlike the `flushright` environment, the `\raggedleft` command does not start a new paragraph; it simply changes how LaTeX formats paragraph units. To affect a paragraph unit's format, the scope of the declaration must contain the blank line or `\end` command (of an environment like quote) that ends the paragraph unit.

2.5.10 itemize

```
\begin{itemize}
\item First item
\item Second item
.
.
.
\end{itemize}
```

The `itemize` environment produces a "bulleted" list. Itemizations can be nested within one another, up to four levels deep. They can also be nested within other paragraph-making environments.

Each item of an `itemized` list begins with an `\item` command. There must be at least one `\item` command within the environment.

2.5.11 list

```
\begin{list}{label}{spacing}
\item First item
\item Second item
.
.
.
\end{list}
```

The `{label}` argument specifies how items should be labelled. This argument is a piece of text that is inserted in a box to form the label. This argument can and usually does contain other LaTeX commands.

The `{spacing}` argument contains commands to change the spacing parameters for the list. This argument will most often be null, i.e., `{}`. This will select all default spacing which should suffice for most cases.

2.5.12 minipage

```
\begin{minipage}[position]{width}
 text
\end{minipage}
```

The `minipage` environment is similar to a `\parbox` command. It takes the same optional `position` argument and mandatory `width` argument. You may use other paragraph-making environments inside a minipage.

Footnotes in a `minipage` environment are handled in a way that is particularly useful for putting footnotes in figures or tables. A `\footnote` or `\footnotetext` command puts the footnote at the bottom of the minipage instead of at the bottom of the page, and it uses the `mpfootnote` counter instead of the ordinary `footnote` counter.

NOTE: Don't put one minipage inside another if you are using footnotes; they may wind up at the bottom of the wrong minipage.

2.5.13 picture

```
\begin{picture}(width,height)(x offset,y offset)
.
 picture commands
.
\end{picture}
```

The `picture` environment allows you to create just about any kind of picture you want containing text, lines, arrows and circles. You tell LaTeX where to put things in the picture by specifying their coordinates. A coordinate is a number that may have a decimal point and a minus sign - a number like `5`, `2.3` or `-3.1416`. A coordinate specifies a length in multiples of the unit length `\unitlength`, so if `\unitlength` has been set to `1cm`, then the coordinate 2.54 specifies a length of 2.54 centimeters. You can change the value of `\unitlength` anywhere you want, using the `\setlength` command, but strange things will happen if you try changing it inside the picture environment.

A position is a pair of coordinates, such as `(2.4,-5)`, specifying the point with x-coordinate `2.4` and y-coordinate `-5`. Coordinates are specified in the usual way with respect to an origin, which is normally at the lower-left corner of the picture. Note that when a position appears as an argument, it is not enclosed in braces; the parentheses serve to delimit the argument.

The `picture` environment has one mandatory argument, which is a `position`. It specifies the size of the picture. The environment produces a rectangular box with width and height determined by this argument's x- and y-coordinates.

The `picture` environment also has an optional `position` argument, following the `size` argument, that can change the origin. (Unlike ordinary optional arguments, this argument is not contained in square brackets.) The optional argument gives the coordinates of the point at the lower-left corner of the picture (thereby determining the origin). For example, if `\unitlength` has been set to `1mm`, the command...

```
\begin{picture}(100,200)(10,20)
```

...produces a picture of width 100 millimeters and height 200 millimeters, whose lower-left corner is the point (10,20) and whose upper-right corner is therefore the point (110,220). When you first draw a picture, you will omit the optional argument, leaving the origin at the lower-left corner. If you then want to modify your picture by shifting everything, you just add the appropriate optional argument.

The environment's mandatory argument determines the nominal size of the picture. This need bear no relation to how large the picture really is; LaTeX will happily allow you to put things outside the picture, or even off the page. The picture's nominal size is used by TeX in determining how much room to leave for it.

Everything that appears in a picture is drawn by the `\put` command. The command...

```
\put (11.3,-.3){...}
```

...puts the object specified by `...` in the picture, with its reference point at coordinates (11.3,-.3). The reference points for various objects will be described below.

The `\put` command creates an `LR box`. You can put anything in the text argument of the `\put` command that you'd put into the argument of an `\mbox` and related commands. When you do this, the reference point will be the lower left corner of the box.

2.5.13.1 \circle

```
\circle[*]{diameter}
```

The `\circle` command produces a circle of the specified diameter. If the `*`-form of the command is used, LaTeX draws a solid circle.

2.5.13.2 \dashbox

```
\dashbox{dash_length}(width,height){...}
```

The `\dashbox` has an extra argument which specifies the width of each dash. A dashed box looks best when the `width` and `height` are multiples of the `dash_length`.

2.5.13.3 \frame

```
\frame{...}
```

The `\frame` command puts a rectangular frame around the object specified in the argument. The reference point is the bottom left corner of the frame. No extra space is put between the frame and the object.

2.5.13.4 \framebox

```
\framebox(width,height)[position]{...}
```

The `\framebox` command is exactly the same as the `\makebox` command, except that it puts a frame around the outside of the box that it creates.

The `framebox` command produces a rule of thickness `\fboxrule`, and leaves a space `\fboxsep` between the rule and the contents of the box.

2.5.13.5 \line

```
\line(x slope,y slope){length}
```

The `\line` command draws a line of the specified `length` and `slope`.

2.5.13.6 \linethickness

```
\linethickness{dimension}
```

Declares the thickness of horizontal and vertical lines in a picture environment to be `dimension`, which must be a positive length. It does not affect the thickness of slanted lines and circles, or the quarter circles drawn by `\oval` to form the corners of an oval.

2.5.13.7 \makebox

```
\makebox(width,height)[position]{...}
```

The `\makebox` command for the picture environment is similar to the normal `\makebox` command except that you must specify a `width` and `height` in multiples of `\unitlength`.

The optional argument, `[position]`, specifies the quadrant that your text appears in. You may select up to two of the following:

- `t` - Moves the item to the top of the rectangle
- `b` - Moves the item to the bottom
- `l` - Moves the item to the left
- `r` - Moves the item to the right

See `\makebox-snt [\makebox], page \makebox-pg`.

2.5.13.8 \multiput

```
\multiput(x coord,y coord)(delta x,delta y){number of copies}{object}
```

The `\multiput` command can be used when you are putting the same object in a regular pattern across a picture.

2.5.13.9 \oval

```
\oval(width,height)[portion]
```

The `\oval` command produces a rectangle with rounded corners. The optional argument, `[portion]`, allows you to select part of the oval.

- `t` - Selects the top portion
- `b` - Selects the bottom portion
- `r` - Selects the right portion
- `l` - Selects the left portion

2.5.13.10 \put

```
\put(x coord,y coord){ ... }
```

The `\put` command places the item specified by the mandatory argument at the given coordinates.

2.5.13.11 \shortstack

```
\shortstack[position]{... \\ ... \\ ...}
```

The `\shortstack` command produces a stack of objects. The valid positions are:

- `r` - Moves the objects to the right of the stack
- `l` - Moves the objects to the left of the stack
- `c` - Moves the objects to the center of the stack (default)

2.5.13.12 \vector

```
\vector(x slope,y slope){length}
```

The `\vector` command draws a line with an arrow of the specified length and slope. The `x` and `y` values must lie between -4 and +4, inclusive.

2.5.14 quotation

```
\begin{quotation}
 text
\end{quotation}
```

The margins of the `quotation` environment are indented on the left and the right. The text is justified at both margins and there is paragraph indentation. Leaving a blank line between text produces a new paragraph.

2.5.15 quote

```
\begin{quote}
 text
\end{quote}
```

The margins of the `quote` environment are indented on the left and the right. The text is justified at both margins. Leaving a blank line between text produces a new paragraph.

2.5.16 tabbing

```
\begin{tabbing}
text \= more text \= still more text \= last text \\
second row \>  \> more \\
.
.
.
\end{tabbing}
```

The `tabbing` environment provides a way to align text in columns. It works by setting tab stops and tabbing to them much the way you do with an ordinary typewriter.

2.5.16.1 \=

The `\=` command sets the tab stops.

2.5.16.2 \>

The `\>` command causes LaTeX to advance to the next tab stop.

2.5.16.3 \<

The `\<` command allows you to put something to the left of the local margin without changing the margin.

2.5.16.4 \+

The `\+` command moves the left margin of the next and all the following commands one tab stop to the right.

2.5.16.5 \-

The `\-` command moves the left margin of the next and all the following commands one tab stop to the left.

2.5.16.6 \'

The `\'` command moves everything that you have typed so far in the current column, everything starting from the most recent `\>`, `\<`, `\'`, `\\`, or `\kill` command, to the right of the previous column, flush against the current column's tab stop.

2.5.16.7 \`

The ``\` `` command allows you to put text flushed right against any tab stop, including tab stop 0. However, it can't move text to the right of the last column because there's no tab stop there. The ``\` `` command moves all the text that follows it, up to the `\\` or `\end{tabbing}` command that ends the line, to the right margin of the tabbing environment. There must be no `\>` or `\'` command between the ``\` `` and the command that ends the line.

2.5.16.8 \kill

The `\kill` command allows you to set tab stops without producing text. It works just like the `\\` except that it throws away the current line instead of producing output for it. The effect of any `\=`, `\+` or `\-` commands in that line remain in effect.

2.5.17 table

```
\begin{table}[placement]

 body of the table

\caption{table title}
\end{table}
```

Tables are objects that are not part of the normal text, and are usually "floated" to a convenient place, like the top of a page. Tables will not be split between two pages.

The optional argument `[placement]` determines where LaTeX will try to place your table. There are four places where LaTeX can possibly put a float:

- `h` : Here - at the position in the text where the table environment appears.
- `t` : Top - at the top of a text page.
- `b` : Bottom - at the bottom of a text page.
- `p` : Page of floats - on a separate float page, which is a page containing no text, only floats.

The standard `report` and `article` styles use the default placement `[tbp]`.

The body of the table is made up of whatever text, LaTeX commands, etc., you wish. The `\caption` command allows you to title your table.

2.5.18 tabular

```
\begin{tabular}[pos]{cols}
column 1 entry & column 2 entry ... & column n entry \\
.
.
.
\end{tabular}

               or

\begin{tabular*}{width}[pos]{cols}
column 1 entry & column 2 entry ... & column n entry \\
.
.
.
\end{tabular*}
```

These environments produce a box consisting of a sequence of rows of items, aligned vertically in columns. The mandatory and optional arguments consist of:

`width` Specifies the width of the `tabular*` environment. There must be rubber space between columns that can stretch to fill out the specified width.

`pos` Specifies the vertical position; default is alignment on the center of the environment.

- `t` - align on top row
- `b` - align on bottom row

`cols` Specifies the column formatting. It consists of a sequence of the following specifiers, corresponding to the sequence of columns and intercolumn material.

- `l` - A column of left-aligned items.
- `r` - A column of right-aligned items.
- `c` - A column of centred items.
- `|` - A vertical line the full height and depth of the environment.
- `@{text}` - This inserts text in every row. An `@`-expression suppresses the intercolumn space normally inserted between columns; any desired space between the inserted text and the adjacent items must be included in text. An `\extracolsep{wd}` command in an `@`-expression causes an extra space of width `wd` to appear to the left of all subsequent columns, until countermanded by another `\extracolsep` command. Unlike ordinary intercolumn space, this extra space is not suppressed by an `@`-expression. An `\extracolsep` command can be used only in an `@`-expression in the `cols` argument.
- `p{wd}` - Produces a column with each item typeset in a parbox of width `wd`, as if it were the argument of a `\parbox[t]{wd}` command. However, a `\\` may not appear in the item, except in the following situations:
  1. inside an environment like `minipage`, `array`, or `tabular`.
  2. inside an explicit `\parbox`.
  3. in the scope of a `\centering`, `\raggedright`, or `\raggedleft` declaration. The latter declarations must appear inside braces or an environment when used in a `p-column` element.
- `*{num}{cols}` - Equivalent to `num` copies of `cols`, where `num` is any positive integer and `cols` is any list of column-specifiers, which may contain another `*-expression`.

2.5.18.1 \cline

```
\cline{i-j}
```

The `\cline` command draws horizontal lines across the columns specified, beginning in column `i` and ending in column `j`, which are identified in the mandatory argument.

2.5.18.2 \hline

The `\hline` command will draw a horizontal line the width of the table. It's most commonly used to draw a line at the top, bottom, and between the rows of the table.

2.5.18.3 \multicolumn

```
\multicolumn{cols}{pos}{text}
```

The `\multicolumn` is used to make an entry that spans several columns. The first mandatory argument, `cols`, specifies the number of columns to span. The second mandatory argument, `pos`, specifies the formatting of the entry; `c` for centred, `l` for flushleft, `r` for flushright. The third mandatory argument, `text`, specifies what text is to make up the entry.

2.5.18.4 \vline

The `\vline` command will draw a vertical line extending the full height and depth of its row. An `\hfill` command can be used to move the line to the edge of the column. It can also be used in an `@`-expression.

2.5.19 thebibliography

```
\begin{thebibliography}{widest-label}
\bibitem[label]{cite_key}
.
.
.
\end{thebibliography}
```

The `thebibliography` environment produces a bibliography or reference list. In the `article` style, this reference list is labelled "References"; in the `report` style, it is labelled "Bibliography".

- `widest-label`: Text that, when printed, is approximately as wide as the widest item label produces by the `\bibitem` commands.

2.5.19.1 \bibitem

```
\bibitem[label]{cite_key}
```

The `\bibitem` command generates an entry labelled by `label`. If the `label` argument is missing, a number is generated as the `label`, using the `enumi` counter. The `cite_key` is any sequence of letters, numbers, and punctuation symbols not containing a comma. This command writes an entry on the '`.aux`' file containing `cite_key` and the item's `label`. When this '`.aux`' file is read by the `\begin{document}` command, the item's `label` is associated with `cite_key`, causing the reference to `cite_key` by a `\cite` command to produce the associated `label`.

2.5.19.2 \cite

```
\cite[text]{key_list}
```

The `key_list` argument is a list of citation keys. This command generates an in-text citation to the references associated with the keys in `key_list` by entries on the '`.aux`' file read by the `\begin{document}` command.

2.5.19.3 \nocite

```
\nocite{key_list}
```

The `\nocite` command produces no text, but writes `key_list`, which is a list of one or more citation keys, on the '`.aux`' file.

2.5.20 theorem

```
\begin{theorem}
 theorem text
\end{theorem}
```

The `theorem` environment produces "Theorem x" in boldface followed by your theorem text.

2.5.21 titlepage

```
\begin{titlepage}
 text
\end{titlepage}
```

The `titlepage` environment creates a title page, i.e. a page with no printed page number or heading. It also causes the following page to be numbered page one. Formatting the title page is left to you. The `\today` command comes in handy for title pages.

2.5.22 verbatim

```
\begin{verbatim}
 text
\end{verbatim}
```

The `verbatim` environment is a paragraph-making environment that gets LaTeX to print exactly what you type in. It turns LaTeX into a typewriter with carriage returns and blanks having the same effect that they would on a typewriter.

2.5.22.1 \verb

`\verb char literal_text char \verb*char literal_text char`

Typesets `literal_text` exactly as typed, including special characters and spaces, using a typewriter (`\tt`) type style. There may be no space between `\verb` or `\verb*` and `char` (space is shown here only for clarity). The `*-form` differs only in that spaces are printed.

2.5.23 verse

```
\begin{verse}
 text
\end{verse}
```

The `verse` environment is designed for poetry, though you may find other uses for it.

2.6 Footnotes

`Footnotes` can be produced in one of two ways. They can be produced with one command, the `\footnote` command. They can also be produced with two commands, the `\footnotemark` and the `\footnotetext` commands. See the specific command for information on why you would use one over the other.

2.6.1 \footnote

```
\footnote[number]{text}
```

The `\footnote` command places the numbered footnote `text` at the bottom of the current page. The optional argument, `number`, is used to change the default footnote number. This command can only be used in outer paragraph mode.

2.6.2 \footnotemark

The `\footnotemark` command puts the footnote `number` in the text. This command can be used in inner paragraph mode. The text of the footnote is supplied by the `\footnotetext` command.

2.6.3 \footnotetext

```
\footnotetext[number]{text}
```

The `\footnotetext` command produces the `text` to be placed at the bottom of the page. This command can come anywhere after the `\footnotemark` command. The `\footnotetext` command must appear in outer paragraph mode.

The optional argument, `number`, is used to change the default footnote number.

2.7 Lengths

A `length` is a measure of distance. Many LaTeX commands take a length as an argument.

2.7.1 \newlength

```
\newlength{\gnat}
```

The `\newlength` command defines the mandatory argument, `\gnat`, as a `length` command with a value of `0in`. An error occurs if a `\gnat` command already exists.

2.7.2 \setlength

```
\setlength{\gnat}{length}
```

The `\setlength` command is used to set the value of a `length` command. The `length` argument can be expressed in any terms of length LaTeX understands, i.e., inches (`in`), millimeters (`mm`), points (`pt`), etc.

2.7.3 \addtolength

```
\addtolength{\gnat}{length}
```

The `\addtolength` command increments a "length command" by the amount specified in the `length` argument. It can be a negative amount.

2.7.4 \settowidth

```
\settowidth{\gnat}{text}
```

The `\settowidth` command sets the value of a `length` command equal to the width of the `text` argument.

2.8 Letters

You can use LaTeX to typeset letters, both personal and business. The `letter` document style is designed to make a number of letters at once, although you can make just one if you so desire.

Your '`.tex`' source file has the same minimum commands as the other document styles, i.e., you must have the following commands as a minimum:

```
\documentstyle{letter}
\begin{document}
 ... letters ...
\end{document}
```

Each letter is a `letter` environment, whose argument is the name and address of the recipient. For example, you might have:

```
\begin{letter}{Mr. Joe Smith\\ 2345 Princess St.
     \\ Edinburgh, EH1 1AA}
  ...
\end{letter}
```

The letter itself begins with the `\opening` command. The text of the letter follows. It is typed as ordinary LaTeX input. Commands that make no sense in a letter, like `\chapter`, don't work. The letter closes with a `\closing` command.

After the `closing`, you can have additional material. The `\cc` command produces the usual "cc: ...". There's also a similar `\encl` command for a list of enclosures.

2.8.1 \opening

`\opening{text}`

The letter begins with the `\opening` command. The mandatory argument, `text`, is whatever text you wish to start your letter, i.e.,

```
\opening{Dear Joe,}
```

2.8.2 \closing

```
\closing{text}
```

The letter closes with a `\closing` command, i.e.,

```
\closing{Best Regards,}
```

2.8.3 Declarations

The following commands are declarations which take a single argument.

2.8.4 \address

```
\address{Return address}
```

The return address, as it should appear on the letter and the envelope. Separate lines of the address should be separated by `\\` commands. If you do not make an `\address` declaration, then the letter will be formatted for copying onto your organisation's standard letterhead. (See ⟨undefined⟩ [Overview], page ⟨undefined⟩, for details on your local implementation). If you give an `\address` declaration, then the letter will be formatted as a personal letter.

2.8.5 \signature

```
\signature{Your name}
```

Your name, as it should appear at the end of the letter underneath the space for your signature. Items that should go on separate lines should be separated by `\\` commands.

2.8.6 \location

```
\location{address}
```

This modifies your organisation's standard address. This only appears if the `firstpage` pagestyle is selected.

2.8.7 \telephone

`\telephone{number}`

This is your telephone number. This only appears if the `firstpage` pagestyle is selected.

2.9 Line & Page Breaking

The first thing LaTeX does when processing ordinary text is to translate your input file into a string of glyphs and spaces. To produce a printed document, this string must be broken into lines, and these lines must be broken into pages. In some environments, you do the line breaking yourself with the `\\` command, but LaTeX usually does it for you.

2.9.1 \\

```
\\[*][extra-space]
```

The `\\` command tells LaTeX to start a new line. It has an optional argument, `extra-space`, that specifies how much extra vertical space is to be inserted before the next line. This can be a negative amount.

The `\\*` command is the same as the ordinary `\\` command except that it tells LaTeX not to start a new page after the line.

2.9.2 \-

The `\-` command tells LaTeX that it may hyphenate the word at that point. LaTeX is very good at hyphenating, and it will usually find all correct hyphenation points. The `\-` command is used for the exceptional cases.

2.9.3 \cleardoublepage

The `\cleardoublepage` command ends the current page and causes all figures and tables that have so far appeared in the input to be printed. In a two-sided printing style, it also makes the next page a right-hand (odd-numbered) page, producing a blank page if necessary.

2.9.4 \clearpage

The `\clearpage` command ends the current page and causes all figures and tables that have so far appeared in the input to be printed.

2.9.5 \hyphenation

```
\hyphenation{words}
```

The `\hyphenation` command declares allowed hyphenation points, where `words` is a list of words, separated by spaces, in which each hyphenation point is indicated by a `-` character.

2.9.6 \linebreak

```
\linebreak[number]
```

The `\linebreak` command tells LaTeX to break the current line at the point of the command. With the optional argument, `number`, you can convert the `\linebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

The `\linebreak` command causes LaTeX to stretch the line so it extends to the right margin.

2.9.7 \newline

The `\newline` command breaks the line right where it is. The `\newline` command can be used only in paragraph mode.

2.9.8 \newpage

The `\newpage` command ends the current page.

2.9.9 \nolinebreak

```
\nolinebreak[number]
```

The `\nolinebreak` command prevents LaTeX from breaking the current line at the point of the command. With the optional argument, `number`, you can convert the `\nolinebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

2.9.10 \nopagebreak

```
\nopagebreak[number]
```

The `\nopagebreak` command prevents LaTeX from breaking the current page at the point of the command. With the optional argument, `number`, you can convert the `\nopagebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

2.9.11 \pagebreak

```
\pagebreak[number]
```

The `\pagebreak` command tells LaTeX to break the current page at the point of the command. With the optional argument, `number`, you can convert the `\pagebreak` command from a demand to a request. The number must be a number from 0 to 4. The higher the number, the more insistent the request is.

2.10 Making Paragraphs

A paragraph is ended by one or more completely blank lines – lines not containing even an `%`. A blank line should not appear where a new paragraph cannot be started, such as in math mode or in the argument of a sectioning command.

2.10.1 \indent

```
\indent
```

This produces a horizontal space whose width equals the width of the paragraph indentation. It is used to add paragraph indentation where it would otherwise be suppressed.

2.10.2 \noindent

```
\noindent
```

When used at the beginning of the paragraph, it suppresses the paragraph indentation. It has no effect when used in the middle of a paragraph.

2.10.3 \par

Equivalent to a blank line; often used to make command or environment definitions easier to read.

2.11 Math Formulae

There are three environments that put LaTeX in math mode:

`math` For Formulae that appear right in the text.

`displaymath`
For Formulae that appear on their own line.

`equation` The same as the displaymath environment except that it adds an equation number in the right margin.

The `math` environment can be used in both paragraph and LR mode, but the `displaymath` and `equation` environments can be used only in paragraph mode. The `math` and `displaymath` environments are used so often that they have the following short forms:

```
\(...\)      instead of      \begin{math}...\end{math}
\[...\]      instead of      \begin{displaymath}...\end{displaymath}
```

In fact, the `math` environment is so common that it has an even shorter form:

```
$ ... $      instead of      \(...\)
```

2.11.1 Subscripts & Superscripts

To get an expression `exp` to appear as a subscript, you just type `_{exp}`. To get `exp` to appear as a superscript, you type `^{exp}`. LaTeX handles superscripted superscripts and all of that stuff in the natural way. It even does the right thing when something has both a subscript and a superscript.

2.11.2 Math Symbols

LaTeX provides almost any mathematical symbol you're likely to need. The commands for generating them can be used only in math mode. For example, if you include `$\pi$` in your source, you will get the symbol π in your output.

2.11.3 Spacing in Math Mode

In a `math` environment, LaTeX ignores the spaces you type and puts in the spacing that it thinks is best. LaTeX formats mathematics the way it's done in mathematics texts. If you want different spacing, LaTeX provides the following four commands for use in math mode:

1. `\;` - a thick space
2. `\:` - a medium space
3. `\,` - a thin space
4. `\!` - a negative thin space

2.11.4 Math Miscellany

`\cdots` The `\cdots` command produces a horizontal ellipsis where the dots are raised to the center of the line.

eg. $\cdots$

`\ddots` The `\ddots` command produces a diagonal ellipsis.

eg. $\ddots$

`\frac` `\frac{num}{den}` The `\frac` command produces the fraction `num` divided by `den`.

eg. $\frac{1}{4}$

`\ldots` The `\ldots` command produces an ellipsis. This command works in any mode, not just math mode.

eg. $\ldots$

`\overbrace`

`\overbrace{text}` The `\overbrace` command generates a brace over text.

eg. $\overbrace{x + \cdots + x}^{k \text{ times}}$

`\overline`

`\overline{text}` The `\overline` command causes the argument text to be overlined.

eg. $\overline{x}$

`\sqrt` `\sqrt[root]{arg}` The `\sqrt` command produces the square root of its argument. The optional argument, `root`, determines what root to produce, i.e., the cube root of `x+y` would be typed as `$\sqrt[3]{x+y}$`.

eg. $\sqrt{x-1}$

`\underbrace`

`\underbrace{text}` The `\underbrace` command generates text with a brace underneath.

eg. $\underbrace{x+y+z}_{>0}$

`\underline`

`\underline{text}` The `\underline` command causes the argument text to be underlined. This command can also be used in paragraph and LR modes.

eg. $\underline{z}$

`\vdots` The `\vdots` command produces a vertical ellipsis.

eg. $\vdots$

2.12 Modes

When LATEX is processing your input text, it is always in one of three modes:

- Paragraph mode
- Math mode
- Left-to-right mode, called LR mode for short

LATEX changes mode only when it goes up or down a staircase to a different level, though not all level changes produce mode changes. Mode changes occur only when entering or leaving an environment, or when LATEX is processing the argument of certain text-producing commands.

"Paragraph mode" is the most common; it's the one LATEX is in when processing ordinary text. In that mode, LATEX breaks your text into lines and breaks the lines into pages. LATEX is in "math mode" when it's generating a mathematical formula. In "LR mode", as in paragraph mode, LATEX considers the output that it produces to be a string of words with spaces between them. However, unlike paragraph mode, LATEX keeps going from left to right; it never starts a new line in LR mode. Even if you put a hundred words into an `\mbox`, LATEX would keep typesetting them from left to right inside a single box, and then complain because the resulting box was too wide to fit on the line.

LATEX is in LR mode when it starts making a box with an `\mbox` command. You can get it to enter a different mode inside the box - for example, you can make it enter math mode to put a formula in the box. There are also several text-producing commands and environments for making a box that put LATEX in paragraph mode. The box make by one of these commands or environments will be called a `parbox`. When LATEX is in paragraph mode while making a box, it is said to be in "inner paragraph mode". Its normal paragraph mode, which it starts out in, is called "outer paragraph mode".

2.13 Page Styles

The `\documentstyle` command determines the size and position of the page's head and foot. The page style determines what goes in them.

2.13.1 \maketitle

```
\maketitle
```

The `\maketitle` command generates a title on a separate title page - except in the `article` style, where the title normally goes at the top of the first page. Information used to produce the title is obtained from the following declarations:

See ⟨undefined⟩ [Page Styles], page ⟨undefined⟩ for the commands to give the information.

2.13.2 \author

```
\author{names}
```

The `\author` command declares the author(s), where `names` is a list of authors separated by `\and` commands. Use `\\` to separate lines within a single author's entry – for example, to give the author's institution or address.

NOTE: The `milstd` and `book-form` styles have re-defined the `\maketitle` command. The `\title` declaration is the only command of those shown below that has any meaning.

2.13.3 \date

```
\date{text}
```

The `\date` command declares text to be the document's date. With no `\date` command, the current date is used.

2.13.4 \thanks

```
\thanks{text}
```

The `\thanks` command produces a `footnote` to the title.

2.13.5 \title

```
\title{text}
```

The `\title` command declares `text` to be the title. Use `\\` to tell LATEX where to start a new line in a long title.

2.13.6 \pagenumbering

```
\pagenumbering{num_style}
```

Specifies the style of page numbers. Possible values of `num_style` are:

- `arabic` - Arabic numerals
- `roman` - Lowercase roman numerals
- `Roman` - Uppercase roman numerals
- `alph` - Lowercase letters
- `Alph` - Uppercase letters

2.13.7 \pagestyle

```
\pagestyle{option}
```

The `\pagestyle` command changes the style from the current page on throughout the remainder of your document.

The valid options are:

- `plain` - Just a plain page number.
- `empty` - Produces empty heads and feet - no page numbers.
- `headings` - Puts running headings on each page. The document style specifies what goes in the headings.
- `myheadings` - You specify what is to go in the heading with the `\markboth` or the `\markright` commands.

2.13.8 \mark

```
\markboth{left head}{right head}
\markright{right head}
```

The `\markboth` and `\markright` commands are used in conjunction with the page style `myheadings` for setting either both or just the right heading. In addition to their use with the `myheadings` page style, you can use them to override the normal headings in the `headings` style, since LaTeX uses these same commands to generate those heads. You should note that a "left-hand heading" is generated by the last `\markboth` command before the end of the page, while a "right-hand heading" is generated by the first `\markboth` or `\markright` that comes on the page if there is one, otherwise by the last one before the page.

2.13.9 \thispagestyle

```
\thispagestyle{option}
```

The `\thispagestyle` command works in the same manner as the `\pagestyle` command except that it changes the style for the current page only.

2.14 Sectioning

Sectioning commands provide the means to structure your text into units.

- `\part`
- `\chapter` (report style only)
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`
- `\subsubparagraph` (milstd and book-form styles only)
- `\subsubsubparagraph` (milstd and book-form styles only)

All sectioning commands take the same general form, i.e.,

```
\chapter[optional]{title}
```

In addition to providing the heading in the text, the mandatory argument of the sectioning command can appear in two other places:

1. The table of contents
2. The running head at the top of the page

You may not want the same thing to appear in these other two places as appears in the text heading. To handle this situation, the sectioning commands have an `optional` argument that provides the text for these other two purposes.

The "sectioning commands" have `*-forms` that print a `title`, but do not include a number and do not make an entry in the table of contents. For example, the `*-form` of the `\subsection` command could look like:

```
\subsection*{Example subsection}
```

2.14.1 \appendix

```
\appendix
```

The `\appendix` command changes the way sectional units are numbered. The `\appendix` command generates no text and does not affect the numbering or parts.

2.15 Spaces & Boxes

2.15.1 \addvspace

```
\addvspace{length}
```

The `\addvspace` command normally adds a vertical space of height length. However, if vertical space has already been added to the same point in the output by a previous `\addvspace` command, then this command will not add more space than needed to make the natural length of the total vertical space equal to `length`.

2.15.2 \bigskip

The `\bigskip` command is equivalent to `\vspace{bigskipamount}` where `bigskipamount` is determined by the document style.

2.15.3 \dotfill

The `\dotfill` command produces a "rubber length" that produces dots instead of just spaces.

2.15.4 \fbox

```
\fbox{text}
```

The `\fbox` command is exactly the same as the `\mbox` command, except that it puts a frame around the outside of the box that it creates.

2.15.5 \framebox

```
\framebox[width][position]{text}
```

The `\framebox` command is exactly the same as the `\makebox` command, except that it puts a frame around the outside of the box that it creates.

The framebox command produces a rule of thickness `\fboxrule`, and leaves a space `\fboxsep` between the rule and the contents of the box.

2.15.6 \hfill

The `\hfill` fill command produces a "rubber length" which can stretch or shrink horizontally. It will be filled with spaces.

2.15.7 \hrulespace

The `\hrulefill` fill command produces a "rubber length" which can stretch or shrink horizontally. It will be filled with a horizontal rule.

2.15.8 \hspace

```
\hspace[*]{length}
```

The `\hspace` command adds horizontal space. The length of the space can be expressed in any terms that LaTeX understands, i.e., points, inches, etc. You can add negative as well as positive space with an `\hspace` command. Adding negative space is like backspacing.

LaTeX removes horizontal space that comes at the end of a line. If you don't want LaTeX to remove this space, include the optional `*` argument. Then the space is never removed.

2.15.9 \makebox

```
\makebox[width][position]{text}
```

The `\makebox` command creates a box just wide enough to contain the `text` specified. The width of the box is specified by the optional `width` argument. The position of the text within the box is determined by the optional `position` argument.

- `c` - centered (default)
- `l` - flushleft

- `r` - flushright

See `\makebox (picture)-snt [\makebox (picture)], page \makebox (picture)-pg`.

2.15.10 \mbox

```
\mbox{text}
```

The `\mbox` command creates a box just wide enough to hold the text created by its argument.

2.15.11 \medskip

The `\medskip` command is equivalent to `\vspace{medskipamount}` where `medskipamount` is determined by the document style.

2.15.12 \newsavebox

```
\newsavebox{cmd}
```

Declares `cmd`, which must be a command name that is not already defined, to be a bin for saving boxes.

2.15.13 \parbox

```
\parbox[position]{width}{text}
```

A `parbox` is a box whose contents are created in `paragraph` mode. The `\parbox` has two mandatory arguments:

- `width` - specifies the width of the parbox, and
- `text` - the text that goes inside the parbox.

LaTeX will position a `parbox` so its center lines up with the center of the text line. An optional first argument, `position`, allows you to line up either the top or bottom line in the parbox.

A `\parbox` command is used for a parbox containing a small piece of text, with nothing fancy inside. In particular, you shouldn't use any of the paragraph-making environments inside a `\parbox` argument. For larger pieces of text, including ones containing a paragraph-making environment, you should use a `minipage` environment.

2.15.14 \raisebox

```
\raisebox{distance}[extend-above][extend-below]{text}
```

The `\raisebox` command is used to raise or lower text. The first mandatory argument specifies how high the text is to be raised (or lowered if it is a negative amount). The text itself is processed in `LR mode`.

Sometimes it's useful to make LaTeX think something has a different size than it really does - or a different size than LaTeX would normally think it has. The `\raisebox` command lets you tell LaTeX how tall it is.

The first optional argument, `extend-above`, makes LaTeX think that the text extends above the line by the amount specified. The second optional argument, `extend-below`, makes LaTeX think that the text extends below the line by the amount specified.

2.15.15 \rule

```
\rule[raise-height]{width}{thickness}
```

The `\rule` command is used to produce horizontal lines. The arguments are defined as follows:

- `raise-height` - specifies how high to raise the rule (optional)
- `width` - specifies the length of the rule (mandatory)
- `thickness` - specifies the thickness of the rule (mandatory)

2.15.16 \savebox

```
\sbox{cmd}[text]
\savebox{cmd}[width][pos]{text}
```

These commands typeset `text` in a box just as for `\mbox` or `\makebox`. However, instead of printing the resulting box, they save it in bin `cmd`, which must have been declared with `\newsavebox`.

2.15.17 \smallskip

```
\smallskip
```

The `\smallskip` command is equivalent to `\vspace{smallskipamount}` where `smallskipamount` is determined by the document style.

2.15.18 \usebox

```
\usebox{cmd}
```

Prints the box most recently saved in bin `cmd` by a `\savebox` command.

2.15.19 \vfill

The `\vfill` fill command produces a rubber length which can stretch or shrink vertically.

2.15.20 \vspace

```
\vspace[*]{length}
```

The `\vspace` command adds vertical space. The length of the space can be expressed in any terms that LaTeX understands, i.e., points, inches, etc. You can add negative as well as positive space with an `\vspace` command.

LaTeX removes vertical space that comes at the end of a page. If you don't want LaTeX to remove this space, include the optional `*` argument. Then the space is never removed.

2.16 Special Characters

The following characters play a special role in LaTeX and are called "special printing characters", or simply "special characters".

`# $ % & ~ _ ^ \ { }`

Whenever you put one of these special characters into your file, you are doing something special. If you simply want the character to be printed just as any other letter, include a `\` in front of the character. For example, `\$` will produce `$` in your output.

The exception to the rule is the `\` itself because `\\` has its own special meaning. A `\` is produced by typing `$\backslash$` in your file.

2.17 Splitting the Input

A large document requires a lot of input. Rather than putting the whole input in a single large file, it's more efficient to split it into several smaller ones. Regardless of how many separate files you use, there is one that is the root file; it is the one whose name you type when you run LaTeX.

2.17.1 \include

`\include{file}`

The `\include` command is used in conjunction with the `\includeonly` command for selective inclusion of files. The `file` argument is the first name of a file, denoting '`file.tex`'. If `file` is one the file names in the file list of the `\includeonly` command or if there is no `\includeonly` command, the `\include` command is equivalent to

```
\clearpage \input{file} \clearpage
```

except that if the file '`file.tex`' does not exist, then a warning message rather than an error is produced. If the file is not in the file list, the `\include` command is equivalent to `\clearpage`.

The `\include` command may not appear in the preamble or in a file read by another `\include` command.

2.17.2 \includeonly

`\includeonly{file_list}`

The `\includeonly` command controls which files will be read in by an `\include` command. It can only appear in the preamble.

2.17.3 \input

`\input{file}`

The `\input` command causes the indicated `file` to be read and processed, exactly as if its contents had been inserted in the current file at that point. The file name may be a complete file name with extension or just a first name, in which case the file '`file.tex`' is used.

2.18 Starting & Ending

Your input file must contain the following commands as a minimum...

```
\documentstyle{style}
\begin{document}
  ... your text goes here ...
\end{document}
```

...where the `style` selected is one the valid styles for LaTeX . See ⟨undefined⟩ [Document Styles], page ⟨undefined⟩, and also see ⟨undefined⟩ [Overview], page ⟨undefined⟩, for details of the various document styles available locally.

You may include other LaTeX commands between the `\documentstyle` and the `\begin{document}` commands.

2.19 Table of Contents

A table of contents is produced with the `\tableofcontents` command. You put the command right where you want the table of contents to go; LaTeX does the rest for you. It produces a heading, but it does not automatically start a new page. If you want a new page after the table of contents, include a `\newpage` command after the `\tableofcontents` command.

There are similar commands `\listoffigures` and `\listoftables` for producing a list of figures and a list of tables, respectively. Everything works exactly the same as for the table of contents.

NOTE: If you want any of these items to be generated, you cannot have the `\nofiles` command in your document.

2.19.1 \addcontentsline

```
\addcontentsline{file}{sec_unit}{entry}
```

The `\addcontentsline` command adds an entry to the specified list or table where...

- `file` is the extension of the file on which information is to be written: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).
- `sec_unit` controls the formatting of the entry. It should be one of the following, depending upon the value of the file argument:
  1. `toc` - the name of the sectional unit, such as part or subsection.
  2. `lof` - figure
  3. `lot` - table
- `entry` is the text of the entry.

2.19.2 \addtocontents

```
\addtocontents{file}{text}
```

The `\addtocontents` command adds text (or formatting commands) directly to the file that generates the table of contents or list of figures or tables.

- `file` is the extension of the file on which information is to be written: `toc` (table of contents), `lof` (list of figures), or `lot` (list of tables).
- `text` is the information to be written.

2.20 Terminal Input/Output

2.20.1 \typeout

```
\typeout{msg}
```

Prints `msg` on the terminal and in the `log` file. Commands in `msg` that are defined with `\newcommand` or `\renewcommand` are replaced by their definitions before being printed.

LaTeX's usual rules for treating multiple spaces as a single space and ignoring spaces after a command name apply to `msg`. A `\space` command in `msg` causes a single space to be printed.

2.20.2 \typein

```
\typein[cmd]{msg}
```

Prints `msg` on the terminal and causes LaTeX to stop and wait for you to type a line of input, ending with return. If the `cmd` argument is missing, the typed input is processed as if it had been included in the input file in place of the `\typein` command. If the `cmd` argument is present, it must be a command name. This command name is then defined or redefined to be the typed input.

2.21 Typefaces

The `typeface` is specified by giving the "size" and "style". A typeface is also called a "font".

2.21.1 \Styles

The following type style commands are supported by LaTeX .

`\rm`	Roman.
`\it`	Italics.
`\em`	Emphasis (toggles between \it and \rm).
`\bf`	Boldface.
`\sl`	Slanted.
`\sf`	Sans serif.
`\sc`	Small caps.
`\tt`	Typewriter.

2.21.2 Sizes

The following type size commands are supported by LaTeX .

`\tiny`	
`\scriptsize`	
`\footnotesize`	
`\small`	
`\normalsize`	(default)
`\large`	
`\Large`	(capital "l")
`\LARGE`	(all caps)
`\huge`	
`\Huge`	(capital "h")

3 Parameters

The input file specification indicates the file to be formatted; TeX uses '`.tex`' as a default file extension. If you omit the input file entirely, TeX accepts input from the terminal. You specify command options by supplying a string as a parameter to the command. eg.

```
latex "\scrollmode\input foo.tex"
```

...will process '`foo.tex`' without pausing after every error.

Output files are always created in the current directory. When you fail to specify an input file name, TeX bases the output names on the file specification associated with the logical name TEX_OUTPUT, typically `texput.log`.

3.1 Alphabetical List of Commands

3.1.1 _{exp} (subscript)

To get an expression `exp` to appear as a subscript, you just type `_{exp}`. Use in math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Subscripts & Superscripts], page ⟨undefined⟩.

3.1.2 ^{exp} (superscript)

To get an expression `exp` to appear as a superscript, you just type `^{exp}`. Use in math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Subscripts & Superscripts], page ⟨undefined⟩.

3.1.3 \;

Include a thick space in math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Spacing in Math Mode], page ⟨undefined⟩.

3.1.4 \:

Include a medium space in math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Spacing in Math Mode], page ⟨undefined⟩.

3.1.5 \,

Include a thin space in math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Spacing in Math Mode], page ⟨undefined⟩.

3.1.6 \!

Include a negative thin space in math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Spacing in Math Mode], page ⟨undefined⟩.

3.1.7 \bf

Boldface typeface.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.8 \cdots

The `\cdots` command produces a horizontal ellipsis where the dots are raised to the center of the line.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.9 \ddots

The `\ddots` command produces a diagonal ellipsis.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.10 \em

Emphasis (toggles between `\it` and `\rm`).

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.11 \footnotesize

Third smallest of 10 typefaces available. This is the default size for footnotes.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.12 \frac

```
\frac{num}{den}
```

The `\frac` command produces the fraction `num` divided by `den`.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.13 \huge

Second largest of 10 typefaces available.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.14 \Huge

Largest of 10 typefaces available. All fonts may not be available in this size.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.15 \it

Italics typeface.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.16 \large

Slightly larger than default typeface size.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.17 \Large

Fourth largest of typefaces available. Is generally the default for titles.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.18 \LARGE

Third largest of typefaces available.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.19 \ldots

The `\ldots` command produces an ellipsis. This command works in any mode, not just math mode.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.20 \normalsize

The size of `\normalsize` is defined by as 10pt unless the 11pt or 12pt document style option is used.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.21 \overbrace

```
\overbrace{text}
```

The `\overbrace` command generates a brace over text.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.22 \overline

```
\overline{text}
```

The `\overline` command causes the argument text to be overlined.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.23 \rm

Roman typeface (default).

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.24 \sc

Small caps typeface.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.25 \scriptsize

Second smallest of 10 typefaces available.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.26 \sf

Sans serif typeface.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.27 \sl

Slanted typeface.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.28 \small

Slightly smaller than default typeface size.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.29 \sqrt

```
\sqrt[root]{arg}
```

The `\sqrt` command produces the square root of its argument. The optional argument, `root`, determines what root to produce, i.e. the cube root of `x+y` would be typed as `$\sqrt[3]{x+y}$`.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.30 \tiny

Smallest of 10 typefaces available. All fonts may not be available in this size.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Sizes], page ⟨undefined⟩.

3.1.31 \tt

Typewriter typeface.

See ⟨undefined⟩ [Typefaces], page ⟨undefined⟩.

See ⟨undefined⟩ [Styles], page ⟨undefined⟩.

3.1.32 \underbrace

```
\underbrace{text}
```

The `\underbrace` command generates text with a brace underneath.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.33 \underline

```
\underline{text}
```

The `\underline` command causes the argument text to be underlined. This command can also be used in paragraph and LR modes.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

3.1.34 \vdots

The `\vdots` command produces a vertical ellipsis.

See ⟨undefined⟩ [Math Formulae], page ⟨undefined⟩.

See ⟨undefined⟩ [Math Miscellany], page ⟨undefined⟩.

Index

(Index is nonexistent)

Table of Contents