$\LaTeX\,2_\varepsilon$ Commands

July 14, 2003

Contents

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 `dvips`, 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.

A LaTeX command begins with the command name, which consists of a \ followed by either (1) a string of letters or (2) 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 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 classes to control numbering.

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

2.1 `\addtocounter`

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

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

2.2 `\alph, \Alpha`

```
\alph{counter}    \Alph{counter}
```

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

2.3 `\arabic`

```
\arabic{counter}
```

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

2.4 `\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.

2.5 `\newcounter`

```
\newcounter{foo}[counter]
```

The `\newcounter` command defines a new counter named `foo`. The counter is initialized to zero.

The optional argument `[counter]` causes the counter `foo` to be reset whenever the counter named in the optional argument is incremented.

2.6 `\refstepcounter`

```
\refstepcounter{counter}
```

The `\refstepcounter` command works like `\stepcounter`, except it also defines the current `\ref` value to be the result of `\thecounter`.

2.7 `\roman \Roman`

```
\roman{counter}    \Roman{counter}
```

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

2.8 `\stepcounter`

```
\stepcounter{counter}
```

The `\stepcounter` command adds one to the `counter` and resets all subsidiary counters.

2.9 `\setcounter`

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

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

2.10 `\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.

2.11 `\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.

3 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." This is called *cross reference*, is done by the following commands:

- `\label`: Assign a symbolic name, or *label*, to a piece of text.
- `\pageref`: Refer to a page number.
- `\ref`: Refer to a section, figure or similar.

To avoid accidentally creating two labels with the same name, it is common to use labels consisting of a prefix and a suffix separated by a colon. The prefixes conventionally used are

- `cha` for chapters
- `sec` for lower-level sectioning commands
- `fig` for figures
- `tab` for tables
- `eq` for equations

Suppose the first figure in Section 1, Chapter2, is labelled as `\label{fig:2.1.1}`. Then we can refer this figure in any place through the input `Fig. \ref{fig:2.1.1} in page \pageref{fig:2.1.1}`.

3.1 \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.

3.2 \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.

3.3 \ref

```
\ref{key}
```

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

4 Definitions

The following commands define some new commands:

- `\newcommand`: Define a new command.
- `\newenvironment`: Define a new environment.
- `\newtheorem`: Define a new theorem-like environment.
- `\newfont`: Define a new font name.

Note that LaTeX will automatically check whether the name of the newly-defined command is the same as some known command; if so, the system will emit an error message. So the definition here is safer than using `\def`.

4.1 `\newcommand`

```
\newcommand{cmd}[args]{definition}
\newcommand{cmd}[args][default]{definition}
\renewcommand{cmd}[args]{definition}
\renewcommand{cmd}[args][default]{definition}
```

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`: if this optional parameter is present, it means that the command's first argument is optional. The default value of the optional argument is `def`.
- `definition`: 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 n-th argument when this substitution takes place.

4.2 `\newenvironment`

```
\newenvironment{nam}[args]{begdef}{enddef}
\newenvironment{nam}[args][default]{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.

- `default`: If this is specified, the first argument is optional, and `default` gives the default value for that argument.
- `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 n-th argument when this substitution takes place.
- `enddef`: The text substituted for every occurrence of `\end{nam}`. It may not contain any argument parameters.

4.3 `\newtheorem`

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

This command defines a theorem-like environment. The `\newtheorem` command may have at most one optional argument.

- `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.

4.4 `\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.

5 Document Classes:`\documentclass,\usepackage`

LaTeX has the following standard document classes: `article, report, letter, book, slides`. You can find other document classes in your local TeX directories. Document classes do many things for typesetting a particular document, including controlling its format. A document class is selected with the following command:

```
\documentclass[options] {class}
```

`class` is in fact a file name with extension `.cls`.

All the standard classes (except `slides`) accept the following options for selecting the typeface size (10 pt is default) for main text: `10pt, 11pt, 12pt`.

All classes accept the following options for selecting the paper size (default is letter): `a4paper, a5paper, b5paper, letterpaper, legalpaper, executivepaper`.

Miscellaneous options:

- landscape: selects landscape format. Default is portrait.
- titlepage, notitlepage: selects if there should be a separate title page.
- leqno: equation number on left side of equations. Default is right side.
- fleqn: displayed formulas flush left. Default is centred.
- openbib: use “open” bibliography format.
- draft, final: mark/do not mark overfull boxes with a rule. Default is final.

The following options are not available with the slides class:

- oneside, twoside: selects one- or twosided layout. Default is oneside, except for the book class.
- openright, openany: determines if a chapter should start on a right-hand page. Default is openright for book.
- onecolumn, twocolumn: one or two columns. Defaults to one column.

The slides class offers the option `clock` for printing the time at the bottom of each note.

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

Additional packages are loaded by a

```
\usepackage[options]{packages}
```

command. A package is a file with extension `.sty`. Such files add some useful tool for your typesetting task. If you specify more than one package, they must be separated by a comma.

Any options given in the `\documentclass` command that are unknown by the selected document class are passed on to the packages loaded with `\usepackage`.

Miscellaneous commands control the general layout of a page:

- `\flushbottom`: Make all text pages the same height.
- `\onecolumn`: Use one-column layout.
- `\raggedbottom`: Allow text pages of differing height.
- `\twocolumn`: Use two-column layout.

5.1 `\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 if twocolumn mode is selected.

5.2 `\onecolumn`

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

5.3 \raggedbottom

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

5.4 \twocolumn

The declaration

```
\twocolumn[text]
```

starts a new page and produces two-column output. If the optional `text` argument is present, it is typeset in one-column mode.

6 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}
```

6.1 array

Math arrays are produced with the `array` environment:

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

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 flush left
- `r` – for flush right

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 `\\`, or `cr`, `crcr`.

Note that the `array` environment can only be used in math mode, so normally it is used inside an `equation` environment.

6.2 center

The `center` environment

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

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 `\\`.

6.2.1 `\centering`

This declaration corresponds to the `center` environment. It can be used inside an environment such as `quote` or in a `parbox`. The text of a figure or table can be centred 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.

6.3 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 boldface and flushed right.

6.4 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. The `enumerate` environment uses the `enumi` through `enumiv` counters. The type of numbering can be changed by redefining `\theenumi` etc.

6.5 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. The `itemize` environment uses the `itemi` through `itemiv` counters. The type of numbering can be changed by redefining `\theitemi` etc. The `list` environment is a generic environment which is used for defining many of the more specific environments. It is seldom used in documents, but often in macros.

6.6 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.

6.7 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.

The command `\lefteqn` is used for splitting long formulas across lines. It typesets its argument in display style flush left in a box of zero width.

`eqnarray*` is equivalent to `eqnarray` with each line ended by `\nonumber\\`.

6.8 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.

6.9 figure

```
\begin{figure}[placement]
        body of the figure
\caption{figure title}
\end{equation}
```

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:

- `h` (Here) - at the position in the text where the figure 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 classes 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.

6.10 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 \\.

6.10.1 `\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.

6.11 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 `\\`.

6.11.1 `\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.

6.12 minipage

```
\begin{minipage}
   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.

6.13 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, e.g., `5, 2.3, -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 2.54cm. 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 100mm and height 200mm, 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 LaTeX 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 (*hbox*). 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.

6.13.1 `\circle`

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

The `\circle` command produces a circle with a diameter as close to the specified one as possible. If the `*`-form of the command is used, LaTeX draws a solid circle. Note that only circles up to 40 pt can be drawn.

6.13.2 `\dashbox`

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

This command draws a box with a dashed line. It 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`.

6.13.3 `\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.

6.13.4 `\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.

6.13.5 `\line`

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

The `\line` command draws a line of the specified `length` and `slope`. Note that LaTeX can only draw lines with slope = x/y, where x and y have integer values from -6 through 6.

6.13.6 `\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.

6.13.7 `\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

6.13.8 `\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.

6.13.9 `\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

Note that the “corners” of the oval are made with quarter circles with a maximum radius of 20 pt, so large ”ovals” will look more like boxes with rounded corners.

6.13.10 `\put`

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

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

6.13.11 `\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 centre of the stack (default)

6.13.12 `\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.

6.14 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.

6.15 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.

6.16 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. It is best suited for cases where the width of each column is constant and known in advance. This environment can be broken across pages, unlike the `tabular` environment.

The following commands can be used inside a `tabbing` enviroment:

- `\=`: Sets a tab stop at the current position.
- `\>`: Advances to the next tab stop.
- `\<`: This command allows you to put something to the left of the local margin without changing the margin. Can only be used at the start of the line.
- `\+`: Moves the left margin of the next and all the following commands one tab stop to the right.
- `\-`: Moves the left margin of the next and all the following commands one tab stop to the left.
- `\'`: Moves everything that you have typed so far in the current column, i.e. everything from the most recent `\>`, `\<`, `\'`, `\\`, or `\kill` command, to the right of the previous column, flush against the current column's tab stop.

- `\``: Allows you to put text flush 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.
- `\kill`: Sets tab stops without producing text. Works just like `\\` 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.
- `\pushtabs`: Saves all current tab stop positions. Useful for temporarily changing tab stop positions in the middle of a `tabbing` environment.
- `\pushtabs`: Restores the tab stop positions saved by the last `\pushtabs`.
- `\a`: In a `tabbing` environment, the commands `\=`, `\'` and `\`` do not produce accents as normal. Instead, the commands `\a=`, `\a'` and `\a`` are used.

The following example typesets a Pascal function in a traditional format:

```
\begin{tabbing}
function \= fact(n : integer) : integer;\\
         \&#62; begin \= \+ \\
               \&#62; if \= n $&#62;$ 1 then \+ \\
                        fact := n * fact(n-1) \- \\
                  else \+ \\
                        fact := 1; \-\- \\
            end;\\
\end{tabbing}
```

6.17 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` classes 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.

6.18 tabular

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

or

```
\begin{tabular*}[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:

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

2. `pos`: Specifies the vertical position; default is alignment on the centre of the environment.

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

3. `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:

     (a) inside an environment like `minipage`, `array`, or `tabular`.

(b) inside an explicit `\parbox`.

(c) 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`.

The following commands can be used inside a `tabular` environment: `\cline`, `\hline`, `\multicolumn`, `\vline`.

6.18.1 `\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.

6.18.2 `\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.

6.18.3 `\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.

6.18.4 `\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.

6.19 `\thebibliography`

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

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

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

6.19.1 `\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`.

6.19.2 `\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. The optional `text` argument will appear after the citation, i.e. `\cite[p. 2]{knuth}` might produce '`[Knuth, p.2]`'.

6.19.3 `\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.

6.19.4 Using BibTeX

If you use the BibTeX program by Oren Patashnik (highly recommended if you need a bibliography of more than a couple of titles) to maintain your bibliography, you don't use the `thebibliography` environment. Instead, you include the lines

```
\bibliographystyle{style}
\bibliography{bibfile}
```

where `style` refers to a file `style.bst`, which defines how your citations will look. The standard styles distributed with BibTeX are:

- `alpha`: Sorted alphabetically. Labels are formed from name of author and year of publication.
- `plain` ¡DD¿ Sorted alphabetically. Labels are numeric.
- `unsrt`: Like `plain`, but entries are in order of citation.
- `abbrv`: Like `plain`, but more compact labels.

In addition, numerous other BibTeX style files exist tailored to the demands of various publications.

The argument to `\bibliography` refers to the file `bibfile.bib`, which should contain your database in BibTeX format. Only the entries referred to via `\cite` and `\nocite` will be listed in the bibliography.

6.20 theorem

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

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

6.21 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. Note that you can use the `\maketitle` command to produce a standard title page.

6.22 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.

6.22.1 \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 as ‘␣’.

6.23 verse

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

The `verse` environment is designed for poetry, though you may find other uses for it. The margins are indented on the left and the right. Separate the lines of each stanza with `\\`, and use one or more blank lines to separate the stanzas.

7 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.

7.1 `\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; i.e., you cannot use it in sectioning commands like `\chapter`, in figures, tables or in a `tabular` environment.

7.2 `\footnotemark`

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

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.

This command can be used to produce several consecutive footnote markers referring to the same footnote by using

```
\footnotemark[\value{footnote}]
```

7.3 `\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.

8 Lengths

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

8.1 `\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.

8.2 `\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`), millimetres (`mm`), points (`pt`), etc.

8.3 `\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.

8.4 `\settodepth`

```
\settodepth{\gnat}{text}
```

The `\settodepth` command sets the value of a `length` command equal to the depth of the `text` argument.

8.5 `\settoheight`

```
\settoheight{\gnat}{text}
```

The `\settoheight` command sets the value of a `length` command equal to the height of the `text` argument.

8.6 `\settowidth`

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

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

8.7 Predefined lengths

```
\width    \height    \depth    \totalheight
```

These length parameters can be used in the arguments of the box-making commands. They specify the natural width etc. of the text in the box. `\totalheight` equals `\height` + `\depth`. To make a box with the text stretched to double the natural size, e.g., say

```
\makebox[2\width]{Get a stretcher}
```

9 Letters

You can use LaTeX to typeset letters, both personal and business. The `letter` document class 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 classes, i.e., you must have the following commands as a minimum:

```
    \documentclass{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`, do not 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. With both these commands, use \\ to separate the items.

9.1 \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 organization's standard letterhead. If you give an `\address` declaration, then the letter will be formatted as a personal letter.

9.2 \cc

```
\cc{Kate Schechter\\Rob McKenna}
```

Generate a list of other persons the letter was sent to. Each name is printed on a separate line.

9.3 \closing

```
\closing{text}
```

The letter closes with a `\closing` command, i.e., `\closing{Best Regards,}`.

9.4 `\encl`

```
\encl{CV\\Certificates}
```

Generate a list of enclosed material.

9.5 `\location`

```
\location{address}
```

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

9.6 `\makelabels`

```
\makelabels{number}
```

If you issue this command in the preamble, LaTeX will create a sheet of address labels. This sheet will be output before the letters.

9.7 `\name`

```
\name{June Davenport}
```

Your name, used for printing on the envelope together with the return address.

9.8 `\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,}.

9.9 `\ps`

Use `\ps` before a postscript.

9.10 `\signature`

```
\signature{Harvey Swick}
```

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.

9.11 `\startbreaks`

`\startbreaks` is used after a `\stopbreaks` command to allow page breaks again.

9.12 `\stopbreaks`

`\stopbreaks` inhibits page breaks until a `\startbreaks` command occurs.

9.13 `\telephone`

```
\telephone{number}
```

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

10 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.

10.1 `\\`

```
\\[*][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.

10.2 `\-`

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.

Note that when you insert `\-` commands in a word, the word will only be hyphenated at those points and not at any of the hyphenation points that LaTeX might otherwise have chosen.

10.3 `\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.

10.4 `\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.

10.5 `\enlargethispage`

```
\enlargethispage{size}    \enlargethispage*{size}
```

Enlarge the `\textheight` for the current page by the specified amount; e.g. `\enlargethispage{\baselineskip}` will allow one additional line.

The starred form tries to squeeze the material together on the page as much as possible. This is normally used together with an explicit `\pagebreak`.

10.6 `\fussy`

This declaration (which is the default) makes TEX more fussy about line breaking. This can avoids too much space between words, but may produce overfull boxes. This command cancels the effect of a previous `\sloppy` command.

10.7 `\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.

10.8 `\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.

10.9 `\newline`

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

10.10 `\newpage`

```
\linebreak[number]
```

The `\newpage` command ends the current page.

10.11 `\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.

10.12 `\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.

10.13 `\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.

10.14 `\sloppy`

This declaration makes TeX less fussy about line breaking. This can prevent overfull boxes, but may leave too much space between words.

Lasts until a `\fussy` command is issued.

11 Making Paragraphs

A paragraph is ended by one or more completely blank lines – lines not containing even a `%`. 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.

11.1 `\indent`

This command 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.

11.2 `\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.

11.3 `\par`

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

12 Margin Notes

The command `\marginpar[left]{right}` creates a note in the margin. The first line will be at the same height as the line in the text where the `\marginpar` occurs. When you only specify the mandatory argument `right`, the text will be placed

- in the right margin for one-sided layout
- in the outside margin for two-sided layout
- in the nearest margin for two-column layout.

By issuing the command `\reversemarginpar`, you can force the marginal notes to go into the opposite (inside) margin.

When you specify both arguments, `left` is used for the left margin, and `right` is used for the right margin.

The first word will normally not be hyphenated; you can enable hyphenation by prefixing the first word with a `\hspace{0pt}` command.

13 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     \(...\)
```

13.1 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.

13.2 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 in your output.

13.3 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:

- `\;` - a thick space
- `\:` - a medium space
- `\,` - a thin space
- `\!` - a negative thin space

13.4 Math Miscellany

- `\cdots`: Produces a horizontal ellipsis where the dots are raised to the centre of the line, e.g., $\cdots$.
- `\ddots`: Produces a diagonal ellipsis, e.g., $\ddots$.
- `\frac{num}{den}`: Produces the fraction `num` divided by `den`, e.g., $\frac{1}{4}$.
- `\ldots`: Produces an ellipsis. This command works in any mode, not just math mode, e.g., $\ldots$.
- `\overbrace{text}`: Generates a brace over text.
- `\overline{text}`: Causes the argument text to be overlined.
- `\sqrt[root]{arg}`: 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 `$\sqrt[3]{x+y}$`.
- `\underbrace{text}`: Generates text with a brace underneath.
- `\underline{text}`: Causes the argument text to be underlined. This command can also be used in paragraph and LR modes.
- `\vdots`: Produces a vertical ellipsis.

14 Modes

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

- Paragraph mode (vertical mode)
- Math mode
- Left-to-right mode, called LR mode for short (horizontal mode)

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".

15 Page Styles

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

15.1 `\maketitle`

The `\maketitle` command generates a title on a separate title page - except in the `article` class, where the title normally goes at the top of the first page.

15.1.1 `\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.

15.1.2 `\date`

```
\date{text}
```

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

15.1.3 `\thanks`

```
\thanks{text}
```

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

15.1.4 `\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.

15.1.5 `\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

15.1.6 `\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.

15.1.7 `\markboth`

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

The `\markboth` command is used in conjunction with the page style `myheadings` for setting both the left and the right heading. 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.

15.1.8 `\markright`

```
\markright{right head}
```

The `\markright` command is used in conjunction with the page style `myheadings` for setting the right heading, leaving the left heading unchanged. 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.

15.1.9 `\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.

16 Sectioning

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

- `\part`
- `\chapter` (report and book class only)
- `\section`
- `\subsection`
- `\subsubsection`
- `\paragraph`
- `\subparagraph`

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:

- The table of contents
- 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.

All 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.

16.1 `\appendix`

The `\appendix` command changes the way sectional units are numbered. The `\appendix` command generates no text and does not affect the numbering of parts. The normal use of this command is something like

```
\chapter{The First Chapter}
  ...
\appendix
\chapter{The First Appendix}
```

17 Spaces & Boxes

All the predefined length parameters can be used in the arguments of the box-making commands.

17.1 `\dotfill`

```
\thispagestyle{option}
```

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

17.2 `\hfill`

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

17.3 `\hrulefill`

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

17.4 \hspace

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

`\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.

17.5 \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`.

17.6 \bigskip

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

17.7 \medskip

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

17.8 \smallskip

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

17.9 \vfill

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

17.10 \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.

17.11 `\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.

17.12 `\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.

17.13 `\lrbox`

```
\begin{lrbox}{cmd} text \end{lrbox}
```

This is the environment form of `\sbox`. The text inside the environment is saved in the box `cmd`, which must have been declared with `\newsavebox`.

17.14 `\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` – centred (default)
- `l` – flushleft
- `r` – flushright
- `s` – stretch from left to right margin. The text must contain stretchable space for this to work.

17.15 `\mbox`

The `\mbox` command creates a box just wide enough to hold the text created by its argument. Use this command to prevent text from being split across lines.

17.16 `\newsavebox`

```
\newsavebox{cmd}
```

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

17.17 `\parbox`

```
\parbox[position][height][inner-pos]{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 centre lines up with the centre of the text line. The optional *position* argument allows you to line up either the top or bottom line in the parbox (default is top).

If the *height* argument is not given, the box will have the natural height of the text.

The *inner-pos* argument controls the placement of the text inside the box. If it is not specified, *position* is used.

- `t` – text is placed at the top of the box.
- `c` – text is centred in the box.
- `b` – text is placed at the bottom of the box.
- `s` – stretch vertically. The text must contain vertically stretchable space for this to work.

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.

17.18 `\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.

17.19 `\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)

17.20 `\savebox`

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

This command typeset `text` in a box just as for `\makebox`. However, instead of printing the resulting box, it saves it in bin `cmd`, which must have been declared with `\newsavebox`.

17.21 `\sbox`

```
\sbox{text}
```

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

17.22 `\usebox`

```
\usebox{cmd}
```

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

18 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.

One exception to this rule is the \ itself because \\ has its own special meaning. A \ is produced by typing `\char‘\\` in your file.

Also, `\~` means ‘place a tilde accent over the following letter’, so you will probably want to use `\verb` instead.

In addition, you can access any character of a font once you know its number by using the `\symbol` command. For example, the character used for displaying spaces in the `\verb*` command has the code decimal 32, so it can be typed as `\symbol{32}`.

You can also specify octal numbers with `'` or hexadecimal numbers with `"`, so the previous example could also be written as `\symbol{'40}` or `\symbol{20}`.

19 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.

19.1 `\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.

19.2 `\includeonly`

```
\includeonly{file_list}
```

The `\includeonly` command controls which files will be read in by an `\include` command. `file_list` should be a comma-separated list of filenames. Each filename must match exactly a filename specified in a `\include` command. This command can only appear in the preamble.

19.3 `\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.

20 Starting & Ending

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

```
\documentclass{class}
  \begin{document}
     ... your text goes here ...
\end{document}
```

where the `class` selected is one of the valid classes for LaTeX. You may include other LaTeX commands between the `\documentclass` and the `\begin{document}` commands (i.e., in the 'preamble').

21 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` 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.

21.1 `\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:
  - `toc` – the name of the sectional unit, such as part or subsection.
  - `lof` – figure
  - `lot` – table
- `entry` is the text of the entry.

21.2 `\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.

22 Terminal Input/Output

22.1 `\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.

22.2 `\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. A `^^J` in `msg` prints a newline.

23 Typefaces

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

23.1 `\Styles`

The following type style commands are supported by LaTeX. These commands are used like `\textit{italics text}`. The corresponding command in parenthesis is the "declaration form", which takes no arguments. The scope of the declaration form lasts until the next type style command or the end of the current group. The declaration forms are cumulative; i.e., you can say `\sffamily\bfseries` to get sans serif boldface. You can also use the environment form of the declaration forms; e.g. `\begin{ttfamily}...\end{ttfamily}`.

- `\textrm` (`\rmfamily`): Roman.
- `\textit` (`\itshape`):
- `\emph`: Emphasis (toggles between `\textit` and `\textrm`).
- `\textmd` (`\mdseries`): Medium weight (default). The opposite of boldface.
- `\textbf` (`\bfseries`): Boldface.
- `\textup` (`\upshape`): Upright (default). The opposite of slanted.
- `\textsl` (`\slshape`): Slanted.

- `\textsf` (`\sffamily`):Sans serif.
- `\textsc` (`\scshape`): Small caps.
- `\texttt` (`\ttfamily`): Typewriter.
- `\textnormal` (`\normalfont`): Main document font.
- `\mathrm`: Roman, for use in math mode.
- `\mathbf`: Boldface, for use in math mode.
- `\mathsf`: Sans serif, for use in math mode.
- `\mathtt`: Typewriter, for use in math mode.
- `\mathit`: Italics, for use in math mode, e.g. variable names with several letters.
- `\mathnormal`: For use in math mode, e.g. inside another type style declaration.
- `\mathcal`: 'Calligraphic' letters, for use in math mode.

In addition, the command `\mathversion{bold}` can be used for switching to bold letters and symbols in formulas. `\mathversion{normal}` restores the default.

23.2 Sizes

The following standard type size commands are supported by LaTeX. The commands as listed here are "declaration forms". The scope of the declaration form lasts until the next type style command or the end of the current group. You can also use the environment form of these commands; e.g. `\begin{tiny}...\end{tiny}`.

- `\tiny`
- `\scriptsize`
- `\footnotesize`
- `\small`
- `\normalsize`: (default)
- `\large`
- `\Large`
- `\LARGE`
- `\huge`
- `\Huge`

23.3 Low-level font commands

These commands are primarily intended for writers of macros and packages. The commands listed here are only a subset of the available ones. For full details, you should consult Chapter 7 of The LaTeX Companion.

`\fontencoding{enc}`: Select font encoding. Valid encodings include `OT1` and `T1`.

`\fontfamily{family}`: Select font family. Valid families include:

- `cmr` for Computer Modern Roman
- `cmss` for Computer Modern Sans Serif
- `cmtt` for Computer Modern Typewriter

and numerous others.

`\fontseries{series}`: font series. Valid series include:

- `m`: Medium (normal)
- `b`: Bold
- `c`: Condensed
- `bc`: Bold condensed
- `bx` Bold extended

and various other combinations.

`\fontshape{shape}`:Select font shape. Valid shapes are:

- `n`: Upright (normal)
- `it` Italic
- `sl`: Slanted (oblique)
- `sc`: Small caps
- `ui`: Upright italics
- `ol`: Outline

The two last shapes are not available for most font families.

`\fontsize{size}{skip}`: Set font size. The first parameter is the font size to switch to; the second is the `\baselineskip` to use. The unit of both parameters defaults to pt. A rule of thumb is that the baselineskip should be 1.2 times the font size.

`\selectfont`: The changes made by calling the four font commands described above do not come into effect until `\selectfont` is called.

`\usefont{enc}{family}{series}{shape}`: Equivalent to calling `\fontencoding`, `\fontfamily`, `\fontseries` and `\fontshape` with the given parameters, followed by `\selectfont`.