Typesetting in LaTeX 2ε

Version 0.1 **CHEAT SHEET** Jacob House

Paragraphs and Alignment

To tell La(TeX) that you would like to create a new paragraph, press the return key twice. That is, leave a blank line between the paragraphs in your `.tex` or `.ltx` file.

Similarly, one may use the TeX primitive `\par` at the end of a paragraph. This prevents the need of a blank line in the source code. For example,

```
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. \par
The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog.
```

Two consecutive empty lines generate two `\par` tokens. For all practical purposes this is equivalent to one `\par`, because after the first one TeX enters vertical mode, and in vertical mode a `\par` only exercises the page builder, and clears the paragraph shape parameters.

By default, LaTeX uses full justification; both the left and right edges of the text are smooth. One can disable full justification in favour of left-aligned text by using the `\raggedright` command. Similarly, default vertical alignment settings attempt to avoid large white spaces at the bottom of pages by stretching page contents. This may be disabled with the `\raggedbottom` command.

Additionally, LaTeX provides `center`, `flushleft`, and `flushright` environments which do exactly what one may expect.

Line Breaks

To manually break a line, one may the `\newline` command. Note that `\\` is an alias of `\newline`; the two have identical meaning. As we can see, when a line
is manually broken, the following line is not indented. Also notice that the broken line was not stretched to alignment with the paragraph. We can also use `\hfill\break` to create a non-stretching linebreak. This is contrasted by the `\linebreak` command, which *will* stretch or shrink whitespace between text when breaking lines to compensate for the text that has been forced out to the next line. Text, combined with an adjustable white space, is called glue and TeX may find the required stretch intolerable and deny the requested line break. To suppress TeX's fussiness over line breaking badness, use `\sloppy`. Issuing a `\fussy` returns TeX to its ordinary compulsive self.

Do *not*, under any circumstances, use `\newline`, `\\`, `\hfill\break`, or `\linebreak` to insert line breaks between paragraphs. Always insert a blank line or use `\par`.

Control sequences `\newpage` and `\pagebreak` behave similar so their horizontal equivalents; `\newpage` will immediately switch to the next page whereas `\pagebreak` will stretch page contents then break.

Boxes and Glue

A box is the TeX term for an invisible container that can hold a visible element, nothing, or other boxes. Glue is the TeX term for an invisible connector that determines the separation between boxes. Each separate visible element contained within a TeX document is contained within a box. A visible element can be a letter, image, geometric shape, etc. TeX builds pages by gluing boxes together according to the default TeX rules, default LaTeX rules, or document commands. In a typical document, letter boxes are glued to other letter boxes to form words, which are then elastically glued to other words to form sentences. Sentences are broken into lines and placed in paragraph boxes. Elastic glue is squeezed or stretched to fully justify lines within paragraph boxes. Paragraph boxes are glued to diagram boxes, and so on.

Producing Boxes

The `\makebox` control sequence may be used to create a box whose contents will not be broken, so it is often used to prevent hyphenation or to group text that should not be broken across several lines. It takes two optional parameters, width and position:

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

These parameters allow `\makebox` to be used in many ways, for example,

```
\makebox[9ex][s]{Bad text}%
\hskip-9ex%
\makebox[9ex][s]{X X X X}

Text \makebox[1.5\width][r]%
{running away}
```

```
Bad text X X X X
Text          running away
```

The control sequence `\mbox{text}` is the shorthand no-option version of `\makebox`.

Framed Boxes

The command `\framebox` behaves identically to `\makebox` except that it additionally draws a box around its contents. So we have

```
\framebox[width][pos]{text}
\fbox{text}
```

Inserting Vertical and Horizontal Glue

The general form to express a glue is: `<fixed part> plus <stretchable part> minus <shrinkable part>`. Each of these parts can be expressed in any of LaTeX units (mm, cm, pt, pc, em, etc.). For example `2cm plus 2mm minus 1mm`.

When composing a box which contains glues, TeX uses first their "natural dimensions" which is the fixed part (2cm in the above example). If the resulting box is underfull, then TeX expands all glue which has a non-zero stretchable part, up to the amount specified in that glue. In our example, the glue can stretch 2mm at maximum. If the box contains several glues with different stretchability, each one is stretched proportionally to the given stretchability. If the box is still underfull after stretching all glue to its maximum, a warning about "Underfull box" is issued.

Analogously, if the box is overfull, TeX tries to reduce the space by shrinking that glue. So, in our example, the final inserted glue can vary between 1.9cm and 2.2cm, depending on the size of the box which contains that glue.

The `plus` part in the glue can specify the value "infinite", through one of the following keywords: `fil`, `fill` or `filll`. Each of these infinites is infinitely greater than the preceding one.

Now that we have a basic understanding of glue, we can describe its insertion in a document. We use the control sequence

```
\hspace{<length>}
```

so insert a horizontal rubber length of `<length>`. We can utilize infinite stretch with the control sequences `\hfil` and `\hfill`, defined as

```
\hfil = \hskip 0pt plus 1fil minus 0pt
\hfill = \hskip 0pt plus 1fill minus 0pt
```

and, though it is not predefined as a macro, we can also use

```
\hskip 0pt plus 1filll
```

to get that third level of infinity. If glue is to be inserted at the beginning of a line, the starred variant of `\hspace{}`, `\hspace*{}`, is to be used.

Similarly, we use `\vspace{}` when inserting a vertical rubber length. Like its horizontal cousin, `\vspace{}` accepts standard glue lengths as an argument. For example,

```
\vspace{2in plus 1in minus 0.5in}
```

produces a vertical space ranging between 1.5 and 3 inches, depending on surrounding text. As before, there is also a `\vspace*{}` command. This is because the command `\vspace{}` has no effect at the top of a page or at the bottom. Why would you want space when you are about to move to a new page? If you insist, you must use `\vspace*{}` to force LaTeX to make space.

Indentation

The horizontal distance by which the first line of paragraphs is indented is stored in `\parindent` which can be set to a constant or to a multiple of another length.

```
\parindent=0pt
\parindent=1.5\parindent
```

To produce a zero `\parindent`, one may also load the parskip package.

Typography

LaTeX, being a markup language, uses special syntax to denote text styles such as *italicised*, **bold-face**, and sans-serif. In particular, we have the commands shown below.

italic	`\textit{...}`	`{\itshape ...}`
bold-face	`\textbf{...}`	`{\bfseries ...}`
slanted	`\textsl{...}`	`{\slshape ...}`
SMALL CAPS	`\textsc{...}`	`{\scshape ...}`
roman	`\textrm{...}`	`{\rmfamily ...}`
sans-serif	`\textsf{...}`	`{\sffamily ...}`
`monospaced`	`\texttt{...}`	`{\ttfamily ...}`
emphasised	`\emph{...}`	`{\em ... }`

Notice that there are two ways to encode each font style: as a command with an argument (e.g., `\textit{...}`) and as a group (`{}`) containing a switch (e.g., `\itshape`). For short texts, the command variant is often more useful whereas for longer texts the switch variant is more aptly suited (see Defining Macros).

LaTeX uses braces (`{}`) to denote what are referred to as *groups*. Looking at the command variant of the font control sequences above, one may mistake the `...` to be the argument passed to the command. In fact, the argument is what immediately follows the command; that is, the entire group `{...}`. What this means in terms of how commands behave, however, is that a command such as `\textbf` affects only the one argument that follows it. So,

```
\textit italics
```

would in fact only print "*i*talics," not *italics*. Conversely, a switch will affect *all* text that follows it, up to the end of the group. For this reason, we may also use `\bgroup` and `\egroup` rather than opening and closing braces,

```
\textit\bgroup some ↩
  italics text\egroup
```

some italics text

though this is shown here as an example and rarely used in practice.

It is recommended to use `\emph` to emphasise text, not `\textit`. `\emph` will italicise normal font text and, when invoked from an already-italicised context, convert text to a normal font for emphasis.

Older LaTeX2.09 documents may contain switches such as `\it`, `\bf`, `\sl`, `\sc`, `\rm`, `\sf` and `\tt` to denote the above font shapes. These commands are obsolete in LaTeX 2_ε and should not be used.

Legacy Support

The obsolescent TeX commands `\rm`, `\it`, `\bf`, etc. are declared in class files to function as their modern equivalents.

```
\DeclareOldFontCommand{\rm}{\normalfont\rmfamily}{\mathrm}
\DeclareOldFontCommand{\bf}{\normalfont\bfseries}{\mathbf}
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

So, writing

```
{\bf\it some text}
```

is equivalent to writing

```
{\normalfont\bfseries\normalfont\itshape some text}
```

As a result, the above code produces *some text*, not ***some text***, as intended. This is because `\normalfont` negated the affect `\bfseries` had.

The moral here is to *never* use the old font commands. You gain nothing and lose much of the flexibility of the new ones. (Well, after 25 years they aren't *really* new.)

We also have the following sizing commands, each of which have a corresponding environment.

tiny	`{\tiny ...}`
scriptsize	`{\scriptsize ...}`
footnotesize	`{\footnotesize ...}`
small	`{\small ...}`
normalsize	`{\normalsize ...}`
large	`{\large ...}`
Large	`{\Large ...}`
LARGE	`{\LARGE ...}`
huge	`{\huge ...}`
Huge	`{\Huge ...}`

Special Characters

LaTeX defines several special symbols and characters using combinations of other keyboard characters, either because the actual symbol is reserved for LaTeX syntax or keyboards do not have the symbol in question.

Opening quotation marks are produced using the backtick (`` ` ``) key while closing quotation marks are produced using the vertical quote (`'`) key. The double quote character is never used. Rather, to produce double quotation marks, use two backticks or vertical quotes in succession. A `\thinspace` (aliased to `\,`) can be used to separate double and single quotation marks that come one after another.

```
``\,`I'm positive,' he said, `I can do it.'\,''
```

To produce dashes of varying sizes, different numbers of hyphens must be used in the TeX source. One hyphen (`-`) is used to typeset a hyphen, used in compound words like 'over-the-counter.' Two hyphens (`--`) are used to create an en-dash that may be used to denote a range of numbers (1994–2019, for example), and three hyphens (`---`) are used to typeset em-dashes — used in paragraphs for interjections.

The tilde (`~`) character is used in TeX to denote what is referred to as a non-breaking space — known as a *tie*. That is, a space that cannot be used as a place to break a line.

```
\catcode`\~=\active
\def~{\penalty10000\ }
```

This should be used whenever a label and a number follow one another, as well as other situations such as phone numbers.

```
See Section~3.1 or call (234)~555-6789 for more details.
```

Ellipsis points may be produced using the `\ldots` macro and letter accents may be produced as in the following example.

```
H\^otel, na\"ive, ↩
  \'el\=eve,\\
sm\o rrebr\o d, ↩
  !`Se\~norita!,\\
Sch\"onbrunner Schlo\ss{}
Stra\ss e
```

Hôtel, naïve, élēve,
smørrebrød, ¡Señorita!,
Schönbrunner Schloß Straße

The below table shows a more comprehensive list of accents and non-English characters with their control symbols.

ò	`` \`o ``	ó	`\'o`	ô	`\^o`	õ	`\~o`
ō	`\=o`	ȯ	`\.o`	ö	`\"o`	ç	`\c c`
ọ	`\d o`	o̲	`\b o`	o͡o	`\t oo`	ß	`\ss`
œ	`\oe`	Œ	`\OE`	æ	`\ae`	Æ	`\AE`
å	`\aa`	Å	`\AA`	ı	`\i`	ȷ	`\j`
ø	`\o`	Ø	`\O`	ł	`\l`	Ł	`\L`

We also note that the ten special characters

$ % & ~ _ ^ \ { }

are produced by preceding the symbol with a backslash (with the exception of '\', which is typeset using `\textbackslash`.)

Ligatures

TeX also produces ligatures for certain characters (shown below).

ﬀ ﬁ ﬂ ﬃ

This may be suppressed by inserting an empty group `{}` between the characters:

```
f{}f \quad f{}i \quad f{}l \quad f{}f{}i
```

produces

ff fi fl ffi.

Modes in TEX

Math Mode

Inline v. Display Math Mode

Equation Environments

Math Symbols

Theorems

Floating Bodies

Graphics

Graphic Formats

Encapsulated PostScript

Portable Document Format, JPEG, and PNG

Cross-Referencing

Indexing

Sectioning

TEX uses a counter for each of its headings. Due to the hierarchical nature of document headings, the counter for a given heading is required to be reset to 1 each time the next higher-level number is incremented. The nesting of headers is as follows:

```
\newcounter{part}
\newcounter{chapter}   %% Book and report classes only
\newcounter{section}[chapter]
\newcounter{subsection}[section]
\newcounter{subsubsection}[subsection]
\newcounter{paragraph}[subsubsection]
\newcounter{subparagraph}[paragraph]
```

What this means is that each time the `chapter` counter is incremented, the `section` counter resets, each time the `section` counter is incremented, the `subsection` counter is reset, and so on.

Defining Macros

The words 'control sequence,' 'macro,' and 'command' all reference the same thing in LATEX. That is, a shorthand way of repeating much longer and more cumbersome code very easily. For example, `\TeX` is defined in `latex.ltx` (from `ltlogos.dtx`) as

```
\def\TeX{T\kern-.1667em\lower.5ex\hbox{E}\kern-.125emX\@}
```

Users of TEX can also define macros. Suppose we wish to use $\mathbb{R}$ to denote the real numbers. Ordinarily this would need to be typeset as `\mathbb{R}` in math mode or even more complicatedly as `$\mathbb{R}$` in text mode. Instead, define `\R` as follows.

```
\def\R{\ifmmode\mathbb{R}\else$\mathbb{R}$\fi}
```

Then we can write simply `\R`, either in math mode or horizontal mode, to get $\mathbb{R}$.

2019/01/22:92:00

Delimited Arguments

User-defined control sequences can be even more flexible than the example above through the use of delimited arguments.

The `\makeatletter` and `\makeatother` Macros

The `\makeatletter` macro changes the category code of '@' character to 11 (which is the catcode of ordinary characters a-z, A-Z). `\makeatother` reverts this to its original catcode of 12.

Knuth assigns a category code for each and every character like 0 for escape '\', 1 for begining of a group '{', 2 for end of group '}', 3 for math shift '$', 4 for alignmet tab '&', 5 for end of line, 6 for paramter '#', 7 for superscript '^', 8 for subscript '_', 9 for ignored character, 10 for space, 11 for letters, 13 for active character '~', 14 for comment character '%', 15 for invalid character and 12 for characters other than the above.

Knuth gives the freedom to change the catcode of any character anywhere. One could change the catcode of \ to 11 (i.e., a letter) and assign the catcode 0 to | so that `|section` becomes a function or control sequence.

You may have noted that an escape character combined with the characters of catcode 11 becomes a control sequence. As such, all the user defined control sequences or macros will be of this nature. This raises the issue of a user-defined macro having the same name as that of a macro in a package or even the TEX kernel. This can break packages and cause unpredictable behaviour.

In order to circumvent this foreseeable problem, package writers always use the character '@' in their control sequences by using `\makeatletter` to change the catcode of '@' character to 11 which is the catcode of alpha characters. At the end of the package, the author will revert the catcode of '@' to 12 with the command `\makeatother`. As a result, these macros cannot be redefined within the document without changing the catcode of '@' to 11 and novice users cannot accidentally create macros that might clash with kernel macros.

For completeness we also remark that `\makeatletter` and `\makeatother` are defined as below and hence the following definitions may be used instead of the control sequences.

```
\def\makeatletter{\catcode`\@11\relax}
\def\makeatother{\catcode`\@12\relax}
```

For example, suppose we wish to make a counter that the user can increment and get the value of, but not redefine or decrement. We might use

```
\makeatletter
\newcount\@counta
\@counta=0
\def\addtocounta{\advance\@counta by 1\relax}
\def\countaval{\the\@counta}
\makeatother
```

Registers and Tokens

Flow Control

Like any programming language, TEX provides all the standard mechanisms for flow control.

If ... Then ... Else ...

Of these flow-control mechanisms, the most commonly used is likely the *if-then* and *if-then-else* construct. TEXnicians creating documents designed for modularity and flexibility in reuse will find TEX's *if* construct of particular interest.

The ifthen package

The optional package

Troubleshooting

Overfull and Underfull `hboxes` and `vboxes`

File ended errors

Unresolved References

Undefined Control Sequence

There are two likely scenarios where one may encounter an 'undefined control sequence' error. The first is that not all of the correct macro packages have been loaded. For example, the `\mathscr{}` control sequence will throw this error if the mathrsfs has not been loaded.

The second and more complex reason that one may encounter this error is that a macro or definition was created within a group and TEX is now outside that group. For example,

```
{\def\a{b}}\a
```

will produce such an error because `\a` is only defined within the group.

For completeness, we also note that

```
\count0=1 {\count0=2 } \showthe\count0
```

will display the value 1; the assignment made inside the group is undone at the end of the group. Moreover, The choice of the brace characters for the beginning and end of group characters is not hard-wired in TEX. It is arranged like this in the plain format:

```
\catcode`\{=1 % left brace is begin-group character
\catcode`\}=2 % right brace is end-group character
\let\bgroup={ \let\egroup=}
```

Paragraph Ended Before ...Was Complete