Chapter 1

Getting Acquainted

Chapter 2

Getting Started

Chapter 3

Carrying On

Three modes [1] :

1. **paragraph mode** input as a sequence of words and sentences to be broken into lines, paragraphs and pages.
2. **math mode** Math mode begins with a command like $ or \(or \[or \begin{equation}, and leaves when finding the corresponding command that ends the formula.
3. **left-to-right mode or LR mode** LR mode consisers your input to be a string of words with spaces between them. It keeps going from left to right; it never starts a new line.

3.1 Changing the Type Style

Type style is used to indicate logical structure. In this book, emphasized text appears in *italic* style type and LaTeX input in `typewriter` style. In LaTeX, a type style is specified by three components: shape, series, and family.

Shapes

- Upgright shape (default). \textup{Upgright shape...}
- *Italic shape*. \textit{Italic shape...}
- *Slanted shape*. \textsl{Slanted shape...}
- Small Caps Shape. \textsc{Small caps shape...}

Series

- Medium series (default). \textmd{Medium series...}
- **Boldface series**. \textbf{Boldface series...}

Family

- Roman family (default). \textrm{Roman family...}
- Sans serif family. \textsf{Sans serif family....}
- `Typewriter family`. \texttt{Typewriter family...}

These commands can be combined in a logical fashion to produce a wide vaiety of type styles.

Who on **Earth** is ***ever*** going to use boldface sans serif or an italic typewriter type style?

[1] Paragraph mode corresponds to the vertical and ordinary horizontal modes in *The TeXbook*, and LR mode is called restricted horizontal mode there. LaTeX also has a restricted form of LR mode called *picture* mode that is described in Section 7.1.

Each of the text-style commands described above has a corresponding declaration. Boldface text can be obtained with either the \textbf text-producing command or the \bfseries declaration.

More and **more** armadillos are crossing the road.

The declarations corresponding to the text-producing commands are:

- **cmd decl**
- \textup \upshape
- \textit \itshape
- \textsl \slshape
- \textsc \scshape
- \textmd \upshape
- \textbf \upshape
- \textrm \upshape
- \textsf \upshape
- \texttt \upshape
- \textup \upshape

None of test text-producing commands or declarations can be used in math mode. Section 3.3.8 explains how to change type style in a mathematical formula.

Type style is a visual property. Commands to specify visual properties belong not in the text, but in the definitions of commands that describe logical structure. LaTeX provides the **\emph** command for emphaiszed text; Section 3.4 explains how to define your own commands for the logical structure in your document.

3.2 Symbols from Other Languages

The `babel` package allows you to produce documents in languages other than English, as well as multilanguage documents.

3.2.1 Accents

Note: While LaTeX accents annotations work, .tex files also support Unicode. This file is UTF-8.

El señor está bien, garçon.

El señor está bien, garçon.

The letters i and j need special treatment because they should lose their dots when accented. The commands \i and \j produce a dotless i and j, respectively.

Él está aquí.

3.2.2 Symbols

The commands in Table 3.2 can appear only in paragraph and LR modes; use an command to put one inside a mathematical formula.

The following six special punctuation symbosl can be used in amy mode:

- †\dag
- ‡\ddag
- §\S
- ¶\P
- ©\copyright
- £\pounds

3.3 Mathematical Formulas

A formula that appears in the running text, called an *in-text* formula, is produced by the **math** environment. This environment can be invoked with either of the two shortforms \(... \) or $...$ as well as by the usual **begin** ... **end** construction.

The **displaymath** enviornment, which has the short form \[...\], produces an unnumbered displayed formula. The short forms $...$, \(...\), and \[...\] act as full-fledged environments. A numbered displayed formula is produced by the **equation** environment. Section 4.2 describes commands for assigning names to equation numbers and referring to the numbers by name, so you don't have to keep track of the actual numbers.

The **math**, **displaymath**, and **equation** environments put TEX in math mode. TEX ignores spaces in the input when it's in math mode (but space characters ma still be needed to mark the end of a command name). Section 3.3.7 describes how to add and remove space in formulas. Remember that TEX is in LR mode, where spaces in the input generate space in the output, when it begins processing the argument of an \mbox command–even one that appears inside a formula.

All the commands introduced in this section can be used only in math mode, unless it is explicitly stated that they can be used elsewhere. Except as noted, they are all robust. However, \begin, \end, \(, \), \[, and \] are fragile commands.

3.3.1 Some Common Structures

Subscripts and Superscripts

Subscripts and superscripts are made with the _ and ^ commands.

- x^{2y} – x^{2y}
- x_{2y} – x_{2y}
- x^{y^2}
- x^{y_1}
- x_1^y
- x_1^y

Fractions

Fractions denoted by the / symbol are made in the obvious way.

Multiplying by $n/2$ gives $(m+n)/n$.

Most fractions in the running text are written this way. The \frac command is used for large fractions in displayed formulas; it has two arguments: the numerator and the denominator.

$$x = \frac{y + z/2}{y^2 + 1}$$

$$x = \frac{(y+z)/2}{y^2+1}$$

$$\frac{x+y}{1+\frac{y}{z+1}}$$

The \frac command can be used in an in-text formula to produce a fraction like $\frac{1}{2}$ (by typing $\frac{1}{2}), but this is seldom done.

Roots

The \sqrt command produces the square root of its argument; it has an optional first argument for other roots. It is a fragile command.

A square root $\sqrt{x+y}$ and an nth root $\sqrt[n]{2}$.

Ellipsis

The commands \ldots between commas produce two different kinds of ellipsis.

A low ellipsis: $x_1, \ldots, x_n$.

A centered ellipsis: $a + \cdots + z$.

Use \ldots between commas and between juxtaposed symbols like $a \ldots z$; use \cdots between symbols like +, -, and =. TEX can also produce verticaland diagonal ellipsis, which are used mainly in arrays.

$\vdots$ \vdots

$\ddots$ \ddots

3.3.2 Mathematical Symbols

Remember that mathematical symbols can be used only in math mode.

Greek Letters

The command to produce a lowercase Green letter is obtained by adding a \to the name of the letter. For an uppercase Greek letter, just capitalize the first letter of the command name.

Making Greek letters is as easy as π (or Π).

If the uppercase Greek letter is the same as its Roman equivalent, as in uppercase alpha, then there is no command to generate it. A complete list of commands for making Greek letters appears in Table 3.3. Note that some of the lowercase letters may have variant forms, made by commands beginning with

\var. Also, observe that there's no special command for an omicron, you just use an o.

Lowercase

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

Uppercase

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

Calligraphic Letters

TEX provides twenty-size calligraphic letters $\mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z}$, also called script letters. They are provided by a special type style invoked with the \mathcal command.

The shaded symbols require the latexsym package to be loaded with a \usepackage command.

Additional symbols can be made by stacking one symbol on top of another with the \stackrel command of Section 3.3.6 or the array environment of Section 3.3.3.

If $x \not\preceq y$ then $x \not\leq y - 1$.

Binary Operation Symbols

- $\pm$ \pm
- $\mp$ \mp
- $\times$ \times
- $\div$ \div
- $*$ \ast
- $\star$ \star
- $\circ$ \circ
- $\bullet$ \bullet
- $\cdot$ \cdot
- $\cap$ \cap
- $\cup$ \cup
- $\uplus$ \uplus
- $\sqcap$ \sqcap
- $\sqcup$ \sqcup
- $\vee$ \vee
- $\wedge$ \wedge
- $\setminus$ \setminus
- $\wr$ \wr
- $\diamond$ \diamond
- $\bigtriangleup$ \bigtriangleup
- $\bigtriangledown$ \bigtriangledown
- $\triangleleft$ \triangleleft
- $\triangleright$ \triangleright
- $\lhd$ \lhd
- $\rhd$ \rhd

- $\trianglelefteq$ `\unlhd`
- $\trianglerighteq$ `\unrhd`
- $\oplus$ `\oplus`
- $\ominus$ `\ominus`
- $\otimes$ `\otimes`
- $\oslash$ `\oslash`
- $\odot$ `\odot`
- $\bigcirc$ `\bigcirc`
- † `\dagger`
- ‡ `\ddagger`
- ⨿ `\amalg`

Here's how they look when displayed:

$$\sum_{i=1}^{n} x_i = \int_0^1 f$$

and in the text: $\sum_{i=1}^{n} x_i = \int_0^1 f$

Section 3.3.8 tells how to coerce TEX into producing $\sum_{i=1}^{n}$ in a displayed formula and $\sum_{i=1}^{n}$ in an in-text formula.

Log-like Functions

Logarithms obey the law: $\log xy = \log x + \log y$.

$\gcd(m, n) = a \bmod b$

$x \equiv y \pmod{a+b}$

Note that `\pmod` has an argument and produces parentheses, while `\bmod` produces only the "mod".

Some log-like functions act the same as the variable-sized symbols of Table 3.8 with respect to subscripts.

As a displayed formula:

$$\lim_{n\to\infty} x = 0$$

but in text: $\lim_{n\to\infty} x = 0$

3.3.3 Arrays

The `array` Environment

Arrays are produced with the `array` environment. It has a single argument that specifies the number of columbs and the alignment of items within the columns. For each column in the array, there is a single letter in the argument that specifies how items in the column should be positioned: `c` for centered, `l` for flush left, or `r` for flush right. Within the body of the environment, adjacent frows are separated by a \\command and adjacent items with a row are separated by an `&` character.

$$\begin{array}{ccrr} a+b+c & uv & x-y & 27 \\ a+b & u-v & z & 134 \\ a & 3u+vw & xyz & 2,978 \end{array}$$

There must be no `&` after the last item in a row and no \\after the last row. TEX is in math mode when processing each array element, so it ignores spaces. Don't put any exprac space in the argument.

In mathematical formulas, array columns are usually centered. However, a column of numbers often looks best flush right. Section 3.3.4 describes how to put large parentheses or vertical lines around an array to make a matrix or determinant.

Each item in an array is a separate formula, just as if it were in its own `math` environment. A declaration that appears in an item is local to the item; its scope is ended by the `&`, \\, or `\end{array}`

The \\is fragile.

Vertical Alignment

TEX draws an imaginary horizontal line through every formula, at the height where the minus sign at the beginning of the formula would go. An individual array item is itself a formula with a center line. The items in a row of an array are positioned vertically so their center lines are all at the same height.

Normally, the center line of an array lies where you would expect it, halfway between the top and bottom. You can change the position of an array's center line by giving an optional one-letter argument to the `array` environment: the argument `t` makes it line up with the top row's center line, while `b` makes it line up with the bottom row's center line.

The box around each array in the following formula is for clarity; it is not produced by the output:

$$x - \begin{array}{c} a_1 \\ \vdots \\ a_n \end{array} - \begin{array}{r} u-v \\ 12 \\ u+v \end{array} \begin{array}{r} 13 \\ \\ -345 \end{array}$$

More Complex Arrays

Visual formatting is sometimes necessary to get an array to look right. Section C.1.6 explains how to change the vertical space between two rows; Sections 3.3.7 eand 6.4.2 describe commands for adding horizontal space within a n item; and Section C.10.2 tells how to add horizontal space between columns. The `array` environment has a number of additional features for making more complex array;s they are described in Section C.10.2. The LATEX *Companion* describes packages that provide additional features for the `array` environment.

The `array` environment can be used only in math mode and is meant for arrays of formulas; Section 3.6.2 describes an analogous `tabular` environment for making arrays of ordinary text items. The `array` environment is almost always used in a displayed formula, but it can appear in an in-text formula as well.

3.3.4 Delimiters

A *delimiter* is a symbol that acts logically like a parenthesis, with a pair of delimiters enclosing an expression. Table 3.10 lists every symbol that TEX regards as a delimiter, together with the command or input character that produces it. These commands and characters produce delimiters of the indicated size. However, delimiters in formulas should be big enough to "fit around" the expressions that they delimit. To make a delimiter the right size, type a \left or \right command before it.

Big delimiters are most often used with arrays.

$$\left(\left| \begin{array}{cc} x_{11} & x_{12} \\ x_{21} & x_{22} \end{array} \right| \begin{array}{c} y \\ z \end{array} \right)$$

The \left and \right commands must come in matching pairs, but the mataching delimiters need not be the same.

$$x + y + z = \left(\begin{array}{c} a \\ b \end{array} \right[$$

Some formulas require a big left delimiter with no matching right one, or vice versa. The \left and \right commands must match, but you can make an invisible delimiter by typing a "." after the \left or \right command.

$$x = \left\{ \begin{array}{ll} y & \text{if } y > 0 \\ x + y & \text{otherwise} \end{array} \right.$$

3.3.5 Multiline Formulas

The `displaymath` and `equation` environments make one-line formulas. A formula is displayed across multiple lines if it is a sequence of separate formulas or is too long to fit on a single line. A sequence of equations or inequalities is displayed with the `eqnarray` environment. It is very much like a three-column `array` environment, with consecutive rows separated by \\and consecutive items with a row separated by `&` (Secxtion 3.3.3). However, an equation number is put on every line unless that line has a \nonumber command.

The middle column can be anything, not just a '='.

$$\begin{aligned} x &= 17y && (3.1) \\ y &> a+b+c+d+e+f+g+h+i+j+ \\ &\quad k+l+m+n+o+p && (3.2) \end{aligned}$$

Section 4.2 describes how to let LaTeX handle referenes to equations so you don't have to remember equation numbers.

The `eqnarray*` environment is the same as the `eqnarray` except it does not generate equation numbers.

$$\begin{aligned} x &= 17y \\ y &> a+b+c+d+e+f+g+h+i+j+ \\ &\quad k+l+m+n+o+p \end{aligned}$$

A `+` or `-` that begins a formula (or certain subformulas) is assumed to be a unary operator, so typing `$-x$` produces $-x$ and typing `$\sum -x_{i}$` produces $\sum -x_i$, with no space between the '-' and the "x". If the forula is part of a larger one that is being split across lines TeX must be told that the `+` or `-` is a binary operator. This is done by starting the formula with an invisible first term, produced by an `\mbox` command with a null argument.

$$\begin{aligned} x &= 17y \\ y &> a+b+c+d+e+f+g+h+i+j \\ &\quad +k+l+m+n+o+p \end{aligned}$$

A formula can often be split across lines using a `\lefteqn` command in an `eqnarray` or `eqnarray*` environment, as indicated by the following example.

$$\begin{aligned} w+x+y+z = & \\ & a+b+c+d+e+f+g+h+i+j+ \\ & k+l+m+n+o+p \end{aligned}$$

The `\lefteqn` command works by making TeX think that the formula it produces has zero width, so the left-most column of the `eqnarray` or `eqnarray*` environment is made suitably narrow. The indentation of the following lines can be increased by adding space (with the commands of Section 6.4.2) between the `\lefteqn` command and the \\.

Breaking a single formula across lines in this way is visual formatting, and I wish LaTeX could do it for you. However, doing it well requires more intelligence that LaTeX has, and doing it poorly can make the formula hard to understand, so you must do it yourself. This means that the formula may have to be reformatted if you change notation (changing the formula's length) or if you change the style of your document (changing the line length).

3.3.6 Putting One Thing Above Another

Symbols in a formula are sometimes placed one above another. The `array` environment is good for vertically stacking subformulas, but not smaller pieces – you wouldn't use it to put a bar over an x to form $\overline{x}$. LaTeX provides special commands for doing this and some other common kinds of symbol stacking.

Over- and Underlining

The `\overline` command puts a horizontal line above its argument.

You can have nexted overlining: $\overline{\overline{x}^2 + 1}$.

There's an analogous `\underline` command for underlining that works in paragraph and LR mode as well as math mode, but it's seldom used in formulas.

$\underline{\text{The}}$ value is $\underline{3x}$.

The `\underline` command is fragile.

Horizontal braces are put above or below an expression with the `\overbrace` and `\underbrace` commands.

$\overbrace{a+\underbrace{b+c}+d}$

In a displayed formula, a subscript or superscript puts a label on the brace.

$$\underbrace{a+\overbrace{b+\cdots+y}^{24}+z}_{26}$$

Accents

The accent commands described in Section 3.2.1 are for ordinary text and cannot be used in math mode. Accents in formulas are produced with the commands shown in Table 3.11. The letter a is used as an illustration: the accents work with any letter.

Wide versions of the `\hat` and `\tilde` are produced by the `\widehat` and `\widetilde` commands. These commands try to

choose the appropriate sized accent to fit over their argument, but they can't produce very wide accents.

$\hat{a}$ `\hat{a}` $\acute{a}$ `\accute{a}` $\bar{a}$ `\bar{a}` $\dot{a}$ `\dot{a}`
$\check{a}$ `\check{a}` $\grave{a}$ `\grave{a}` $\vec{a}$ `\vec{a}` $\ddot{a}$ `\ddot{a}`
$\breve{a}$ `\breve{a}` $\tilde{a}$ `\tilde{a}`

Here are two sizes of wide hat: $1\widehat{-x} = \widehat{-y}$.

The letters i and j should lose their dots when accented. The commands `\imath` and `\jmath` produce a dotless i and j, respectively.

There are no dots in $\vec{\imath} + \vec{\jmath}$.

Stacking Symbols

The `\stackrel` command stacks one symbol above another.

$A \xrightarrow{a'} B \xrightarrow{b'} C$
$\vec{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_n)$

See Section 3.3.8 for an explanation of the `\mathrm` command. The `\stackrel` command's first argument is printed in small type, like a superscript; use the `\texxtstyle` declaration of Section 3.3.8 to print it in regular size type.

3.3.7 Spacing in Math Mode

In math mode, TEX ignores the spaces you type and formats the formula the way it thinks is best. Some authors feel that TEX cramps formulas, and they want to add more space. TEX knows more about typesetting formulas than many authors do. Adding extra space usually makes a formula prettier but harder to read, because it visually "fractures" the formula into separate units. Study how forumulas look in ordinary mathematics texts before trying to improve TEXś formatting.

Although fiddling with the spacking is dangerous, you sometimes have to do it to make a formula look just right. One reason is that TEX may not understand the formula's logical structure. For example, it interprets (y dx) as the product of three quantities rather than as y times the differential dx, so it doesn't add the little space after the y that appears in $y\,dx$. Section 3.4 explains how to define your own commands for expressing this kind of logical structure, so you need to worry about the proper spacing only when defining the commands and when writing the formulas.

Like any computer program that makes aesthetic decisiosn, TEX sometimes needs human assistance. You'll have to examine the output to see if the spacing needs adjustment. Pay special attention to square root signs, integral signs, and quotient symbols (/).

The following five commands add the amount of horizontal space shown between the vertical lines:

- \, thin space
- \! negative thin space
- \: medium space
- \; thick space
- \␣ interword space

The \! acts like a backspace, removing the same amount of space that \, adds. The \, command can be used in any mode; the others can appear only in math mode. Here are some examples of their use, where the result of omitting the spacing commands is also shown.

$\sqrt{2}\,x$ instead of $\sqrt{2}x$
n/logn instead of $n/logn$
$\iint z\,dx\,dy$

As with all such fine tuning, you should not correct the spacing in formulas until you've finished writing the document and are preparing the final output.