An Introduction to TEX and LATEX

Peter M. Neumann
Mathematical Institute, Oxford
and The Queen's College, Oxford

Notes for three lectures given in Michaelmas Term 2016

Version of October 2016

Contents

Oxford University Mathematical Institute

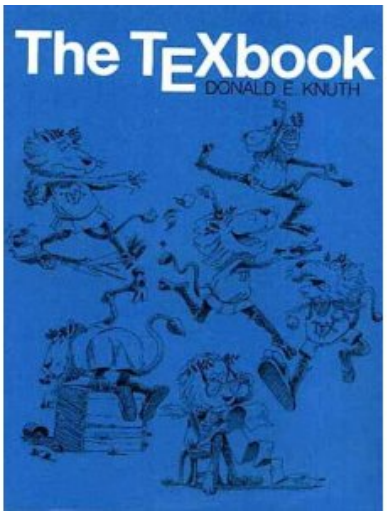
An Introduction to TEX and LATEX

1 Basics

1.1 Background

TEX is a typesetting system—**not** a word-processing system—designed with mathematics particularly in mind. Certainly it processes words, but it does much more. In particular, it typesets formulae following the typographical conventions that were developed over a period of hundreds of years to make complex mathematics relatively readable to the trained mathematician, and it typesets them to the highest standards of mathematical typography. It was created by the great Donald Knuth and published in his beautiful work *The TEXbook* (Addison-Wesley, Reading MA, 1984).

Donald Knuth of Stanford University



The TeXbook.

In use TEX calls on style files, font packages, specialised packages and other such paraphernalia to produce the particular formatting and style that an author or editor seeks. The main dialects are PlainTEX, AMSTEX and LATEX. The third of these is perhaps more than just a dialect of the language. It was created by Leslie Lamport and published in his book *LATEX—A Document Preparation System* (Addison-Wesley, Reading MA, 1985). It is an overlay of TEX designed to give colleagues facilities for organising their documents as preprints, articles, books, letters, and the like. It can be a bit bossy, but an experienced user can control that bossiness to produce beautiful and beautifully readable output. For some years it has been the industry standard. Most journals and many book publishers require authors to submit LATEX source material, as does the Oxford Mathematical Institute for lecture notes, exercise sheets, examination papers and theses.

Before Knuth created TEX mathematicians would type their articles on an ordinary typewriter, or on a typewriter (such as the IBM golfball machine or the Olivetti dual carriage machine—both famous in their time, forgotten now) modified in some way to offer a range

of mathematical symbols in addition to the standard QWERTY keyboard. That range was, however, very small compared with the ingenuity, sometimes perverse but always strongly defended, of mathematicians for inventing new notation. Most formulae and symbols had to be inserted by hand. Then specialist and highly skilled compositors turned the typescript (or, sometimes, manuscript when it was sufficiently legible and editors were sufficiently permissive) into print. For hundreds of years that meant moveable lead type, though for just a few years before the advent of TEX and laser-printing there were some photo-setting techniques in use. For mathematics the most common system was the *Monotype* system for casting the lead. This produced an approximation to the form that could be used for printing, but complicated formulae, especially displayed 'two-line' material, had to be adjusted by hand. At that time the typesetting of mathematics was such a specialised and skilled trade that costs were between 4 and 10 times the costs per page of ordinary books. For an account of what was involved the reader is referred to a third classic: *The printing of mathematics* by T. W. CHAUNDY, P. R. BARRETT, AND CHARLES BATEY (Oxford University Press, 1954).[1]

Nowadays mathematicians are their own typesetters. Many have understood enough of the conventions of mathematical typography and the needs of their readers to produce admirable output. Many, however, use LATEX to produce horrible travesties of mathematical typesetting. It is my fond hope that, after reading these notes or attending my three lectures, you will join yourself to the former group.

1.2 Getting equipped

In order to use TEX or LATEX you need an editor to produce a plain-text input file (often known as the *source-file*), and you need a compiler (which you summon to process it—although some systems provide for almost simultaneous automatic compilation). Both are available on the Mathematical Institute computing system

`<https://www.maths.ox.ac.uk/members/it/faqs/latex>`

but many people prefer to have a personal version on their own personal computer. There are several versions available free of charge for download from the web. The web-page given above leads one to

The Comprehensive TeX Archive Network (CTAN) `http://www.tex.ac.uk/`

and from there to

Repositories of TeX material: `http://www.tex.ac.uk/FAQ-archives.html`

Myself, I recently downloaded the free latest edition of the TEXLive system from `http://www.tug.org/texlive/` (MikTEX is a very good alternative). The TEXLive editor, though, has given me some problems (for example, I could not configure it to give the right version of an apostrophe), so I have downloaded **TEXmaker** as the front end. By a front end I mean not just an editor into which I can type my plain-text but more than that, namely, a program which also gives me buttons to press (click) to compile my documents and then view them. All such systems are available for MS-Windows, Apple Mac, Linux and most other operating systems.

[1] Theo Chaundy was reader in Mathematics in the University of Oxford, Student—that is, Tutorial Fellow—in Mathematics at Christ Church, and editor of the *Quarterly Journal of Mathematics* (Oxford), Barrett was the mathematical proof-reader at OUP, and Batey was Printer to the University, that is, the chief compositor at OUP.

There are also web-based LaTeX systems, such as **ShareLaTeX** and **Overleaf**. The latter is used by the UK Mathematics Trust to allow collaborative editing, which, to my mind, is the main value of such systems.

So your second step (after reading a brief introduction such as this) is to get access to, or equip yourself with, a suitable LaTeX system. You will also need a manual. You can download manuals of varying degrees of helpfulness from various sources—one of them being the Mathematical Institute web page (address above). It is probably worth investing in a real book, however. The one that I find pretty congenial and helpful is *A Guide to LaTeX* by Helmut Kopka and Patrick W. Daly [3] (ISBN-10: 0321173856, ISBN-13: 978-0321173850). When I bought my copy a few years ago it came with a TeXLive cd inside the back cover.

1.3 Commands in TeX and LaTeX

It is characteristic of TeX and LaTeX that they are programming languages in which commands are introduced with a backslash. Thus, for example `\TeX` tells the compiler (that is, the typesetting program) to produce TeX on the page, and `\noindent` at the start of a paragraph ensures that that particular paragraph will not be indented in the output.

Warning: All commands are of the form `\string` where '`string`' is a string of letters. When dealing with instructions the TeX system seeks the first non-letter key-stroke after the backslash and stops there. That key-stroke might be a space, a numeral, a punctuation mark—anything non-literal. And TeX takes this literally.

Moreover: a space after a command is taken simply as the command terminator and therefore does not appear as a space. Thus, for example, since many spaces are equivalent to just one (see below, p. 4),

```
\LaTeX            is a wonderful system.
```

produces

LaTeXis a wonderful system.

1.4 Getting started in LaTeX

The basic structure of a LaTeX input file consists of just three command lines with lines of other material between them:

```
\documentclass[X]{Y}

   [Preamble]

\begin{document}

   Your text

\end{document}
```

Here `X` is optional, `Y` is not. Almost always something between square brackets after a command is an optional parameter. The options `X` give the compiler such information as font size, paper size, and the like. For the full set of options available to `\documentclass` see a LaTeX manual (such as [3] for example). The possibilities for `Y` tell the compiler what kind of document is to be produced. It is a name for a *class*-file or *style*-file, such as `book`, `article`, `report`. Most publishers will have their own class- or style-file which they ask their authors to use. The first line of the file I am presently typing is

```
\documentclass[a4paper,11pt]{article}
```

The preamble which follows that first line gives further information about preferences. For example, the next three lines in my file are

```
\usepackage{latexsym,amssymb}
\usepackage{graphicx}
\usepackage{hyperref}
```

The first of these says that I want the system to summon up the LaTeX and AMS special symbol files and commands (which permit such constructs as `$\mathcal{A}$` to yield $\mathcal{A}$, `$\mathbb{R}$` to yield $\mathbb{R}$, `$\leqslant$` to yield $\leqslant$, for example) the second asks the system to summon the package that allows me to insert pictures, the third allows hyper-references. The remainder of my preamble contains my personal definitions of commands that I find useful. We'll return to this point later.

1.5 The basics of TeX: textmode; mathmode

The TeX program distinguishes formulae from text, **mathmode** from **textmode**. Text is the natural mode—as one would expect—and all that the compiler does is set it into the chosen type-fount, choose suitable line-breaks to justify the page, choose suitable page-breaks, and the like. When compiling what you have typed the system will treat any positive number of inter-word spaces in your input file as just one space. It will treat a single line break as an inter-word space. When it meets one or more blank lines it takes this simply as an instruction to start a new paragraph. Thus if you really want extra horizontal or vertical space you must use special instructions to impose your will. We'll come to such instructions later.

Mathmode comes in two forms. The first is 'textstyle', the form used for incorporating formulae such as $ax^2 + bx + c$ into text. The second is 'displaystyle', the form used for complicated or lengthy formulae such as

$$f(a) = \int_{\Gamma_r(a)} \frac{f(z)}{z-a}\,\mathrm{d}z$$

or

$$f(a) = f(0) + f'(0)a + \frac{f''(0)}{2!}a^2 + \cdots + \frac{f^{(n-1}(0)}{(n-1)!}a^{n-1} + R_n(f,a),$$

which tend to be too cramped or hard to read if incorporated into text.

Textstyle mathmode is created by enclosing the required formulae between dollar symbols. The quadratic formula above, for example, came from `$ax^2 + bx+ c$`. In LaTeX one has the alternative versions

`\(ax^2 + bx + c\)` and `\begin{math}ax^2 + bx + c\end{math}`.

I do not know what the rationale for these variants is and I myself never use them.

Displaymath is created by enclosing the required formula-describing text between double dollar signs: for example

```
$$
 f(a) = \int_{\Gamma_r(a)} {f(z)\over z - a}\ \mathrm{d}z
$$
```

or

```
$$
 f(a) = f(0) + f'(0)a + {f''(0) \over 2!}a^2 + \cdots
 + {f^{(n-1}(0) \over (n-1)!} a^{n-1} + R_n(f,a),
$$
```

There is a minor but useful point to note here. Following a suggestion that I got either from Knuth's *TEXbook* or from Michael Spivak's original AMSTEX manual *The Joy of TEX* [5]—I cannot now remember which—I start new lines in my input file for each instance of `$$`. It helps greatly to make the input file readable and editable.

As with textstyle mathmode, LATEX has alternative versions for displaymath that replace `$$ ... $$` with

`\[ ... \]` or with `\begin{displaymath} ... \end{displaymath}`.

Again, although I am told that these are the modern usages, no-one has been able to explain to me why they are better than what Knuth introduced. To my mind they are not: mathematics already uses brackets ( ) [ ] all over the place whereas $ symbols stand out to make my page easier to decode with the naked eye. So these are constructs that I myself do not use, but there are other variants that I do find useful and to which we'll return later.

2 The interior and the exterior of documents

2.1 The interior: text in TEX

Ordinary text is typed as ordinary text for TEX, and by default it is printed in roman type. Sometimes, however, you'll want variations such as *italic* or **bold face** or CAPS AND SMALL CAPS (usually known nowadays simply as SMALL CAPS) or `typewriter` fonts. To achieve these I use `{\it italics}`, `{\bf bold face}`, `{\sc small caps}`, and `{\tt typewriter}` respectively. LATEX has alternatives such as `\textit{italic}`, `\textbf{bold face}`, `\textsc{Small caps}`, `\texttt{typewriter}`. If, within a passage of some other font, you wish to return temporarily to roman type you can use `{\rm roman}` or `\textrm{roman}`. A related command is `\emph{text}`. The effect of this is to turn 'text' into italic type if the ambient type is roman and *vice-versa*.

Just as you can change the type font so you can change the type size. Thus

`{\large large}`, `{\Large larger}`, `{\LARGE very large}`,

produce large, larger, very large, respectively. Similarly

`{\small small}`, `{\footnotesize smaller}`, `{\tiny too small}`,

produce small, smaller, too small.

Notice here the use of braces `{ }`. They delimit blocks. Sometimes those blocks are there to limit the effect of a command such as in `{\it italics}` or `{\LARGE large}` and stop it from continuing in an unwelcome fashion; sometimes the block becomes the argument to a command such as `\textit` that requires input; and sometimes, as in `{\TeX}`, the closing bracket provides the non-letter character needed to terminate the command, in which case it has to be matched with its paired opening bracket—since in a civilised world all brackets (of all types) always come in pairs.

We now have a problem, however. Since symbols such as `\`, `{`, `}`, `$` have special meanings we cannot type them into our input file and expect them to appear type-set in the output. These are not the only so-called **command** characters. The full list is `\ $ # ~ _ ^ % { }`. All except the backslash can be obtained by printing the character preceded by a backslash. Thus `\$` produces $. A similar technique produces characters that are used in print but do not appear on your keyboard. Thus, for example, `\S` produces §, `\dag` produces †, `\ddag` produces ‡, and (because it does not appear on other than British keyboards), `\pounds` produces £.

How do we get the backslash? In mathmode we have `\backslash`, not to be confused with `\setminus` which puts the space appropriate for a binary operator fore and aft as in $A \setminus B$. (For comparison: `$A \backslash B$` produces $A\backslash B$.) There is, however, a text alternative.

One can go into 'verbatim' mode and type `\verb=\=`. The LaTeX compiler takes the command `\verb=characterstring=` and prints the character string `characterstring` like that, exactly as you typed it. There are three rules associated with this command. The first is that the delimiters which bracket the character string may be any symbol other than a letter or a space and must be the same symbol fore and aft of the string. Secondly, that symbol must of course not occur in the character string itself. Third, the whole string `\verb=characterstring=` must be short enough that it fits into one line of typing. Verbatim mode is indispensable when, as here, one wishes to write about TeX and LaTeX. It comes in another form, the verbatim environment similar to other environments (to be discussed below). I'll not treat this: see a LaTeX manual.

Let's return to those nine special characters and ask a different question about them, namely, the obvious one, what are they all for? The first two, `\`, `$` and the braces `{ }` have already been explained; `#` is used as a place-holder when in the definition of commands that take one or more arguments; `~` provides a protected space (see below, p. 7); the characters `_`, `^` are used to get subscripts and superscripts in mathmode; and finally, `%` allows one to make comments in one's input file. Any typing appearing on a line after this symbol is ignored by the type-setting program. This applies only to the line in which the symbol `%` appears. Thus if one wants to make a comment that takes several lines, each of those lines must start with the symbol `%`.

Accents are also produced using special commands:

`\'` puts an acute accent over the following letter as in é, ź;

`\‘` puts a grave accent over the next letter as in ù, x̀;

`\^` provides a circumflex as in â, ŵ; and

`\"` provides an umlaut as in ö, v̈.

There are many more. For a full list of accents see any TeX or LaTeX manual.

2.2 The interior: environments in LaTeX

Environments provide a very useful way to organise blocks of special material in LaTeX. What you type is always of the form

```
\begin{environmentname}
    Your special material
\end{environmentname}
```

Thus if you type, for example,

```
\begin{theorem}
    All groups of order $1$ are purple.
\end{theorem}
```

you should get something like

THEOREM 2.1 *All groups of order* 1 *are purple.*

Notice that this particular environment is a little on the bossy side. It has introduced a line space fore and aft of the statement of the theorem, it has set the statement in italic type, it has given the theorem a number. Most of the time that is exactly what you want. For those rare occasions when you want something else you can personalise environments. How to do that is not something to be treated here: see a LaTeX manual.

It is worth remembering that one can label words and items. Thus, if I had typed

```
\begin{theorem}\label{Some purple groups}
    All groups of order $1$ are purple.
\end{theorem}
```

(as indeed, I secretly did) then I would have got exactly the same arrangement of my theorem, but I would now be able to cross-reference it elsewhere in my document. Typing

```
Most groups are green but by Theorem~\ref{Some purple groups}
certainly not all.
```

yields

Most groups are green but by Theorem 2.1 certainly not all.

Notice the tie character ~ (known technically as a 'protected space'). It produces an interword space and also ensures that the reference number is not separated from the word 'Theorem' by a line-end. One gets page references in a very similar way:

```
By the theorem on p.\,\pageref{Some purple groups}
not all groups are green.
```

yields

By the theorem on p. 6 not all groups are green.

Again there is a special construction here: backslash+comma produces a thin space that ties (in the sense of not letting the TEX compiler put a line-break at just this spot).

Many environments are standard for all classes of document. You'll find

- theorem, lemma, proposition, definition [you or your publisher define these];
- itemize, enumerate, description [for creating lists like this one];
- large, Large, LARGE [for increasing type size];
- small, footnotesize, scriptsize, tiny [for decreasing type size];
- titlepage;
- quotation, quote, verse

and many, many more—as always, consult a proper manual.

A very important environment is `thebibliography`, which I use to deal with reference lists. It is perhaps a little old-fashioned now. Many colleagues prefer to use BibTEX, which, however, I myself found cumbersome and somewhat overbearing (and inflexible) when I tried it some years ago. If you want to use BibTEX read the manuals or get advice from one of its fans. I have used `thebibliography` for these lecture notes by typing

```
\begin{thebibliography}{10}

  \bibitem{Chaundy-etal-1954}
    {\sc T.~W.~Chaundy, P.~R.~Barrett, and Charles Batey}
    {\it The printing of mathematics},
    Oxford University Press, 1954.

  \bibitem{Knuth1984}
    {\sc Donald Knuth},
    {\it The {\TeX}book},
    Addison-Wesley, Reading MA, 1984.

\end{thebibliography}
```

In the first line `{10}` is a sample label. It tells the system how much width to allow for item labels in the bibliography; `\bibitem` indicates the start of the next item in the reference list; its argument, here `Chaundy-etal-1954` or `Knuth1984`, is a key or label, which may be called with the command `\cite`. Thus if I type

```
This command is treated in Knuth's classic \cite[p.\,45]{Knuth1984}.
```

then I should get

This command is treated in Knuth's classic [2, p. 45].

By default bibliography labels are arabic numbers but in fact `\bibitem` permits an optional label for the reference. If, for example, I had typed

```
\bibitem[Knuth (1984)]{Knuth1984}
```

in the list in the bibliography environment then the example would have come out as

This command is treated in Knuth's classic [Knuth (1984), p. 45].

2.3 The interior: formulae in TEX

Let's return to mathmode. Recall (from Section 1.5) that `$formula$` produces a textstyle formula that will be incorporated into the text and `$$formula$$` will give you the formula in displaystyle. Recall also that there are variants of each of these. In the displaymath case, however, there more possibilities than were mentioned before. One of them is

`\begin{equation} ... \end{equation}`,

which behaves differently in that it automatically adds a sequential equation number. There are, in fact, two more constructions that are used for displaying complex mathematics. They are the environments

```
\begin{eqnarray} ... \end{eqnarray}
```

and

```
\begin{eqnarray*} ... \end{eqnarray*}
```

which are used to produce multiline formulae. An example is

```
\begin{eqnarray*}
  f(x) \kern-6pt&=&\kern-6pt (x - 1)(x - 2)(x - 3)(x - 4)(x - 5)\\
   \kern-6pt&=&\kern-6pt x^5 - 15 x^4 + 85x^3 - 225x^2 + 274x - 120,
\end{eqnarray*}
```

yielding

$$\begin{aligned} f(x) &= x(x-1)(x-2)(x-3)(x-4)(x-5) \\ &= x^5 - 15x^4 + 85x^3 - 225x^2 + 274x - 120, \end{aligned}$$

which is not guaranteed to be correct, but illustrates what is possible. The former adds equation numbers; the * form (used for this little example) does not. Notice that the construction uses an array with three columns. The symbol & separates those columns (it acts as a sort of tabulator stop). But because the columns offer too much space (10pt, I believe) round the = symbol, I correct with `\kern-6pt`.

Typing formulae for TEX to compile is pretty intuitive. It is important to remember, though, that in mathmode the system ignores spaces and, unless instructed otherwise, prints all letters in the 'mathitalic' font. Thus

```
$|x| = x if x \geq 0 and |x| = -x if x < 0$.
```

produces

$|x| = xifx \geqslant 0and|x| = -xifx < 0$.

which no-one could possibly want. In this particular case it is clear that the sentence contains four separate formulae, each of which is a clause in its own right, so it should be typed as

```
$|x| = x$ if $x \geq 0$ and $|x| = -x$ if $x < 0$.
```

This yields

$|x| = x$ if $x \geqslant 0$ and $|x| = -x$ if $x < 0$.

Sometimes one wishes (or even needs) to move into textmode within a formula. This is done with `\mbox{yourtext}`. And of course within an mbox one can move back into mathmode (textstyle mathmode, not displaymath). Thus

```
$$
  \mathbb{Q} := \{x \in \Reals \mid \mbox{$x$ is rational}\}.
$$
```

produces

$$\mathbb{Q} := \{x \in \mathbb{R} \mid x \text{ is rational}\}.$$

Most commands are intuitive and easily learned. Greek letters are `\alpha`, `\beta`, ..., with `\Alpha`, `\Beta`, ... giving their capital versions; `\int` gives an integral sign; `\sum` gives a summation symbol $\sum$ (not to be confused with Greek Σ), `\prod` gives a product symbol $\prod$ (not to be confused with Greek Π). For a superscript use `^`, for a subscript use `_` (the underline key on your keyboard). Note, however, that these require multi-symbol arguments to be put into a block picked out with braces `{ }`. This is not necessary if the exponent or subscript consists of a single symbol. Thus we type `x^{-2}gx^2` to get $x^{-2}gx^2$ or `$\sum_{i = 1}^n x_i$` to get $\sum_{i=1}^{n} x_i$.

There are various fonts available in mathmode. For example `\mathrm{something}` will print what you have written as something in roman type within your formula. The rules of mathmode still apply, however, so `$\mathrm{some thing}$` ignores the space and also produces something. Besides `\mathrm`, the ones I find worth remembering are

- `\mathbb` (blackboard bold) which yields $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$ and other such letters;
- `\mathbf` which gives boldface type within formulae;
- `\mathcal` which offers $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, ..., $\mathcal{Z}$ ('caligraphic') and
- `\mathfrak` which yields $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, ..., $\mathfrak{Z}$ ('Fraktur'—sometimes called 'old German' in English and 'Altenglisch' (noun) or 'altenglisch' (adjective) auf Deutsch).

For the full range see your LaTeX manual.

There are many kinds and sizes of brackets (also known as 'delimiters') available in mathmode. The obvious ones are `( )`, `[ ]`, `| |`, and these are typed as you would expect. To get braces `{ }` however you must type `\{ \}` (see p. 5). And to get angle brackets $\langle\,,\rangle$ one types `\langle`, `\rangle`. All such brackets can be made in various larger-than-normal sizes if preceded with `\big`, `\Big`, `\bigg` or `\Bigg`. Thus

```
$\Bigg(\bigg(\Big(\big((x)\big)\Big)\bigg)\Bigg)$
```

produces $\Big(\bigg(\Big(\big((x)\big)\Big)\bigg)\Big)$. (I have never understood why `\big` produces no noticeable enlargement.) We also have `\left` and `\right`. Placed in front of delimiters they produce growth to accommodate the bracketed formula. Thus

```
$\left(\left(\left(\left(\left((x^2)^2
\right)^2\right)^2\right)^2\right)^2\right)^2$
```

produces $\left(\left(\left(\left(\left((x^2)^2\right)^2\right)^2\right)^2\right)^2\right)^2$.

This facility to get delimiters to grow must be used in pairs. Just once in a while one wants an unpaired bracket that grows. Then `\left bracket` is paired with `\right.` to keep the balance. One situation where this is very useful is with arrays. They are made with the array environment. You'll find details in a LaTeX manual. Here is a simple example:

```
$$
 |x| := \left\{ \begin{array}{rl}
 x &\mbox{if $x \geq 0$},\\
 -x &\mbox{if $x \leq 0$}.
 \end{array} \right.
$$
```

produces

$$|x| := \begin{cases} x & \text{if } x \geqslant 0, \\ -x & \text{if } x \leqslant 0. \end{cases}$$

Here the argument `rl` tells the compiler that the array is to have two columns, the first justified right, the second justified left.

2.4 Tables and diagrams

I do not need tables much myself, and therefore am not very familiar with the way in which they are built in LaTeX. On those rare occasions when I do need them I rely heavily on my manual [3, Sect. 4.8]. The words to look for in an index are 'table' and 'tabular'.

One can create simple diagrams in LaTeX using the 'picture' environment (see [3, Sect. 6.1], for example). I find it clumsy and difficult, though, for anything even slightly sophisticated. Therefore I now create geometrical diagrams using some other system (such as GEOGEBRA), save them as JPG or PDF files and import them as pictures. To import pictures I have the line

```
\usepackage{graphicx}
```

in the preamble of my source files. Then a command such as

```
\includegraphics[width=200pt]{TeXbookCover.jpg}
```

does the job (see [3, Sect. 6.2])—indeed, it was this that did the job for my p. 1 above. There are many parameters of placement, size, and the like, that are under your control.

2.5 Controlling spaces

As has already been mentioned twice, multiple inter-word spaces in your TeX source file count as just one space, and multiple inter-line spaces count as just one (giving a new paragraph in textmode). Therefore extra space has to be inserted manually where it is wanted. To understand properly how spacing works in TeX and LaTeX one should know about horizontal boxes, horizontal glue, vertical boxes and vertical glue. I do not propose to go into such matters here: once you have some facility with using the system in practice you will probably wish to learn more about it from the manuals quoted in §1 and cited in the reference list below. I shall say here just enough to get you started.

Measures of length are given as $x$pt (printers' points, of which there are 72 to the inch, so 1pt = 0.35mm and there are about 30pt to 1cm), $x$mm, $x$cm or $x$in (inches). Here x need not be an integer and (usually) need not be positive.

There are several commands that give horizontal space. There are `\ `, a backslash followed by a space, which produces what would be an inter-word space except that it is fixed and does not compress or expand; `\,` which is the same as `\thinspace`, producing a thin space needed sometimes to make formulae more readable. It is used, for example between $f(x)$ and $\mathrm{d}x$ in the construction $\int f(x)\,\mathrm{d}x$. For larger spaces I know of `\hglue15pt`, `\hspace{15pt}`, `\kern15pt`, and perhaps there are more. They are not quite interchangeable. I need the first if I want space at the beginning of a line; I use the second in most other contexts; I use the third

in mathmode (often for small negative amounts of space). But my usage may be no more than habit. To learn their proper use you should read the manuals and you should experiment.

There are two special spaces that can be useful, especially in mathmode. They are `\quad` which inserts a space as wide as the current type size (a printer's quad space used to be as wide as M, which was supposed to be as wide as it was high), and `\qquad` which is twice as wide. This is 11pt type, so here a quad is 11pt wide and `\qquad` should produce a fixed space 22pt wide.

The command `\break` at the end of a line will spread that line out and start a new one. There are also `\hfil` and `\hfill` which fill out a line with space. In particular, `\hfill\break` can be used as a sort of 'return' that starts a new line without trying to justify the current line. A double backslash `\\` has the same effect in textmode.

For vertical space there are similar commands. We have `\vglue15pt`, `\vspace{15pt}`, and three commands that can be used to give space before a new paragraph, namely `\smallskip`, `\medskip` and `\bigskip`. Of these, `\bigskip` gives a vertical space that is as deep as a single line of type (so it is the vertical equivalent of the horizontal space `\quad`). It is what usually appears before and after a theorem, a lemma, a quotation, or the like, when the appropriate environment is invoked. It is not always quite the same as a line-space because it can expand or contract a little depending on where the type-setting program calculates optimal page-breaks. As with horizontal space, we also have `\vfil` and `\vfill` (which differ only a little—see a manual for details). These will fill out a page and help you control a page-break. In fact, however, we have `\newpage` (equivalent to `\vfil\eject`) to force a page-break.

Spacing of paragraphs has its own commands. The paragraph indent `\parindent` is set to 15 pt by default in most document classes. If you come to a paragraph that you do not want indented you can use the command `\noindent` immediately preceding it. The default is no vertical space between paragraphs, but you can change that with `\setlength{\parskip}{x}`, where x must be a length such as 3 pt.

There is a very versatile command intended to produce horizontal or vertical lines ('rules' was the technical term used by printers), but which can also be used to control spacing. The command `\rule[r pt]{w pt}{h pt}` produces a 'line' that is raised r pt above the baseline (below if r is negative), is w pt wide, and is h pt high. Thus:

`\rule[5pt]{10pt}{1pt}` produces —

`\rule[-5pt]{1pt}{10pt}` produces |

`\rule{10pt}{10pt}` produces ■.

If the width w is set as 0 pt (or 0 cm, or whatever your preferred unit of length may be) then the rule is invisible, but still has the height and the depth that you have set. Thus for example `$\left(1\rule[-5pt]{0pt}{20pt}\right)$` produces $\left(1\right)$. This particular example is no more than an illustration, but rules of zero width or height can be remarkably useful sometimes.

2.6 The exterior: how to format a document in LaTeX

Let's return to the LaTeX document structure described in Subsection 1.4 on p. 3.

```
\documentclass[X]{Y}
   [Preamble]
\begin{document}
   Your text
\end{document}
```

and consider the preamble and the main text in a little more detail. As I have already said, the preamble might begin with some `\usepackage` commands summoning up a selection from the huge collection of packages that has accumulated over the years. To begin with you will find that a couple of font packages (such as the ones I illustrated, or `\usepackage{times}`) and a graphics package are quite sufficient. Once you have become a TEX expert you can be more adventurous.

What follows in my own preamble is a sequence of lines that override the default for the 'article' class. Thus I have

```
\setlength{\textwidth}{450pt}
\setlength{\textheight}{720pt}
\setlength{\topmargin}{-50pt}
\setlength{\oddsidemargin}{12pt}
\setlength{\parskip}{1pt plus 1pt}
\setlength{\mathsurround}{1pt}
\renewcommand{\baselinestretch}{1.05}
```

which re-define the page parameters. Most of them should be self-explanatory. Note, though, that lengths can be pt (points, of which there are 72 to the inch), mm, cm, or in (inches). I myself acquired a background knowledge of typography many years ago in the days of hot-metal type-casting, so I have become used to working in printers' points, tiny though they are. Most people prefer centimetres, I think. You need to experiment to find the units that you are comfortable with and the settings that you like. Those settings may be different for different documents to ensure good page-breaks, for example. Both `\oddsidemargin` and `\evensidemargin` exist in LATEX because for some documents you may need a different left margin for odd-numbered pages (rectos, those that appear on the right in an open book) and even-numbered pages (versos, those that appear on the left).

The length `\parskip`, paragraph skip, is the extra space (or none) between paragraphs; the specification `{1pt plus 1pt}` says that a 1-point space shall be expected, but may expand up to 2 points if TEX needs flexibility to cope with a difficult page break. The command `\mathsurround` puts just a little extra space fore and aft of in-line formulae. The command `\baselinestretch` expands (or contracts if the parameter is < 1) the spacing between lines. In particular, for example `\renewcommand{\baselinestretch}{1.09}` would increase spacing by 9%, so would give what in the days of hot metal (lead type) would have been described (given that one of my document class options is `11pt`) as '11-point type on a 12-point body'.

Next my preamble has the following three lines:

```
\newtheorem{theorem}{\indent\sc Theorem}[section]
\newtheorem{lemma}{\indent\sc Lemma}[section]
\newcommand{\Thmstop}{\hglue-6pt.\kern6pt}
```

The first of these commands defines the theorem environment for me and customises it just a little. The first input parameter `theorem` or `lemma` names the environment; then it is called with the command

```
\begin{theorem} . . . \end{theorem}
\begin{lemma} . . . \end{lemma}
```

respectively. Then comes the structure name, Theorem or Lemma in these cases, that is printed with a number to form the name of the theorem, the lemma, or whatever. The default is non-indented and boldface so I have customised it a little to be indented and printed in small caps

to pander to my preference. Finally, the option `section` tells the system to print the number as $a.b$ where a is the number of the current section and b is the serial number of the theorem within the section. I hold that there should be a full stop after the theorem number. It would have been easy enough to adjust the theorem environment in my personal style files to get that, but I have been too lazy. Instead I use the last of the three lines shown above as in the example below:

```
\begin{theorem}\Thmstop
   All groups of order $27$ are green.
\end{theorem}
```

which yields

THEOREM 2.2. *All groups of order* 27 *are green.*

After those commands my preamble has a few definitions of notation, mostly not used in this document, but used in a problem sheet that I used as a template. I have, for example,

```
\newcommand{\Half}{{1\over2}}
[this is old-fashioned PlainTeX; modern usage in AMSLaTeX is \frac{1}{2}]
\newcommand{\Nats}{\mathbb N}
\newcommand{\Sym}{\mathrm{Sym}\kern1pt}
```

Instead of putting such definitions into the preamble one could put them all into a separate file called something like `macros.tex` or `mystyle.tex` and then call this file at the start of the main body of text, right after `\begin{document}`, with the command `\include{macros}` or `\include{mystyle}` .

As I have already indicated, the main body of text comes between `\begin{document}` and `\end{document}`. For a short article (like this one) it is reasonable to type everything into a master file (as I have done). For a long article, a dissertation or a book, however, it is wiser to split the work into sections or chapters typed into separate files, with commands in the (relatively short) master file that summon them up. Then what appears between `\begin{document}` and `\end{document}` will be a series of commands of the form `\input{file-name}`, one for each section or chapter, as necessary.

LaTeX has commands such as `\chapter`, `\section`, `\subsection` to ensure that when the master file is compiled the chapters, sections and subsections all fit together properly. Each of these elements comes with a counter so that the chapters, sections, subsections, theorems, lemmas, etc., get numbered automatically. There are two sensible ways to understand how this works: read a good manual; experiment for yourself.

The filenames for chapters or sections should be something like `ChIntegration.tex` or `SectIntro.tex`. As was indicated above, these are summoned into your main source file (the one that begins`\documentclass[X]{Y}` and ends `\end{document}`) by the command `\include{SectIntro}` or `\include{ChIntegration}`. It may be tempting to give those files names like `Sect1.tex` or `Ch5.tex`, but in my experience that can be unwise because section and chapter numbers tend to change as a long document grows.

Finally, of course, comes the bibliography, which is essential for any work of scholarship. A convenient way of producing it was discussed above (see p. 7).

3 Sense and sensibility

It is good practice to ensure that all your work has title, author (yourself), date. For a long work, such as a dissertation, a version number can also help you to know where you have reached with your revisions. This may seem pompous, but a small amount of pomposity can be pretty helpful.

3.1 Hyphens and dashes

Do not misuse hyphen -, en-dash –, em-dash —. Hyphens are traditionally used if a word has to be broken at the end of a line of text to ensure good spacing; they are also used in compound adjectival phrases made from nouns as in 'a white-knuckle ride', 'a red-nosed reindeer'; also in double-barrelled names such as Heath-Brown and Swinnerton-Dyer.

An en-dash, typed as two adjacent hyphens `--` in TEX, is used for ranges such as 1984–2009 or pp. 123–254.

The em-dash, typed as three adjacent hyphens `---` in TEX, is a form of punctuation used—much as parentheses are—to mark off certain kinds of subordinate clauses.

Mathematicians have acquired a habit of using the en-dash to create eponymous names for theorems, conjectures, and the like, as in the Schröder–Bernstein Theorem or the Cantor–Schröder–Bernstein Theorem.[2] In recent years I have sometimes seen authors go a step further and use a construction that makes no sense at all: 'the theorem of Cantor–Bernstein' or 'Cantor–Bernstein's theorem'. In that context the conjunction *must* be 'and' or '&' and the possessive *must* be in the 'of' form: 'the theorem of Cantor and Bernstein'.

3.2 Stops

The full stop (period in US English) and the abbreviation stop are typographically different. Typographical convention expects extra space after a full stop and before the start of the following sentence, and TEX inserts such space automatically. For an abbreviation stop, as seen in 'the works of D. E. Knuth' for example, we type `.\ ` (using a fixed space after the stop) , or, if we wish to inhibit line-breaking after the stop, we type `.\~`, using the protected space (tie). It often looks better if thin spaces are used after initials, however.

3.3 Correct use of symbols

In mathmode we have all the correct symbols for mathematics: do not use the wrong ones. I have seen such horrors as

$$< a, b | a^2 = b^2 = (ab)^3 = 1 > \cong Sym(3), \quad < X, Y >= X^T A Y \quad \text{and} \quad e^{iy} = cosy + isiny.$$

Compare them with

$$\langle a, b \mid a^2 = b^2 = (ab)^3 = 1 \rangle \cong \mathrm{Sym}\,(3), \quad \langle X, Y \rangle = X^T A Y \quad \text{and} \quad \mathrm{e}^{\mathrm{i}y} = \cos y + \mathrm{i} \sin y.$$

The latter are much more professional, much more easily readable.

- Never use $<$, $>$ as angle brackets. Since they are binary relation symbols they come with spaces fore and aft. We have `\langle, \rangle` to give us $\langle$, $\rangle$, which are genuine delimiters (brackets).

[2] Myself, I have never quite come to terms with the mild pottiness of using something that looks similar to a subtraction symbol when the symbols & and + are available.

- Use `\mid` in contexts such as $\{x \in \mathbb{R} \mid x^2 < 2\}$: it is spaced correctly, whereas a vertical bar is not.

- Functions such as log, exp, cos, sin, max, min, ker, det, dim are always printed in roman type in formulae. Use `\log`, `\exp`, `\cos`, `\sin`, `\max`, `\min`, `\ker`, `\det`, `\dim` and other such commands that are built into TEX.

- Functions like Hom, Aut, End, Tor, Sym, Alt should likewise be printed in roman type in formulae, as in $\mathrm{Aut}(\mathrm{Alt}(n)) \cong \mathrm{Sym}(n)$. Since TEX does not have built in commands for most of these, either use `\mathrm` each time or make yourself definitions such as `\newcommand{\Aut}{\mathrm{Aut}}`.

3.4 Style

The following is a selection of the advice that was written in about 1980 by the London Mathematical Society editors of the time. Much of it is still to be found on the LMS publications website.

- Mathematics should be written in grammatically correct language and should be properly punctuated, even in sentences that include formulae or displayed material.

- Words such as 'assume', 'suppose', 'show' and 'imply' should usually be followed by 'that'.

- Where 'if' introduces a conditional clause, it should usually be followed by 'then', as in 'if $x = 3$, then $y = 4$'. This is essential where omitting 'then' would result in juxtaposition of formulae, as in 'if $x = 3$, $y = 4$' (where the comma could be misread as 'and').

- Adjectives should not usually be used as nouns. For example, 'a unitary' should not be used when what is meant is 'a unitary operator'. It is particularly obnoxious to use proper names such as Frobenius, Sylow as ordinary nouns by taking their eponymous adjectival uses and dropping the noun as in '(the) Frobenius' where what is meant is 'the Frobenius automorphism' or 'a Sylow' where what is meant is 'a Sylow subgroup'. Perhaps even worse is to use a proper name as an adjective without its noun, as in 'we see that (s_n) is Cauchy and therefore converges'.

- Sentences should begin with words rather than symbols; it is particularly bad to start a sentence with a lower case symbol. This holds especially when the preceding sentence ends with a formula or symbol, so that two formulae could apparently coalesce to make one which, at first reading, is nonsense. One can always rephrase sentences using constructions such as 'It follows that', 'Since', 'Now', 'The function' to avoid having a sentences that starts with a symbol or a formula and to avoid formulae being separated only by a comma.

- Formulae should never be separated by punctuation marks only, except in lists. Try to avoid having quotation marks, reference citations or footnote symbols adjacent to formulae. Do not use abbreviations such as 'e.g.', 'i.e.', 's.t.' adjacent to formulae.

- Never use 'apostrophe s' for plurals. One writes 'the 1960s', 'the variables x_i' (never 'the x_is' or 'the x_i's'). For the possessive with symbols use 'of', as in 'the range of f is A' (never 'f's range is A').

- Use abbreviation sparingly. Mathematical writing is already very concentrated.

- Avoid the use of symbols such as $=$, $<$ as abbreviations in text. The scope of such binary relation symbols should be made clear: usages such as 'The number of prime divisors of $30 = 3$' are unnecessarily disturbing.
- Never use symbols such as $\Rightarrow$, $\forall$, $\exists$ as lazy abbreviations in your text. The abbreviation 'iff' is best avoided in print. The full form 'if and only if' is easier to read (and looks less like a misprint).

Enough. Enjoy making professional mathematical text. May your essays, reports, dissertations, articles give pleasure to you and to your readers.

* * * * *

References

[1] T. W. Chaundy, P. R. Barrett, and Charles Batey *The printing of mathematics*, Oxford University Press, 1954.

[2] Donald Knuth, *The TEXbook*, Addison-Wesley, Reading MA, 1984.

[3] Helmut Kopka and Patrick W. Daly, *A Guide to LATEX* (Fourth edition), Addison-Wesley, 2003.

[4] Leslie Lamport, *LATEX—A Document Preparation System*, Addison-Wesley, Reading MA, 1985.

[5] Michael Spivak, *The Joy of TEX*, Amer. Math. Soc., Providence RI, 1986.

ΠMN: Mathematical Institute and Queen's

Version of MT 2016, revised 11.x.2016