

1 Writing documents

L^AT_EX documents are all *plaintext* files. This means printable characters only (in whatever writing system is native to your language and culture), no hidden internal binary gubbins like fonts or formatting (except for spaces and linebreaks). If you haven't seen a plaintext file before, it looks like this:

```
This means printable characters only (in whatever
writing system is native to your language and culture),
no hidden internal binary gubbins like fonts or
formatting (except for spaces and linebreaks).
```

By comparison, wordprocessor files saying the same thing often actually look something like this inside:

```
@A@[@O@B@@@@@h@@PñÿB@h@@@M@D@e@f@a@u@l@t@ @S@t@y@l
@e@@@*$A3@B*@OJC@QJC@CJX@mHIXsHIXKHA@PJD@nHDHtHDHJE@
aJX@_H9D@@@@@@@@@@@@@@@@@@@@@@@@@F@p_A@BAF@@@@G@H@e@z@d@
i@n@g@@@@M@O@Sf@A@X@OJF@QJF@CJ\@PJD@JE@aJ\@.@BPA@BA.@
@@I@T@e@x@t@ @B@o@d@k@j@mHIXs@X@OJF@aJX@_H9D@@@@
```

The big advantage of plaintext is not just that it's readable; it's that the files can be copied, downloaded, or uploaded to any computer system running L^AT_EX and they will typeset exactly the same. Because they are plain text they cannot corrupt your system, and they cannot be used for hiding virus infections in the

way that **binary** (coded non-plaintext) files can be. Everything you can see is in the file and everything in the file is there for you to see: there is nothing hidden or secret and there are no manufacturers' proprietary 'gotchas' like suddenly going out of date with a new version or imposing selective [Digital Restrictions Management \(DRM\)](#), leaving you unable to open your files.

Exercise 1 – Plaintext and wordprocessor files

1. Open your favourite wordprocessor (eg *Libre Office*, *Microsoft Word*, *Apple Pages*, *Google Docs*, etc);
2. Create a new, completely empty document (no template);
3. Type the single word **LaTeX**;
4. Save the file (call it **LaTeX-test** or something obvious) in an obvious folder that you will remember (eg **Home**, **Documents**, **Desktop**, **My Documents**, **~/**, or similar);
5. Close and quit the wordprocessor completely;
6. Open your text editor (eg *Emacs*, *Linux vi*, *Microsoft Notepad*, *Apple TextEdit*, *VS Code*, *Sublime*, etc);
7. Open the document you just saved;
8. See if you can find the word **LaTeX** that you typed in it;
9. If it's not visible, that's because wordprocessors don't use plaintext.

So, you may ask, if \LaTeX files are all plaintext, how does \LaTeX know how to format them? The answer is that it uses **markup**: a system of labels which identifies what's what in your document. \LaTeX and its packages recognise the labels and know how to format them, so you don't usually need to add formatting by hand unless you want to do something very special or invent something out of the ordinary.

Wordprocessors use markup too (*Libre Office* and *Microsoft Word* actually use **XML** internally nowadays) but it's extraordinarily complex, and encoded to prevent casual inspection. It's then

packaged up (often into a zip file) with all the stylesheets, images, and other bits and pieces. L^AT_EX prefers to keep everything out in the open, making it very obvious how you are constructing your documents, so you can come back to them in the future and not have to worry about what you did.

1.1 Markup

In a L^AT_EX document, you type your text along with *markup* to identify the important bits by name, for example 'title', 'author', 'chapter', 'section', 'figure', etc. L^AT_EX does all the typesetting for you automatically, using the markup to apply the formatting rules (styles) you tell it to use.

Exercise 2 – Clarity check

See if you can remember, work out, or guess what the L^AT_EX markup for these document elements is:

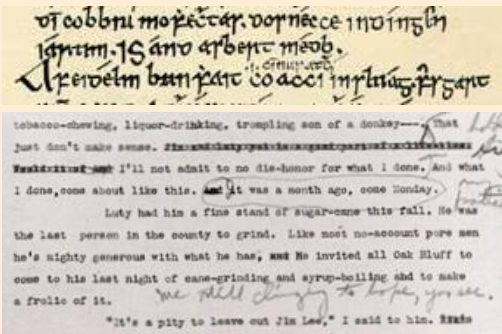
1. a footnote;
2. a section heading;
3. the Table of Contents;
4. an item in a list;
5. a URL (web link).

In the web version of this exercise you can click on the button after each one to see the answer at the place in this book where it is explained.

In the panel 'Markup' on p.4 you can see some examples of markup over the years. In the manuscript era (before printing) scribes often added extra information to what they were copying. In the days of the typewriter, publishers would add handwritten markup to the author's typescript so the printer would know what to do with it. When computers started being used for text

Markup

The term ‘markup’ editorial notes or proofreader’s corrections, but the practice goes back to the beginning of writing (a very long time). It now means information added to a document for identification or formatting.



Anon, ‘Táin bó Cúailnge’ 1100

Rawlings, ‘Varmints’ 1932

```
.h1 Interest Rates
:h1. Interest Rates
@Heading[Interest Rates]
\section{Interest Rates}
```

Runoff, Script 1960s
GML/DCF c.1975
Scribe c.1976
L^AT_EX 1984

```
<SEC><TTL>Interest Rates</TTL>...
```

SGML (American Association of Publishers (AAP) DTD) 1985

```
<div1><head>Interest Rates</head>...
```

SGML (TEI) 1989

```
<H1>Interest Rates</H1>
```

SGML (HTML) 1989

```
<sect1><title>Interest Rates</title>
```

XML (DocBook) 1995

```
<h1>Interest Rates</h1>
```

XML (HTML5) 2005

See the history of computer markup in the names (h1 and H1, section, sec, sect1, etc).

processing, systems tended to follow the established conventions: the similarity between the computer forms is striking, and not coincidental.

L^AT_EX markup is all in (American) English, with a few abbreviations for long words to minimise typing. Most people use an editor with a menu or toolbar button which knows about L^AT_EX markup, so actually typing it by hand is rare.

You do not need to format any of your text *in your editor*, because L^AT_EX does the formatting all by itself when it typesets. You can of course regularise or neaten its appearance *in your editor* for your own ease of editing (for example, keeping each item in a list on a separate line), but this is not required.

You will often hear L^AT_EX markup referred to as ‘commands’ or sometimes ‘control sequences’ (the proper T_EXnical term for them). For all practical purposes these terms all mean the same thing.

1.2 Choosing your L^AT_EX processor

Before you go any further there is one configuration you SHOULD check. As you may have seen in the list [on page xxiii](#), there isn’t just one flavour of L^AT_EX.

Plain L^AT_EX: For many years, there was only L^AT_EX, which (like T_EX) produced a [.dvi](#) (Device-Independent) file, which had to be converted to Postscript in an additional step;

pdfL^AT_EX: In the 1990s, Hàn Thế Thành developed PDFT_EX, which (along with *pdfL^AT_EX*) produced [PDF](#) directly, as well as adding the possibility of benefits like microtypographic adjustments;

ConT_EXt: Around 1996, Hagen Hans and Ton Otten released ConT_EXt, using L^AT_EX-like controls for more advanced classes of work, especially typographical manipulation for education, with most features built in, rather than via a package system;

X₃L^AT_EX: More recently, Jonathan Kew developed X₃T_EX, which not only recognises [Unicode Transformation Format — 8-bit](#) (UTF-8) characters directly, but can also use your system’s

natively-installed [TrueType Fonts](#) (TTFs) and [OpenType Fonts](#) (OTFs) as well as those which come with \LaTeX ;

Lua \LaTeX : Even more recently, Hans Hagen, Hartmut Henkel, Taco Hoekwater and Luigi Scarso have developed $\text{Lua}\text{\LaTeX}$, an extended version of $\text{PDF}\text{\LaTeX}$ using *Lua* as an embedded scripting language, so you can write scripts inside your \LaTeX document to generate your content programmatically. This also has some of the features of $\text{X}\text{\LaTeX}$ such as support for TTF and OTF fonts.

In this book I recommend that you use $\text{X}\text{\LaTeX}$ unless you have a compelling reason not to. In my view the ability to handle natively-installed system fonts as well as UTF-8 characters while retaining the flexibility of the \LaTeX package system sets them well above the other processors (I'm not going to be dealing with Lua's scripting or $\text{ConT}\text{\LaTeX}$'s typographical abilities here).

Exercise 3 – Set your \LaTeX and $\text{BibT}\text{\LaTeX}$ processors

1. Open your \LaTeX editor and set the processor to be $\text{X}\text{\LaTeX}$. In most cases this is a configuration setting in the editor (see Figure 1.1 on page 8 for examples).
2. Repeat the process to set your $\text{BibT}\text{\LaTeX}$ processor to *biber*.

Note that in *Emacs* you edit the `~/ .emacs` configuration file and add these lines:

```
(setq latex-run-command "xelatex")  
(setq bibtex-dialect 'biblatex)  
(setq tex-bibtex-command "biber")
```

Save the file and type `M-x eval-buffer RET` to implement it for this session (in future sessions it will be automatic).

There are still a few reasons some users stay with $\text{pdf}\text{\LaTeX}$ or even the original \LaTeX . These include:

- ☐ a few packages (now a very small number) which positively require a processor which creates an old-style `.dvi` file;
- ☐ some specific packages still rely on raw Postscript features which need DVI-to-Postscript conversion first, before the PS output can be converted to PDF; and some DVI-to-Postscript converters cannot handle the (eXtended DVi (XDV)) format produced by $\text{X}\text{\LaTeX}$;
- ☐ there are some other toolchains which depend on DVI files;
- ☐ some older editors do not yet make it possible to select $\text{X}\text{\LaTeX}$ or $\text{Lua}\text{\LaTeX}$ as the processor¹

There is a separate but related setting to choose the bibliographic formatter (old-style $\text{Bib}\text{\TeX}$ `.bst` files or the more recent `biblatex` package) and which bibliographic processor to use (*bibtex* or *biber*). If your documents don't use bibliographic references, this will not be a concern for you.

The relationship is that the *biblatex* package and the *biber* program, like $\text{X}\text{\LaTeX}$, deal natively with UTF-8 characters and are actively supported, whereas the `.bst` files and the *bibtex* processor have known problems with multibyte (accented and non-Latin) characters, and are no longer being developed; this makes the reference and citation of works in many languages difficult, if not impossible with `.bst` files and the *bibtex* processor. We will be dealing with this choice in more detail in section 5.3.2.1 on page 142.

1.3 Picking an Editor

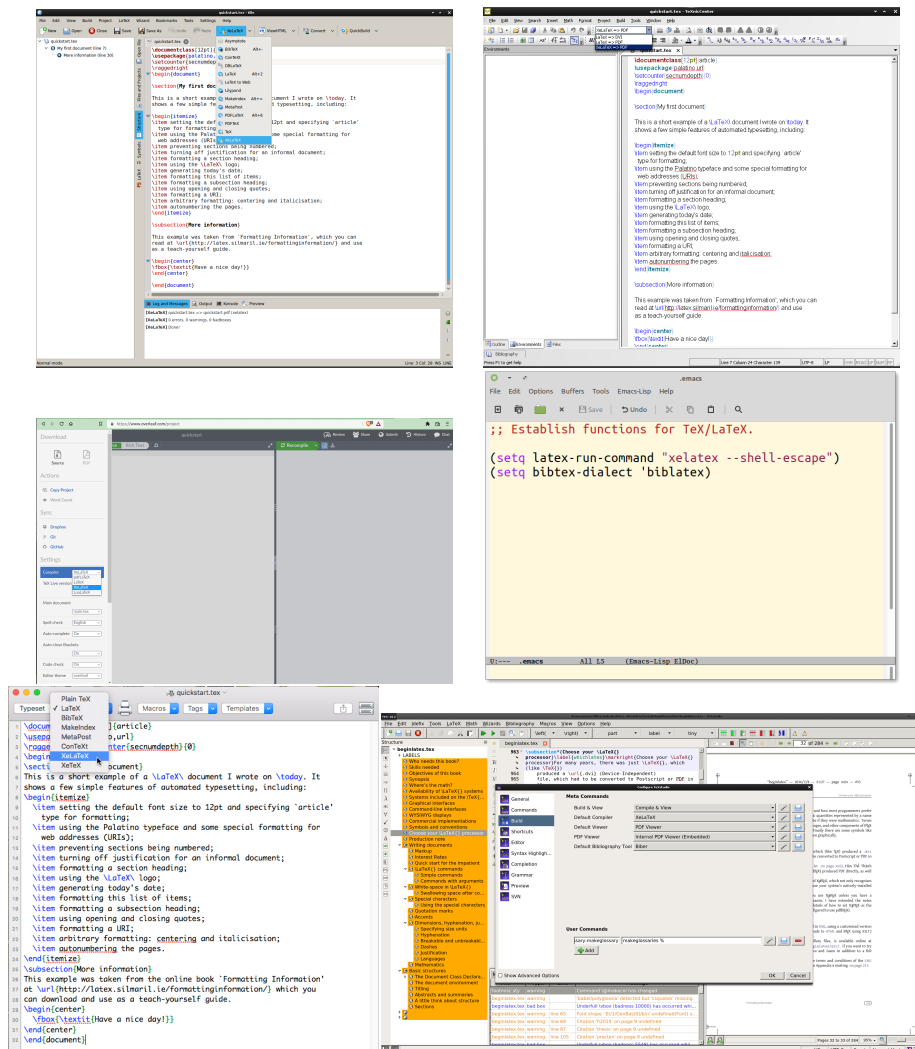
One of the best features of $\text{T}\text{\E}\text{\X}$ -based systems like $\text{L}\text{\A}\text{\T}\text{\E}\text{\X}$ is that they don't force you to use any particular editor or viewer: you can pick one that you're comfortable with.

One of the worst features (for a beginner) is not understanding this: many new users have never come across this kind of flexibility in software before, and may be unfamiliar with the

¹ Some which are straightforward are shown in below.

CHAPTER 1. WRITING DOCUMENTS

Figure 1.1 – Some \LaTeX editors being configured to use \XeLaTeX



idea that you don't have to do what a vendor says: you can pick and choose.

Nevertheless, I'm solving this by edict for beginners here: *unless you already have a favourite editor or viewer suitable for \LaTeX* , just use one of those shown below. If you're an experienced computer user, see the comments at the end of this section.

Table 1.1 – Default editors with different distributions of T_EX

System	Package	Engine	Editor	Viewer
Microsoft Windows	<i>ProT_EXt</i>	<i>MiK_T_EX</i>	<i>T_EXStudio</i>	Adobe Acrobat Reader
Apple Mac OS X	<i>MacT_EX</i>	<i>T_EX Live</i>	<i>T_EXshop</i>	<i>Preview</i>
Unix and GNU/Linux	<i>T_EX Live</i>	<i>T_EX Live</i>	<i>Kile</i> , <i>T_EXStudio</i> , or <i>Emacs</i>	<i>Okular</i> , <i>Evince</i> , <i>qpdfview</i> , etc

So for beginners, in the case of *ProT_EXt* (Windows) and *MacT_EX* (Macs) ‘the editor’ is the one that comes with the distribution (*T_EXStudio* and *T_EXshop* respectively, see below). Unix and GNU/Linux users need to choose and install an editor separately.

However, there are many other editors, if you want to try them out and pick one you are comfortable with. Below are a few that I have either tried or had good reports of. There is an excellent page all about L^AT_EX editors maintained by Fabrizio Musacchio at www.fabriziomusacchio.com/blog/2021-08-19-LaTeX_editors/.

Editor	Mac OS X	Linux	Windows
<i>Emacs</i>	✓	✓	✓
<i>T_EXMaker</i>	✓	✓	✓
<i>L_AX</i>	✓	✓	✓
<i>texifier</i>	✓	✗	✗
<i>Kile</i>	✗	✓	✓

Have a look also at Barbara Beeton’s slide on *Features of a good editor*, [slide 4](#) from her presentation at TUG 2017 (Beeton 2017).

Choose your editor carefully

Experienced computer users should also read these very sensible and relevant comments from Michael Sperberg-McQueen:

Twenty years ago, when my computer center required us all to start moving our data from VM/CMS to Unix, and we were all trying to decide which of the available text editors to invest time in learning, I formulated one of the few general principles of computer usage that has ever been revealed to me:

1. If you are a serious computer user, you will spend more time in your text editor than in any other single application. Choosing well is a good idea.
2. It takes a long time to learn an editor well. You will not learn 2000 editing programs between now and the day you stop using computers at all — you may only learn two or three. Choosing wisely is important to avoid wasting time.
3. The first question to ask, when considering any candidate editor, is 'when I write a macro for this editor, what is the name of the language I'm writing in?'

There are three classes of answer:

- (a) Answers of the form 'you don't, there are no macros in — '. These won't normally occur, since such editors won't ever actually be candidates for serious day-in day-out use.
- (b) Answers of the form 'It's a specialized macro lang-[...]'. You needn't listen to the end of these; the editor is not worth investing your time to learn.
- (c) Answers of the form 'X', where 'X' is the name of a 'real' programming language, with variables and functions and flow control and all of those things.

It was on this basis that I chose *Emacs* over *vi*. (A decision, I may say, that I have never regretted.)

The importance of macros was pretty clear to me because over my years of using VM, I had written dozens of macros for *XEdit*, the system editor, and the fact that I could write them in a well designed programming language (*Rexx*) was really important.

The funny thing is that having chosen *Emacs* over *vi* because it has a real programming language for macros, I then used *Emacs* for ten years or more before I ever actually used *Lisp* to write any macros. I decided that this was because all the macros I needed seemed to have been written by someone else already. (Sperberg-McQueen 2016)



1.4 Quick start for the impatient

If you already know all about editors and plaintext files and markup and how to run programs, and you know that \LaTeX is already fully installed (including an editor that you know how to use), you'd probably like to type something in and see \LaTeX do its job. This is probably specially true if you've just signed up with *Overleaf* or another of the online \LaTeX services.

Exercise 4 – Quick start

1. Open your \LaTeX editor (or log into *Overleaf*) and create a new, empty document (delete any auto-template material if present);
2. Make sure your editor is set up to use \XeLaTeX ;
3. Copy and paste the text from Figure 1.2 on page 13. Make sure you get all of it (apart from the caption), and don't change anything yet.
4. Save the document as `quickstart.tex` in your *Documents* folder or wherever you normally keep your documents (*Overleaf* already asked for the name in step above);
5. Click on the `Run`, `Build`, `Typeset`, `Recompile` or `\TeX` `File` menu; or on the toolbar icon for your editor, as indicated by the black arrow in the illustrations in Figure 1.3 on page 14.
6. Some editors open the preview automatically when a new document is typeset. If this does not open automatically for you, click on the `View` or `Preview` toolbar item (usually next to the `Typeset` icon).
7. If you have a printer installed, you can click on the Print toolbar icon in your viewer (or open the PDF in any suitable viewer).

If you don't know this stuff yet, then by all means do this section now, but treat it as part of the learning experience. Then you can read in [section 1.5](#) more about how \LaTeX works, and come back to this section to check it.

You can check to see if \LaTeX is already installed on your computer by opening a command window and typing `xelatex` and pressing the  or  key.

If you're using an online system like *Overleaf* instead, make sure you have created an account.

If you encounter any errors, it means you *do* need to read the rest of this chapter after all! There is a list of common error messages in [section B.3.1](#) on [page 291](#). You need to fix the errors in your document and click on the Typeset button again (or whatever your editor uses).

1.5 \LaTeX commands

Now that you have seen \LaTeX working, let's have a closer look at what it's actually doing.

1.5.1 Commands used in the example

\LaTeX commands all begin with a *backslash* (`\`) and normally consist of lowercase letters only (there are a few which have uppercase letters). Going through the [quickstart.tex](#) document in [Figure 1.2](#) on the facing page, we can see the following commands being used:

`\documentclass` specifies the class of document ([article](#)) and the size of type for the text (*12pt*);

`\usepackage` tells \LaTeX to use the named packages (plugins), here [fontspec](#) (to use all fonts) and [url](#) (provides a way to format URIs);

`\setmainfont` lets you give the name of the typeface to use as the main font; XCharter is an extended version of Matthew Carter's 1987 Charter typeface, based on Pierre-Simon Fournier's characters from the 18th century (see [section 6.2](#) on [page 168](#) for how to find and specify others);

Figure 1.2 – Quick-start example document text

```
\documentclass[12pt]{article}
\usepackage{fontspec,url}
\setmainfont{XCharter}
\setcounter{secnumdepth}{0}
\begin{document}

\section{My first document}

This is a short example of a \LaTeX\ document I wrote
on \today. It shows a few simple features of automated
typesetting, including:

\begin{itemize}
  \item setting the font size to 12pt for the 'article'
    class;
  \item using any font, not just the default;
  \item using the special formatting for URLs (web
    addresses);
  \item using the XCharter typeface;
  \item preventing sections from being numbered;
  \item formatting a section heading;
  \item using the \LaTeX\ logo;
  \item generating 'todays date;
  \item formatting this list of items;
  \item formatting a subsection heading;
  \item using opening and closing quotes;
  \item formatting a URI;
  \item boxing, centering, and italicisation;
  \item autonumbering the pages.
\end{itemize}

\subsection{More information}

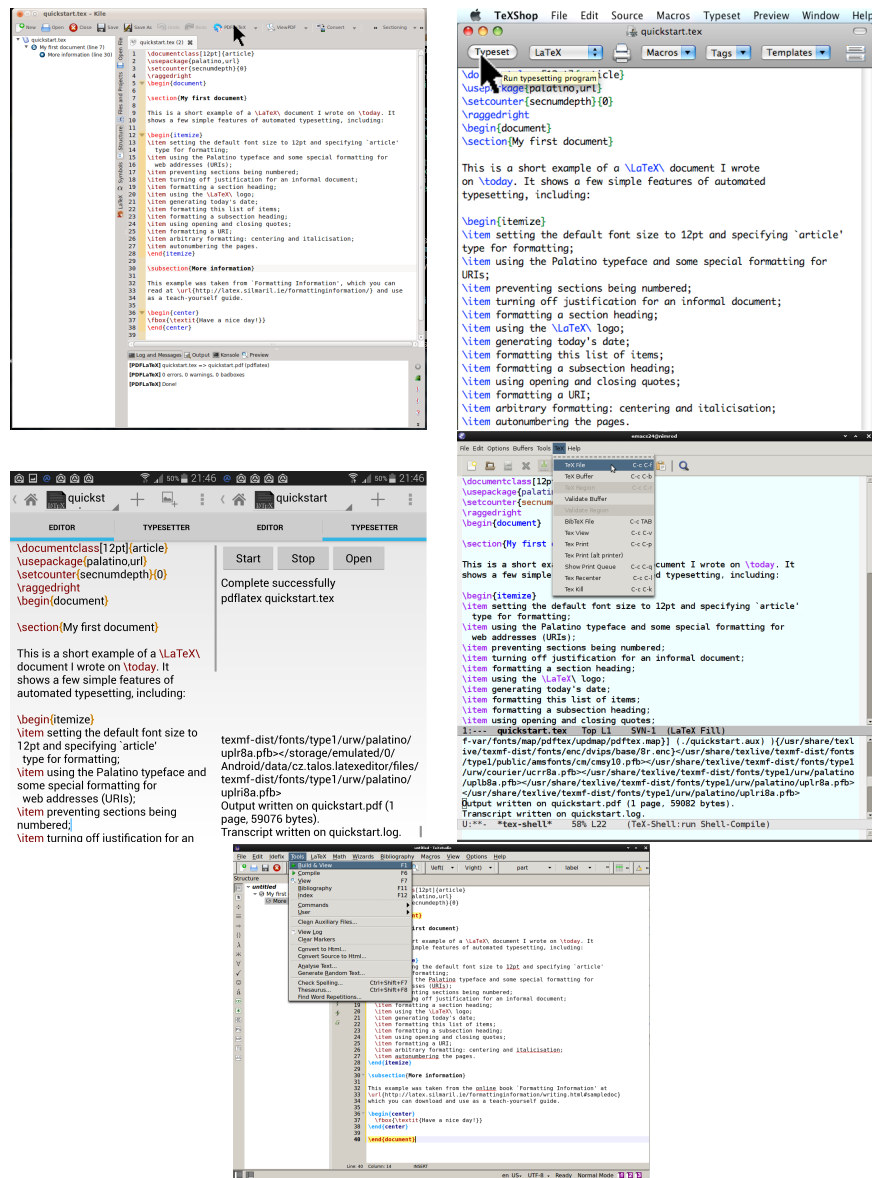
This example was taken from the book 'Formatting
Information', which you can read online at
\url{http://latex.silmaril.ie/formattinginformation/}
and use as a teach-yourself guide.

\begin{center}
  \fbox{\textit{Have a nice day!}}
\end{center}

\end{document}
```

CHAPTER 1. WRITING DOCUMENTS

Figure 1.3 – What to click on to typeset a document



If you are using an editor not shown here, look for a menu or toolbar button marked `Typeset` or `XYLaTeX` or `pdfLaTeX` or `Build` or `Compile`.

`\setcounter` sets the value of a *counter*, here `secnumdepth`, which is the depth to which sections are autonumbered. Setting it to zero prevents sections being numbered at all.

`\begin` marks the beginning of an *environment*, here `document`, which contains the whole text of the document. It's terminated by a matching `\end{document}` at the bottom of the file.

Everything up to this point is called the Preamble, and this is where you set up how the document looks

The Preamble

Settings which you want to affect a whole document go at the start of your L^AT_EX file, immediately after the `\documentclass` line and before the `\begin{document}` line. In the Quick Start example, we wanted to use some packages, set the font, and set a counter:

```
\documentclass[12pt]{article}
\usepackage{fontspec,url}
\setmainfont{XCharter}
\setcounter{secnumdepth}{0}
\begin{document}
...
\end{document}
```

This position, between the Document Class Declaration and the beginning of the `document` environment, is called the **Preamble**.

`\section` identifies a section heading;

`\LaTeX` typesets the L^AT_EX logo;

`\today` typesets today's date;

`\begin` marks the beginning of an *itemize* environment, which is an itemized (bulleted) list. It's terminated by a matching `\end{itemize}` command at the end of the list;

`\item` marks the start of a new list item;

`\end` ends an environment, here the itemized list;

`\subsection` identifies a subsection heading;

`\url` typesets a URL in a monospace font, allowing line-breaks *only* at slashes or dots;

`\begin` begins another environment, *center*, which centres the material within it;

`\fbox` typesets the material in curly braces in a framed box;

`\textit` typesets the material in curly braces in italic type;

`\end` ends the *center* environment;

`\end` ends the *document* environment, and thereby terminates the whole document. Anything after this line gets saved but ignored.

1.5.2 Simple commands

Simple commands are just the command name on its own, after the backslash, for example:

```
\today
```

This example is an instruction to L^AT_EX to insert the current date at that point. You would usually use this in a draft article or report somewhere close to the beginning, so that you have a record of when it was last typeset. You don't have to do anything else, although there are packages like [datetime2](#) for changing the format of the date.

Other simple commands include `\tableofcontents` which inserts a Table of Contents, and `\LaTeX`, which creates the L^AT_EX logo.

Most commands, however, need some information to work on, called an ***argument***, which you put in curly braces after the command name.

Backslashes and forward slashes

Do not confuse the backslash (\) with the forward slash (/). They are two different characters.

- ☐ The forward slash is used in Unix-based systems (including Mac OSX and GNU/Linux) to separate directory names and file names;
- ☐ The forward slash is also used on the Web to separate the directory names and file names in a URI;
- ☐ The backslash is used to separate directory names and file names *only* in the Microsoft Windows file system;
- ☐ The backslash is used to signal the start of a L^AT_EX command in *all* systems.

When you refer to directory and file names in L^AT_EX (eg image files), you MUST use the forward slash, even in Microsoft Windows.

1.5.3 Commands with arguments

Most L^AT_EX commands are followed by one or more **arguments**, meaning information to be acted upon. Here are two examples, a chapter title (see section 2.6 on page 56) and a cross-reference label (see section 5.3.1 on page 138):

```
\chapter{Poetic Form}\label{pform}  
The shape of poetry when written or printed  
distinguishes it from prose.
```

When an argument is needed (as here) it always goes in {curly braces} like those shown above.

Exercise 5 – Braces or not

1. Which of the commands listed in section 1.5.1 on page 12 do *NOT* need any argument in curly braces or square brackets after them?
2. Which of the commands listed in section 1.5.1 on page 12 allow an *optional* argument in square brackets?

Warning

Be careful not to confuse the curly braces on your keyboard with (round) parentheses, [square] brackets, <less-than or greater-than> signs, <typographic angled brackets>, or «guillemets» (French quotes, not guillemots; those are sea-birds). They are all quite different and they mean different things.

(Embarrassingly, the \LaTeX command for guillemets was mis-cloned as 'guillemot' when it was created, apparently from an earlier error by Adobe (Beeton 2005) and no-one seems to have the nerve to change it. Albatross!)

1.6 White-space in \LaTeX

\LaTeX does its own spacing and alignment based on the layout defined in the templates (classes) and stylesheets (packages) you give it, plus the commands that you type in.

You *cannot* force white-space into your output by typing it into your editor, as you do with a wordprocessor. If you need extra white-space in your document, use a horizontal or vertical spacing command, or change the spacing parameters or the document or the environment involved (see section 6.1.1 on page 161).

There are three simple rules for spacing in \LaTeX (see the Note on p.19) — the first two say that, essentially, multiple spaces

and blank lines are ignored in the typesetting. That means you are free to use them *in your editor* for optical ease and personal convenience when editing.

Three rules for spacing in L^AT_EX documents

1. All consecutive spaces and TAB characters are treated as if they were a *single* space during typesetting;
2. All consecutive newlines (linebreaks) are treated as if they were just two newlines (a paragraph break);
3. Any white-space after a command ending in a letter is discarded when there is no argument present.

The third rule means that any simple command which ends in a letter and has no argument **MUST** be followed by ***white-space*** or an empty pair of ***curly braces*** before the text which follows it, to keep it separate. The following examples will all produce identical output.

```
\tableofcontents Thanks to Aunt Mabel for all her  
help with this book.
```

```
\tableofcontents      Thanks to Aunt Mabel for  
all her help with this book.
```

```
\tableofcontents  
Thanks to Aunt Mabel for her help with this book.
```

```
\tableofcontents{}Thanks to Aunt Mabel for all her  
help with this book.
```

```
\tableofcontents
```

Thanks to Aunt Mabel for her help with this book.

The additional spacing or braces is not needed if

- ☐ the command name ends with a non-letter, or;
- ☐ it is directly followed by another command, or;
- ☐ it occurs immediately before a closing curly-brace, or;
- ☐ it is followed by a double newline (paragraph break).

If you forget the white-space, like this:

```
\tableofcontentsThanks to Aunt Mabel for all her  
help with this book.
```

then \LaTeX will treat everything up to the next non-letter as a command, so it will end up trying to make sense of a ‘command’ apparently called `\tableofcontentsThanks`. There’s no such command, of course, so \LaTeX will complain by displaying an error message about an ‘undefined control sequence’ (see section B.3.3.2 on page 293).

With commands that take arguments you do *not* need to use extra white-space or curly braces after the command, because the curly braces will keep the command separate from any normal text which comes after it. The following example is therefore *exactly* equivalent to the one we just saw in section 1.5.3 on page 17, and will typeset identically despite the absence of spaces between commands.

```
\chapter{Poetic Form}\label{pform}The shape of poetry  
when written or printed distinguishes it from prose.
```

By the same token, the following example is therefore also *exactly* equivalent (although rather unusual!):

```
\chapter      {Poetic Form}          \label  
              {pform} The shape of  
poetry when written                or  
              printed distinguishes it from prose.
```

That is, it will get typeset exactly the same. Here’s what you would normally type:

```
\chapter{Poetic Form}
```

```
\label{pform}
```

The shape of poetry when written or printed
distinguishes it from prose.

Exercise 6 – To space or not to space

Which of the following commands needs to be followed by white-space (or another command, or an empty pair of braces)?

1. `\begin{document};`
2. `\LaTeX;`
3. `\chapter{Introduction};`
4. `\today;`
5. `\textit{Star Trek}.`

Why would you want all that spacing in the examples (or none)? The answer is usually never, although extra blank lines *in your editor* between chapters or sections make editing easier. But a lot of L^AT_EX is not typed by hand: it is generated by computer programs from other systems such as web scripts, XML documents, databases, filestores, mashup engines and other processes, and it makes life easier for the programmers if they don't have to worry about the odd space or two creeping in here and there in normal text: it simply won't have any effect. It also means that if you want to use extra spacing to make your text easier to edit, you don't have to worry about unwanted linebreaks coming out between sections or paragraphs, tabbing in tables, or indentation in list items.

Table 1.2 – Special characters in L^AT_EX

Key	Special meaning	<i>If you need the actual character itself, type it like this:</i>	Example
<code>\</code>	The command character	<code>\textbackslash(\)</code>	
<code>\$</code>	Math typesetting delimiter (obsolescent: <code>***</code>)	<code>\\$</code>	<code>\\$37.46</code>
<code>%</code>	The comment character	<code>\%</code>	<code>42\%</code>
<code>^</code>	Math superscript character	<code>\^</code>	<code>\^{ }</code>
<code>&</code>	Tabular column separator	<code>\&</code>	<code>AT\&T</code>
<code>_</code>	Math subscript character	<code>_</code>	<code>A_B</code>
<code>~</code>	Non-breaking space	<code>\~</code>	<code>\~{ }</code>
<code>#</code>	Macro parameter symbol	<code>\#</code>	<code>\#42</code>
<code>{</code>	Argument start delimiter	<code>\{</code>	<code>\{ \\$</code>
<code>}</code>	Argument end delimiter	<code>\}</code>	<code>\} \\$</code>

1.7 Special characters

There are ten keyboard characters which have special meanings to L^AT_EX, and *cannot* be used on their own except for the purposes shown in below.

These characters were deliberately chosen, either because they are rare in normal text, or (in the case of `$`, `#`, `&`, and `%`) they already had an established special meaning on computers as **metacharacters** (characters standing as symbols for something else) when T_EX was written.

We saw at the start of this section how to use the backslash to begin a command, and how to use curly braces to delimit an argument, and we are not covering the mathematical uses in this book. The only special characters remaining from the list in above are therefore:

- `%` The **comment character** makes L^AT_EX ignore the remainder of the line in your document, so you can see it in your editor, but it will never get typeset. For example:

```
Today's price per kilo is €22.70
% get Mike to update this daily
```

As with all comments in documents, don't forget to remove them before sending the original document source to someone else!

```
and must never be allowed authority to create
a charge on the department again.
% and fire those idiots down in Finance!
```

- ~ The ***tilde*** in \LaTeX prints as a normal space, but prevents a linebreak ever occurring at that point. It's often used between a person's initials and their surname, such as `P.~Flynn`, in case it might fall at the end of a line where a linebreak would make it harder to read.
- & The ***ampersand*** is used in tabular setting (rows and columns) to separate the cell values within each row. We'll see how this works in [section 4.2 on page 95](#).
- # The ***hash mark*** or ***octothorpe*** is the American 'pound' [weight] or 'number' sign.

For the pound (sterling) currency sign use the `\textsterling` command if there is no £ on your keyboard.²

While we're on the subject of money, not every font has a Euro character (€), especially those designed before the currency came into being. The default € sign in many fonts is a crummy design based on the letter C instead of a rounded E. The official (sans-serif) Euro sign € is in the [marvosym](#) package and is done with the `\EUR` command. A slightly unusual but more interesting serif Euro sign € is in the [textcomp](#) package using the `\texteuro` command with the the TS1 font encoding (see

² The £ sign is now nearly obsolete except in the [United Kingdom \(UK\)](#) and some of its former colonial dependencies; in Egypt, Sudan, and Syria; and some other countries for historical purposes. It should not be confused with ₤, used in countries using the Lira.

section 6.2.3 on page 187 for details of switching typefaces and settings in mid-text).

Ordinal superscripts

Don't use them in normal English text. Superscripted ordinals like 21st are a historical relic of Victorian and earlier typography. Unless done with skill by a typographer, they are usually ugly and unnecessary, and are almost never used in modern professional typesetting. They were re-introduced by Microsoft *Word* apparently because some American corporations liked their wordprocessing to look like what they fondly imagine print used to be.

If you want to try to mimic low-grade wordprocessing, or if you are trying to make a genuine typographic facsimile of antiquarian typesetting, use the `\textsuperscript` command from the `textcomp` package. Never, *never*, NEVER use math mode superscripting for text superscripts — it's the wrong height, wrong size, and the wrong font.

In non-English languages (European ones, at least) the position is completely different: the ordinal feminine and masculine characters like 2^a ('secunda') or 8^o ('octavo') are the normal method of representation, and their use in Latin-derived terminology in printing and binding remains standard.

1.8 Quotation marks

If you are using X_YL^AT_EX and UTF-8, you can use your operating system's curly 'open-quote' and 'close-quote' characters.

Otherwise, use the ``` key (grave-accent or 'backtick') for the opening quote, and the `'` (apostrophe) key for the closing quote, doubled if you want double quotes; L^AT_EX will automatically typeset these as real quotes:


```
He said, ``I'm just going out.''
```

He said, "I'm just going out."

This ensures you get real left-hand (opening) and right-hand (closing) ‘curly quotes’, usually shaped like tiny ⁶⁶ and ⁹⁹ characters, or as symmetrically-balanced strokes in sans-serif or script typefaces.

Do NOT use the unidirectional typewriter single-quote `'` key (apostrophe) or double-quote `"` key (quotes) for opening quotes: L^AT_EX treats these as closing quotes only.

However, if you are using *Emacs* as your editor, the `"` key is specially programmed in *latex-mode* to think for itself and produce correct ``` and `'` characters automatically.

When typing one quotation inside another, they need to be separated by a small amount of space. The Thin Space character used for this is code `0x2009` but it's not on most keyboards, so L^AT_EX provides the command `\thinspace`. This leaves just enough separation between double and single quotes — leaving no space would make them look like triple-quotes, and using a normal space is too much and could allow an unwanted linebreak:

```
He said, `Her answer was ``never'\thinspace', and she
meant it.
```

He said, 'Her answer was "never"', and she meant it.

1.9 Accents

For accented letters in Latin-alphabet languages, use the accented keys from your keyboard.

- If you can't see any (eg [United States \(US\)](#) and [UK](#) keyboards), they are probably available using the `AltGr` key or some other control key combination: see the documentation which came with your operating system.

Browser fonts and spacing

If you are reading this in a browser, or if you have typeset the document yourself using different fonts, it may not show you real quotes (some older browser fonts are defective) and the `\thinspace` may look too wide. Download the typeset (PDF) version of this document to see the real effect, and switch to a standards-compliant browser.

- ☐ If you really don't have any, you can use your computer's or editor's character map to pick them from a pop-up window (may be under the `Insert` `Special Characters` menu).
- ☐ If you can't find the right combination of keystrokes to generate the characters you want, or you simply can't generate those characters from your keyboard, use below.

For language-specific hyphenation and cultural adaptation (including the correct language headings for all the parts of your document) use the `babel` or `polyglossia` packages (see section 1.10.6 on page 37).

For non-Latin typefaces you will also need the appropriate package for the language and the fonts which actually contain the characters (see section 6.2 on page 168).

Failing all this, if you don't have accented letter keys on your keyboard, or you can't find the codes to type, or if you need additional accents or symbols which are not in any of the keyboard tables, you can use the symbolic notation in below. In fact this can be used to put any accent over any letter (for example, Welsh users can get a `ŵ` with `\^w`), even for combinations which only rarely exist in any language: if you particularly want a `ğ`, for example, you can have one with the command `\~g`.

Before the days of keyboards and screens with their own real accented characters, the symbolic notation in above was the *only* way to get accents, so you may come across a lot of older documents (and users!) using this method all the time: it does have the advantage in portability that the `LATEX` file

If you don't have accented letters on your keyboard

If your keyboard does not have native accent characters, and the control functions don't provide them, see your Operating System manual for details. Here are some common examples:

- On GNU/Linux systems the letter é is usually got with **AltGr** + **e** (and áíóúÁÉÍÓÚ similarly), assuming you set up the keyboard for a European language.

If you know the Unicode hexadecimal code-point (number) of the character you want, press **Ctrl** + **↑** + **U** and release, followed by the number, and press Enter.

Refer to the *xkeycaps* utility for a table of key codes and combinations (install it with your system's package manager or get it from www.jwz.org/xkeycaps/).

- On Apple Mac OS X systems, the letter é is got with **Alt** + **e**.

- Under Microsoft Windows the letter é is got with **Ctrl** + **'** + **e**.

If you know the Microsoft encoding (number) of the character you want, hold down the **Alt** key and type the number on the numeric keypad (*not* the top row of shifted numerals), then release the **Alt** key.

Refer to the *charmap* utility for a table of key codes and combinations.

remains plain ASCII, which will work on all machines everywhere, regardless of their internal encoding, and even with very old T_EX installations.³

³ Remember not everyone is lucky enough to be able to install new software: many users on business and academic networks still use old versions of T_EX because they or their system managers don't know how to update them. Local user groups may be able to provide help and support here.

Table 1.3 – Symbolic notation for Latin-alphabet accents

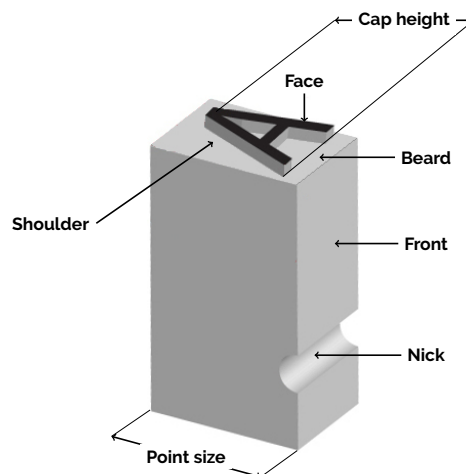
Accent	Example	Characters to type
Acute (fada)	é	<code>\'e</code>
Grave	è	<code>\`e</code>
Circumflex	ê	<code>\^e</code>
Umlaut or diæresis	ë	<code>\"e</code>
Tilde	ñ	<code>\~n</code>
Macron	ō	<code>\=o</code>
Bar-under	ṁ	<code>\b o</code>
Dot-over (séimíú)	ṁ	<code>\.m</code>
Dot-under	ș	<code>\d s</code>
Breve	ū	<code>\u u</code>
Háček (caron)	ň	<code>\v n</code>
Long umlaut	ö	<code>\H o</code>
Tie-after	öö	<code>\t oo</code>
Cedilla	ç, Ç	<code>\c c, \c C</code>
O-E ligature	œ, Œ	<code>\oe, \OE</code>
A-E ligature	æ, Æ	<code>\ae, \AE</code>
A-ring	å, Å	<code>\aa, \AA</code>
O-slash	ø, Ø	<code>\o, \O</code>
Soft-l	ł, Ł	<code>\l, \L</code>
Ess-zet (scharfes-s)	ß	<code>\ss</code>

Irish and Turkish dotless-ı can be done with the special command `\i`, so an í (which is normally typed with `\acute{i}`) may require `\'\i{}` if you need to type it in the long format—remembering that dummy pair of curly braces if there is no punctuation, because of the rule that L^AT_EX control sequences which end in a letter always absorb any following space (see the Note on p.19swallow). So what you see as *Rí Teanraic* (‘King of Tara’) when typeset would have to be `R\'\i\ Tea\.mra\.c` when typed in full (there are not usually any dedicated keyboard keys for the dotless-ı or for aspirated or lenited characters). A similar rule applies to dotless-j and to uppercase Í.

1.10 Dimensions, hyphenation, justification, and breaking

L^AT_EX's internal measurement system is extremely accurate. The underlying T_EX engine conducts all its business in units smaller than the wavelength of visible light, so if you ask for 15mm space, that's what you'll get — within the limitations of your screen or printer, of course. While modern high-resolution displays use pixels smaller than you can easily see, many older screens cannot show dimensions of less than $\frac{1}{96}$ " without resorting to magnification or scaling; and on printers, even at 600dpi, fine oblique lines or curves can still sometimes be seen to stagger the dots.

Figure 1.4 – Some parts of a piece of metal type



At the same time, many dimensions in L^AT_EX's preprogrammed formatting are specially set up to be flexible: so much space, plus or minus certain limits to allow the system to make its own adjustments to accommodate variations like overlong lines, unevenly-sized images, and non-uniform spacing around headings. This is quite different from the 'grid' system used in many other typesetting and DTP systems.

T_EX uses a very sophisticated justification algorithm to achieve a smooth, even texture to normal paragraph text by justifying a

whole paragraph at a time, quite unlike the line-by-line approach used in most wordprocessors and DTP systems.

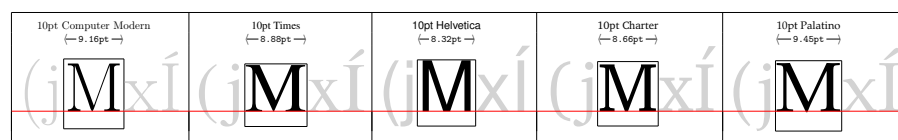
Occasionally, however, you will need to hand-correct an unusual word-break or line-break, and there are facilities for doing this on individual occasions as well as automating it for use throughout a document.

1.10.1 Specifying size units

Most people in the printing and publishing businesses in English-speaking cultures habitually use the traditional printers' *points*, *picas* and *ems* as well as cm and mm when dealing with clients. Many older English-language speakers (and most North Americans) still use inches. In continental European and related cultures, Didot points and Ciceros (Didot picas) are also used professionally, but cm and mm are standard everywhere else: inches are only used now when communicating with North American cultures.

You can specify lengths in L^AT_EX in any of these units, plus some others (see below).

Figure 1.5 – An M of type of different faces boxed at 1em



The red line is the common baseline. Surrounding letters in grey are for illustration of the actual extent of the height and depth of one em of the current type size.

The em can cause beginners some puzzlement because it's based on the 'point size' of the type, which is itself misleading. The point size refers to the depth of the metal body on which foundry type was cast in the days of metal typesetting, *not* the printed height of the letters themselves (see Figure 1.4 on the previous page). Thus the letter-size of 10pt type in one typeface

can be radically different from 10pt type in another (look at above, where the widths are given for 10pt type). An em is the height of the type-body in a specific size, so 1em of 10pt type is 10pt and 1em of 24pt type is 24pt. A special name is given to the 12pt em, a 'pica' em, and a pica has become a fixed measure in its own right. An old name for a 1em space is a 'quad', and L^AT_EX has a command `\quad` for leaving exactly that much horizontal space.

Table 1.4 – Units in L^AT_EX

Unit	Size
Printers' fixed measures	
pt	Anglo-American standard points (72.27 to the inch)
pc	Pica ems (12pt)
bp	Adobe's 'big' points (exactly 72 to the inch)
sp	T _E X's internal 'scaled' points (65,536 to the pt)
dd	Didot (European standard) points (67.54 to the inch)
cc	Ciceros (European pica ems), 12dd)
Printers' relative measures	
em	Ems of the current point size (historically the width of a letter 'M' but see Figure 1.5 on the facing page)
ex	x-height of the current font (height of a letter 'x')
Other measures	
cm	centimeters (2.54 to the inch)
mm	millimeters (25.4 to the inch)
in	inches (obsolete except in UK and parts of North America)

To highlight the differences between typefaces at the same size, above shows five capital Ms in different faces, surrounded by a box exactly 1em of those sizes wide, and showing the actual width of each M when set in 10pt type. Because of the different ways in which typefaces are designed, none of them is exactly 10pt wide.

If you are working with other DTP users, watch out for those who think that Adobe points (bp) are the only ones. The difference between an Adobe big-point and the standard point is only .27pt per inch, but in 10" of text (a full page of A4) that's 2.7pt, which is

nearly 1mm, enough to be clearly visible if you're trying to align one sample with another.

1.10.2 Hyphenation

L^AT_EX hyphenates automatically according to the language you use (see [section 1.10.6](#) on [page 37](#)). To specify different breakpoints for an individual word, you can insert soft-hyphens (discretionary hyphens), done with the `\-` command (backslash-hyphen) wherever you need them, for example:

```
When in Mexico, we visited Popo\ -ca\ -tépetl by  
helicopter.
```

If the words needs to be hyphenated, the best-fit of the points will be used, and the rest ignored.

To specify hyphenation points for *all* occurrences of a word in the document, use the `\hyphenation` command in your Preamble (see the panel 'The Preamble' on [p.15preamble](#)) with one or more words as patterns in its argument, separated by spaces; in this case using the normal hyphen to indicate permitted break-points. This will even let you break 'helicopter' correctly.

```
\hyphenation{helico-pter Popo-ca-tépetl vol-ca-no}
```

If you have frequent hyphenation problems with long, unusual, or technical words, ask an expert about changing the value of `\spaceskip`, which controls the flexibility of the space between words. This is not something you would normally want to do without advice, as it can change the appearance of your document quite significantly.

If you are using a lot of unbreakable text (see the next section and also [section 4.7.1](#) on [page 129](#)) it may also cause justification problems: you can turn justification off with `\raggedright`.

1.10.3 Breakable and unbreakable text

Unbreakable text is the opposite of discretionary hyphenation. To force L^AT_EX to treat a word as unbreakable, use the `\mbox` command:


```
\mbox{pneumonoultramicroscopicsilicovolcanoconiosis}
```

This may have undesirable results, however, if you subsequently change margins or the size of the text: `pneumonoultramicroscopicsilicovolcanoconiosis`, although if you're reading this in a browser, you probably won't see the effect properly: look at the [PDF](#).

Another option, for reoccurring words, is to use the `\hyphenation` command as shown in [section 1.10.2](#) on the [preceding page](#), but give the word[s] with no hyphens at all, which stops them having any break-points.

To tie two words together with an unbreakable space (hard space), use a tilde (~) instead of the space (see the list in [section 1.7](#) on [page 22](#)). This will print as a normal space but \LaTeX will never break the line at that point.

A normal space between words is always a candidate for a place to break the text into lines, and the word-spacing gets evened-out between all the remaining words in the paragraph (not just the line)...with one exception: a full point (period) after a lowercase letter is treated in \LaTeX as the end of a sentence, and it automatically gets a little more space before the next word. You do not (indeed SHOULD NOT) type any extra space yourself between sentences.

However, after abbreviations in mid-sentence like 'Prof.', it's *not* the end of a sentence, so we need a way to tell \LaTeX that this should be a normal space. The command for doing this is the `_` (backslash-space — I have made the space visible here so you can see it, but it's just a normal space). This prevents \LaTeX from adding the extra sentence-space and it also means it becomes a normal breakpoint (otherwise you would use the tilde as described above).

For example, it would look odd to split the author's name Prof. D.E. Knuth over a line-end. It's a good idea to make adding the non-sentence space standard typing practice for things like people's initials followed by their surname, as `Prof._D.E._Knuth` (I've used a visible space character here for emphasis but you just type a normal space).

1.10.4 Dashes

The hyphen (-) is only used for hyphenated compound words like editor-in-chief. L^AT_EX inserts its own hyphens when it needs to break a word at right right-hand margin.

Dashes are different: they're longer and they are used in different places. Check the panel 'If you don't have accented letters on your keyboard' on p. 27 for keystrokes for how to find these characters in your computer's character-map.

Long dash: The long dash — what printers call an 'em rule' like this — is used to separate a short phrase from the surrounding text in a similar way to parentheses. If you're using X_YL^AT_EX, you can just type the long dash on your keyboard.

- ☐ If you can't find the character, type three hyphens typed together, `like---this`: L^AT_EX will recognise this combination and replace it with a real em rule.
- ☐ If you want space either side, bind the first hyphen to the preceding word with a tilde `like~---_this` and use a normal space after the third hyphen (shown as a visible space here, but it's just a normal space). This avoids the line being broken before the dash.

The difference between spaced and unspaced rules is purely æsthetic, but different cultures have different conventions (see below). NEVER use a single hyphen for this purpose.

Short dash: The short dash is used between digits like page ranges (35–47). Printers call this an 'en-rule' and if you're not using X_YL^AT_EX you can get it by typing two hyphens together, as in `35--47`. NEVER use a single hyphen for this purpose either.

Minus sign: If you want a minus sign, use math mode (see section 1.11 on page 38) where you type a normal hyphen as part of a mathematical expression, so it occurs between math delimiters like `\(x=y-z\)` for $x = y - z$. DO NOT use the hyphen for a minus sign outside math mode.

There are other dashes for special purposes in the Unicode repertoire, but they are out of scope for this document.

Em rules vs En rules

In a discussion on the TYPO-L mailing list, Yateendra Joshi observed:

[...] unspaced em dashes are standard in US publishing, whether the dashes occur in pairs enclosing parenthetical matter or come singly before the last part of a sentence. In the UK and Europe, I often see spaced en dashes when they occur in pairs but an unspaced em dash when it occurs singly.

Leila Singleton wrote:

[...] unspaced dashes are the standard for the US publishing industry, as it typically references the MLA Handbook (used by books + journals) to establish stylistic conventions. It's worth mentioning that the Associated Press Stylebook (used for newspapers and sometime magazines) instead calls for spaces. It's my understanding that an en dash in British usage is equivalent to an em dash in American usage, and that it's spaced whether it appears as a single or a pair ...

Christopher Maden wrote:

[I learned] that Jan Tschichold's influential design for Penguin Books included spaced en-dashes instead of em-dashes, and that directive (and a few others) saw wide uptake throughout British typography.

1.10.5 Justification

The default mode for typesetting in L^AT_EX is justified (two parallel margins, with word-spacing adjusted automatically for the best optical fit). In justifying, L^AT_EX will never add space between letters, only between words. The `soul` package can be used if you need letter-spacing ('tracking'), but this is best left to the expert.

There are two commands `\raggedright` and `\raggedleft` which typeset with only one margin aligned. Ragged-right has the text ranged (aligned) on the left, and ragged-left has it aligned on the right. They MUST be used inside a *group* (curly-braces, for example: see the panel 'Grouping' on p.202grouping) to confine their action to a part of your text, otherwise all the rest of the document will be done that way. Put the command in your Preamble if you want the whole document like that. This paragraph is set ragged-right.

These modes also exist as *environments* called *raggedright* and *raggedleft* which are more convenient when applying this formatting to a whole paragraph or more, like this one, set ragged-left.

```
\begin{raggedleft}
These modes also exist as environments
called raggedright and raggedleft which is more
convenient when applying this formatting to a
whole paragraph or more, like this one.
\end{raggedleft}
```

Ragged setting turns off hyphenation and indentation. There is a package `ragged2e` providing the command `\RaggedRight` (note the capitalisation) which retains hyphenation in ragged setting, useful when you have a lot of long words. There's a `\RaggedLeft` and a `\RaggedCenter`, too.

To centre text, which is in effect both ragged-right and ragged-left at the same time, use the `\centering` command inside a *group*, or use the *center* environment.

Be careful when centering headings or other display-size material: it's one of the rare occasions when you may need to add

a premature linebreak or forced newline (the double-backslash `\`) to make the lines break at sensible pauses in the meaning (Flynn 2012). *Never* rely on the automated line-breaking of editors in these cases.

White-space and the double backslash

The `\` command is *not* the same as a paragraph break:

it's just a premature linebreak *within* the current paragraph. The double backslash command can have an optional argument (in square brackets) giving an amount of extra white-space to leave, if you need to, eg

```
not the same as a paragraph break\\[3mm]
it's just a premature linebreak
```

(If you need to start the new line with a square bracket for some reason, you will need to prefix it with an empty group (`{ }`) to prevent it being interpreted as the optional argument to `\`.)

1.10.6 Languages

L^AT_EX can typeset in the native manner for several dozen languages. This affects hyphenation, word-spacing, indentation, and the automatic labelling of the parts of documents displayed in headings such as Chapter, Appendix, References, etc (but not the commands used to produce them).

Most distributions of L^AT_EX come with US English and one or more other languages installed by default, but it is easy to use the `polyglossia` or `babel` package and specify any of the supported languages or variants, for example:

```
\usepackage[german,frenchb,english]{babel}
...
As one writer has noted, \selectlanguage{german}``Das
berühmte Voltaire-Zitat, \emph{\foreignlanguage{frenchb}
```

```
{il est bon de tuer de temps en temps un amiral pour  
encourager les autres}}, ist ein Beispiel sarkastischer  
Ironie.''
```

Make sure that the base language of the document comes *last* in the list. The list of supported languages is in the package documentation.

Changing the language with `babel` or `polyglossia` is a cultural shift: it changes the hyphenation patterns, word-spacing, the way in which indentation is used, and the names of the structural units and identifiers like ‘Abstract’, ‘Chapter’, and ‘Index’, etc. For example, using French as the default, chapters will start with ‘Chapitre’.

The `\selectlanguage` lets you tell \LaTeX when to switch to the language specified in the argument. If you have only a small fragment in another language (a word or two, maybe a sentence, but less than a paragraph), you can use the command `\foreignlanguage` to change the language just for that text. The first argument gives the language; the second contains the word or phrase.

The `babel` package uses the hyphenation patterns provided with your version of \LaTeX (see the start of your document log files for a list). For other languages you need to set the hyphenation separately (outside the scope of this book).

1.11 Mathematics

As explained on p. xxii, \TeX was originally written to automate the typesetting of books containing mathematics, because mathematics is typeset differently from normal text. This book does not cover mathematical typesetting, which is explained in detail in many other books and Web pages, so all we will cover here is the existence of the math mode commands, and some characters which have special meaning, so they don’t trip you up elsewhere.

In addition to the 10 special characters listed in section 1.7 on page 22, there are three more characters which only have any meaning inside mathematics mode:

Key	Meaning
	Vertical bar
<	Less-than
>	Greater-than

If you type any of these in normal text (that is, outside math mode), you will get very weird things happening and lots of error messages. If you need to print these characters, you must type them using math mode, or use the symbolic names from the `textcomp` package (`\textbrokenbar`, `\textlangle`, and `\textrangle`).

The hyphen also has a different meaning in math mode, as we saw in the previous section: it typesets as a minus sign, which is both wider and longer than a hyphen, so if you want to write about negative numbers in normal text, you should type the number between inline math delimiters (eg `\(-37^\circ\)`)C.

Delimiters

Plain T_EX used the dollar sign to start and end mathematics, and this was carried over into earlier versions of L^AT_EX, so you'll see a lot of equations in older documents and web pages like `$E=mc^2$` (and a double-dollar for displayed equations).

L^AT_EX defines `\(` and `\)` for inline mathematics like `\(E=mc^2\)` and `\[` and `\]` for displayed mathematics.

You SHOULD therefore use the L^AT_EX method of `\(...\)` and `\[...\]` — Barbara Beeton writes:

The `\(...\)` and `\[...\]` make it easier to diagnose beginning/end matches. Remember that when T_EX was created, memory was severely limited. so terseness was important. Of course, `$` and `$$` are still used under the covers, but the L^AT_EX notation, especially the 'environment' structure, makes it easier to provide many features not at all easy in basic T_EX. For math, I include in that automatic numbering and cross referencing.

Mathematics markup in L^AT_EX is actually not hard, there's just a lot to learn. Most of it is fairly obvious and straightforward: if you compare the equation below carefully with the code, you'll see:

- ☐ the `\[` to start display math;
- ☐ the `\bar` for the \bar{n} ;
- ☐ the caret (^) for a superscript or upper limit (here, an asterisk (*));
- ☐ the underscore (_) for the subscript or lower limit (here, a j for example);
- ☐ the (s) =;
- ☐ a large fraction using `\frac{denominator}{enumerator}`.

and so on. If you know mathematics, you can probably work out the rest by inspection.

```
\[ \bar{n}^{*j}(s) = \frac{\left\{ s \sum_{i=1}^k n_i(0) p_{i,k+1}^{*}(s) + M^{*}(s) \right\} \sum_{i=1}^k p_{0i} p_{ij}^{*}(s)}{\sum_{i=1}^k p_{0i} p_{ij}^{*}(s)} + \frac{\{1 - s \sum_{i=1}^k p_{0i} p_{i,k+1}^{*}(s)\}}{\sum_{i=1}^k p_{0i} p_{ij}^{*}(s)}, \quad (j=1, 2, \dots, k). \]
```

$$\bar{n}_j^{*}(s) = \frac{\left\{ s \sum_{i=1}^k n_i(0) p_{i,k+1}^{*}(s) + M^{*}(s) \right\} \sum_{i=1}^k p_{0i} p_{ij}^{*}(s)}{1 - s \sum_{i=1}^k p_{0i} p_{i,k+1}^{*}(s)} + \sum_{i=1}^k n_i(0) p_{ij}^{*}(s), \quad (j = 1, 2, \dots, k).$$

To use math mode inline (that is, within a paragraph), enclose your math expression in `\(` and `\)` commands (round parentheses). You can get the much-quoted equation $E = mc^2$ by typing `\(E=mc^2\)`, and to get a temperature like -40° you need to type `\(-40^\circ\)` in order to get the minus sign and the right spacing.⁴ Never use the math superscript and a letter o for degrees: all that gets you is a raised italic letter o.

To typeset a math expression as ‘displayed math’ (centered between paragraphs, like the huge equation above), enclose it in

⁴ Bear in mind that the degree symbol is a non-ASCII character, so you must specify what input encoding you are using if you want to type it: see the example of the `inputenc` package in section 1.9 on page 25. If you don't want to use non-ASCII characters (or if you are using a system which cannot generate them), you can use the command `\textdegree` to get the degree sign.

the commands `\[` and `\]` (square brackets)⁵. Displayed equations can be auto-numbered with the *equation* environment instead of the `\[` and `\]` commands. The [American Mathematical Society \(AMS\)](#) document classes and styles provide many extended mathematical features.

For a long time, there were very few typefaces with mathematics fonts (section 6.2 on page 168), basically Computer Modern, Times, Lucida, and Concrete/Euler. This is now changing, and there are new typefaces with math characters, as well as mathematics add-on packages which work with a dozen or more faces (Hartke 2006), and more are being announced (the [notomath](#) package was announced as I wrote this).

⁵ You will also see dollar signs used for math mode. This is quite common but deprecated: it's what plain $\text{T}_{\text{E}}\text{X}$ used in the days before $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, and the habit got ingrained in many mathematicians. It still works as a convenient shorthand like $\text{\$x=y\$}$, as do double-dollars for display-mode math like $\text{\$\$E=mc^2\$\$}$, but they are only mentioned here to warn readers seeing them in other authors' work that `\(... \)` and `\[... \]` are the proper $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ commands.

