Beginner's LaTeX

Electronic Publishing Unit
UCC Computer Centre

v2.2 December 2001

Summary

This booklet is designed to accompany a 2–day course on using the LaTeX typesetting system under Linux or Microsoft *Windows* (but most of it applis to any LaTeX system on any platform, including Apple Macs).

The audience is assumed to be computer-literate and composed of technical professional, academic, or administrative users.

The objective is to enable the users to create their own typeset documents from scratch. The course covers the following topics:

- Where to get and how to install LaTeX (fpTeX from the TeX Live CD-ROM);
- How to type in LaTeX: using an editor to create files (*WinEdt* or *Emacs*);
- Basic document structures (the Document Class Declaration and its layout options; the document environment with sections and paragraphs);
- Typesetting, viewing, and printing;
- The use of packages and the CTAN to adapt formatting using standard tools;
- Other document structures (lists, tables, figures, images, and verbatim text);
- Textual tools (footnotes, marginal notes, cross-references, indexes and glossaries, and bibliographic work);
- Typographic considerations (spacing, font installation and changes, inline markup);
- Programmability (macros and modifying LaTeX's behaviour);
- Compatibility with other systems (XML, *Word*, etc).

It does not cover mathematical typesetting, complex tabular material, the design of user-written macros, or the finer points of typography or typographic design, although it does refer to the existence of these topics.

Foreword

This document grew out of a 1–day introductory training course I ran in the late 1990s in UCC and elsewhere. It became obvious from repeated questions in class and afterwards, as well as from general queries on `comp.text.tex` that many people were not reading the FAQs, not buying the manuals, not downloading the free documentation, and instead trying to get by using the training technique known as 'sitting by Nelly', which involves looking over a colleague's shoulder and absorbing all his or her bad habits.

In the summer of 2001 I presented a short proposal on the marketing of LaTeX to the annual conference of the TeX Users Group held at the University of Delaware, and showed an example of a draft brochure[1] designed to persuade newcomers to try LaTeX for their typesetting. As a result of questions and suggestions, I agreed to make available a revised form of this present training document, expanded to include those topics on which I have had most questions from users over the years.

It turned out to mean a significant reworking of a lot of the standard material which appears in almost every manual on LaTeX, but I took the opportunity to revise the structure to match more closely the expansion of the original training course to two days, and to include a more comprehensive index. It is by no means perfect, and I would be grateful for comments and bugs to be sent to me at `peter@silmaril.ie`.

I had originally hoped that the source code of the document would be processable by any freshly-installed LaTeX system, but the need to include font samples beyond the default installation, and to use some packages which the new user is unlikely to have installed, means that this document itself is not really a simple piece of LaTeX, however simply it may describe the process itself.

If you are just starting with LaTeX, at an early opportunity you should acquire a copy of *LaTeX, a document preparation system*[2] which is the author's manual. More advanced users should get the *The LaTeX Companion*[3] or one of its successors. In the same series there are also the *The LaTeX Graphics Companion*[4] and the *The LaTeX Web companion*[5].

[1] `http://www.silmaril.ie/documents/latex-brochure/leaflet.pdf`
[2] Lamport, (1994)
[3] Goossens *et al*., (1994)
[4] Goossens/Rahtz/Mittelbach, (1997)
[5] Goossens/Rahtz, (1999)

Contents

Preface

This is a technical document. LaTeX is a very easy system to learn, but it assumes that you are completely fluent and familiar with using a computer before you start. Specifically, it assumes that you know and understand the following thoroughly:

- How to run and use a plaintext editor (*not* a wordprocessor);
- Where all 96 of the printable ASCII characters are on your keyboard;
- How to open, save, and close files in a windowing environment (the File menu);
- How to create, name, and move files and directories;
- How to use a Web browser to download and save files to disk;

Just to clear up any misunderstandings:

- TeX is a typesetting engine, written by Prof Don Knuth (Stanford) in 1978. It implements a typesetting programming language of some 900 basic operations and has formed the core of dozens of DTP systems. Although it is possible to write in 'raw' TeX, you need to study it first, and you need to be able to write macros (mini command programs) to perform even the simplest of tasks.
- LaTeX is a user interface for TeX, designed by Leslie Lamport (DEC) in 1985 to automate all the common tasks of document preparation. LaTeX is the recommended system for all users except professional typographic programmers and computer scientists who want to study the internals of TeX.

Both TeX and LaTeX have been constantly updated since their inception. Knuth has now frozen development of the TeX engine so that users have a virtually bug-free, stable platform; typographic programming development continues with the NTS (New Typesetting System). The LaTeX3 project has taken over development of LaTeX, and the current version is LaTeX 2_ε. Details of all developments can be had from the TeX Users Group (TUG) at `http://www.tug.org`, which all users should consider joining.

SESSION I

Where to get and how to install LaTeX

The installation CD-ROM and documentation are available from the UCC Computer Centre.

This course is based on using Thomas Esser's fpTeX (for Linux and other Unixes) and François Popineau's fpTeX (for Microsoft *Windows*) from the TeX Live CD-ROM (fpTeX is in fact a Windows implementation of fpTeX).

This CD-ROM is issued annually by the TeX Users Group, of which UCC is an institutional member, and the disk is republished by many local user groups around the world (see `http://www.tug.org/lugs.html` for addresses).

There is a local installation guide which gives extensive and very explicit step-by-step instructions on how to install the Microsoft *Windows* version (it's actually very simple: you accept the default for almost everything, but a lot of new users are uneasy with the succession of little windows that pop up). The Linux version is usually installed from the operating system installation disks by default (using the RPM or other packaged files provided), but can also be installed from the TeX Live CD-ROM.

Other versions of TeX, including those for other platforms entirely, can be downloaded from the Comprehensive TeX Archive Network (CTAN: see session 5).

Two warnings: the TeX Live CD-ROM no longer includes the *GSview* preview program or the *WinEdt* editor as these are not freely distributable any more. It is strongly recommended you download *GSview* from the links at `http://www.ghostscript.com` and *WinEdt* from `http://www.winedt.com` and install them yourself first.

SESSION II

Using an editor to create files

This course assumes that users will have either *WinEdt* or *Emacs* on their system. *WinEdt* is available for download; *Emacs* comes on most Linux systems by default these days (and is available for Microsoft *Windows* and other platforms as well, should you prefer it). Both are discussed briefly in section 2.3 and the menus and toolbars for running LaTeX are explained in section 4.

LaTeX documents are all plaintext files: it keeps your document text separately from the WYSIWYG typeset display. Most versions of LaTeX do not have a synchronous display to let you interact with the typeset characters while you type, like a graphical wordprocessor does, because LaTeX's formatting is done on an asynchronous basis which provides you with greater controllability.[1] This is a small penalty to pay for the difference in power and typeset quality, which is immediately obvious to most readers.

In a LaTeX document, you type your text along with commands which identify the important parts of your document by name ('title', 'section', 'list', etc), and LaTeX does all the formatting for you automatically, using these commands to guide its internal rules for typesetting. You do not need to format any of your text by hand in your editor, because LaTeX does it all by itself. You can of course regularise or neaten its appearance *in your editor* for ease of editing (for example, keeping each item in a list on a separate line), but this is not required.

You may hear commands referred to as 'control sequences', which is the proper TeXnical term for them.

The biggest advantage of LaTeX's approach is consistency: so long as you identify each element of your document correctly, it will be typeset identically to all other elements of the same type, so that you achieve a professional finish with minimum effort.

2.1 LaTeX commands

LaTeX commands all begin with a **backslash** character (\) and are mostly made up

Do not confuse the backslash (\) with the forward slash (/).

[1] The exceptions are the Mac version, *Textures*, and one of the PC versions, *Scientific Word*, both of which are commercial products; and *Lyx*, which is free.

of lowercase letters only, for example **\clearpage**.[2] These commands that are followed directly by more text must be kept separate from that text by a space or linebreak, for example.

Spaces, TABs, and linebreaks ('newlines' in Unix terms) are collectively referred to in Document Engineering as **white-space**.

```
\clearpage The importance of poetic form must not
```

Omitting the white-space would confuse LaTeX because it would see what looks like a command: `\clearpageThe` but which is not. The separating space you type gets swallowed up by the command, so you don't get unwanted space in your typesetting.

Many LaTeX commands are followed by **arguments** (text to act upon, or additional identifying information), for example

```
\chapter{Poetic Form}    or    \cite{maxwell96}
```

These arguments always go in **curly braces** like those shown. In this case you do *not* need to add white-space after the command name, because there is an argument following.

Do not confuse these on your keyboard with round parentheses (), square brackets [], or angle brackets ⟨⟩.

In LaTeX documents, multiple spaces, linebreaks, and TABs are all treated as if they were a single space during typesetting, so you can use white-space for optical ease and convenience when editing. For example, the following is exactly equivalent to the preceding example:

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

2.2 Special characters

There are ten keyboard characters which have special meaning to LaTeX, and cannot be used on their own except for these purposes:

Key	Meaning	*If you need the actual character itself, type this:*	Typeset Character
\	The command character	`$\backslash$`	\
$	Math typesetting delimiter	`\$`	$
%	The comment character	`\%`	%
^	Math superscript character	`\^{}`	^
&	Tabular column separator	`\&`	&
_	Math subscript character	`\_`	_
~	Non-breaking space	`\~{}`	~
#	Macro parameter symbol	`\#`	#
{	Argument start delimiter	`$\{$`	{
}	Argument end delimiter	`$\}$`	}

[2] The **\clearpage** command is a manual instruction to the typesetter to start a new page. Page-breaking is usually automatic in LaTeX.

(These were deliberately chosen because they are rare in normal text, with the exception of $, #, &, and %.)

So if you want $35.99 you type `\$35.99`;[3] if you want AT&T you type `AT\&T`; if you want 45% you type `45\%`; and if you want an American number sign (#) you type `\#`.

The comment character (%) makes LaTeX ignore the remainder of a line in your document, so you can see it in your editor, but it will never get typeset, for example:

```
Today's price per Kg is £22.70 % get Mike to update this
```

2.2.1 Quotation marks

Do *not* use the keyboard " key for quotation marks. Correct typographic quotes are got with the ` key and the ' key, doubled if you want double quotes:

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

This ensures you get left-hand and right-hand (opening and closing) quotes. When typing one quotation inside another, there is a special command **`\thinspace`** which provides just enough separation between double and single quotes (a normal space is too much):

```
He said, `Her answer was ``never''\thinspace', and
```
He said, 'Her answer was "never"', and

2.2.2 Accents

For accented letters in ISO 8859–1 (Western European) or any other non-ASCII encoding just type your standard keyboard Ctrl or Alt sequences or use the accented keys on your keyboard if you have them. To make LaTeX recognise them, use the package inputenc with the relevant option (eg `\usepackage[latin1]{inputenc}`).

We haven't covered the use of packages yet: see session 5 if you're curious.

Examples (see your Operating System manual for full details):

The letter 'é' on a Linux PC is AltGr–; e

The letter 'é' on a Microsoft *Windows* PC is Ctrl–' e or Alt–0 1 3 0.

If you cannot generate ISO 8859–1 characters on your keyboard, or you need additional accents, you can use the symbolic notation below. In fact, this can be used to put any accent over any letter: if you particularly want a g̃ you can have one with the command `\~g` (and Welsh users can at last get ŵ with `\^w`).

[3] An unusual but interesting serif-font Euro sign € is got with **`\texteuro`** from the textcomp package. The standard sans-serif € needs the marvosym package and is done with the **`\EUR`** command. The European Commission specifies that everyone use the sans-serif design even in serif text, but this is amazingly ugly.

Accent	Example	Characters to type
Acute (fada)	é	`\'e`
Grave	è	`\`e`
Circumflex	ê	`\^e`
Umlaut or diæresis	ë	`\"e`
Tilde	ñ	`\~n`
Macron	ō	`\=o`
Bar-under	o̲	`\b o`
Dot-over (séimiú)	ṁ	`\.m`
Dot-under	ṣ	`\d s`
Breve	ŭ	`\u u`
Háček	ǔ	`\v u`
Long umlaut	ő	`\H o`
Tie	o͡o	`\t oo`
Cedilla	ç	`\c c`
O-E ligature	œ, Œ	`\oe`, `\OE`
A-E ligature	æ, Æ	`\ae`, `\AE`
A-ring	å, Å	`\aa`, `\AA`
O-slash	ø, Ø	`\o`, `\O`
Soft-l	ł, Ł	`\l`, `\L`
Ess-zet (scharfes-S)	ß	`\ss`

Irish and Turkish dotless-ı is done with the special command **`\i`**, so an í-fada which is normally typed `í` requires `\'\i` if you need to type it in long format. Because of the rule that LaTeX control sequences which end in a letter (see page 9) always absorb any following spaces, this `\i` needs to be followed by a space before the next letter, or a backslash-space or dummy pair of curly braces when at the end of a word not followed by punctuation: what you normally type as `Rí Teamhrac` has to be `R\'\i\ Tea\.mra\.c` when typed in full (there are no keyboard keys for the dotless-ı or the lenited characters). A similar rule applies to dotless-ȷ.

2.2.3 Hyphenation, justification, and breaking

LaTeX hyphenates automatically. To specify different breakpoints for a word, you can insert soft-hyphens (discretionary hyphens, done with **`\-`**) wherever you want, such as in `Popa\-cata\-petl`. To specify hyphenation points for all occurrences of a word, use the **`\hyphenation`** command in your preamble (see box on page 20) with one or more words in its argument. This will even let you break 'helicopter' correctly.

```
\hyphenation{helico-pter Popa-cata-petl im-mer-sion}
```

To force LaTeX to treat a word as unbreakable, use **`\mbox`** with the word in curly braces: `\mbox{pneumonoultramicroscopicsilicovolcanoconiosis}`. This may have undesirable results, however, if you change margins: pneumonoultramicroscopicsilicovolcanocon

To tie two words together with an unbreakable space (hard space), use a tilde instead of the space. This will print as a space but LaTeX will never break the line at that point. You should make this standard typing practice for things like people's initials followed by their surname, as in Prof. D. E. Knuth: `Prof.\ D.~E.~Knuth`. A full

point after a lowercase letter is treated as the end of a sentence, and creates more space before the next word. Here, it's *not* the end of a sentence, and the backslash-space forces LaTeX to insert just an ordinary word-space because it's OK to break the line after 'Prof.', whereas it would look wrong to have initials separated with Prof. D. E. Knuth broken over a line-end.

For a long dash — what printers call an 'em rule' like this — use three hyphens typed together, `like~--- this`, and bound to the preceding word with a tilde to avoid the line being broken before the dash. It's also common to see the dash printed without spaces—like that: the difference is purely æsthetic. *Never* use a single hyphen for this purpose. Between digits like page ranges (35–47), it is normal to use the short dash (en-rule) which you get by typing two hyphens together, as in `35--47`. If you want a minus sign, use math mode (next section): do *not* use the normal hyphen.

The default mode for typesetting is justified (two parallel margins, with word-spacing adjusted automatically for the best optical fit). In justifying, LaTeX will never add space between letters, only words (there is a special so [space-out] package if you need special effects like letter-spacing).

There are two commands **`\raggedright`** and **`\raggedleft`** which set ragged-right (ranged left) and ragged-left (ranged right). Use them inside a group (see page 53) to confine their action to a part of your text.

These modes also exist as environments (see page 17) called raggedright and raggedleft which is more convenient when applying this formatting to many paragraphs.

```
\begin{raggedleft}
These modes also exist as environments called raggedright
and raggedleft which is more convenient when applying this
formatting to many paragraphs.
\end{raggedleft}
```

2.2.4 Mathematics

There are three characters which only have meaning inside mathematics mode:

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

To get $|$, $<$, and $>$, type `$|$`, `$<$`, and `$>$` respectively (but you shouldn't need these outside math mode much anyway).

The hyphen has an extra meaning in math mode: it turns into a minus sign, so if you want $-3°$, type `\(-3\)\textdegree`.

The careful reader will already have noticed that mathematics is handled differently from normal text. This document does not explain math mode in detail, but it is useful to know that inline math (printed as part of the surrounding text) is enclosed in `\(`[4] and `\)`, so $E = mc^2$ is got with `\(E=mc^2\)`. Note the special use of ^ to produce a superscript.

Display math uses `\[` and `\]`, and prints the formula on a line by itself, in larger type (auto-numbering is also possible):

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

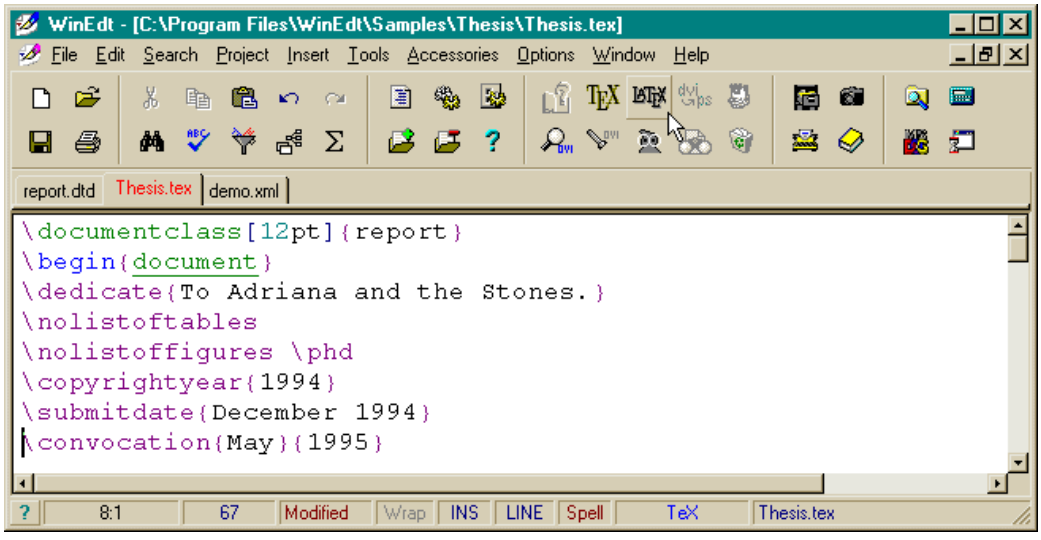
2.3 Editors

All the preceding can be typed into your LaTeX document from a standard keyboard using any decent plaintext editor. However, it is more convenient to use an editor with special features to make using LaTeX easier, and two of the most popular are *WinEdt* and *Emacs*.

2.3.1 WinEdt

WinEdt is a windowing plaintext editor for Microsoft *Windows* (all versions). It lets you create and maintain files of plain text (*not* wordprocessing files) for any application. Download it from `http://www.winedt.com`

WinEdt has a built-in toolbar of configurable buttons, and it provides automatic syntactic colourising of LaTeX commands in files you edit.



[4] The use of dollar signs shown earlier is common but deprecated: it's what plain TeX used in the days before LaTeX, but the habit got ingrained in many mathematicians, so it still works as a convenient shorthand, as do double-dollars for display-mode math, and they are shown here to accustom the reader to seeing them in other authors' work even though `\(`...`\)` and `\[`...`\]` are the formal commands.

Apart from standard features like the File menu, Edit menu, searching, etc, *WinEdt* has default buttons on its toolbar for one-click typesetting, previewing, and PostScript or PDF generation from LaTeX documents.

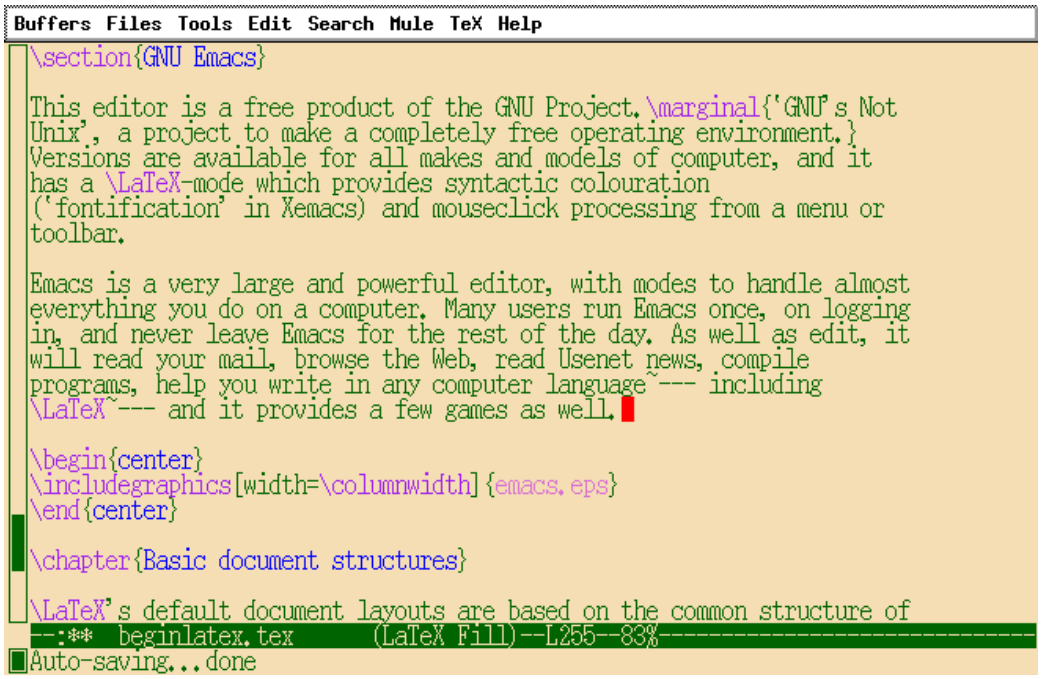
WinEdt comes configured for the MikTeX version of LaTeX, rather than fpTeX, so some editing of the menus is required (explained in the local installation document) after finishing the fpTeX installation. Alternatively, you could install MikTeX instead.

WinEdt is group-licensed in UCC. If you install it from CD-ROM or from the network, you should obtain your license key code from the Computer Centre. Used on its own it otherwise starts to nag you to register after the 1–month free trial period is over.

2.3.2 GNU Emacs

Emacs is a free product of the GNU Project. Versions are available for all makes and models of computer, and it has a LaTeX-mode which provides syntactic colouration ('fontification' in *Xemacs*) and mouseclick processing from a menu or toolbar.

'GNU's Not Unix', a project to make a completely free operating environment.



Emacs is a very large and powerful editor, with modes to handle almost everything you do on a computer. Many users run *Emacs* once, on logging in, and never leave *Emacs* for the rest of the day. As well as edit, you can use it to read your mail, browse the Web, read Usenet news, do wordprocessing and spreadsheets, compile programs, help you write in any computer language — including LaTeX — and it provides a few games as well.

Emacs, like *WinEdt*, knows about LaTeX and how to process it, so there is a menu full of LaTeX operations to click on. If you are editing more complex documents,

there is an add-on package ('mode' in *Emacs*-speak) called *AUCTEX* which has more functionality.

Because *Emacs* runs on Microsoft *Windows*, Macs, Linux, VMS, and other platforms, many LaTeX users who use multiple machines prefer it to other editors because it provides the same environment regardless of which platform they are using.

SESSION III

Basic document structures

LaTeX's default document layouts are based on the common structure of parts, chapters, sections, subsections, subsubsections, and so on (the exception is the 'letter' class of documents, as we shall see below).

3.1 The Document Class Declaration

To tell LaTeX what class of document you are going to create, you type a special first line in your file which identifies it, eg

Readers familiar with HTML or XML will recognize the concept as similar to the Document Type Declaration.

```
\documentclass{report}
```

There are four default classes provided:

`report` for business, technical, and laboratory reports;

`article` for articles, reviews, and research notes;

`book` for books and theses;

`letter` for letters.[1]

There are many more you can download, but these are the most common. The Article class in particular can be used (some would say 'abused') by omitting the formal titling and abstract layout to leave it looking like an undistinguished piece of wordprocessing (albeit neater and better typeset!).

[1] The Letter class is not well-defined: there are much better ones you can download and install yourself.

3.1.1 Layout options

The default layouts are designed for US 'Letter' size paper. To create documents with the proper margins for ISO A4 paper, you specify the paper size in an optional argument in square brackets before the document class name, eg

```
\documentclass[a4paper]{report}
```

There are other options as well:

The default type size is 10pt, which in some typeface designs is too small for A4 paper.

11pt to specify 11pt type (headings etc get scaled up in proportion);

12pt to specify 12pt type (headings scale);

oneside for one-sided printing;

Reports and books assume two-sided printing, which means the text area is shifted slightly from left-hand page to right-hand page.

twoside to force articles to two-sided print shifting;

titlepage to force articles to have a separate title page.

Reports and books always have a separate title page by default.

The settings used for this document are based on:

```
\documentclass[11pt,a4paper,oneside]{report}
```

There are add-on packages which can automate hundreds of other layout and formatting variants without you having to program anything by hand or even change your text.

3.2 The document environment

After the Document Class Declaration, the text of your document is enclosed between two commands which identify the beginning and end of the actual document:

```
\documentclass[11pt,a4paper,oneside]{report}
\begin{document}
...
\end{document}
```

This is an example of a common LaTeX technique called an **environment**. Environments enclose text which is handled in a specific manner. They all start with `\begin{...}` and end with `\end{...}` (putting the name of the environment in curly braces).

3.2.1 Titling

The first few lines (replacing the ellipsis [...] in the example above) are always used to specify the document title, the author's name, and the date (except in letters, which have a special set of commands for addressing).

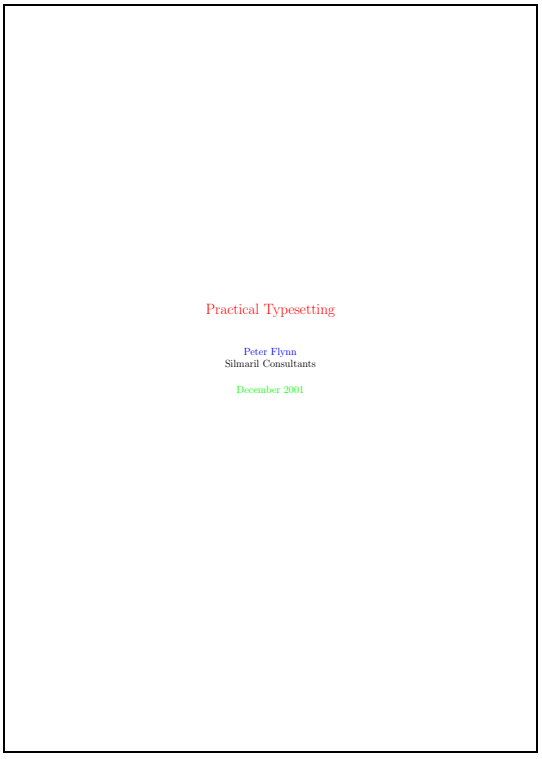
```
\documentclass[11pt,a4paper,oneside]{report}
\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2001}
\maketitle

\end{document}
```

The **\title**, **\author**, and **\date** commands are mostly self-explanatory: the only restriction is that you *must* finish the title block with the **\maketitle** command, otherwise the titling will never be typeset. This is what signals the end of the title page and the start of the actual text.

Note the double-backslash in the **\author** command argument. This is LaTeX's shorthand for 'start a new line here'. When this file is typeset, you get something like this:



Before we see how to get this printed, however, there are just four more elements to cover: abstracts, sectioning, the Table of Contents, and paragraphs.

3.2.2 Abstracts

In reports and articles it is normal for the author (you) to provide an Abstract, in which you summarise what you have written about and explain its importance. Abstracts in articles are usually only a few paragraphs long; Abstracts in reports can be several pages. In both cases the Abstract is optional, but conventional.

Immediately after the **\maketitle** you can use the abstract environment, in which you simply type your Abstract, leaving a blank line between paragraphs (see page 21 for this convention).

```
\documentclass[11pt,a4paper,oneside]{report}
\usepackage[latin1]{inputenc}
\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2001}
\maketitle

\begin{abstract}
This report presents the basic concepts of typesetting in a
form usable by non-specialists. It is aimed at those who find
themselves (willingly or unwillingly) asked to undertake work
previously sent out to a professional printer, and who are
concerned that the quality of work (and thus their corporate
image) does not suffer unduly.

The topics cover layout, the need for accuracy, the choice of
typeface, arrangement of the document, adherence to
specifications, and the production process. No foreknowledge
of printing or publishing is needed, but an eye for detail,
a feeling for æsthetics, and some fluency with a computer is
expected.
\end{abstract}

\end{document}
```

In business and technical reports, the Abstract is often called a Summary, Executive Summary, Preview, or some similar phrase (see page 2). LaTeX lets you redefine the Abstract name to any kind of title you want, using a command of the form:

```
\renewcommand{\abstractname}{Executive Summary}
```

Put this line in your preamble (see box on page 20), which is at the start of the document, after the **\documentclass** but before the **\begin{document}**.

3.2.3 Sections

In the remainder of your document, LaTeX provides seven levels of sectioning for you to use in structuring your text. They are all optional: it is perfectly possible to write

a document consisting solely of paragraphs of undivided text — just unusual: even novels are normally divided into chapters, although short stories are often written as single collections of paragraphs.

Two of these divisions, Parts and Chapters, are only available in the 'book' and 'report' document classes, because they don't have any use or meaning in articles and letters.[2]

Depth	Division	Command	Notes
−1	Part	**\part**	Only in books and reports
0	Chapter	**\chapter**	Only in books and reports
1	Section	**\section**	
2	Subsection	**\subsection**	
3	Subsubsection	**\subsubsection**	
4	Titled paragraph	**\paragraph**	
5	Titled subparagraph	**\subparagraph**	

In each case the title of the part, chapter, section, etc goes in curly braces after the command. LaTeX automatically calculates the correct numbering and prints the title in bold, like this document. Parts get Roman numerals (Part I, Part II, etc); chapters and sections get decimal numbering, and Appendixes (which are just specialised form of chapters, and share the same structure) are lettered (A, B, C, etc).

You can change the depth to which numbering occurs, so if you only wanted chapters, sections, and subsections numbered, you could change the value of the *secnumdepth* counter, using the depth value from the table above:

```
\setcounter{secnumdepth}{2}
```

LaTeX has many counters calculating all sorts of values while it typesets, and this command is just one of many that can be used to change LaTeX's behaviour.

All modifications like this, which affect a whole document, go at the *start* of your LaTeX file, *immediately after* the **\documentclass** line and *before* the \begin{document} line:

```
\documentclass[11pt,a4paper,oneside]{report}
\setcounter{secnumdepth}{2}
\begin{document}
...
\end{document}
```

This position, between the Document Class Declaration and the beginning of the document text, is called the **preamble**, and it is used for all kinds of modifications to the style and behaviour of the document.

[2] It is arguable that chapters also have no place in reports, as these are conventionally divided into sections as the top-level division. However, this is only the default, and can be changed very simply (see section 9).

A related counter which can also be set in the preamble is *tocdepth*, which specifies what depth to take the Table of Contents to. It can be set in exactly the same way as *secnumdepth*. The current setting for this document is 1.

To get an unnumbered section heading which does not go into the Table of Contents, follow the command name with an asterisk before the opening curly brace:

```
\subsection*{Shopping List}
```

All the sectional commands from **\part*** to **\subparagraph*** have this 'starred' version which can be used on special occasions for an unnumbered heading when it would normally have been numbered.

3.2.4 Table of contents

All auto-numbered headings get entered in the Table of Contents (ToC) automatically. You don't have to print a ToC, but if you want to, add the single command **\tableofcontents** at the point where you want it printed. Entries are gathered each time you process the document, and reproduced the next time you process the document, so it takes two runs of LaTeX to be certain that the page number calculations are correct.

There are commands to add extra lines to the ToC, and features to let unnumbered section headings get added, and to add a text slightly different from the one printed.

The commands **\listoffigures** and **\listoftables** work in exactly the same way to automatically list all your tables and figures.

3.2.5 Ordinary paragraphs

After section headings comes your text. Just type it and leave a blank line between paragraphs. That's all LaTeX needs.

The blank line means 'start a new paragraph here': it does *not* (repeat: NOT) mean you get a blank line in the typeset output.

The spacing between paragraphs is a controllable amount, a **dimension** called \parskip, normally zero, but you can set its **length** to any size you want:

```
\setlength{\parskip}{1cm}
```

This will set the space between paragraphs to 1cm.[3] Leaving multiple blank lines between paragraphs means nothing: all consecutive blank lines get squeezed to one. To change the space between paragraphs, specify it with the command as shown above.

The same applies to the indentation. Normally, the first paragraph after a heading follows the standard publishers' practice with *no* indentation. Subsequent paragraphs

[3] See section 3.2.6 for details of the various size units LaTeX can use.

are indented by the length value of `\parindent` (default 18pt). You can change this in the same way as any other length, normally in the preamble:

```
\setlength{\parindent}{6mm}
```

To turn off indentation completely, set it to zero (but you still have to provide units: it's still a measure!)

```
\setlength{\parindent}{0in}
```

There are many hundreds of such lengths controlling aspects of LaTeX's typesetting, and they can all be set and reset by this method.

> Now you've got enough information to start typing your first report. Use the template we've looked at so far, add a **`\chapter`** command to provide your top-level heading, then follow it with a couple of paragraphs of text and maybe a **`\section`** heading and more text.

3.2.6 Specifying size units

You can specify the length in any of the following units:

Unit	Size
	Printers' fixed measures
pt	Anglo-American standard points (72.27 to the inch)
pc	pica ems (12pt)
bp	Adobe 'big' points (72 to the inch)
sp	TeX 'scaled' points (65536 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 (approx width of letter 'M')
ex	x-height of the current font (approx height of letter 'x')
	Other measures
cm	centimeters (2.54 to the inch)
mm	millimeters (25.4 to the inch)
in	inches

Most people in printing and publishing habitually use points and picas and ems. Some designers use cm and mm. Many Irish, British, and Americans still use inches.

Watch out for DTP users who think that Adobe points (bp) are the only ones. The difference is only .27pt per inch, but in 10″ of text that's 2.7pt, which is nearly 1mm, enough to be visible.

SESSION IV

Typesetting, viewing and printing

We've now got far enough to typeset what you've entered. I'm assuming at this stage that you have typed some sample text in the format specified in the previous chapter, and you've saved it in a plaintext file with a filetype of `.tex` and a name of your own choosing.

> Never, ever use directories (folders) or file names which contain spaces. Although your operating system probably supports them, some don't, and they will only cause grief and tears.
>
> Make filenames as short or as long as you wish, but strictly avoid spaces. Best of all, stick to upper- and lower-case letters without accents (A–Z and a–z), the digits 0–9, the hyphen (`-`), the underscore (_), and the period (`.`), as this is like the specification for a Web URL: it will let you refer to TEX files over the Web more easily and make your files more portable.

4.1 Typesetting

To run LATEX on your file, click the LATEX toolbar icon (*WinEdt*) or pick `TeX File` from the TEX menu (*Emacs*). Your editor may suggest you save your file if you haven't already done so.

LATEX will process your file and display its log. This is to let you see where (if!) there are any errors. Don't panic if you see errors: it's very common for learners to mistype or mis-spell commands, forget curly braces, or type a forward slash instead of a backslash.

If there were no errors, the file is ready for displaying or printing. In *WinEdt* you have to press the Enter key to dismiss the log window.

4.1.1 *pdflatex*

If you have configured your system to generate PDF files direct instead of DVI files (see page 25 for details of DVI), then you can skip section 4.2.1 and go straight to 4.2.2.

4.1.2 Error messages

Most error messages are self-explanatory, but because some errors can only be righted by humans who can read the text and understand what it's supposed to mean, some errors don't get spotted by LaTeX until much later, leading to several error messages in a row.

Fortunately the layout of an error message is the same all the time. Error messages begin with an exclamation mark and a description of the error, followed by a line starting with the line number in your document file where the error was spotted.

```
! Too many }'s.
l.6 \date December 2001}
```

In the example above, the reason there are too many }'s is that the opening curly brace after **\date** and before the word December is missing, so the closing curly brace is seen as one too many.

```
! Undefined control sequence.
l.6 \dtae
         {December 2001}
```

In this second example, LaTeX is complaining that it has no such command ('control sequence') as \dtae (it's been mistyped, but only a human can detect that).

```
Runaway argument?
{December 2001 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
                   \par
l.8
```

In this final example of an error, the closing curly brace has been omitted from the date, resulting in **\maketitle** trying to format the title page while LaTeX is still expecting more text for the date! As **\maketitle** creates new paragraphs on the title page, this is detected and LaTeX complains that the previous paragraph has ended but **\date** is not yet finished.

If you find an error message you can't understand, ask for help. See the section on online help (section 5.2 on page 31) for details.

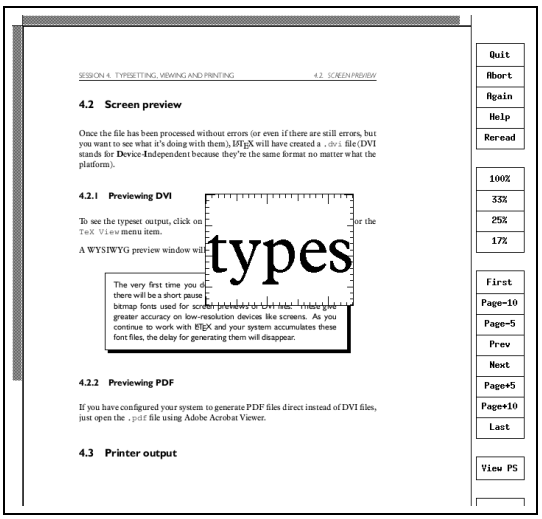
4.2 Screen preview

Once the file has been processed without errors (or even if there are still errors, but you want to see what it's doing with them), standard LaTeX will have created a `.dvi` file (DVI stands for **D**evice-**I**ndependent because these files are the same format no matter what the platform). If you're creating PDF, skip the next subsection.

4.2.1 Previewing with DVI

To see the typeset output, click on the Preview toolbar icon dvi (*WinEdt*) or the `TeX View` menu item.

A WYSIWYG preview window will appear with your typeset display.



Many previewers have a wide range of scaling, zooming, and measuring functions, but you cannot edit the image. To make changes, always go back to your LaTeX file and change the source, then re-process.

> The very first time you do this with a new installation of TeX, there will be a short pause while the previewer creates the special bitmap fonts used for screen previews of DVI files. These give greater accuracy on low-resolution devices like screens. As you continue to work with LaTeX and your system accumulates these font files, the delay for generating them will disappear.

4.2.2 Previewing with PDF

If you have configured your system to generate PDF files direct instead of DVI files, just open the `.pdf` file using Adobe Acrobat Reader.

4.2.3 Previewing with PostScript

PostScript is a page description language invented by Adobe and used in laser printers and high-end typesetters. It's been the universal standard for electronically-formatted print files for over a decade, and all printers and publishers are accustomed to using them. PDF is slowly taking over, but PostScript is still very common.

An alternative to viewing the DVI file direct is to generate a PostScript file, especially if you're going to have to do this for your publisher anyway, and many editors can be configured to do this by default. Look for a toolbar icon such as dvips.

It's very simple to do manually anyway, using the *dvips* program. Let's assume your LaTeX file was called `mydoc.tex`, so processing it has created `mydoc.dvi`. Just type:

```
$ dvips -o mydoc.ps mydoc
```

and *dvips* will create `mydoc.ps` which can be used both for previewing *and* printing. To view a PostScript file, you need a PostScript previewer like *GSview*, which works with *GhostScript*, which should have been installed along with your whole TeX system (if not, install both now). The command

```
$ gv mydoc
```

will pop up the *GSview* previewer window.

4.3 Printer output

Printing varies slightly according to how you view and display your typesetting:

If you are using PDF, you can print directly from the Adobe Acrobat Reader preview screen.

If you are using DVI, and you have a previewer which has a print function configured for your printer, you can use that.

If you are using PostScript, use the print function in *GSview*.

If you have a real PostScript printer, or you are using a system with built-in PostScript printing support (such as Linux), you can create and send a `.ps` printfile directly to the printer without the need to view it first. In *Emacs*, you do this with the `TeX Print` menu.

Otherwise, create a PostScript file (see section 4.2.3) and use *GSview* to print it (*GSview* can print PostScript files to almost any make or model of non-PostScript printer);

Or, for non-PostScript printers, install or configure a TeX print driver for your printer (as supplied with your TeX installation, and there are dozens on CTAN: their names all start with dvi and are followed by an abbreviation for the printer

make or model like dvieps for Epsons). Configure this driver to print directly to the print queue, or pipe it to the print queue manually. On Linux this would be:

```
$ dvixxx mydoc  or  $ dvixxx mydoc | lpr
```

Microsoft *Windows* has no easy way to bypass the print spool, but you can do it from an MS-DOS window with the command:

```
C:\texdocs\> dvixxx mydoc     (this creates mydoc.xxx)
C:\texdocs\> print /b mydoc.xxx
```

Both the *dvips* program and all the previewers that print should have facilities for printing selected pages, printing in reverse, scaling the page size, and even for imposing multiple page images on a single sheet.

Prove that you have understood the process of typesetting, previewing, and printing, by displaying a page of your document and printing it.

SESSION V

The use of packages and the CTAN

The Comprehensive TEX Archive Network is a collection of Web and FTP servers worldwide which contain copies of every free piece of software related to TEX and LATEX. The CTAN is rooted at `http://www.ctan.org` and there are several online indexes.

To find out what packages are available, you should browse the index at any of the linked servers.

To list the packages installed on your system, look for files ending in `.sty` within the `texmf/tex/latex` subdirectory of your installation. See section 5.1.2 for details of package documentation.

> The CTAN should *always* be your first port of call when looking for a software update or a feature you want to use. Please don't ask the network help resources until you have checked CTAN first.

5.1 Packages

Add-on features for LATEX are known as 'packages'. Dozens of these are pre-installed with LATEX and can just be used in your documents without further work.

Additional packages can be downloaded from CTAN and installed on your computer either to update a copy you already have (perhaps a copy that was installed along with your version of LATEX), or to add to your resources.

There is no theoretical limit to the number of packages you can have, but there is probably a physical limit to the number that can be active inside any one LATEX document, although it depends on how big each package is. In practice there is no problem in having even a couple of dozen packages active.

5.1.1 Using an existing package

To use a package already installed on your system, refer to its name with the **\usepackage** command in your document preamble. For example, to use the color package, which lets you typeset in colours:

```
\documentclass[11pt,a4paper,oneside]{report}
\usepackage{color}
\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2001}
\maketitle

\end{document}
```

You can include several package names in one **\usepackage** command by separating the names with commas, and you can have more than one **\usepackage** command.

```
\documentclass[11pt,a4paper,oneside]{report}
\usepackage{pslatex,palatino,avant,graphicx,color}
\usepackage[margin=2cm]{geometry}
\begin{document}

\title{\color{red}Practical Typesetting}
\author{\color{blue}Peter Flynn\\Silmaril Consultants}
\date{\color{green}December 2001}
\maketitle

\end{document}
```

Many packages can have additional formatting specifications in optional arguments in square brackets, in the same way as the **\documentclass** command does. Read the documentation for the package concerned to find out what can be done.

5.1.2 Package documentation

To find out what commands a package provides (and thus how to use it), you need to read the documentation. In the `texmf/doc` subdirectory of your installation there should be `.dvi` files for every package installed. These can be previewed or printed as normal (see section 4.2.1).

Before using a package, you should read carefully the documentation, especially the subsection usually called 'User Interface', which describes the commands the package makes available. You cannot just guess and hope it will work: you have to read it and find out.

See the next section for details of how to read documentation on new packages you install yourself.

5.1.3 Downloading and installing packages

Once you have identified a package you need and haven't already got (or you have got it and need to update it), use the indexes on any CTAN server to find the directory where the package lives. You can use the index at `http://www.ucc.ie/cgi-bin/ctan` to search for the filename.

What you must look for is always two files, one ending in `.dtx` and the other in `.ins`. The first is the DOCTEX file, which combines the package code and its documentation. The second is the installer routine (always much smaller). You *must* download *both* files.

Some rare or obsolete packages are still supplied as a single `.sty` file intended for the obsolete LATEX 2.09 version. You can try to use these if you wish but they are not guaranteed to work. Always look for the `.dtx` and `.ins` pair of files first.

Download both files to a temporary directory. Keep `C:\tmp` for this in Windows; all Linux systems already have a `/tmp` directory. There are four steps to installing LATEX packages:

1. Run LATEX on the `.ins` file. That is, open the file in your editor and process it as if it were a LATEX document, or if you prefer, type `latex` followed by the filename in a command shell or MS-DOS window.

   This will extract all the files needed from the `.dtx` file (which is why you must have both of them present in the temporary directory). Note down the names of the files created (read the log file if you want to see them).

2. Run LATEX on the `.dtx` file, twice. This will create a `.dvi` files of documentation explaining what the package is for and how to use it. Twice is needed to resolve any internal crossreferences in the text (a feature we'll come onto later).

3. While that is printing, move or copy the files created by Step 1 from your temporary directory to the correct place in your TEX installation directories. In an installation that conforms to the TEX Directory Structure (TDS), this is:

   For new, additional packages: `texmf.local/tex/latex/`
   (you can create subdirectories if you wish);

   For upgrades of standard packages: `texmf/tex/latex/...`
   (whichever subdirectory the original `.sty` file was in: look for it).

   Normally there is just a `.sty` file to move but in the case of complex packages there may be more. If you didn't note down the names from Step 1, you may be able to identify them from a directory listing sorted by reverse date. Under Linux, for example, type `ls -lt|more`

4. Run your TEX indexer program to update the package database. This program comes with every modern version of TEX and is variously called *texhash*, *mktexlsr*, or even *configure* or it might just be a mouse click on a menu in your editor. Read the documentation that came with your installation to find out which one it is.

   This step is *utterly essential*, otherwise nothing will work.

The reason this process has not been automated widely is that there are still thousands of installations which do not conform to the TDS, so there is no way for an installation program to guess where to put the files.

If you are a user on a shared system (Unix, NT, etc) and do not have access to the main TEX installation directory tree, you can put the files in your local (personal) tree under `texmf.local` (called `texmf-local` on some systems). Your system should be configured to look there first, so any updates to standard packages will be found there before the (old) copies in the normal `texmf` tree.

5.2 Online help

The indexes and documentation files on CTAN are the primary online resource for self-help and you should read these carefully before asking questions. You should most especially read the various FAQs (lists of Frequently-Asked Questions) so that you avoid wasting online time asking about things for which there is already an easily-accessible answer.

The Usenet newsgroup `comp.text.tex` is the principal forum for questions and answers about LATEX. Feel free to ask questions, but please do not ask FAQs: read the documentation instead. People who answer the questions do so voluntarily, unpaid and in their own time, so don't treat this as a commercial support service. To access Usenet news, type the following URL into your browser's 'Location' or 'Address' window:

```
news:comp.text.tex
```

Questions can also be asked of the TEX Users Group, but this is normally restricted to members. Institutional members (including UCC) are expected to provide local expertise. In UCC's case, you can call or email the Computer Centre HelpDesk (×567, `helpdesk@ucc.ie`) or contact the Electronic Publishing Unit (×2609, `elecpub@ucc.ie`).

If you need commercial support, buy one of the many commercial versions of TEX, or contact a consultancy which deals with TEX (details on the TUG Web site). A local consultancy which handles LATEX is Silmaril Consultants (`http://www.silmaril.ie`).

SESSION VI

Other document structures

Sections and paragraphs are not the only things you need to write a document, although it's undoubtedly possible to do so.

This session covers the most common features: lists, tables, figures (including images), verbatim text (program listings), and sidebars like boxes and panels.

6.1 Lists

Lists are useful tools for arranging thoughts in a digestible format, usually a small piece of information at a time.

There are four basic types of list:

Random or arbitrary lists (sometimes called 'itemized' or 'bulleted' lists) where the order of items is irrelevant or unimportant. The items are often prefixed with a bullet or other symbol for clarity or decoration, but are sometimes simply left blank, looking like miniature paragraphs.	**Enumerated or sequential lists** where the order of items is critical, such as sequences of instructions or rankings of importance. The enumeration can be numeric (Arabic or Roman), or lettered (uppercase or lowercase), and can also be independent (1, 2, 3, etc) or hierarchical (1.a.viii, 2.3.6, etc).
Descriptive or labelled lists (sometimes called 'discussion' lists), which are composed of unnumbered subheadings or topic labels each followed by one or more indented paragraphs of discussion or explanation.	**Inline lists,** which are sequential in nature but are *a*) formatted within their paragraph, and *b*) labelled with letters, like this example. The items are often Boolean, with the final item prefixed by 'and' or 'or'.

The structure of lists in LaTeX is identical for each type, but with a different keyword. Lists are another example of LaTeX environments, the pair of matched commands surrounding some text which needs special treatment.

Within a list environment, list items are always identified by the command **\item** (optionally followed by an item label). Normally you don't type the bullet or the number, it's all automated.

6.1.1 Itemized lists

To create an itemized list, use the the environment itemize:

```
\begin{itemize}

\item Itemized lists usually have a bullet;

\item Long items use 'hanging indentation', whereby the text
is wrapped with a margin which brings it clear of the bullet
used in the first line of each item;

\item The bullet can be changed for any other symbol, for
example from the \textsf{bbding} or \textsf{pifont} package.

\end{itemize}
```

- ✏s Itemized lists usually have a bullet;
- ✏s Long items use 'hanging indentation', whereby the text is wrapped with a margin which brings it clear of the bullet used in the first line of each item;
- ✏s The bullet can be changed for any other symbol, for example from the bbding or pifont package.

See session 9 for details of how to change the settings for list item bullets.

6.1.2 Enumerated lists

To create an enumerated list, use the environment enumerate:

```
\begin{enumerate}

\item Enumerated lists use numbering on each item;
```

```
\item Long items use hanging indentation just the same as for
itemized lists;

\item The numbering system can be changed for any level.

\end{enumerate}
```

1. Enumerated lists use numbering on each item;
2. Long items use 'hanging indentation', just the same as for itemized lists;
3. The numbering system can be changed for any level.

6.1.3 Description lists

To create a description list, use the environment description:

```
\begin{description}

\item[Identification:] description lists require a topic for
each item given in square brackets;

\item[Hanging indentation:] Long items use this in the same
way as all lists;

\item[Reformatting:] Long topics can be reprogrammed to fold
onto multiple lines.

\end{description}
```

Identification: description lists require a topic for each item given in square brackets;

Hanging indentation: Long items use this in the same way as all lists;

Reformatting: Long topics can be reprogrammed to fold onto multiple lines.

6.1.4 Inline lists

Inline lists are a special case as they require the use of the paralist package and the inparaenum environment (with an optional formatting specification in square brackets):

```
\usepackage{paralist}
...
\textbf{Inline lists}, which are sequential in nature but are
\begin{inparaenum}[\itshape a\upshape)]
\item formatted within their paragraph, and
\item labelled with letters,
\end{inparaenum}
like this example. The items are often Boolean, with the final
item prefixed by ‘and’ or ‘or’.
```

Inline lists, which are sequential in nature but are *a*) formatted within their paragraph, and *b*) labelled with letters, like this example. The items are often Boolean, with the final item prefixed by ‘and’ or ‘or’.

6.1.5 Lists within lists

You can start a new list environment within the item of an existing list, so you can embed one list inside another up to four deep. The lists can be of any type, so you can have a description list containing an item in which there is a numbered sub-list, within which there is an item containing a bulleted sub-sub-list.

Multiply-embedded lists change bullet or numbering scheme so that the levels don't get confused: an outer enumerated list is numbered in Arabic; an embedded enumerated list is lettered in lowercase, and so on, using Roman numerals and uppercase letters.

If you feel you need more than this, you're probably in need of sections and subsections, rather than lists.

Treat lists with care: people sometimes use tables for information which is really a list and would be better handled as such. They often do this because their wordprocessor has no way to do what they want *except* by using a table, hence they are misled into believing that their text is essentially tabular when it's not.

6.2 Tables

Tabular typesetting is the most complex and time-consuming of all textual features to get right. This holds true whether you are typing in plaintext form, using a wordprocessor, using LaTeX, using HTML or XML, using a DTP system, or some other text-handling package.

Terminology: LaTeX, in common with standard typesetters' practice, uses the word 'Table' to mean a formal textual feature, numbered and with a caption, referred to from the text (as in 'See Table 5').

The arrangement of information in rows and columns *within* a Table is called a 'tabulation' or 'tabular matter'.

It is critical to keep this distinction firmly in mind for this section.

6.2.1 Formal Tables

Tables and Figures are what printers refer to as 'floats'. This means they are not part of the normal stream of text, but separate entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the designer specifies). They always have a caption describing them and they are always numbered so they can be referred to from more than one place in the text.

LaTeX automatically floats Tables and Figures, depending on how much space is left on the page at the point that they are processed. If there is not enough room on the current page, the float is moved to the top of the next page. This can be changed by moving the Table or Figure definition to an earlier or later point in the text, and by adjusting some of the parameters which control automatic floating.

Authors sometimes try to have too many floats occur too soon after one another, without any thought for how they are supposed to fit on the page and still leave room for text. In this case, LaTeX stacks them all up and prints them together if possible, or leaves them to the end of the chapter in protest. The skill is to space them out within your text so that they intrude neither on the thread of your argument or discussion, nor on the visual balance of the typeset pages. But this is a skill few authors have, and it's one point at which professional typographic advice may be needed.

To create a Table, use the table environment, and add the **`\caption`** command to provide the caption.

```
\begin{table}
\caption{Project expenditure to year-end 2001}
\label{ye2001exp}
...
\end{table}
```

Numbering is automatic, and includes the chapter number in document classes where this is appropriate.

The **`\label`** command is for automated crossreferencing, as in 'see Table 6.1 on page 38', which we will look at in the next Session. If used, this command must *follow* the **`\caption`** command.

6.2.2 Tabular matter

Within a Table, you can either typeset the tabular matter using LaTeX, or include a table captured as an image from elsewhere. We will see how to include images in the next section on Figures, where they are more common.

To typeset tabular matter in LaTeX, use the tabular environment. The command `\begin{tabular}` must be followed by a second argument in curly braces giving the alignment of the columns. These are specified as a combination of the single letters `l`, `c`, and `r`, or the letter `p` followed by a width argument:

`l` left-aligned column, single line per cell;

`c` centred column, single line per cell;

`r` right-aligned column, single line per cell;

`p{...}` justified multi-line paragraph per cell.

TeX's original tabular settings were designed for classical numerical tabulations, where each cell contains a single value. The `p` specification allows multi-line cells like miniature paragraphs, but the width has to be specified in advance. Auto-adjusting cell sizes are possible, but these are often inelegant in print, however convenient they may be in a HTML browser. The array package provides for left, right, and centred multi-line columns with many other typographic variations. Other packages provide decimal-aligned columns, row-spanning, and multi-page tables.

A tabular setting with three columns, the first one centered, the second left-aligned, and the third one right-aligned, would be specified as `{clr}`, as in the example below. Note the use of indentation to make the elements of the table clear, and how it has no effect on the formatting.[1]

```
\begin{table}
  \caption{Project expenditure to year-end 2001}
  \label{ye2001exp}
  \begin{center}
    \begin{tabular}{clr}
      &Item&Amount\\
    \hline
      a)&Salaries (2 research assistants)&28,000\\
      &Conference fees and travel expenses&14,228\\
      &Computer equipment (5 workstations)&17,493\\
      &Software&3,562\\
      b)&Rent, light, heat, power, etc&1,500\\\cline{3-3}
      &Total&64,783
    \end{tabular}
    \par\medskip\footnotesize
The Institute also contributes to (a) and (b).
  \end{center}
\end{table}
```

It is conventional to centre the tabular setting within the Table, using the center environment (note US spelling). The entries for each cell are separated by an ampersand character (`&`) and the end of a row is shown by the double-backslash (`\\`):

The **`\hline`** command draws a rule across all columns and the `\cline{`*x*`-`*y*`}` draws

There are some extra formatting commands after the tabular material in the example. These are explained in Session 8.

[1] In fact, you could visually line up the & column separators with spaces if you wanted to keep the data clear for editing. LaTeX doesn't mind, so long as you get the syntax of the tabular setting right.

Table 6.1: Project expenditure to year-end 2001

	Item	Amount
a)	Salaries (2 research assistants)	28,000
	Conference fees and travel expenses	14,228
	Computer equipment (5 workstations)	17,493
	Software	3,562
b)	Rent, light, heat, power, etc	1,500
	Total	64,783

The Institute also contributes to (a) and (b).

a rule from column *x* to column *y*. If used, both these commands *follow* the \\ of the previous row. You do not need to format the tabular data: LaTeX does this for you, using the column specifications. Extra space is automatically added between columns.

If there is no data for a cell, just don't type anything — but you still need the & separating it from the next column's data. The astute reader will already have deduced that for a table of n columns, there must always be $n-1$ ampersands in each row.[2]

There is much more to tabular setting for which there is no space here. Full details are in the manuals mentioned on page 3. Note that because the table is labelled, it can now be referenced from anywhere in the document as Table 6.1 just by using `\ref{ye2001exp}`.

6.3 Figures

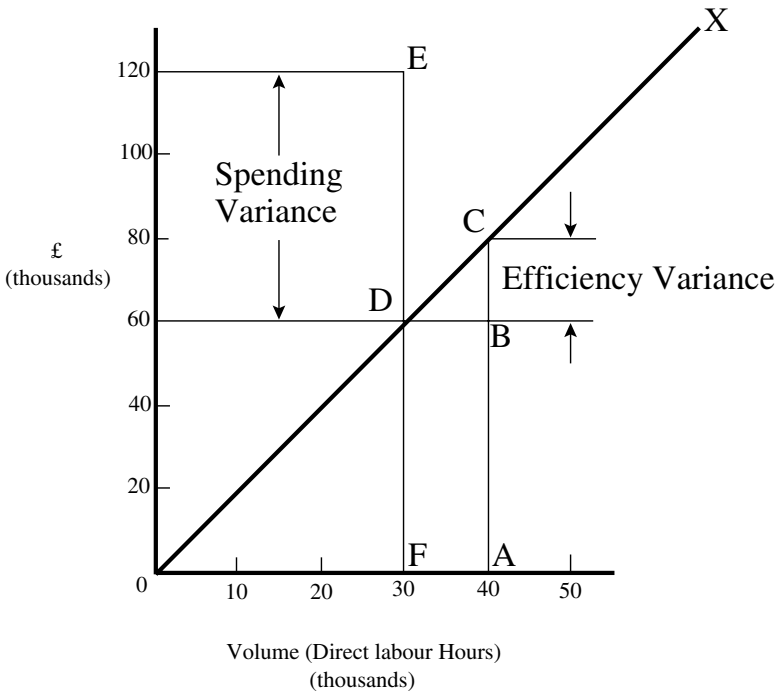
As explained on page 36, Figures and Tables float to a vacant part of the page, as they are not part of the sequence of sentences making up your document.

Figures can contain text, diagrams, pictures, or any other kind of illustration. Like Tables, they automatically get numbered, and must include a caption. To create a figure, use the figure environment:

```
\begin{figure}
\caption{Work efficiency of expenditure}
\label{workeff}
\begin{center}
\includegraphics[width=.75\columnwidth]{diagram}
\end{center}
\end{figure}
```

[2]Not absolutely true, as the `\multicolumn` command can be used to create cells which span multiple rows. There is also a package to enable cells to span multiple rows.

Figure 6.1: Work efficiency of expenditure (after Heffen [1984])



You can see that the structure is very similar to the tables environment, but in this case we have a graphic included. Details of this command (`\includegraphics`) are in the next section.

The content of the Figure could of course also be textual, in the form of a list or a text diagram. LaTeX does have a rather simple drawing feature, which lets you create a limited set of lines and curves (PICTeX), but for any diagram of complexity, it is *strongly* recommended that you use a vector drawing program like *tkpaint* or *Corel Draw*, as these allow the file to be saved as Encapsulated PostScript (EPS) or PDF, which means they can be scaled arbitrarily to any size with no loss of resolution.

Never, ever (except in the direst necessity) save a diagram as a bitmap (BMP, GIF, JPG, etc) as these become blurred and jagged when scaled.

6.3.1 Images

Images (graphics) can be included anywhere in a LaTeX document, although in most cases they will occur in Figures (see preceding section). To use graphics, you need to use the graphicx package in your preamble:

```
\usepackage{graphicx}
```

This enables the command `\includegraphics` which is used to insert an image where you put it. The command is followed by the name of your graphics file *without the filetype*, for example:

```
\includegraphics{myhouse}
```

For standard LaTeX, graphics files *must* be in Encapsulated PostScript (`.eps`) format: this is the publishing industry standard for portable graphics, and no other format is acceptable in standard LaTeX.

All good graphics packages can save images as EPS, but be very careful because some packages, especially on the Micosoft *Windows* platform, use very poor quality drivers which create very poor quality EPS files. If in doubt, check with an expert.

For *pdflatex*, graphics files can be in JPG, PNG, or PDF format, *not* EPS. This means if you want to use both standard LaTeX as well as *pdflatex*, you need to keep your graphics in two formats, EPS and one of the others.[3]

Both standard LaTeX and *pdflatex* know what file types they can accept and will search accordingly. This is the reason why it is desirable to omit the file type from the end of your graphics filename in the **`\includegraphics`** command, so your document can be processed without error by either program.

The **`\includegraphics`** command can take optional arguments within square brackets before the filename, eg

```
\includegraphics[width=3in]{myhouse}
```

For details of these, see the documentation on the graphicx package or a copy of the *LaTeX Graphics Companion*. This package also include commands to rotate, mirror, and **scale** text.

6.4 Verbatim text

If you are documenting computer procedures, you probably need fixed-width type for examples of programming or data input or output.

Standard LaTeX includes two features for this, and there are dozens more available in packages.

To specify a word or phrase as verbatim text in typewriter type within a sentence, use the special command **`\verb`**, followed by your piece of text surrounded by any suitable character which does *not* occur in the text itself. I often use the plus sign for this, for example to show a LaTeX command, I type

```
\verb+\includegraphics[width=3in]{myhouse}+
```

This command has the advantage that it turns off all special characters (see page 9) except the one you are using as the delimiter, so you can easily quote sequences of characters in any computer syntax without problems. However, LaTeX will never break the argument of **`\verb`** at a line-end when formatting a paragraph, so if it happens to be long, and falls towards the end of a line, it will stick out into the margin.

[3] Actually, it's possible to tell LaTeX to generate the right format by itself, but this requires an external command-line graphics converter, and as it gets done afresh each time, it slows things down rather a lot.

The url package avoids this with by providing the command **\url** which works in the same way as **\verb** but performs a hyphenless break at punctuation characters, as in `http://www.ucc.ie/doc/ucc/siteowner.xml`. It was designed for Web URLs, so it understands their syntax and will never break mid-way through an unpunctuated word. Bear in mind, however, that URLs are forbidden by their specifiation to include spaces, so using spaces in **\url** will fail.

For longer chunks of fixed-format text, use the verbatim environment:

```
\begin{verbatim}
\documentclass[11pt,a4paper,oneside]{report}
\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2001}
\maketitle

\end{document}
\end{verbatim}
```

Like **\verb**, this turns off all special characters, so you can include anything at all in the verbatim text except the exact line

`\end{verbatim}`

For more control over formatting, however, I recommend the use of the fancyvrb package, which adds an environment Verbatim (note the capital letter) which lets you draw a rule round the verbatim text, change the font size, and even have typographic tricks inside the verbatim environment.

6.5 Boxes, sidebars and panels

LaTeX has a simple command for putting a box round some text:

```
\fbox{box round}
```

This works for a few words, but the box won't break over the end of a line. To typeset multiline text in a box, put it in a tabular setting:

```
\fbox{\begin{tabular}{p{3cm}}
Multiline text in a box typeset using \textsf{tabular}
\end{tabular}}
```

Multiline text in a box typeset using tabular

You can also use the minipage environment, which gives you a typeset box of text of any size (in which you have all the normal document objects) and enclose that in a box:

```
\fbox{\begin{minipage}{3in}
This multiline text is more flexible than a tabular setting:
\begin{itemize}
\item it can contain any type of normal \LaTeX{} typesetting;
\item it can be any specified width;
\item it can even have its own footnotes\footnote{Like this}.
\end{itemize}
\end{minipage}}
```

This multiline text is more flexible than a tabular setting:

- it can contain any type of normal LaTeX typesetting;
- it can be any specified width;
- it can even have its own footnotes[a].

[a] Like this

The spacing between text and box is controlled by the value of `\fboxsep`, and the thickness of the line by `\fboxrule`. The following values were used above:

```
\setlength{\fboxsep}{1em}
\setlength{\fboxrule}{2pt}
```

The fancybox package lets you surround text in square, round-corner, and drop-shadow boxes, so you can create panels of any size or shape anywhere you want, using the minipage environment to capture the text and format it to the right size before putting the box round it.

SESSION VII

Textual tools

Every text-handling system supports a repertoire of tools for doing things with text, and LaTeX is no exception.

7.1 Footnotes

The command **\footnote**, followed by the text of the footnote in curly braces, will produce an auto-numbered footnote anywhere you type it.[1] The number is reset to 1 at the start of each chapter (but you can override this).

Because of the way LaTeX reads ahead, you can't use **\verb** in footnotes (use **\url** instead, especially for Web addresses!).

Footnotes inside minipages produce lettered notes instead of numbered ones, and they get printed at the bottom of the minipage, not the bottom of the page (but this too can be changed).

There are packages to hold over your footnotes and make them print at the end of the chapter instead (endnotes) or at the end of the whole document, and there is a package to print many short footnotes in several columns so they take up less space.

There are also ways to refer more than once to the same footnote, and to defer the positioning of the footnote if it occurs in a float like a Table or Figure.

7.2 Marginal notes

If you use the geometry package to allocate enough space for a decent margin, you can add marginal notes. There are several packages to help with this, but the simplest way is to add this new command to your preamble:

Like this.

[1] Like this

```
\newcommand{\marginal}[1]{\leavevmode\marginpar{\tiny\raggedright#1}}
```

and use `\marginal{Some text}` where you need it. Be careful, however, because marginal notes are aligned with the line where the command starts, so a very long one followed too closely by another will cause LaTeX to try and adjust the position so they don't overlap. See section 9 for more information about making up your own commands.

Some text

7.3 Cross-references

This is one of the most powerful features of LaTeX. You can label any point in a document with the command **`\label`** followed by a short name you make up in curly braces:

This section is labelled 'xrefs'.

```
\label{newresearch}
```

You can then refer to this point from anywhere in the same document with the command **`\ref`** followed by the same name, eg

```
\ref{newresearch}
```

If the label was placed in normal text, the reference will provide the chapter and section number. If the label was inside a Table or Figure, the reference provides the Table number or Figure number. A label in an enumerated list will provide a reference to the item number.

Thus I can refer to `\ref{xrefs}` and get the value 7.3.

The command `\pageref{...}` using any of your label values will provide the page number where the label occurred.

Because LaTeX records the label values each time the document is processed, they get picked up the *next* time the document is processed, so you always need to process the document one extra time when you are finished, before printing or viewing it, to make sure the cross-references are correctly resolved.

Unresolved references cause a warning message at the end of the log file.

7.4 Bibliographic work

Exactly the same mechanism is used for references to reading lists and bibliographies. Although it is possible to type these manually, there is a companion program to LaTeX called BIBTeX, which manages bibliographic references automatically.

It works exactly the same as many other bibliographic databases: you keep details of every document you want to refer to in a separate file, using BIBTeX's format (see below). Many writers make a habit of adding the details of every book and article they read, so that when they write, these entries are always available. You give each entry a short label, and it is this label you use to refer to in your own documents when you cite the work:

`\cite{...}`

By default, this creates a cross-reference in square brackets [1], and the details of every document you cite will be extracted from your database, formatted according to the specification you supply, and listed at the end of your document with a number corresponding to the number printed at the point of citation.

The specifications (BIBTEX styles) are based on standard settings for journals and books from dozens of publishers: you pick the one you want.

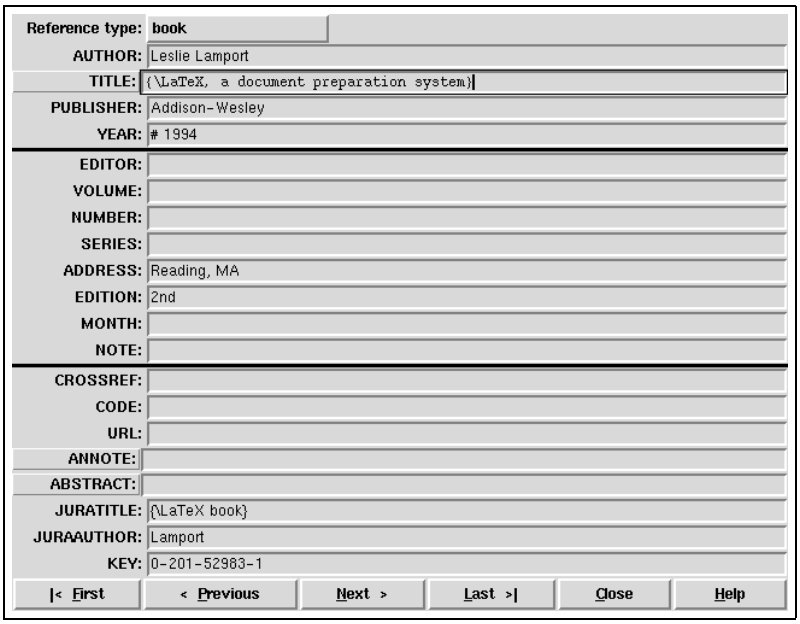
To make this work, just enter the document details in the format specified in the BIBTEX documentation into a file ending with `.bib`, for example:

```
@book{latexbook,
   title        = {{\LaTeX, a document preparation system}},
   author       = {Leslie Lamport},
   edition      = {2nd},
   publisher    = {Addison-Wesley},
   year         = {1994},
   address      = {Reading, MA},
   key          = {0-201-52983-1}
}
```

There is a prescribed set of fields for each of a dozen or so types of document: book, article, thesis, report, etc. Each entry identifies the document type after the '@' sign, followed by the entry label, and then each field in the form

`keyword = {value},`

omitting the comma only after the last field. Titles may get auto-capitalised: to prevent this, enclose the title in double curly braces. Fields can occur in any order but the format must be *strictly* observed. To help with this, there are several interfaces to creating and maintaining BIBTEX files, such as *tkbibtex*, which runs on most platforms.



This mechanism lets you maintain an unlimited number of entries, and refer to them by their labels from anywhere in any document, using the **\cite** command.

To print the bibliographic details, add two lines at the end of your document (or wherever you want it printed):

```
\bibliography{mybib}
\bibliographystyle{ieeetr}
```

The **\bibliography** command is followed by the filename of your `.bib` file *without* with `.bib` extension. The **\bibliographystyle** command is followed by the name of any of LaTeX's supported bibliography styles, of which there are many dozen: `plain` and `alpha` are two common generic styles.

The style shown in the example here provides formatting according to the specifications for Transactions of the IEEE.

After processing your file with LaTeX, run BIBTeXon it by using the BIBTeX toolbar icon (*WinEdt*) or the `BIBTeX File` menu entry (*Emacs*). This extracts the details and formats them. Then run LaTeX again to resolve the references, and finally run LaTeX again to create the correctly cross-referenced list.

In practice, authors tend to run LaTeX from time to time during writing, so they can check the formatting in a previewer. Just run BIBTeX after adding a new **\cite** command, and subsequent runs of LaTeX will incrementally incorporate all references. You only need to follow the formal sequence in the preceding paragraph when you have finished writing and want to check that all references have been resolved.

See the `texmf/bib/bst` subdirectory of your installation for the names of other installed formats, or search on CTAN for others (`.bst` files). Always read the documentation, because most of the formats are very specific to a journal, and have fairly absolute requirements. It is possible (but quite difficult) to write your own `.bst` file, but rarely needed.

The method of citing a work by numeric reference is common in the Natural Sciences but is unheard-of in the Humanities and Law. In these fields, citations produce short references (author/short-title/year) in a numbered footnote, and the bibliography at the back of your document is called 'References' and is printed unnumbered in alphabetic order of author. The jurabib package (originally intended for German law articles) has extensive features for doing this style of citation.

The big advantage of using LaTeX and BIBTeX in this way is that your document text remains unchanged but you can vary the formatting of the bibliography for different publications simply by changing the style name.

7.5 Indexes and glossaries

LaTeX has a powerful, automated indexing facility which uses the standard *makeindex* program. To use indexing, use the package makeidx and include the command **\makeindex** in your preamble:

```
\usepackage{makeidx}
\makeindex
```

When you want to index something, using the command **\index** followed by the entry as you want it to appear in the index, using one of the following formats:

Plain entry: `\index{beer}` will create an entry for 'beer' with the current page number.

Subindex entry: `\index{beer!lite}`. Subsubentries also work: `\index{beer!lite!American}`.

Cross-references: ('see' entries) are done with `\index{Microbrew|see{beer}}`

Font changes: to change the typographic style of an entry, use the format `\index{bud@\textit{Budweiser}}` (any of the standard font-change commands work here: see section 8 for details).

Out of sequence: the same method can be used as for font changes: `\index{beer@bud}` will place an entry for 'bud' as if it was spelled 'beer'.

When the document has been processed, run the *makeindex* program on your document, eg by typing

```
makeindex mythesis
```

This will look for a `.idx` file and output a `.ind` file which can then be used with the command `\printindex` at the end of your document. The default index format is two columns.

Glossaries are done in a similar manner using the command **\makeglossary** in the preamble and the command **\glossary** in the same way as **\index**. There are some subtle differences in the way glossaries are handled: see the books by Lamport and by Goossens *et al*..

7.6 Multiple columns

LaTeX has built-in support for two-column typesetting via a `twocolumn` option in all the standard Document Class Declaration, but this is relatively inflexible. A much better solution is the multicol package, which will set up to 10 columns, and allows the number of columns to be changed or reset to one in mid-page, so that full-width graphics can be used. The package provides the multicols environment and you follow the `\begin{multicols}` command with the number of columns needed.

```
\usepackage{multicol}
...
\begin{multicols}{3}
...
\end{multicols}
```

SESSION VIII

Typographic considerations

This is the session that most users want first, because they come to structured documents from a wordprocessing environment where the *only* way to convey a change of information is to fiddle with the font and size drop-down menus.

As I hope we have seen, this is usually unnecessary in LaTeX, which does most of the hard work for you. However, there are occasions where you want to make typographic changes, and this session is about how to do them.

8.1 Spacing

We mentioned in section 7.2 the existence of the geometry which lets you change margins and the text-area height and width (read the package documentation for details). The spacing around the standard textual components (headings, paragraphs, lists, footnotes, etc) can also be changed at will, as we saw with paragraph spacing and indentation in section 3.2.5.

Changing the spacing around section headings is a little more complex, as it involves adjusting several parameters including those responsible for detecting and avoiding the abuse of spare space at the bottom of pages. The best way to do this is to use one of the packages designed to let you adjust section-head spacing without having to rewrite the internal LaTeX coding yourself, such as sectsty. The same applies to the spacing for lists, which is also complex (see package mdwlist). In both cases the user with specific requirements should read the relevant sections in the *The LaTeX Companion* or ask an expert.

Manual or single-time spacing changes are easy, however. For vertical space, there are three preset commands **`\smallskip`**, **`\medskip`**, and **`\bigskip`** which output flexible (dynamic, or 'rubber') space, approximately 3pt, 6pt, and 12pt high, but which will automatically compress or expand a little, depending on the demands of the rest of the page (for example to allow one extra line to fit, or a heading to be moved to the next page without anyone except a typographer noticing the change). These

commands can only be used after a paragraph break (a blank line or the command **\par**).

For a fixed-height space, use the command **\vspace** followed by a length in curly braces, eg `\vspace{18pt}`. Bear in mind that extra space which ends up falling at a page-break gets discarded entirely so that the bottom and top lines fall in the correct places. To force a space to remain even in this position (rare), use the 'star' variant: **\vspace***`{18pt}`.

There is a horizontal equivalent, **\hspace**, which works in the same way, so I can force a 1″ space like this in mid-paragraph. There are also some predefined (shorter) spaces available: **\thinspace** (1/12 em), **\enspace** (1/2 em), **\quad** (1em), and **\qquad** (2em).

Double-spacing lines is much frowned upon, as it is incredibly ugly. It is still, sadly, a requirement in some unenlightened universities for thesis submission, a historical relic from the days of typewriters. Nowadays, 1⅓ or 1½ of a line space is considered acceptable. If your institution still requires double line spacing, show them this paragraph and explain that they need to enter the 21st century and adapt to the features of automated typesetting. If they still insist, either change institution or use the setspace package and be prepared for some suboptimal output.

The space between lines is defined by the length of **\baselineskip** multiplied by the value of **\baselinestretch**. In general, don't meddle with these unless you know what you are doing, but they can be used as values in commands like `\vspace*{\baselineskip}`.

The value of **\baselineskip** changes with the font size (see section 8.3.1) but is usually 1.2 times the current font size. This is a value derived from long experience: only change it if you understand what it means and what effect it will have.

8.2 Using fonts

The default typeface in LaTeX is Computer Modern rather than Times. Computer Modern (CM) was designed by Knuth to go with TeX in the days before desktop publishing, and because it is one of the very few book typefaces with a comprehensive set of fonts, including a full suite of mathematics, it has remained the default. CM is based on a 19th-century book typeface from Monotype, which is why it looks a little like a school book. This paragraph is set in Computer Modern so you can see what it looks like. CM is written in METAFONT, a font-drawing program written by Knuth to accompany TeX.

(The rest of this document is set in Bitstream Aldine 721 (Bembo), with Bitstream Humanist 521 (Gill Sans) for some of the headings and Courier for the fixed-width type, but apart from Courier, these are commercial fonts: you have to buy them.)

Typeface (font family)	Example	Package	Family
	Metafont fonts		
⋆ Computer Modern	The quick brown fox jumped over the lazy dog	(default)	`cmr`
Computer Modern Sans	The quick brown fox jumped over the lazy dog	(default)	`cmss`
Computer Modern Typewriter	The quick brown fox jumped over the lazy dog	(default)	`cmtt`
Pandora	The quick brown fox jumped over the lazy dog	pandora	`panr`
Pandora Sans	The quick brown fox jumped over the lazy dog	”	`pss`
Universal	The quick brown fox jumped over the lazy dog	—	`uni`
⋆ Concrete	The quick brown fox jumped over the lazy dog	concrete	`ccr`
BB Dingbats	[illegible]	—	`ding`[c]
Cypriot	[illegible]	—	`cypr`
Éıreannach	Níl aon tinteán mar do tinteán féin	eiad	`eiad`
Etruscan	[illegible]	—	`etr`
Linear ‘B’	[illegible]	—	`linb`
Phoenician	[illegible]	—	`phnc`
Runic	[illegible]	—	`fut`
Uncial	The quick brown fox jumped over the lazy dog	—	`uncl`
Dürer	THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG	—	`zdu`
Yannis Fraktur	The quick brown fox jumped over the lazy dog	—	`yfrak`[c]
Yannis Gothic	The quick brown fox jumped over the lazy dog	—	`ygoth`[c]
Yannis Schwäbische	The quick brown fox jumped over the lazy dog	—	`yswab`[c]
	Adobe fonts		
Bookman	The quick brown fox jumped over the lazy dog	bookman	`pbk`
New Century Schoolbook	The quick brown fox jumped over the lazy dog	newcent[a]	`pnc`
Palatino	The quick brown fox jumped over the lazy dog	palatino[b]	`ppl`
† Times New Roman	The quick brown fox jumped over the lazy dog	times[b]	`ptm`
Avant Garde	The quick brown fox jumped over the lazy dog	avant	`pag`
Helvetica	The quick brown fox jumped over the lazy dog	helvet	`phv`
Zapf Chancery	*The quick brown fox jumped over the lazy dog*	zapfchan	`pzc`
Courier	The quick brown fox jumped over the lazy dog	courier	`pcr`
Zapf Dingbats	[illegible]	pifont	`pzd`[c]
Symbol	Τηε θυιχκ βροων φοξ φυμπεδ οϖερ τηε λαζψ δογ	—	`pzd`[c]
	X Consortium fonts		
Charter	The quick brown fox jumped over the lazy dog	charter	`bch`
Nimbus Roman	The quick brown fox jumped over the lazy dog	—	`unm`
Nimbus Sans	The quick brown fox jumped over the lazy dog	—	`unms`
URW Antiqua	The quick brown fox jumped over the lazy dog	—	`uaq`
URW Grotesk	**The quick brown fox jumped over the lazy dog**	—	`ugq`
Utopia	The quick brown fox jumped over the lazy dog	utopia	`put`

⋆ Full selection of mathematical symbols available.
† Mathematical symbols are a commercial add-on.
a Automatically adds Avant Garde as the sans-serif font and Courier as the monospace font.
b Automatically adds Helvetica as the sans-serif font and Courier as the monospace font.
c Uses specialised font encoding ‘U’.

In addition to CM, there are many other METAFONT fonts which can be downloaded from the CTAN, including a large collection of historical and non-Latin fonts. LaTeX comes with the complete 'Adobe 35' fonts which are built into every laser printer, so you have the same base set as other DTP systems. There are some fonts (in PostScript Type 1 format) donated by the X Consortium which match those distributed free with the X Window system.

Standard LaTeX uses only METAFONT and PostScript Type 1 fonts. To use True-Type fonts as well, you must use *pdflatex* instead.

8.2.1 Changing the default font family

LaTeX expects to work with three font families as defaults:

Font family	Code
Roman (serif, with tails on the uprights), the default	`rm`
Sans-serif, with no tails on the uprights	`sf`
`Monospace (fixed-width or typewriter)`	`tt`

If you `\usepackage` one of the package names listed in the table on page 50, it will replace the default of the same type. For example, `\usepackage{bookman}` makes the default Roman font Bookman but leaves the sans-serif and monospace fonts untouched. Equally, `\usepackage{helvet}` changes the default sans-serif font to Helvetica but leaves the serif (Roman) and monospace fonts untouched. Using both commands will change both defaults because they operate independently.

As it is common to want to change all three defaults at the same time, some of the most common 'suites' of typefaces are provided as packages:

times changes to Times/Helvetica/Courier

pslatex same as times but uses a specially narrowed Courier to save space (normal Courier is rather wide). This should be the preferred setting if you want Times.

newcent changes to New Century Schoolbook/Avant Garde/Courier

palatino changes to Palatino/Helvetica/Courier

palatcm changes to Palatino alone with CM mathematics

Where no package name is given on page 50, it means the font is rarely used as a default by itself except in special cases like users' own homebrew packages.

8.2.2 Changing the font family temporarily

To shift to another font family on a temporary basis, enclose the text in curly braces, and use the commands

```
\fontfamily{...}\selectfont
```

immediately inside the opening curly brace, replacing the '...' with the font family name code, eg

```
Helvetica looks like {\fontfamily{phv}\selectfont this}.
```

> This use of curly braces is slightly different from their use to delimit the argument to a command. This is called 'grouping' and it restricts the effect of changes made *inside* the group so that they do not interfere with the text following. Any font changes made within the curly braces cease when the closing curly brace is processed.

However, most cases where people want to do this involve things like special symbols on a repetitive basis, and LaTeX provides much easier programmable ways to make this into a shorthand command (macro: see section 9).

8.3 Changing font style

Within each typeface (font family) there are usually several different styles of type. LaTeX distinguishes between *family*, *shape*, and *series*:

Type style	Command	Example
Upright (default)	`\upshape`*	The quick brown fox jumped over the lazy dog
Italic	`\itshape`	*The quick brown fox jumped over the lazy dog*
Slanted	`\slshape`*	*The quick brown fox jumped over the lazy dog*
Small Capitals	`\scshape`*	THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG
Bold	`\bfseries`	**The quick brown fox jumped over the lazy dog**
Bold Extended	`\bfseries`*	**The quick brown fox jumped over the lazy dog**
Sans-serif	`\sffamily`	The quick brown fox jumped over the lazy dog
Monospace	`\ttfamily`	`The quick brown fox jumped over the lazy dog`

* Not all typefaces have all variants! Some only have bold and italics.

Shapes and series and family are commutative, so you can combine a shape with a series and/or a family, as in:

```
...{\bfseries\itshape\sffamily bold italic sans-serif type}...
```

bold italic sans-serif type, but it is not normally meaningful (and usually poor typographic practice) to combine one shape or series with another, and an impossibility to combine one family with another. Slanted plus italics, for example, doesn't make sense, as italics are already slanted; and while some typefaces may well possess italic small caps, they are not in common use. Sans-serif and monospace (typewriter) are different typefaces.

There is an alternative syntax for the most common type shape and series commands which uses curly braces in the normal 'argument' manner:

Type style	Command	Example
Italic	`\textit{a few words}`	puts *a few words* into italics
Slanted	`\textsl{a few words}`	puts *a few words* into slanted type
Small Capitals	`\textsc{a few words}`	puts A FEW WORDS into small caps
Bold	`\textbf{a few words}`	puts **a few words** into bold type
Sans-serif	`\textsf{a few words}`	puts a few words into sans-serif type
Monospace	`\texttt{a few words}`	puts `a few words` into typewriter type

You can nest these inside one another too:

```
...\textbf{\itshape\textsf{bold italic sans-serif type}}...
```

for ***bold italic sans-serif type*** but you must make sure the curly braces match up.

8.3.1 Font sizes

LaTeX has built into its defaults a set of predefined font sizes corresponding more or less to the traditional sizes available to metal typesetters. This is deliberate, as these sizes have grown up over 500 years as those which go best together for book printing, which is where TeX originated.

These sizes are reflected in the sizes at which Computer modern was designed (it may come as a shock to new users that the best fonts are not designed as a single pattern and just scaled up or down, but specially designed at different sizes to make them more legible). Here's 11pt Computer Modern and here's 5pt Computer Modern scaled up to 11pt. and here's 17pt Computer Modern scaled down to 11pt. In general, you probably don't want to go scaling fonts too much beyond their design size because they will start to look odd.

The default sizes (and the commands that operate them) are:

Command	Example	Approximate point size *
`\tiny`	The quick brown fox jumped over the lazy dog	5
`\scriptsize`	The quick brown fox jumped over the lazy dog	7
`\footnotesize`	The quick brown fox jumped over the lazy dog	8
`\small`	The quick brown fox jumped over the lazy dog	9
`\normalsize`	The quick brown fox jumped over the lazy dog	10
`\large`	The quick brown fox jumped over the lazy dog	12
`\Large`	The quick brown fox jumped over the l	14
`\LARGE`	The quick brown fox jumped over t	18
`\huge`	The quick brown fox jumped	20
`\Huge`	The quick brown fox jum	24

* based on the 10pt default document font (see page 17). Sizes are scaled up for 11pt and 12pt defaults.

While this to some extent relieves the novice of having to worry about the 'right' size for a given task, when you need a specific size there is the **`\fontsize`** command, which works the same way as **`\fontfamily`** (page 53):

```
\fontsize{22}{28}\selectfont
```

This takes two arguments: the point size and the baseline. The above example gives you 22pt type on a 28pt baseline (ie with 6pt extra space or 'leading' between lines).

Computer Modern fonts come fixed at the named sizes in the earlier table: if you need to generate them at arbitrary sizes it is possible but beyond the scope of this document.

8.3.2 Logical markup

All this playing around with fonts is very pretty but you normally only do it for a reason. Italics, for example, are used for many things:

Cause	Example
Foreign words	*ex officio*
Scientific names	*Ranunculus ficaria*
Emphasis	*must not*
Titles of documents	*The LaTeX Companion*
Product names	Corel's *WordPerfect*
Variables in maths	$E = mc^2$

Humans usually have no problem telling the difference between these meanings, because they can read and understand. Computers cannot, so it has become conventional to use command names which make the distinction explicit, even though the appearance is the same.

LaTeX has only one of these built in, **\emph**, which provides *emphasis*. It's special, however, because *when the surrounding text is already italic*, emphasis *reverts to upright type!*

```
It's special, however, because {\itshape when the surrounding
text is already italic, \emph{emphasis} reverts to upright
type!}
```

This sensitivity to logic is programmed into the meaning of `\emph` and it's not hard to make up other commands of your own which do the same, such as **\foreign** or **\product**.

Why would you bother? In a short document it's probably not important, but if you're writing a long article or report, or a book or a thesis, it makes editing hugely easier if you can control whole groups of special effects with a single command, such as italicising it, indexing it, and cross-referencing it to the glossary.

It also makes it possible to find and act on groups of meanings — such as making an index of scientific names — if they are identified with a special command; otherwise you'd spend weeks hunting manually through every **\textit** command to find the ones you wanted.

Section 9 explains how to make your own simple commands.

8.4 Colour

You can typeset anything in LaTeX in any colour you want. First, you need to add the command

```
\usepackage{color}
```

to your preamble (note the US spelling!). This makes available a default palette of primary colours, so you can use the command `\textcolor{red}{...}` to typeset some words in red. The 'in-group' colour-changing command also works:

```
...{\color{blue}some text in blue}...
```

You do of course need a colour previewer and printer to see it in colour…

You can define any shade you want by giving it a name and providing the Red-Green-Blue (RGB) or Cyan-Magenta-Yellow-Black (CMYK) colour codes expressed as decimal fractions, using the **\definecolor** command. For example, a shade given as (37,125,224) in decimal (#250FE0 in hexadecimal as used on the Web) needs to be given as

```
\definecolor{midblue}{rgb}{0.145,0.490,0.882}
```

(divide each value by 255, the maximum for each of the hues in the Red-Green-Blue colour model). The **\definecolor** command takes three arguments: the name you want to refer to the shade by; the name of the color model (here `rgb`) and the set of decimal-fraction values separated by commas. This midblue looks like this if you're reading in colour.

If you have *dvips* installed, you also get a separate 64-colour palette of predefined names (representing the colours in the big box of Crayola colouring-pencils) to save you having to remember or work out the colour values. This adds a new colour model called `named`, so if you want the Crayola colour RubineRed, you can use

```
\definecolor{myred}{named}{RubineRed}
```

which looks like this. However, to use these with *pdflatex*, you still need the colour codes, as *pdflatex* doesn't understand *dvips* colour names yet. You can look up the CMYK codes in `texmf/dvips/base/color.pro`.

The color package also provides a colour version of **\fbox** (see section 6.5) called **\colorbox**:

```
\colorbox{myred}{\color{midblue}Blue on red}
```

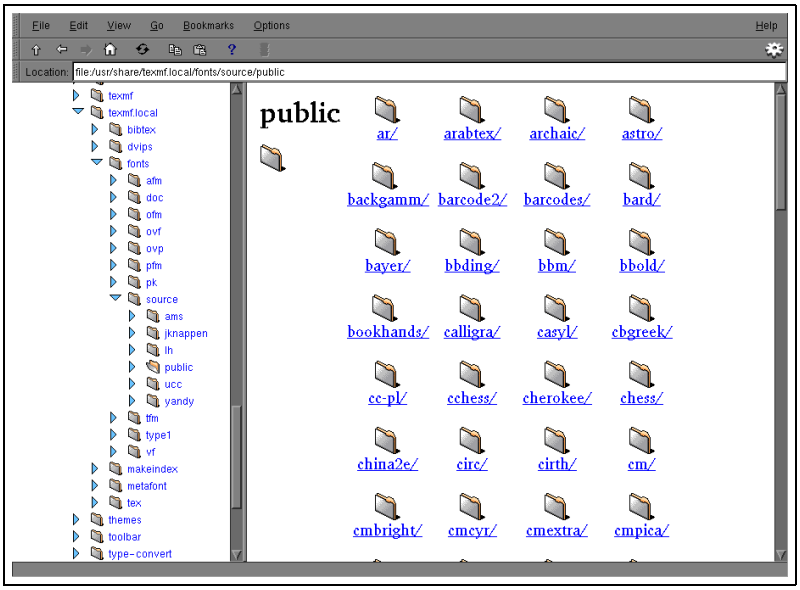
Blue on red

8.5 Installing new fonts

Different fonts come in a variety of packagings: METAFONT fonts, PostScript fonts, and TrueType fonts. How you install them and where they go depends on how you installed LaTeX. All I can deal with here is the standard locations within the TDS.

8.5.1 Installing METAFONT fonts

This is the simplest installation. When you download the fonts from the CTAN, you'll usually find a large number of `.mf` files and maybe some other types as well. Create a directory (named after the font you're installing) in `texmf.local/fonts/source/public/`:



Copy all the files to this directory. You should find among them (but sometimes in a related package) at least one `.fd` (Font Definition) file and one `.sty` (style) file. Move these to a subdirectory of `texmf.local/tex/latex/`...I use `faces` for all additional typefaces, but you could equally create a separate subdirectory for each typeface.

Unlike PostScript fonts, METAFONT fonts can be used automatically to generate the metric (`.tfm`) files on-the-fly, so there should be nothing else to install. However, some METAFONT fonts come with pre-generated `.tfm` files, so if you want to install them as well, put them in `texmf/fonts/tfm/public/` in a subdirectory named exactly the same as the one you created for the `.mf` files.

Run your TEX indexer program to update the package database. This program comes with every modern version of TEX and is variously called *texhash*, *mktexlsr*, or even *configure* or it might just be a mouse click on a menu in your editor. Read the documentation that came with your installation to find out which one it is.

This step is *utterly essential*, otherwise nothing will work.

Now you can `\usepackage{...}` (whatever the `.sty` file was called) in your LATEX file to make it the default font or enable it to be used. Read the documentation (refer to section 5.1.3, item 2) to find out what commands the package provides. The font bitmaps for display and printing will be automatically generated on demand, as you use the font.

If it came without `.fd` or `.sty` files, you'll need to find someone who can make them for you (or follow the outline in section 8.5.2, item 11).

8.5.2 Installing PostScript fonts

[some comments on fonts from CTAN]

This is adapted from a posting on `comp.text.tex`:

```
From: Peter Flynn <peter@silmaril.ie>
Newsgroups: comp.text.tex
Subject: Re: Type 1 fonts and PDFLaTeX: I just don't understand
Date: Sun, 04 Mar 2001 01:53:44 +0000
Organization: Silmaril Consultants
Message-ID: <3AA1A028.AFF4E1FB@silmaril.ie>
```

Luci Ellis wrote:

> Dear TEXers,
> I have large numbers of Type 1 (PostScript) fonts on both my Mac and my PC. I use PDFLATEX successfully within the OzTEX (Mac) and MikTEX (PC) distributions. I understand how to write/modify a style file to select different text fonts as the default roman typeface – I got Utopia working on MikTEX, for example.
> But in general I can't work out how to set up Type 1 fonts outside the standard LaserWriter-35 set + Utopia.

OK. This way has worked for me for years. WARNING — this is not the official way (that's *fontinst* but I have never managed to get it to work for me), and there is a recent problem (see end).

1. Make sure you have `.afm` and `.pfb` files for the font you want. Let's call them `foo.afm` and `foo.pfb` in this example. Put them in your temporary directory (eg `C:\tmp` or `C:\temp` on Microsoft *Windows* or `/tmp` on Linux).

2. Find out or decide on the fontname to use inside LATEX. Note this is not the font family full name (eg 'Baskerville') but the 'Karl Berry' shortened font name in the format `fnnsseec`:

Character	Meaning
`f`	foundry (eg `b`=Bitstream, `m`=Monotype etc)
`nn`	name (eg `ba`=Baskerville)
`ss`	series (eg `bi`=bold italic)
`ee`	encoding (eg `8a`=default8–bit ANSI, `ly`=Y&Y's TEX'n'ANSI)
`c`	smallcaps flag (this is a literal 'c' character)

   The `fontname` directory in a full installation of LATEX has files per foundry giving fully-formed names like these for common fonts. Read the *fontname* documentation to find out how to make up your own. In this example we'll call our font 'zork' (**Z**fonts **OR**dinary Boo**k**face [mythical]). Fontnaming conventions let you leave off the defaults.

3. Decide on your encoding. This is what tripped me up the first few times until someone pointed me at Y&Y's TEX'n'ASNI encoding which (to me) seems to be the only one that includes the glyphs I want where I want them. Your mileage may vary. Their encoding is referred to as `LY1` within LATEX and is in `texmf/dvips/base/texnansi.enc`. Copy this file to the temporary directory where you're doing all this stuff.

4. Run *afm2tfm*[1] on your `.afm` files like this:

   ```
   afm2tfm foo.afm -v zorkly.vpl -p texnansi.enc rzorkly.afm >zork.id
   ```

   This creates a virtual font encoded as LY1 with the 'raw' data in the `.afm` file (hence the `r` prefix). Many people will tell you that virtual fonts are dead and that this is the wrong way to do it, but no-one has ever shown me an alternative that works, so I stick with it.

5. If you want a small caps variant faked up (perhaps because the typeface family doesn't have a special small-caps font), repeat the medicine like this:

   ```
   afm2tfm foo.afm -V zorklyc.vpl -p texnansi.enc rzorkly.tfm >>zork.id
   ```

   Note the capital `V` option here, and yes it *does* overwrite the `rzork.tfm` created in the first command. Let it.

6. Now turn the `.vpl` files into `.vf` and `.tfm` pairs:

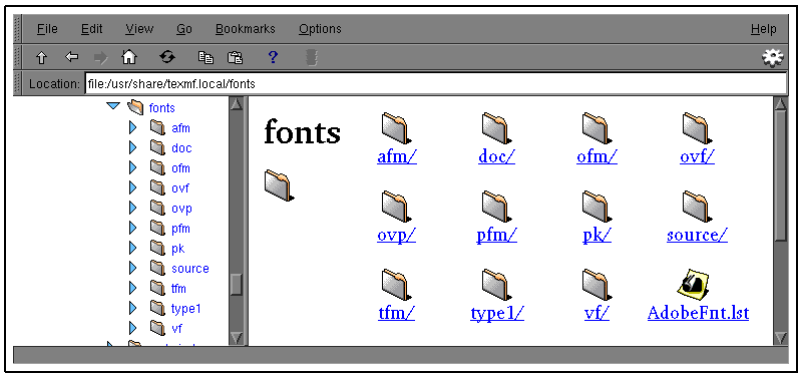
   ```
   vptovf zorkly.vpl zorkly.vf zorkly.tfm
   vptovf zorklyc.vpl zorklyc.vf zorklyc.tfm
   ```

7. Make directories to hold the files: the path may be different on your machine but under your `texmf.local` directory there should be a `fonts` directory, and in there there should be `afm`, `tfm`, `type1` and `vf` subdirectories. Create them if they do not already exist. Also create a subdirectory for the foundry in each, and within those, create a subdirectory for the typeface. In our example, this means:

   ```
   mkdir -p /usr/share/texmf.local/fonts/afm/zfonts/ordinary
   mkdir -p /usr/share/texmf.local/fonts/tfm/zfonts/ordinary
   mkdir -p /usr/share/texmf.local/fonts/type1/zfonts/ordinary
   mkdir -p /usr/share/texmf.local/fonts/vf/zfonts/ordinary
   ```

   For Microsoft *Windows* users, the path will look something like `C:\Local\TeX\texmf-local\fonts\afm...`

   The `-p` is a Unix feature: it automatically creates any missing intervening subdirectories. If your directory-making command doesn't do this, you'll have to do it the slow way and make the intervening directories first.



[1] If you don't have *afm2tfm* then your TEX installation or search path is partial or defective: it's a standard program that should be in the `bin` directory.

8. Copy the files to their rightful places:

```
cp *.afm /usr/share/texmf/fonts/afm/zfonts/ordinary/
cp *.tfm /usr/share/texmf/fonts/tfm/zfonts/ordinary/
cp *.pfb /usr/share/texmf/fonts/type1/zfonts/ordinary/
cp *.vf /usr/share/texmf/fonts/vf/zfonts/ordinary/
```

   The `cp` command is Unix: the equivalent for Microsoft *Windows* users is `copy`.

9. Make an entry in the *dvips* font map file for this foundry so that printing will work. Canonically the file `texmf/dvips/config/config.ps` should contain entries for each foundry's map file, in the form

   ```
   p +zfonts.map
   ```

   so the first time you use a font from a new foundry, create a new, appropriately-named map file in that directory and add a `p` line to `config.ps`.

   The alternative is to stuff everything into `psfonts.map` itself but that is a bit unwieldy, and liable to get overwritten when you upgrade. There is an *updmap* script which is supposed to take care of all this but I haven't tried it.

   The font entries in our hypothetical `zfonts.map` will look like this (one for raw, one for cooked, and each entry is on a *single* line to itself, no wrapping):

   ```
   rzorkly Ordinary-Blackface "TeXnANSIEncoding ReEncodeFont" <texnansi.enc <foo.pfb
   zorkly Ordinary-Blackface "TeXnANSIEncoding ReEncodeFont" <texnansi.enc <foo.pfb
   ```

   You get the full font name from `zork.id` which was created way back when we ran *afm2tfm* in Step 4. You should get this right, because it's the 'official' full name of the font.

10. Next, create a style file for the typeface for LaTeX to use called `foozork.sty` and put it in `texmf.local/tex/latex/faces` or somewhere similarly obvious where you keep home-brew typeface style files. This contains:

```
% foozork - created from foo
\def\fileversion{1.0}
\def\filedate{2001/12/03}
\def\docdate{2001/12/03}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{foozork}[\filedate\space\fileversion\space
                          Zfonts Ordinary PSNFSS2e package]
\RequirePackage[LY1]{fontenc}
\renewcommand{\rmdefault}{zor}
\endinput
```

    Note the argument to **`\ProvidesPackage`** *must* be the same as this style file name, and the font family is now referred to as `zor` in this example, being the foundry letter plus the fontname abbreviation. If this is a typewriter font, make that **`\rmdefault`** into **`\ttdefault`**. If it's a sans-serif font, make it **`\sfdefault`**.

11. Last file to create is the font definition (`.fd`) file. These are named `eeefnn.fd`, following the same conventions as before, by prepending the encoding abbreviation to the foundry letter and fontname abbreviation, so our example would be `ly1zor.fd`, and would contain:

```
\ProvidesFile{ly1zor.fd}[2001/03/03 manual font
                         definitions for LY1/zor.]

\DeclareFontFamily{LY1}{zor}{}
\DeclareFontShape{LY1}{zor}{k}{n}{<-> zorkly}{}
\DeclareFontShape{LY1}{zor}{k}{sc}{<-> zorklyc}{}
```

One **\DeclareFontFamily** command. Then as many pairs (assuming you did both normal and smallcaps for each font: see Step 5) as you converted fonts. The arguments to the **\DeclareFontShape** command to watch are the 3rd (weight/width), 4th (shape), and 5th (fontspec): the rest are static for each `.fd` file and simply identify the encoding and the font family.

The codes to use are given on pages 190–91 of the *LaTeX Companion* and should also be in your copies of `texmf/fontnames/weight.map` and `texmf/fontnames/width.map`. The rules for combining weight and width need care: RTFM. Oddly, there seems to be no `shape.map` online, so here's what the *Companion* says:

Character	Meaning
`n`	normal (upright)
`it`	italic
`sl`	slanted
`sc`	smallcaps
`ui`	upright italic
`ol`	outline

Add your own for other oddities: I use `cu` for cursive (scripts), for example, and `k` for 'black' font weights. The default fontspec parameter above `<->` means all sizes come from the same fontspec (remember if this was a METAFONT font with different design sizes like CM it would be more complex). If you didn't create a smallcaps flavour, then omit the second entry here (and in the foundry `.map` file).

If the face has only a few variants, you can create any 'missing' entries for bold, italic, slanted, etc with the relevant weight and width and shape values, but in the fontspec give the `ssub` substitution command: for example to make references to `sl` (slanted) type use an existing italic font, say

```
{<-> ssub * zor/m/it}
```

If you find the x-height of a font too big or too small to sort well with another font you are using, you can specify an `s` (scale) factor in this argument instead: this example will shrink the result to 80% of normal:

```
{<-> s * [0.8] zorkly}
```

12. Run your TeX indexer program to update the package database. This program comes with every modern version of TeX and is variously called *texhash*, *mktexlsr*, or even *configure* or it might just be a mouse click on a menu in your editor. Read the documentation that came with your installation to find out which one it is.

    This step is *utterly essential*, otherwise nothing will work.

Now you can `\usepackage{foozork}` in your LaTeX file to make it the default font. To use the font incidentally instead of as the default, you can say, eg

```
This is {\fontencoding{LY1}\fontfamily{zor}\selectfont ZORK}
```

In this case you will need to make LaTeX use the fontenc package by adding the command `\usepackage[LY1]{fontenc}` to the preamble of your document, as the one provided in `zork.sty` won't be used.

> Now…the bug. As I said, this has worked for years just fine. Suddenly, when I installed TeX Live 5d, everything fell apart. *Dvips* and the previewer were objecting to the 'lack' of an entry in the foundry `.map` file, because *someone* has changed the way these programs look for font names, ignoring the initial `r`, so where I had `rzorkly` they were taking `rzo` from the virtual font to be the font family instead of `zor`, and therefore failing to find the font entry. I don't know why this was done, and I didn't see any discussion about it, but it's obviously too late, and I'm sure it was done for a reason. The problem is I don't have a solution for it until someone can explain how to use Type 1 fonts without using virtual font mechanisms and the `r` prefix.

As we go to press, Anthony Goreham of Queen's, Oxford, posted to `comp.text.tex`:

> Actually I think this is configurable; there is a setting in the file `mktex.cnf`. (See `MT_FEATURES`, you don't want to have the option 'fontmaps' listed.) Maybe the default changed in the distro at some point?

I need to investigate this.

SESSION IX

Programmability (macros)

We've touched on the ability of LaTeX to be reprogrammed. This is one of its central features, and one that still, after nearly a quarter of a century, puts it well above many other typesetting systems, even those with macro systems of their own.

LaTeX is in fact a collection of macros, little program-like pieces of code which can be used as shorthand for an operation you wish to perform. In the simplest form, a LaTeX macro can just be a straightforward text replacement to avoid misspelling, eg

```
\newcommand{\ucc}{University College Cork}
```

Put this in your preamble, and then you can use `\ucc` in your document and it will typeset in the full version. Remember that after a command ending in a letter you need to leave a space to avoid the next word getting gobbled up as part of the command (see page 9). But when you want to force a space to be printed, use a backslash followed by a space, eg

```
In \ucc\ we have 13,000 students.
```

The **\newcommand** command takes two arguments: the name you want to give the new command, and the expansion to be performed when you use it.

But macros are not limited to text expansion. They can take arguments, so you can define a command to do something with text you give it. We looked earlier (page 56) at making new commands to put specific classes of words into certain fonts, such as product names into italics. Here's an example which also indexes the product:

```
\newcommand{\product}[1]{\textit{#1}\index{#1@\textit{#1}}}
```

If I type `\product{Velcro}` then I get *Velcro* typeset, and if you look in the index, you'll find this page referenced. The macro is specified as having one argument (the `[1]` in the definition). The expansion first prints the value of argument no. 1 (that's the `#1`) in italics, and then performs the index operation on the same value, additionally making sure that it's italicised in the index. Each occurrence of `#1` gets replaced by the value you provide in curly braces after `\product`. Macros can have up to nine arguments (each goes in its own set of curly braces).

Here's a slightly more complex example. It's common to refer in texts to people by their forename and surname (in that order), for example Don Knuth, but to have them indexed as surname,forename. This macro, **`\person`**, does exactly that.

```
\def\reindex #1 #2\sentinel{\index{#2, #1}}
\newcommand{\person}[1]{#1\reindex #1\sentinel}
```

It has one argument, so you put the whole name in a single set of curly braces, eg `\person{Don Knuth}`. The first thing the macro does is output `#1`, so the whole name gets printed exactly as you type it. But then it uses a second new macro called **`\reindex`** and follows it with the name you typed plus a dummy command **`\sentinel`** which is just there to signal the end of the name. The reason is, **`\reindex`** is defined in plain TeX's notation using **`\def`** which lets you specify the layout of the argument[s]. Here we tell **`\reindex`** to expect *two* arguments separated by a space and terminated by a dummy command called **`\sentinel`**, so **`\reindex`** in effect sees '`Don Knuth`' as two arguments. It can therefore output them using **`\index`** in reverse order, which is what we want. A book with a large number of personal names to print and index could make significant use of this to allow them to be typed and printed as `\person{Leslie Lamport}` but have them indexed as Lamport, Leslie.

How would you make it cater for Marie-Claire van Leunen or Blanca Maria Bartosova de Paul?

9.1 Reprogramming LaTeX's internals

LaTeX's internal macros can also be reprogrammed or even rewritten entirely, although that requires some considerable degree of expertise. Simple changes, however, are easily done. In section the footnote mentions that LaTeX's default document structure for the Report document class uses Chapters as the main unit of text, whereas in reality most reports are divided into Sections, not Chapters. The result of this is that if you start off your report with `\section{Introduction}`, it will print as

0.1 Introduction

which is not what you want. The zero is caused by it not being part of any chapter. But it's only a macro called `\thesection` which reproduces the current section number. It's redefined afresh in each document class file, using the command **`\renewcommand`** (in `texmf/tex/latex/base/report.cls` for this example):

```
\renewcommand \thesection {\thechapter.\@arabic\c@section}
```

You can see it invokes **\thechapter** (which is defined elsewhere to reproduce the value of the *chapter* counter); followed by a dot; and then the Arabic value of the counter called *section* (the \c@ notation is LaTeX's internal way of referring to counters). So we can easily redefine this in your preamble to leave out the reference to chapters:

```
\makeatletter
\renewcommand{\thesection}{\@arabic\c@section}
\makeatother
```

There are several things going on here, though:

- an 'at' sign (@) is not normally a valid character in LaTeX command names. It's used in internal commands to stop them being accidentally redefined, but here we are serious about it. There are two small additions, **\makeatletter** and **\makeatother**, which can be used to enclose redefinitions in a preamble which use the @-sign and allow it to be used;
- I've used the more formal method of enclosing the command being redefined in curly braces. For largely irrelevant historical reasons these braces are often omitted in LaTeX's internal code.

Now the introduction to your report will start with:

1 Introduction

What's important is that you *don't* alter the original document class file `report.cls`: you just copy the command you need to change into your own document preamble, and modify that instead. It will then override the default.

As promised on page 33, here's how to redefine a bullet for an itemized list:

```
\usepackage{bbding}
\renewcommand{\labelitemi}{\raisebox{-.25ex}{\PencilRight}}
```

In this case, we are using the ✏ symbol from the bbding package. Read the documentation for a full list of the symbols. For the font used for this document, these symbols are positioned slightly too high, so we use the **\raisebox** to shift the symbol down by ¼ ex (the minus value means 'lower' to **\raisebox**).

SESSION X

Compatibility with other systems

As we saw in section 2, LaTeX uses plaintext files, so they can be read and written by any standard application that can open files. This helps preserve your information over time, as the plaintext format cannot be obsoleted or hijacked by any manufacturer or sectoral interest.

However, LaTeX, as well as being a document preparation system, is also intended as the last stage of the editorial process: formatting for print. If you have a requirement to use the text in some other environment — a database perhaps — then it should be edited, stored, and maintained in something neutral like SGML or XML, and only converted to LaTeX when a typeset copy is needed.

10.1 Converting into LaTeX

There are several systems for converting or saving in LaTeX format. The best known is probably the *Lyx* editor, which is a wordprocessor-like interface to LaTeX. Several maths packages like *Mathematica* and *Maple* can also save material in LaTeX format, but in general, most wordprocessors and DTP systems don't have the level of internal markup needed to create a LaTeX file, although there are some small (oldish) programs on CTAN to convert *Word* and *WordPerfect* to LaTeX.

However, for text in SGML or XML, processors in any of the common languages like DSSSL, XSLT, *Omnimark*, *Balise*, etc can easily be written to output LaTeX, and this approach is extremely common and very productive. TeX4ht and HTML2LaTeX are packages to convert from HTML.

10.2 Converting out of LaTeX

This is harder, as it really requires the LaTeX program itself in order to process all the packages and macros correctly. There is a program on CTAN to do LaTeX-to-*Word* but it only handles the built-in commands of simple LaTeX, not packages.

At worst, the *detex* program will strip a LaTeX file of all markup and leave just the text, which can then be re-edited. There are also programs to extract the raw text from DVI and PS files.

10.3 Going beyond LaTeX

XML is the current target of most activity in this field. It has the advantages that it is based on an international standard (SGML), it works on all platforms, and there is lots of free software supporting it as well as lots of commercial packages. It is a language to let you define your own markup, so it is very flexible, and as mentioned above, there is a good choice of systems for converting into almost any target format, including LaTeX.

See *Understanding SGML and XML tools*[1] for details.

[1] Flynn, (1998)

Bibliography

1. Flynn, Peter: Understanding SGML and XML tools. Dordrecht/Boston: Kluwer Academic Publishers, 1998 432, ISBN 0–7923–8169–6

2. Goossens, Michel/Mittelbach, Frank/Samarin, Alexander: The LaTeX Companion. 1st edition. Reading, MA: Addison-Wesley, 1994

3. Goossens, Michel/Rahtz, Sebastian: The LaTeX Web companion. Reading, MA: Addison-Wesley Longman, 1999, Tools and Techniques for Computer Typesetting 522 , ISBN 0–201–43311–7

4. Goossens, Michel/Rahtz, Sebastian/Mittelbach, Frank: The LaTeX Graphics Companion. Reading, MA: Addison-Wesley, 1997, Tools and Techniques for Computer Typesetting 554, ISBN 0–201–85469–4

5. Lamport, Leslie: LaTeX, a document preparation system. 2nd edition. Reading, MA: Addison-Wesley, 1994

Index