

**format**  
**format**  
**format**  
`\title{`**formatting**  
**information**`}`  
**format**  
**format**  
**format**

**A beginner's introduction to typesetting with  $\text{\LaTeX}$**

**Peter Flynn**

Silmaril Consultants  
*Textual Therapy Division*

## Acknowledgments

Thanks to all the people who sent me corrections and suggestions for improvement or additions to earlier versions. As usual, the problem has been what to leave out, not what to include.

Some of the suggestions were well-intentioned but would have turned the book into a higher-level mathematics treatise. One of my objectives was to omit all maths except for a short example, as all the other books on  $\text{T}\text{E}\text{X}$  and  $\text{L}\text{A}\text{T}\text{E}\text{X}$  already cover mathematical typesetting in finer and better detail than I am capable of.

Some of the suggestions would have taken me down pathways I prefer not to tread. Large software corporations are full of well-meaning, hard-working individuals who genuinely believe that their products make life easier for users. Unfortunately, experience shows that this is often only true in the first hot flush of using a new program: in the long run the winners are those whose data is secure, accessible, and reusable; whose documents can be reformatted at any time, on any platform, without penalty, financial or otherwise.

I make no apology for recommending Unix-like systems running  $\text{L}\text{A}\text{T}\text{E}\text{X}$  as the platform of choice for document-processing applications — if you have a choice — and I'm happy to be able to include the Apple Macintosh in that family. Unfortunately, there are those whose circumstances at home or work require them to use something else, and I am pleased that  $\text{L}\text{A}\text{T}\text{E}\text{X}$  can help them by being available on their platform as well.

I have incorporated all the remaining suggestions except where it would materially distort the objective of being a *beginner's* booklet. A special thank-you goes to Barbara Beeton, Karl Berry, and William Adams for their editorial corrections and contributions, finding all kinds of mistakes from simple slips of the finger to cultural differences to over-complicated ways of explaining things.

It is very difficult for people who write technical documentation to remember how they struggled to learn what has now become a familiar system. So much of what we do is second nature, and a lot of it actually has nothing to do with the software, but more with the way in which we view and approach information, and the general level of knowledge of computing. If I have obscured something by making unreasonable assumptions about *your* knowledge, please let me know so that I can correct it.

Peter Flynn is author of *The HTML Handbook* and *Understanding SGML and XML Tools*, and editor of *The XML FAQ*.

## Technical note

The text for the March 2003 online edition was edited into a customized version of DocBook from the original private format (Read The Fine Markup Language (RTFML)). All subsequent work has been done in DocBook with a customization layer for typographics. XSLT was used to generate HTML (for the Web and plain-text versions) and  $\text{L}\text{A}\text{T}\text{E}\text{X}$  (for PDF and PostScript). The November 2003 edition was published in *TUGboat*<sup>1</sup>.

This document is Copyright © 1999, 2000, 2001, 2002, 2003 by Silmaril Consultants under the terms of what is now the GNU Free Documentation License (copyleft).

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled *The GNU Free Documentation License*<sup>2</sup>.

You are allowed to distribute, reproduce, and modify it without fee or further requirement for consent subject to the conditions in §A.4. The author has asserted his right to be identified as the author of this document. If you make useful modifications you are asked to inform the author so that the master copy can be updated. See the full text of the License in Appendix A.

---

<sup>1</sup>Beeton (Since 1980)

<sup>2</sup>FSF (2003/02/10 23:42:49)

# Contents

Introduction . . . . .	6
Who needs this booklet? . . . . .	6
Skills needed . . . . .	6
Objectives of this booklet . . . . .	6
Synopsis . . . . .	7
Production note . . . . .	8
Foreword . . . . .	10
Preface . . . . .	11
<b>1 Installing T<sub>E</sub>X</b> . . . . .	<b>13</b>
1.1 Editing and display . . . . .	13
1.2 Installation for Linux and Unix . . . . .	14
1.3 Installation for Apple Mac . . . . .	14
1.4 Installation for Microsoft Windows . . . . .	15
<b>2 Using your editor to create documents</b> . . . . .	<b>16</b>
2.1 Quick start for the impatient . . . . .	16
2.2 Editors . . . . .	18
2.2.1 WinEdt . . . . .	18
2.2.2 GNU Emacs . . . . .	19
2.3 L <sup>A</sup> T <sub>E</sub> X commands . . . . .	19
2.3.1 Simple commands . . . . .	20
2.3.2 Commands with arguments . . . . .	21
2.3.3 White-space in L <sup>A</sup> T <sub>E</sub> X . . . . .	21
2.4 Special characters . . . . .	21
2.4.1 Using the special characters . . . . .	22
2.5 Quotation marks . . . . .	22
2.6 Accents . . . . .	23
2.7 Sizes, hyphenation, justification, and breaking . . . . .	24
2.7.1 Specifying size units . . . . .	25
2.7.2 Hyphenation . . . . .	26
2.7.3 Unbreakable text . . . . .	26
2.7.4 Dashes . . . . .	26
2.7.5 Justification . . . . .	26
2.7.6 Languages . . . . .	27
2.8 Mathematics . . . . .	27
<b>3 Basic document structures</b> . . . . .	<b>29</b>
3.1 The Document Class Declaration . . . . .	29
3.1.1 Document class options . . . . .	30
3.2 The document environment . . . . .	31
3.3 Titling . . . . .	31
3.4 Abstracts and summaries . . . . .	33
3.5 Sections . . . . .	34
3.5.1 Section numbering . . . . .	35
3.6 Ordinary paragraphs . . . . .	36
3.7 Table of contents . . . . .	37
<b>4 Typesetting, viewing and printing</b> . . . . .	<b>39</b>
4.1 Typesetting . . . . .	39
4.1.1 Standard L <sup>A</sup> T <sub>E</sub> X . . . . .	39
4.1.2 pdfL <sup>A</sup> T <sub>E</sub> X . . . . .	41
4.1.3 Running L <sup>A</sup> T <sub>E</sub> X from a command window . . . . .	41
4.2 Error messages . . . . .	42
4.3 Screen preview . . . . .	43
4.3.1 Previewing DVI output . . . . .	43
4.3.2 Previewing with PDF . . . . .	44
4.3.3 Previewing with PostScript . . . . .	45

4.4	Printer output . . . . .	45
<b>5</b>	<b>CTAN, packages, and online help</b>	<b>47</b>
5.1	Packages . . . . .	47
5.1.1	Using an existing package . . . . .	47
5.1.2	Package documentation . . . . .	48
5.2	Downloading and installing packages . . . . .	49
5.3	Online help . . . . .	51
<b>6</b>	<b>Other document structures</b>	<b>52</b>
6.1	A brief note on structure . . . . .	52
6.2	Lists . . . . .	53
6.2.1	Itemized lists . . . . .	54
6.2.2	Enumerated lists . . . . .	54
6.2.3	Description lists . . . . .	54
6.2.4	Inline lists . . . . .	55
6.2.5	Reference lists and segmented lists . . . . .	56
6.2.6	Lists within lists . . . . .	56
6.3	Tables . . . . .	57
6.3.1	Floats . . . . .	57
6.3.2	Formal tables . . . . .	58
6.3.3	Tabular matter . . . . .	58
6.3.4	Tabular techniques for alignment . . . . .	60
6.4	Figures . . . . .	61
6.5	Images . . . . .	61
6.6	Verbatim text . . . . .	63
6.6.1	Inline verbatim . . . . .	63
6.6.2	Display verbatim . . . . .	64
6.7	Boxes, sidebars, and panels . . . . .	65
6.7.1	Boxes of text . . . . .	65
6.7.2	Framed boxes . . . . .	66
6.7.3	Sidebars and panels . . . . .	67
<b>7</b>	<b>Textual tools</b>	<b>68</b>
7.1	Quotations . . . . .	68
7.2	Footnotes and end-notes . . . . .	69
7.3	Marginal notes . . . . .	70
7.4	Cross-references . . . . .	70
7.4.1	Normal cross-references . . . . .	70
7.4.2	Bibliographic references . . . . .	71
7.5	Indexes and glossaries . . . . .	75
7.6	Multiple columns . . . . .	76
<b>8</b>	<b>Fonts and layouts</b>	<b>77</b>
8.1	Changing layout . . . . .	77
8.1.1	Spacing . . . . .	78
8.1.2	Headers and footers . . . . .	79
8.2	Using fonts . . . . .	80
8.2.1	Changing the default font family . . . . .	82
8.2.2	Changing the font family temporarily . . . . .	82
8.2.3	Changing font style . . . . .	83
8.2.4	Font sizes . . . . .	84
8.2.5	Logical markup . . . . .	85
8.2.6	Colour . . . . .	86
8.3	Installing new fonts . . . . .	87
8.3.1	Installing METAFONT fonts . . . . .	87
8.3.2	Installing PostScript fonts . . . . .	88
8.3.3	Installing the Type 1 Computer Modern fonts . . . . .	95

<b>9 Programmability (macros)</b>	<b>96</b>
9.1 Simple replacement macros	96
9.2 Macros using information gathered previously	96
9.3 Macros with arguments	98
9.4 Nested macros	99
9.5 Macros and environments	100
9.6 Reprogramming L <sup>A</sup> T <sub>E</sub> X's internals	100
9.6.1 Changing list item bullets	101
<b>10 Compatibility with other systems</b>	<b>102</b>
10.1 Converting into L <sup>A</sup> T <sub>E</sub> X	102
10.2 Converting out of L <sup>A</sup> T <sub>E</sub> X	103
10.3 Going beyond L <sup>A</sup> T <sub>E</sub> X	104
<b>A GNU Free Documentation License</b>	<b>105</b>
A.0 PREAMBLE	105
A.1 APPLICABILITY AND DEFINITIONS	105
A.2 VERBATIM COPYING	106
A.3 COPYING IN QUANTITY	106
A.4 MODIFICATIONS	107
A.5 COMBINING DOCUMENTS	108
A.6 COLLECTIONS OF DOCUMENTS	108
A.7 AGGREGATION WITH INDEPENDENT WORKS	108
A.8 TRANSLATION	108
A.9 TERMINATION	109
A.10 FUTURE REVISIONS OF THIS LICENSE	109
A.11 ADDENDUM: How to use this License for your documents	109
<b>B Configuring T<sub>E</sub>X search paths</b>	<b>110</b>
<b>C T<sub>E</sub>X Users Group membership</b>	<b>111</b>
TUG membership benefits	111
Becoming a TUG member	111
Privacy	111

## Exercises

1 Create a new document	31
2 Add a document environment	31
3 Adding the title block	32
4 Using an Abstract or Summary	34
5 Start your document text	35
6 Start typing!	37
7 Inserting the table of contents	38
8 Saving your file	39
9 Running L <sup>A</sup> T <sub>E</sub> X from the toolbar or menu	40
10 Running L <sup>A</sup> T <sub>E</sub> X in a terminal or console window	41
11 Print it!	46
12 Add colour	48
13 Read all about it	49
14 Install a package	51
15 List practice	55
16 Nesting	57
17 Create a tabulation	60
18 Adding pictures	63
19 Try some fixed-format text	64
20 Other names	100

## Introduction

This booklet originally accompanied a 2-day course on using the  $\LaTeX$  typesetting system. It has been extensively revised and updated and can now be used for self-study or in the classroom. It is aimed at users of Linux or Microsoft Windows but it can be used with  $\LaTeX$  systems on any platform, including other Unix workstations, Apple Macs, and mainframes.

### Who needs this booklet?

The audience for the original training course was assumed to be computer-literate and composed of professional, business, academic, technical, or administrative computer users. The readers of the booklet (you) are mostly assumed to be in a similar position, but may also come from many other backgrounds. You are expected to have one or more of the following or similar objectives:

- production of typesetter-quality formatting;
- formatting of long, complex, highly-structured, repetitive, or generated documents;<sup>3</sup>
- save time and effort by automating common tasks;
- independence from specific makes or models of proprietary hardware, software, or file formats (portability);
- use of Open Source software (free of restrictions, sometimes also free of charge).

### Skills needed

$\LaTeX$  is a very easy system to learn, and requires no specialist knowledge, although some familiarity with the publishing process is useful. It is, however, assumed that you are completely fluent and familiar with using your computer before you start. Specifically, effective use of this document requires that you already know and understand the following thoroughly:

- how to run and use a good plain-text editor (*not* a wordprocessor and not a toy like *Notepad*);
- where all 94 of the printable ASCII characters are on your keyboard and what they mean (and how to type accents and symbols, if you use them);
- how to create, open, save, close, rename, and delete files and folders (directories);
- how to use a Web browser or File Transfer Protocol (FTP) program to download and save files from the Internet;
- how to uncompress and unwrap (unzip) downloaded files.

If you don't know how to do these things yet, it's probably best to go and learn them first. Trying to become familiar with the fundamentals of using a computer *at the same time* as learning  $\LaTeX$  is not likely to be as effective as doing them in order.

### Objectives of this booklet

By the end of this booklet, you should be able to undertake the following tasks:

- use a plain-text editor to create and maintain your  $\LaTeX$  documents;
- add  $\LaTeX$  markup to achieve your formatting requirements;

---

<sup>3</sup> $\LaTeX$  can easily be used for once-off or short and simple documents as well, but its real strength lies in consistency and automation.

- typeset  $\LaTeX$  documents, correct simple formatting errors, and display or print the results;
- identify, install, and use additional formatting packages (using CTAN for downloading where necessary);
- recognise the limitations of procedural markup systems and choose appropriate generic markup where appropriate.

## Synopsis

The original course covered the following topics as separate sessions, now represented in the booklet as chapters:

1. Where to get and how to install  $\LaTeX$  (te $\TeX$  and fp $\TeX$  from  $\TeX$  Live);
2. How to type  $\LaTeX$  documents: using an editor to create files (*WinShell*, *WinEdt* or *Emacs*);
3. Basic structures (the Document Class Declaration and its layout options; the document environment with sections and paragraphs);
4. Typesetting, viewing, and printing;
5. The use of packages and CTAN to adapt formatting using standard tools;
6. Other document structures (lists, tables, figures, images, and verbatim text);
7. Textual tools (footnotes, marginal notes, cross-references, indexes and glossaries, and bibliographic citations);
8. Typographic considerations (white-space and typefaces; inline markup and font changes; extra font installation and automation);
9. Programmability and automation (macros and modifying  $\LaTeX$ 's behaviour);
10. Conversion and compatibility with other systems (XML, *Word*, etc.).

A few changes have been made in the transition to printed and online form, but the basic structure is the same, and the document functions as a workbook for the course as well as a standalone self-teaching guide. It is important to note that the document *does not cover* mathematical typesetting, complex tabular material, the design of large-scale macros and document classes, or the finer points of typography or typographic design, although it does refer to these topics in passing on a few occasions. There are several other guides, introductions, and 'get-started' documents on the Web and on CTAN. Among the more popular are:

- *Getting Started with  $\TeX$ ,  $\LaTeX$ , and friends*<sup>4</sup>, where all beginners should start;
- *The (Not So) Short Guide to  $\LaTeX 2_{\epsilon}$ :  $\LaTeX 2_{\epsilon}$  in 131 Minutes*<sup>5</sup> is a good beginner's tutorial;
- *A Gentle Introduction to  $\TeX$ : A Manual for Self-Study*<sup>6</sup> is a classic tutorial on Plain  $\TeX$ ;
- *Using imported graphics in  $\LaTeX 2_{\epsilon}$* <sup>7</sup> shows you how to do (almost) anything with graphics: side-by-side, rotated, etc.;

---

<sup>4</sup>TUG (November 2003)

<sup>5</sup>Oetiker et al. (2001)

<sup>6</sup>Doob (2002)

<sup>7</sup>Reckdahl (1997)

- *Short Math Guide for L<sup>A</sup>T<sub>E</sub>X*<sup>8</sup> gets you started with the American Math Society's powerful packages;
- *A comprehensive list of symbols in T<sub>E</sub>X*<sup>9</sup> shows over 2,500 symbols available in L<sup>A</sup>T<sub>E</sub>X.

(Taken from the CTAN search page.)





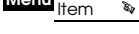
## Production note

This document is written and maintained in XML, using a customized version of the *DocBook* DTD. Conversions were made to HTML and L<sup>A</sup>T<sub>E</sub>X using XSLT scripts and Michael Kay's *Saxon* processor. The complete source, with all ancillary files, is available online at <http://www.ctan.org/tex-archive/info/beginlatex/> but if you want to try processing it yourself you must install *Java* (from Sun, IBM, or a number of others) and *Saxon* (from <http://saxon.sourceforge.net/>), in addition to L<sup>A</sup>T<sub>E</sub>X.

This document is published under the terms and conditions of the GNU Free Documentation License. Details are in Appendix A.

## Symbols and conventions

The following typographic notations are used:

Notation	Meaning
<b>\command</b>	Control sequences which perform an action, e.g. <b>\newpage</b>
<code>\length</code>	Control sequences which can be set to a dimension (measurement in units), e.g. <code>\parskip</code>
<i>counter</i>	Values used for counting (as opposed to measuring in units), e.g. <i>secnumdepth</i>
<i>term</i>	Defining instance of a specialist term
<i>product</i>	program or product name
<b>environment</b>	Formatting environment
package	L <sup>A</sup> T <sub>E</sub> X package (available from CTAN)
<i>mybook</i> or <i>value</i>	Examples of things you have to supply real-life values for
	A key on your keyboard
	Two keys pressed together
	Two keys pressed one after another
	On-screen button to click
	Drop-down menu with items

Examples of longer fragments of input are shown with a border round them. Where necessary, the formatted output is shown immediately beneath. Warnings are shown with a shaded background. Exercises are shown with a double border.

## Availability of L<sup>A</sup>T<sub>E</sub>X systems

The traditional T<sub>E</sub>X implementation is a Command-Line Interface (CLI), that is, a 'console' program which you run from a Unix or Mac shell window or an MS-DOS command window by typing the command `tex` or `latex` followed by the name of your document file. In automated (unattended) systems, this command is issued from within a Unix shell script or MS-DOS/Windows batch file. All the popular distributions for Unix and MS-DOS/Windows, both free and commercial, deliver this interface as standard (t<sub>E</sub>X, f<sub>P</sub>T<sub>E</sub>X, M<sub>I</sub>K<sub>T</sub>E<sub>X</sub>, P<sub>C</sub>-T<sub>E</sub>X, T<sub>R</sub>U<sub>E</sub>T<sub>E</sub>X, etc.).

While it is quite possible to run T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X this way, it is more normal to use an editor as your interface to the program as well as to your document. This allows you to control L<sup>A</sup>T<sub>E</sub>X, the typeset display, and other related programs with a mouse-click or menu item. This is the method assumed in this booklet. In the editors used for examples (*Emacs*, *WinShell*, and *WinEdt*) the typesetting process is logged visibly in an

<sup>8</sup>AMS (2001)

<sup>9</sup>Pakin (2002)



adjoining text window so that you can see the progress of pages being typeset, and any error messages that may occur.<sup>10</sup> This method is called an *asynchronous* typographic display because the typeset window only updates *after* you have typed the text and [re-]processed it, not *while* you are still typing.

Some commercial implementations of T<sub>E</sub>X offer a *synchronous* typographic interface: *Textures* for the Apple Macintosh from Blue Sky Research, *Scientific Word* from MacKichan Software, and *V<sub>T</sub>E<sub>X</sub>* from MicroPress, Inc (both for Microsoft Windows) are three examples. At least one free version (L<sub>y</sub>X, see Figure 2.1 in §2.2) offers the same kind of interface. In these, you type directly into the typographic display, as with a graphical wordprocessor, using the font controls of whatever Graphical User Interface (GUI) are appropriate.

With a synchronous display you get your instant textual gratification, but your level of control is restricted to that of the graphical user interface, which almost certainly does not provide access to everything that L<sub>A</sub>T<sub>E</sub>X can do. For complete control of the formatting you still need access to the L<sub>A</sub>T<sub>E</sub>X language. There are several methods available free for Unix and some other systems for close-to-synchronous updates of the typeset display (including Jonathan Fine’s *Instant Preview* and the T<sub>E</sub>X daemon), and for embedding typographic fragments from the typeset display back into the editor window (David Kastrup’s *preview-latex* package).

Whatever method you choose, T<sub>E</sub>X Live and CTAN are not the only source of software. The following vendors offer robust commercial implementations of T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X, and if you are in a position where their enhanced support and additional features are of benefit, I urge you to support them. In most cases their companies, founders, and staff have been good friends of the T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X communities for many years.

Product	Platform	Company	URI
PCT <sub>E</sub> X	MS-Windows	Personal T <sub>E</sub> X, Inc	<a href="http://www.pctex.com/">www.pctex.com/</a>
TrueT <sub>E</sub> X	MS-Windows	True T <sub>E</sub> X	<a href="http://truetex.com/">truetex.com/</a>
Textures	Apple Mac	Blue Sky Research	<a href="http://www.bluesky.com/">www.bluesky.com/</a>
Scientific Word	MS-Windows	Mackichan Software	<a href="http://www.mackichan.com/">www.mackichan.com/</a>
V <sub>T</sub> E <sub>X</sub>	MS-Windows, Linux, OS/2	MicroPress, Inc	<a href="http://www.micropress-inc.com/">www.micropress-inc.com/</a>
Y&YT <sub>E</sub> X	MS-Windows	Y&Y Software	<a href="http://www.yandy.com/">www.yandy.com/</a>

<sup>10</sup>Some recent versions of *Emacs* hide this window by default unless errors occur in the typesetting.

## Foreword

As noted in the Introduction, this document accompanied a 2-day introductory training course (still does) which I run 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 do not read the FAQs, do not buy the books and manuals, do not download the free documentation, and instead try to get by using the training technique known as ‘sitting by Nelly’, which involves looking over a colleague’s shoulder in the office, lab, library, or classroom, 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<sup>11</sup> designed to persuade newcomers to try  $\LaTeX$  for their typesetting requirements. As a result of questions and suggestions, it was obvious that it needed to include a pointer to some documentation, and I agreed to make available a revised form of this document, expanded to be used outside the classroom, and 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 material, some of which appears in almost every manual on  $\LaTeX$  but which is essential to the beginner and therefore bears repetition. I took the opportunity to revise the structure of the training course in parallel with the booklet (expanding it from its original one day to two days), and to include a more comprehensive index. It is by no means perfect (in both senses), and I would be grateful for comments and bugs to be sent to me at the address given under the credits.

I had originally hoped that the  $\LaTeX$  version of the document would be processable by any freshly-installed default  $\LaTeX$  system, but the need to include font samples which go well 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.

However, as the careful reader will have already noticed, the master source of the document is not maintained in  $\LaTeX$  but in XML. A future task is therefore to compare the packages required with those installed by default, and flag portions of the document requiring additional features so that an abbreviated version can be generated which can be guaranteed to process even with a basic  $\LaTeX$  installation.

If you are just starting with  $\LaTeX$ , at an early opportunity you should try to get hold of a copy of  *$\LaTeX$ : A Document Preparation System*<sup>12</sup> which is the original author’s manual. More advanced users should get the *The  $\LaTeX$  Companion*<sup>13</sup> or one of its successors. In the same series there are also the *The  $\LaTeX$  Graphics Companion*<sup>14</sup> and the *The  $\LaTeX$  Web Companion*<sup>15</sup>. Mathematical users might want to read *Short Math Guide for  $\LaTeX$* <sup>16</sup>.

---

<sup>11</sup><http://www.silmaril.ie/documents/latex-brochure/leaflet.pdf>

<sup>12</sup>Lamport (1994)

<sup>13</sup>Goossens/Mittelbach/Samarin (1993)

<sup>14</sup>Goossens/Rahtz/Mittelbach (1997)

<sup>15</sup>Goossens et al. (1999)

<sup>16</sup>AMS (2001)

## Preface

Many people discover  $\LaTeX$  after years of struggling with wordprocessors and desktop publishing systems, and are amazed to find that  $\TeX$  has been around for nearly 25 years and they hadn't heard of it. It's not a conspiracy, just 'a well-kept secret known only to a few million people', as one anonymous user put it.

Perhaps a key to why it has remained so popular is that it removes the need to fiddle with the formatting while you write. Although playing around with fonts and formatting is attractive to the computer newcomer, it is counter-productive for the serious author or editor who wants to concentrate on writing — ask any journalist or professional author. In response to a beginner's concern about 'learning to write in  $\LaTeX$ ', here's some advice posted (in the `comp.text.tex` newsgroup<sup>17</sup>):

No, the harder part might be *writing*, period.  $\TeX$ / $\LaTeX$  is actually easy, once you relax and stop worrying about appearance as a be-all-and-end-all. Many people have become 'Word Processing Junkies' and no longer 'write' documents, they 'draw' them, almost at the same level as a pre-literate 3-year old child might pretend to 'write' a story, but is just creating a sequence of pictures with a pad of paper and box of *Crayolas* — this is perfectly normal and healthy in a 3-year old child who is being creative, but is of questionable usefulness for, say, a grad student writing a Master's or PhD thesis or a business person writing a white paper, etc. For this reason, I strongly recommend *not* using any sort of fancy GUI 'crutch'. Use a plain vanilla text editor and treat it like an old-fashioned typewriter. Don't waste time playing with your mouse.

Note: I am *not* saying that you should have no concerns about the appearance of your document, just that you should *write* the document (completely) first and tweak the appearance later... *not* [spend time on] lots of random editing in the bulk of the document itself. [Heller, *New To  $\LaTeX$ ... Unlearning Bad Habits* (11 March 2003)]

Knuth originally wrote  $\TeX$  to typeset mathematics for the second edition of his master-work *The Art of Computer Programming*<sup>18</sup>, and it remains pretty much the only typesetting program to include fully-automated mathematical formatting done the way mathematicians want it. Knuth generously placed the entire system in the public domain, so for many years there was no publicity of the commercial kind which would have got  $\TeX$  noticed outside the technical field. Nowadays, however, there are many companies selling  $\TeX$  software or services, dozens of publishers accepting  $\LaTeX$  documents for publication, and hundreds of thousands of users using  $\LaTeX$  for millions of documents.<sup>19</sup>

There is occasionally some confusion among newcomers between the two main products,  $\TeX$  and  $\LaTeX$ :

- $\TeX$  is a typesetting program, originally written by Prof Don Knuth (Stanford) around 1978. It implements a macro-driven typesetters' programming language of some 300 basic operations and it has formed the core of many other desktop publishing (DTP) systems. Although it is still possible to write in the raw  $\TeX$  language, you need to study it in depth, and you need to be able to write macros (subprograms) to perform even the simplest of repetitive tasks.
- $\LaTeX$  is a user interface for  $\TeX$ , designed by Leslie Lamport (Digital Equipment Corporation (DEC)) in 1985 to automate all the common tasks of document preparation. It provides a simple way for authors and typesetters to use the power of  $\TeX$  without having to learn the entire language.  $\LaTeX$  is the recommended system for all users except professional typographic programmers and computer scientists who want to study the internals of  $\TeX$ .

<sup>17</sup>`news:comp.text.tex/MPG.18d82140d65ddc5898968c@news.earthlink.net`

<sup>18</sup>Knuth (1980)

<sup>19</sup>A guesstimate. With free software it's impossible to tell how many people are using it, but it's a *lot*.

### Debunking the mythology

Naturally, over all the years, a few myths have grown up around  $\LaTeX$ , often propagated by people who should know better. So, just to clear up any potential misunderstandings...

**MYTH: ' $\LaTeX$  has only got one font'** Most  $\LaTeX$  systems can use any OpenType, TrueType, Adobe (PostScript) Type1 or Type3, or METAFONT font, among others. This is more than most other known typesetting systems.  $\LaTeX$ 's default font is Computer Modern (based on Monotype Series 8: see Table 8.1), not Times Roman, and some people get upset because it 'looks different'. Typefaces differ: that's what they're for.

**MYTH: ' $\LaTeX$  is a Unix system'** People are also heard saying: ' $\LaTeX$  is a Windows system', ' $\LaTeX$  is a Mac system', etc., etc. *ad nauseam*.  $\TeX$  systems run on almost every computer in use, from some of the biggest supercomputers down to handhelds (Personal Digital Assistant (PDA)s like the Sharp *Zaurus*). That includes Windows and Linux PCs, Macs, and all Unix systems. If you're using something  $\TeX$  doesn't run on, it must be either incredibly new, incredibly old, or unbelievably obscure.

**MYTH: ' $\LaTeX$  is obsolete'** Quite the opposite: it's under constant development, with new features being added almost weekly. Check the `comp.text.tex` for messages about recent uploads to CTAN. It's arguably more up-to-date than most other systems:  $\LaTeX$  had the Euro (€) before anyone else, it had Inuktitut typesetting before the Inuit got their own province in Canada, and it still produces better mathematics than anything else.

**MYTH: ' $\LaTeX$  isn't WYSIWYG'** Simply not true. The DVI and PDF preview is better WYSIWYG than any wordprocessor and most DTP systems.

What critics mean is that  $\LaTeX$ 's typographic display is asynchronous with the edit window. This is only true for the default CLI implementations. See the Introduction for details of synchronous versions.

**MYTH: ' $\LaTeX$  is "too difficult"'** This has been heard from physicists who can split atoms; from mathematicians who can explain why  $\pi$  exists; from business people who can read a balance sheet; from historians who can grasp Byzantine politics; from librarians who can understand LoC and MARC; and from linguists who can decode Linear 'B'. Most people grasp  $\LaTeX$  in 20 minutes or so. It's not rocket science (or if it is, I know any number of unemployed rocket scientists who will teach it to you).

**MYTH: ' $\LaTeX$  is "only for scientists and mathematicians"'** Not at all. Although it grew up in the mathematical and computer science fields, two of its biggest growth areas are in the humanities and business, especially since the rise of XML brought new demands for automated typesetting.

## CHAPTER I

# Installing T<sub>E</sub>X

This course is based on using Thomas Esser's teT<sub>E</sub>X (for Linux and other Unix-like systems, including Mac OS X) and François Popineau's fpT<sub>E</sub>X (for Microsoft Windows), both from T<sub>E</sub>X Live (fpT<sub>E</sub>X is an implementation of teT<sub>E</sub>X). Many other implementations of T<sub>E</sub>X, including Christian Schenk's MikT<sub>E</sub>X for Microsoft Windows, and Tom Kiffe's CMacT<sub>E</sub>X for the Apple Macintosh, can be downloaded from CTAN. L<sup>A</sup>T<sub>E</sub>X is included with all distributions of T<sub>E</sub>X.

T<sub>E</sub>X Live is issued annually as a joint effort by virtually all the local T<sub>E</sub>X user groups around the world (see <http://www.tug.org/lugs.html> for addresses). There are many other distributions of L<sup>A</sup>T<sub>E</sub>X both free and commercial, as described in the Introduction: they all process L<sup>A</sup>T<sub>E</sub>X identically, but there are some differences in size, speed, packaging, and (in the case of commercial distributions) support.

### 1.1 Editing and display

Before you start using L<sup>A</sup>T<sub>E</sub>X you will need to decide on which plain-text editor you want to use to create and maintain your documents. There is a wide range available, and probably no other piece of software causes more flame-wars in Internet and other discussions than your choice of editor. It's a highly personal choice, so feel free to pick the one you like. My personal biases are revealed below.

If you are intending to produce PostScript or PDF (Adobe Acrobat) files, you will need a viewer to display them. *GSview* displays both PostScript and PDF files; *xpdf* and Adobe's own *Acrobat Reader* just display PDF files.

#### Additional downloads

For licensing reasons, the *GSview*, and *Acrobat Reader* viewers and the *WinEdt* editor cannot be distributed on the T<sub>E</sub>X Live media. You have to download and install them separately. This is a restriction imposed by their authors or vendors, not by the T<sub>E</sub>X Live team.

- *GSview* is available for all platforms from <http://www.ghostscript.com/gsview/index.htm> (on Unix and VMS systems it's also available as *GhostView* and *gv*: see <http://www.cs.wisc.edu/~ghost/>)
- *Acrobat Reader* (all platforms) can be downloaded from <http://www.adobe.com/products/acrobat/readstep2.html>
- *Xpdf* (X Window systems only) can be downloaded from <http://www.foolabs.com/xpdf/>
- *WinEdt* (Microsoft Windows only) comes from <http://www.winedt.com>

Editor	Comments
<i>Emacs</i>	Large and powerful. Needs learning (a life-skill like L <sup>A</sup> T <sub>E</sub> X) but well worth it. Multi-platform, it opens and edits anything and everything, with special features for L <sup>A</sup> T <sub>E</sub> X etc., and good productivity tools for writers. Open Source.
<i>WinShell</i>	Simple but effective beginner's tool for MS-Windows. Runs L <sup>A</sup> T <sub>E</sub> X etc. with toolbar buttons. Very easy to use, small footprint. Strongly recommended. Free.
<i>WinEdt</i>	Comprehensive editor aimed at heavy L <sup>A</sup> T <sub>E</sub> X usage, with toolbar, productivity features, and good support. Configurable for almost any distribution of T <sub>E</sub> X (with some effort). Free trial, licensable after 1 month, MS-Windows only.
<i>PFE</i>	Popular general-purpose editor for MS-Windows: no special L <sup>A</sup> T <sub>E</sub> X features but a very configurable launcher and command-line controller. Free but no longer developed.
<i>BBedit</i>	Plain-text editor for Apple Macs, heavily used for text applications. Some T <sub>E</sub> X distributions for the Mac come with their own editor, but this is a popular and useful tool. 30-day demo.
<i>vi</i>	Standard editor on Unix systems. Dual-mode editor (separate text-entry and command modes), now showing its age. Adored by devotees, detested by others, just like <i>Emacs</i> 😊. Free.

## 1.2 Installation for Linux and Unix

Make sure your system libraries and utilities are up to date. If you are using Red Hat Linux, use the *up2date* program to download and install updates. For Debian and other distributions, use *apt-get* or similar utilities. On Red Hat systems, remove (or don't install) the RPM version of teT<sub>E</sub>X and associated utilities to avoid version conflicts.

If you are installing T<sub>E</sub>X Live to a new partition, or if you have the opportunity to reformat the partition before use, use *mkfs* with a granularity as small as it will go (usually 1024 bytes). This avoids the partition running out of inodes because T<sub>E</sub>X uses very large numbers of very small files.

Plan the installation carefully if you are installing for multiple users (a shared machine): read §5.2 for some comments on where to put additional files downloaded later, and see the FAQ on the same topic at <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=wherfiles>

Above all, Read The Fine Manual (RTFM). The documentation is in *readme.html*, and there's a PDF copy for printing.

The installer runs in a shell window, so it can be done even from systems with no X Window installation, but it does need to be installed as root if you want to stick with the default directory locations.

Type the command `sh install-tl.sh` to run it: the options are mostly self-explanatory, and beginners should pick the Recommended scheme and leave all other settings at their defaults.

## 1.3 Installation for Apple Mac

This is exactly the same interface as for the Linux/Unix installation. The documentation does point out that 'most front ends (*T<sub>E</sub>XShop*, *IT<sub>E</sub>XMac*,...) use the teT<sub>E</sub>X default location which is `/usr/local/teTeX`, so, Mac users could find interest in installing T<sub>E</sub>X Live in `/usr/local/teTeX` rather than in `/usr/TeX`.'

## 1.4 Installation for Microsoft Windows

Before you install T<sub>E</sub>X Live, make sure you have enough disk space: the default installation takes about 350Mb on a modern filesystem. The installation assumes you have a fully updated version of Windows, so visit the Microsoft Web site first (<http://www.microsoft.com/>) and click on *Windows Update*. Select and install all the relevant updates for your operating system (Windows 95, 98, ME, 2000, NT, or XP). You should be doing this regularly anyway, to keep your system healthy. You may want to run *ScanDisk* and give your hard disks a full surface check. T<sub>E</sub>X consists of a very large number of quite small files, so it's important that your disk is in good shape. When you insert the T<sub>E</sub>X Live disk, it should start the setup program automatically. If you have auto-run turned off, open *My Computer*, double-click on the relevant drive, and then double-click *Autorun* to start the setup program. Some versions of Microsoft Windows are broken and won't start the program correctly this way. If this happens, use *My Computer* to go to the `bin\win32` subfolder of the T<sub>E</sub>X Live disk and double-click `TeXLive.exe`.

Once the T<sub>E</sub>X Live program is running:

### 1. E<sub>T</sub><sub>E</sub>X

Install E<sub>T</sub><sub>E</sub>X itself from the **TeXLive Software** `Install on Hard Disk` menu. If you're new to E<sub>T</sub><sub>E</sub>X, pick Quick Install on the following screen. This gives you everything you need to get started, and doesn't ask any questions, it just installs it all straight away.

If you're installing under Windows NT, 2000, or XP, you may want to click on the option to install for all users if you have other users on your system.

If you want to use *Emacs* as your editor, click the option for X<sub>e</sub>mT<sub>E</sub>X Support.<sup>1</sup>

### 2. Emacs

After installation, right-click and drag `Xemacs.exe` from the `C:\ProgramFiles\TeXLive\bin\win32` folder out onto your desktop and let go, then pick 'Create Shortcut'. This places *Emacs* on your desktop for easy access.

### 3. WinShell and WinEdt

If you want to install *WinShell*, run the installer program in the `support/winshell` directory. For *WinEdt* you must go to their Web site (<http://www.winedt.com/>) for a downloadable version.

You don't have to install just one editor: if you've got the space, install them all so you can try them out. You can always uninstall the ones you don't want afterwards.

### 4. GSView

Ghostscript is installed automatically, but for *GSView* you need to go to <http://www.cs.wisc.edu/~ghost/gsview/>, and download the most recent version.

If you use *GSView*, please register your copy with Ghostgum, Pty. (<http://www.ghostgum.com.au/>).

Newcomers should note that the downloadable *WinEdt* comes preconfigured for a different distribution of T<sub>E</sub>X (MIK<sub>T</sub><sub>E</sub>X) and needs some careful reconfiguration to work with T<sub>E</sub>X Live. I recommend that beginners use *WinShell* to start with and graduate to *Emacs* when they become more expert. Many other people will doubtless tell you different...

Please read the T<sub>E</sub>X Live update pages at <http://www.tug.org/texlive/bugs.html> for details of any changes since the disks were released, and download and install any additional software required.

<sup>1</sup>Note this is nothing to do with Eberhard Matthes' DOS implementation of T<sub>E</sub>X called emT<sub>E</sub>X — the 'Xem' is short for *Xemacs*.

## CHAPTER II

# Using your editor to create documents

LaTeX documents are all *plain-text* files.<sup>1</sup> You can edit them with any editor, and transfer them to any other computer system running LaTeX and they will format exactly the same. Because they are plain text they cannot corrupt your system, and they cannot be used for virus infections as wordprocessor files can. Everything you can see is in the file and everything in the file is displayed to you: there is nothing hidden or secret and no proprietary manufacturers' 'gotchas' like suddenly going out of date with a new version.

In a LaTeX document, you type your text along with *markup* which identifies the important parts of your document by name, for example 'title', 'section', 'figure', etc. LaTeX does all the formatting for you automatically, using the markup to guide its internal rules and external stylesheets for typesetting.

You will usually hear this markup referred to as 'commands' or sometimes 'control sequences' (which is the proper Texnical term for them). For all practical purposes these terms all mean the same thing.

You do not need to format any of your text by hand *in your editor*, because LaTeX does it all by itself when it typesets. 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.

This course assumes that users have either *WinEdt* (Windows only) or *Emacs* (any platform) installed. Both are discussed briefly in §2.2 and the menus and toolbars for running LaTeX are explained in Chapter 4.

## 2.1 Quick start for the impatient

If you already know all this stuff about editors and plain-text files and running programs, or you know your system is already correctly installed (including your editor), you'd probably like to type something in and see LaTeX do its job. If you don't, then skip forward to §2.3 and read a bit more about LaTeX first.

### 1. Install the software

Make sure you have a properly-installed LaTeX system and a copy of a suitable editor (*Emacs* or *WinEdt* are recommended).

---

<sup>1</sup>'Plain-Text' originally meant just the 94 printable characters of the American Standard Code for Information Interchange (ASCII) but now more commonly includes both these *and* the relevant 8-bit characters from *one* (only) character set such as ISO-8859-1 (Western Latin-1) or ISO-8859-15 (Western Latin plus the Euro sign). These are international standards which work everywhere: try to avoid using manufacturers' proprietary character sets like Microsoft Windows-1252 or Apple Macintosh Roman-8, because they are unusable on some other systems.



## 2. Create a demonstration document

Open your editor and type in the following text *exactly* as given. Do *not* make any changes or miss anything out or add anything different:

```
\documentclass[12pt]{article}
\usepackage{palatino,url}
\begin{document}
\section*{My first document}

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

\begin{itemize}
\item setting the default font size to 12pt;
\item specifying 'article' type formatting;
\item using the Palatino typeface;
\item adding special formatting for URIs;
\item formatting a heading in 'section' style;
\item using the \LaTeX\ logo;
\item generating today's date;
\item formatting a list of items;
\item centering and italicizing;
\item autonumbering the pages.
\end{itemize}

\subsection*{More information}

This example was taken from 'Formatting Information',
which you can download from
\url{http://www.ctan.org/tex-archive/info/beginlatex/}
and use as a teach-yourself guide.

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

\end{document}
```



## 3. Save the document

Save the document as `demo.tex`.

## 4. Run $\LaTeX$


Click on the  toolbar icon or the  menu item; or type `latex demo` in a command window.

## 5. Preview the typesetting

Click on the  toolbar icon or the  menu item; or type `dviview demo` (Microsoft Windows) or `xdvi demo &` (Unix and Linux).

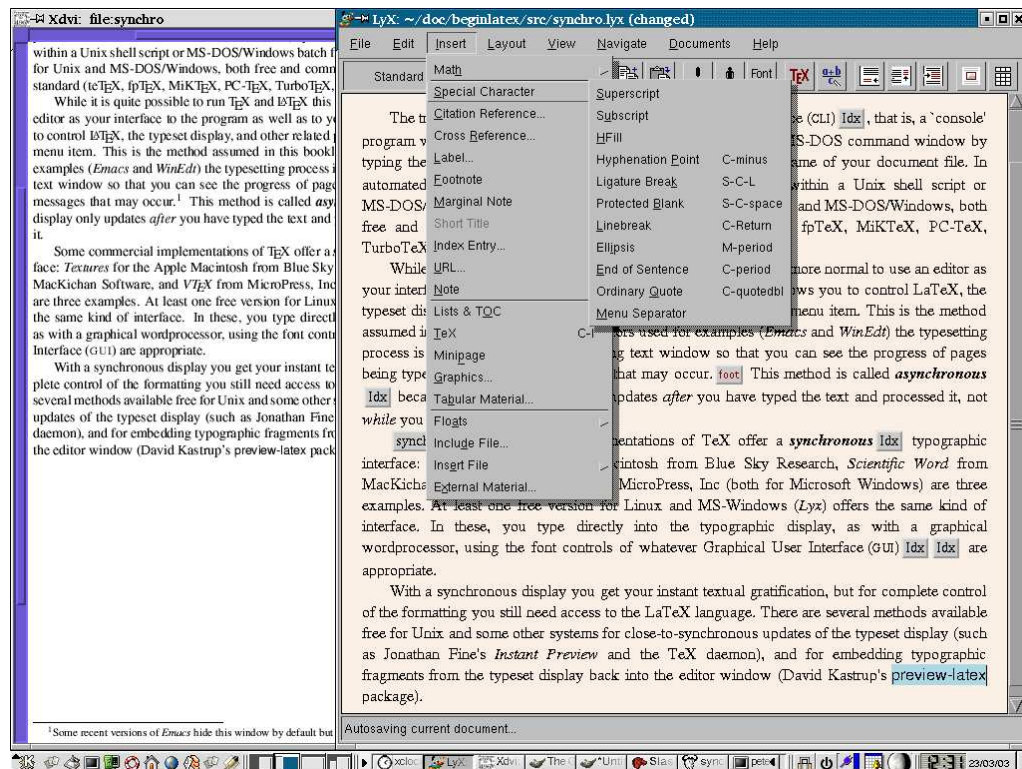
(Note that there may be a pause the first time some fonts are used, while bitmaps are created.)

## 6. Print it

Click on the  toolbar icon within the previewer (Microsoft Windows) or type `dvips demo` (Unix and Linux).

If you encounter any errors, it means you *do* need to study this chapter after all!

Figure 2.1: The LyX document editor



## 2.2 Editors

All the text of your documents can be typed into your  $\LaTeX$  document from a standard keyboard using any decent plain-text editor. However, it is more convenient to use an editor with special features to make using  $\LaTeX$  easier. Two of the most popular are *WinEdt* (Windows only) and *Emacs* (all platforms). The LyX document editor is a special case, as it uses the What You See Is What You Mean (WYSIWYM) model of synchronous typographic editing as opposed to WYSIWYG, and many users prefer this interface (but see the reservations in the Introduction).

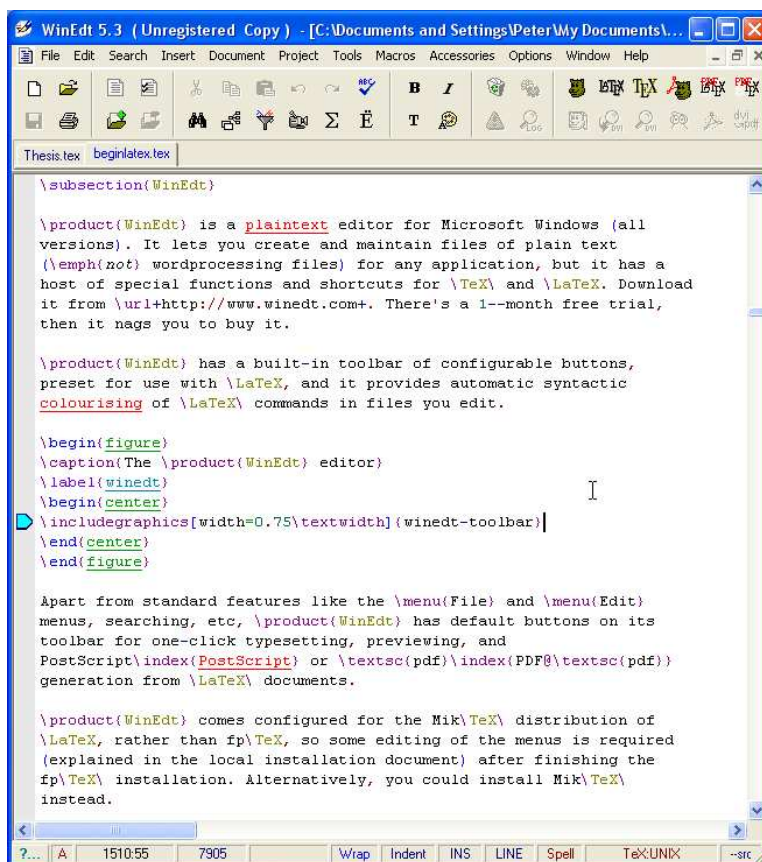
### 2.2.1 WinEdt

*WinEdt* is a plain-text editor for Microsoft Windows (all versions). It lets you create and maintain files of plain text (*not* wordprocessing files) for any application, but it has a host of special functions and shortcuts for  $\TeX$  and  $\LaTeX$ . Download it from <http://www.winedt.com>—there's a 1-month free trial, then it nags you to buy it.

*WinEdt* has a built-in toolbar of configurable buttons, preset for use with  $\LaTeX$ , and it provides automatic syntactic colouring of  $\LaTeX$  commands in files you edit.

Apart from standard features like the **File** and **Edit** menus, 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 Mik $\TeX$  distribution of  $\LaTeX$ , rather than fp $\TeX$ , so some editing of the menus is required (explained in the local installation document) after finishing the fp $\TeX$  installation. Alternatively, you could install Mik $\TeX$  instead.

Figure 2.2: The *WinEdt* editor

## 2.2.2 GNU Emacs

*Emacs* is a product of the GNU Project.<sup>2</sup> Versions are available for all makes and models of computer, and it has a L<sup>A</sup>T<sub>E</sub>X-mode which provides syntactic colouring ('fontification' in *Emacs*-speak) and mouseclick processing from a menu or toolbar.

*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 it for the rest of the day — or month. 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 XML and L<sup>A</sup>T<sub>E</sub>X — and it provides a few games as well.

*Emacs*, like *WinEdt*, knows about L<sup>A</sup>T<sub>E</sub>X and how to process it, so there is a menu full of L<sup>A</sup>T<sub>E</sub>X operations to click on. If you are editing more complex documents, especially with mathematics, there is an add-on package ('mode' in *Emacs*-speak) called *AUCT<sub>E</sub>X* which has even more functionality.

Because *Emacs* runs on Microsoft Windows, Macs, Linux, VMS, and most other platforms, many L<sup>A</sup>T<sub>E</sub>X users who have multiple machines prefer it to other editors because it provides the same environment regardless of which platform they are using.

## 2.3 L<sup>A</sup>T<sub>E</sub>X commands

L<sup>A</sup>T<sub>E</sub>X commands all begin with a *backslash* (`\`) and are usually made up of lowercase letters only, for example:

<sup>2</sup>'GNU's Not Unix (GNU)' is a project to make a computer operating environment completely free of restrictions.

Figure 2.3: Emacs editing L<sup>A</sup>T<sub>E</sub>X

```
\clearpage
```

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

The `\clearpage` command is an instruction to the typesetter to start a new page. It's actually quite rare — page-breaking is automatic in L<sup>A</sup>T<sub>E</sub>X — but it makes a good example of a simple command.

### 2.3.1 Simple commands

Simple one-word commands like `\clearpage` which end with a letter must be separated from any following text with white-space (a newline [linebreak], tab character, or an ordinary space). For example either of these two forms will be fine:

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

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

However, if you omit the white-space, as in the following example, L<sup>A</sup>T<sub>E</sub>X will try to read it as a command called `\clearpageThe`, which doesn't exist, and it will complain at you.

```
\clearpageThe importance of poetic form must not be ignored
```

$\LaTeX$  swallows any white-space which follows a command ending in a letter. It does this automatically, so you won't get unwanted extra space in your typeset output, but it does mean that any simple command which ends in a letter and has no arguments (see below) must be followed by white-space before normal text starts again, simply to keep the command separate from the text.

### 2.3.2 Commands with arguments

Many  $\LaTeX$  commands are followed by one or more *arguments* (a term from the field of Computer Science, meaning information to act upon), for example:

```
\chapter{Poetic Form}
\label{pform}
```

Such arguments always go in *{curly braces}* like those shown above. Be careful not to confuse the curly braces on your keyboard with round parentheses ( ), square brackets [ ], or angle brackets < >. They are all different.

With commands like this (with arguments) you do *not* need to use extra white-space after the command name, because there is an argument following it which will keep it separate from any normal text which follows after that.

### 2.3.3 White-space in $\LaTeX$

In  $\LaTeX$  documents, all *multiple* spaces, newlines (linebreaks), and TAB characters are treated as if they were a *single* space or newline during typesetting.  $\LaTeX$  does its own spacing and alignment using the instructions you give it, so you have extremely precise control. You are therefore free to use extra white-space in your editor for optical ease and convenience when editing.

The following is therefore exactly equivalent to the example in the preceding section:

```
\chapter    {Poetic Form}\label
           {pform}
```

That is, it will get typeset exactly the same. In general, just leave a blank line between paragraphs and a single space between words and sentences.  $\LaTeX$  will take care of the formatting.

## 2.4 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	<i>If you need the actual character itself, type this:</i>	Character
<code>\</code>	The command character	<code>\backslash</code>	<code>\</code>
<code>\$</code>	Math typesetting delimiter	<code>\\$</code>	<code>\$</code>
<code>%</code>	The comment character	<code>\%</code>	<code>%</code>
<code>^</code>	Math superscript character	<code>\^</code>	<code>^</code>
<code>&amp;</code>	Tabular column separator	<code>\&amp;</code>	<code>&amp;</code>
<code>_</code>	Math subscript character	<code>\_</code>	<code>_</code>
<code>~</code>	Non-breaking space	<code>\~</code>	<code>~</code>
<code>#</code>	Macro parameter symbol	<code>\#</code>	<code>#</code>
<code>{</code>	Argument start delimiter	<code>\{</code>	<code>{</code>
<code>}</code>	Argument end delimiter	<code>\}</code>	<code>}</code>

These were deliberately chosen because they are rare in normal text, with the exception of `$`, `#`, `&`, and `%`, some of whose meanings were already established as *metacharacters* (characters standing as symbols for something else) by the time  $\text{\TeX}$  was written.

### 2.4.1 Using the special characters

Because of the special meaning  $\text{\TeX}$  uses for the dollar-sign on its own, if you want to print `$35.99` you type `\$35.99`.

(An unusual but interesting serif-font Euro sign `€` is got with the `\texteuro` command from the `textcomp` package. The standard sans-serif `€` needs the `marvosym` package and is done with the `\EUR` command.<sup>3</sup>)

If you want to print `AT&T` you need to type `AT\&T`; if you want to print `45%` you need to type `45\%`; and if you want a *hash mark* (the *octothorpe* or American number or ‘pound’ [weight] sign `#`) you type `\#`. For a pound (sterling) sign `£`, now nearly obsolete except in the UK and some of its former dependencies, use your `£` key or type `\textsterling`.

The *comment character* (`%`) makes  $\text{\TeX}$  ignore the remainder of the line in your document, so you can see it in your editor, but it will never get typeset, for example:

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

## 2.5 Quotation marks

Do *not* use the unidirectional typewriter 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 real left-hand and right-hand (opening and closing) quotes (usually shaped like tiny <sup>66</sup> and <sup>99</sup> or as similarly symmetrically-balanced strokes). If you are using *Emacs* as your editor, the `”` key (usually Shift 2) is specially programmed

<sup>3</sup>The European Commission has specified that everyone use the sans-serif design, even in serif text, but this is amazingly ugly and most designers rightly ignore it.

in  $\LaTeX$ -mode to think for itself and produce correct `` and '' characters (so this is one occasion when you *can* use the `"` key).

If you are reading this in a browser, or if you have reprocessed the file using different fonts, it may not show you real quotes (some of the browser fonts are defective) and the `\thinspace` below may be too wide. Download the typeset (PDF) version of this document to see the real effect.

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 and could allow an unwanted linebreak):

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

He said, 'Her answer was "never"', and...

## 2.6 Accents

For accented letters in ISO 8859-1 (Latin-1, Western European), 8859-15 (same but with the Euro) or other Latin-alphabet character sets just use the accented keys on your keyboard (if you have them). If you don't, you need to use your operating system's standard keyboard `Ctrl` or `Alt` key combinations to generate the characters (see the panel 'Keystrokes for accented letters' on p.23).

You must also tell  $\LaTeX$  what input encoding you are using. Specify this by using the `inputenc` package<sup>4</sup> in your preamble with the relevant option. For example, to make  $\LaTeX$  understand the codes for ISO Latin-1, use:

```
\usepackage[latin1]{inputenc}
```

### Keystrokes for accented letters

This is for users whose keyboards do not have native accent characters on them. See your Operating System manual for full details. Here are two common examples:

- Under Linux the letter é is usually got with `AltGr-;` `e`. Refer to the *xkeycaps* utility for a table of key codes and combinations (get it from <http://www.jwz.org/xkeycaps/>).
- Under Microsoft Windows the letter é is got with `Ctrl-7` `e` or holding down the `Alt` key and typing `0130` on the numeric keypad (*not* the top row of shifted numerals). Refer to the *charmap* utility for a table of key codes and combinations (find it in the C:\Windows folder).

If you cannot generate ISO 8859-1 characters from your keyboard at all, or you need additional accents or symbols which are not in any of the keyboard tables, use the symbolic notation below. In fact, this can be used to put any accent over any letter: if you particularly want a ã you can have one with the command `\~g` (and Welsh users can get `\^w`).

<sup>4</sup>We haven't covered the use of packages yet. Don't worry, see Chapter 5 if you're curious.

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	ǒ	\b o
Dot-over (séimíú)	ṁ	\.m
Dot-under	ḣ	\d s
Breve	ṁ	\u u
Háček (caron)	ř	\v u
Long umlaut	ö	\H o
Tie-after	oo	\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

If you use this symbolic method only, you do not need to use the `inputenc` package. Before the days of keyboards and screens with their own real accented characters, the symbolic notation was the *only* way to get accents, so you may come across a lot of older documents (and users!) using this method all the time.

Irish and Turkish dotless-ı is done with the special command `\i`, so an í-fada (which is normally typed with `\i`) requires `\' \i` if you need to type it in long format, followed by a backslash-space or dummy pair of curly braces if it comes at the end of a word and there is no punctuation, because of the rule that `\TeX` control sequences which end in a letter (see §2.3.1) always absorb any following space. So what you normally type as Rí Teamraç has to be `R\' \i \ Tea\ .mra\ .c` when typed in full (there are not usually any keyboard keys for the dotless-ı or the lenited characters). A similar rule applies to dotless-j.

## 2.7 Sizes, hyphenation, justification, and breaking

`\TeX`'s internal measurement system is extraordinarily accurate. The underlying `\TeX` engine conducts all its business in units smaller than the wavelength of visible light, so if you ask for 15mm space, that's what you'll get.<sup>5</sup> At the same time, many dimensions in `\TeX`'s preprogrammed formatting are specially set up to be flexible: so much space, plus or minus certain limits to allow the system to make its own adjustments to accommodate variations like overlong lines, unevenly-sized images, and non-uniform spacing around headings.

`\TeX` uses the most sophisticated justification algorithm known to achieve a smooth, even texture to normal paragraph text. The programming for this has been borrowed by a large number of other DTP systems, and users of these are often quite unaware that they are in fact using a significant part of `\TeX` in their work. Occasionally, however, you will need to hand-correct an unusual word-break or line-break, and there are facilities for doing this on individual occasions as well as throughout a document.

<sup>5</sup>Within the limitations of your screen or printer, of course. Most screens cannot show dimensions of less than  $\frac{1}{96}$ " without resorting to magnification or scaling, and even at 600dpi, fine oblique lines or curves on a printer can still sometimes be seen to stagger the dots.



Figure 2.4: Different sizes of type boxed at 1em

**Times New Roman 72pt    Adobe Helvetica 36pt**



### 2.7.1 Specifying size units

Most people in printing and publishing habitually use points and picas and ems. Some designers use cm and mm. Many English-language speakers still use inches. You can specify lengths in  $\LaTeX$  in any of these units, and others:

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 (historically the width of a letter 'M' but see below)
ex	x-height of the current font (height of letter 'x')

*Other measures*

cm	centimeters (2.54 to the inch)
mm	millimeters (25.4 to the inch)
in	inches

The em can cause beginners some puzzlement because it's based on the 'point size' of the type, which is itself misleading. The point size refers to the depth of the metal body on which foundry type was cast in the days of metal typesetting, *not* the height of the letters themselves. Thus the letter-size of 10pt type in one face can be radically different from 10pt type in another (look at Table 8.1, where all the examples are 10pt). An em is the height of the type in a specific size, so 1em of 10pt type is 10pt and 1em of 24pt type is 24pt. Another name for a 1em space is a 'quad', and  $\LaTeX$  has a command `\quad` for leaving exactly that much horizontal space. To make the point, Figure 2.4 shows two capital Ms of 72pt and 36pt type in different faces, surrounded by a box exactly 1em of those sizes wide. A special name is given to the 12pt em, a 'pica' em, as it has become a fixed measure in its own right.

If you are working with other DTP users, watch out for those who think that Adobe points (bp) are the only ones. The difference is only .27pt per inch, but in 10'' of text (a full page of A4) that's 2.7pt, which is nearly 1mm, enough to be clearly visible if you're trying to align one sample with another. Adobe Acrobat will treat

## 2.7.2 Hyphenation

$\LaTeX$  hyphenates automatically according to the language you use (see §2.7.6). To specify different breakpoints for an individual word, you can insert soft-hyphens (discretionary hyphens, done with `\-`) wherever you need them, for example:

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

To specify hyphenation points for all occurrences of a word, use the `\hyphenation` command in your preamble (see the panel ‘The Preamble’ on p.34) with one or more words in its argument, separated by spaces. This will even let you break ‘helicopter’ correctly. In this command you use normal hyphens, not soft-hyphens.

```
\hyphenation{helico-pter Popoca-tépetl im-mer-sion}
```

If you have frequent hyphenation problems with long, unusual, or technical words, ask an expert about changing the value of `\spaceskip`, which controls the flexibility of the space between words. This is not something you would normally want to do, as it can change the appearance of your document quite significantly.

If you are using a lot of unbreakable text (see next section and also §6.6.1) it may also cause justification problems. One possible solution to this is shown in §9.3.

## 2.7.3 Unbreakable text

To force  $\LaTeX$  to treat a word as unbreakable (the opposite of hyphenation), use the `\mbox` command: `\mbox{pneumonoultramicroscopicsilicovolcanoconiosis}`. This may have undesirable results, however, if you change margins: pneumonoultramicroscopicsilicovolcanoconiosis

To tie two words together with an unbreakable space (hard space), use a tilde (`~`) instead of the space. This will print as a normal 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`.

Note that a full point after a lowercase letter is treated as the end of a sentence, and creates more space before the next word. Here, after ‘Prof.’, 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.

## 2.7.4 Dashes

For a long dash — what printers call an ‘em rule’ like this — use three hyphens typed together, like `~---` this, and bind them 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 aesthetic. *Never* use a single hyphen for this purpose.

Between digits like page ranges (35–47), it is normal to use the short dash (what printers call an en-rule) which you get by typing two hyphens together, as in 35--47. If you want a minus sign, use math mode (§2.8).

## 2.7.5 Justification

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 between words. There is a special package called so (‘space-out’) if you need special effects like letter-spacing, but these are best left to the expert.

There are two commands `\raggedright` and `\raggedleft` which set ragged-right (ranged left) and ragged-left (ranged right). Use them inside a group (see the panel ‘Grouping’ on p.83) to confine their action to a part of your text.

These modes also exist as ‘environments’ (see §3.2) called `raggedright` and `raggedleft` which are more convenient when applying this formatting to a whole paragraph or more, like this one.

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

## 2.7.6 Languages

ℒ<sub>TEX</sub> can typeset in the native manner for several dozen languages. This affects hyphenation, word-spacing, indentation, and the names of the parts of documents used as headings (e.g. Table of Contents).

Most distributions of ℒ<sub>TEX</sub> come with US English and one or more other languages installed by default, but it is easy to add the `babel` package and specify any of the supported languages or variants, for example:

```
\usepackage[frenchb]{babel}
...
\selectlanguage{frenchb}
```

Changing the language with `babel` automatically changes the names of the structural units and identifiers like ‘Abstract’, ‘Index’, etc. to their translated version. For example, using French as above, chapters will start with ‘*Chapitre*’.

## 2.8 Mathematics

As explained in the Preface, ℒ<sub>TEX</sub> was originally written to automate the typesetting of books containing mathematics. The careful reader will already have noticed that mathematics is handled differently from normal text, which is why it has to be typeset specially. This document does not cover mathematical typesetting, which is explained in detail in many other books and Web pages, so all we will cover here is the existence of the math mode commands, and some characters which have special meaning, so they don’t trip you up elsewhere.

In addition to the 10 special characters listed in §2.4, there are three more characters which only have any meaning inside mathematics mode:

Key	Meaning
$\bar{v}$	Vertical bar
$\bar{<}$	Less-than
$\bar{>}$	Greater-than

If you type any of these in normal text (ie outside math mode), you will get very weird things happening and lots of error messages. If you need to print these characters, you *must* type them using math mode.

The hyphen also has an extra meaning in math mode: it typesets as a minus sign, so if you want to write about negative numbers you need to type the number in math mode so the minus sign and the spacing come out right.

To use math mode within a paragraph, enclose your math expression in `\ ( and \)` commands. You can get the much-quoted equation  $E = mc^2$  by typing `\(E=mc^2\)`, and to get a temperature like  $-30^\circ$  you need to type `\(-30\)^\circ`.<sup>6</sup>

To typeset a math expression as ‘displayed math’ (centered between paragraphs), enclose it in the commands `\[ and \]`.<sup>7</sup>

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

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

Displayed equations can be auto-numbered by using the `equation` environment instead of the `\[ and \]` commands.

<sup>6</sup>Bear in mind that the degree symbol is a non-ASCII character, so you must specify what input encoding you are using if you want to type it: see the example of the `inputenc` package in §2.6. If you don't want to use non-ASCII characters (or if you are using a system which cannot generate them), you can use the command `\textdegree` to get the degree sign.

<sup>7</sup>You will also see dollar signs used for math mode. This is quite common but deprecated: it's what plain `TEX` used in the days before `LATEX`, and the habit got ingrained in many mathematicians. It still works as a convenient shorthand like `$x=y$`, as do double-dollars for display-mode math like `$$E=mc^2$$`, but they are only mentioned here to warn readers seeing them in other authors' work that `\(...\)` and `\[...\]` are the proper `LATEX` commands.

## CHAPTER III

# Basic document structures

LaTeX's approach to formatting is to aim for consistency. This means that as long as you identify each *element* of your document correctly, it will be typeset in the same way as all the other elements like it, so that you achieve a professional finish with minimum effort.

Elements are the component parts of a document. The popular structure of parts, chapters, sections, subsections, subsubsections, paragraphs, lists, tables, figures, and so on is familiar to everyone from reading books, newspapers, magazines, reports, articles, and other classes of documents.

To achieve this consistency, every LaTeX document must start by declaring what *document class* it belongs to.

### 3.1 The Document Class Declaration

To tell LaTeX what class of document you are going to create, you type a special first line into your file which identifies it.<sup>1</sup> To start a report, for example, you would type the `\documentclass` command like this:

```
\documentclass{report}
```

There are four default classes provided, and many others are available online:

**report** for business, technical, legal, academic, or scientific reports;

**article** for magazine or journal articles, reviews, conference papers, or research notes;

**book** for books and theses;

**letter** for letters.<sup>2</sup>

The `article` class in particular can be used (some would say 'abused') for almost any short piece of typesetting by simply omitting the titling and layout (see below).

The default classes are normally only used for compatibility when exchanging documents with other LaTeX users, as they come with every copy of LaTeX and are therefore guaranteed to format identically everywhere. For most purposes, especially for publication, you can download replacements or extensions of these classes:

- The `memoir` and `komascript` packages contain sophisticated replacements for all the default classes;
- Many academic and scientific publishers provide their own special class files for articles and books (often on their Web sites for download);

<sup>1</sup>Readers familiar with SGML, HTML, or XML will recognize the concept as similar to the Document Type Declaration.

<sup>2</sup>The default `letter` class is rather idiosyncratic: there are much better ones you can download and install yourself, such as the `memoir` and `komascript` packages.

- Conference organisers may also provide class files for authors to write papers for presentation;
- Many universities provide their own thesis document class files in order to ensure exact fulfillment of their formatting requirements;
- Businesses and other organizations can provide their users with corporate stylesheets on a central server and configure  $\LaTeX$  installations to look there first for class files and packages.

It is worth noting that some of the default document classes are intended for final printing on different sizes of paper than A4 or Letter. Books and journals are not usually printed on office-size paper, so printing them on standard office stationery makes them look odd: the margins are too wide, or the positioning is unusual, or the font size is too small. These default classes are adequate for drafts or for sending to another  $\LaTeX$  user (because you know their system will understand them) but they are not really sufficient for professional publishing purposes. For this you need a style file designed by the publisher to fit their series of publications (quite often based on the default classes, but looking very different). As noted earlier, the memoir and komascript packages provide good facilities for helping design these.

### 3.1.1 Document class options

The default layouts are designed for US ‘Letter’ size paper.<sup>3</sup> To create documents with the correct proportions for standard A4 paper, you need to specify the paper size in an optional argument in square brackets before the document class name, e.g.

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

The other default settings are for: *a*) 10pt type (all document classes); *b*) two-sided printing (books and reports) or one-sided (articles and letters); and *c*) separate title page (books and reports only). These can be modified with the following document class options which you can add in the same set of square brackets, separated by commas:

11pt to specify 11pt type (headings, footnotes, etc. get scaled up or down in proportion);

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

oneside to format one-sided printing for books and reports;

twoside to format articles for two-sided printing;

titlepage to force articles to have a separate title page;

draft makes  $\LaTeX$  indicate hyphenation and justification problems with a small square in the right-hand margin of the problem line so they can be located quickly by a human.

There are extra preset options for other type sizes which can be downloaded separately (10pt, 11pt, and 12pt between them cover probably 95% of all document typesetting). In addition there are hundreds of add-in packages which can automate other layout and formatting variants without you having to program anything by hand or even change your text.

---

<sup>3</sup>Letter size is  $8\frac{1}{2}'' \times 11''$ , which is the trimmed size of the old Demi Quarto, still in use in North America. The other common US office size is ‘Legal’, which is  $8\frac{1}{2}'' \times 14''$ , a bastard cutting, but close to the old Foolscap ( $8\frac{3}{4}'' \times 13\frac{3}{4}''$ ). ISO standard paper sizes are still virtually unknown in North America.

## EXERCISE 1

**Create a new document**

1. Use your editor to create a new document.
2. Type in the Document Class Declaration as shown above.
3. Add a font size option if you wish.
4. In North America, omit the `a4paper` option or change it to `letterpaper`.
5. Save the file (make up a name) ensuring the name ends with `.tex`.

## 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}
```

(You would put your text where the dots are.) The reason for marking off the beginning of your text is that  $\LaTeX$  allows you to insert extra setup specifications before it (where the blank line is in the example above: we'll be using this soon). The reason for marking off the end of your text is to provide a place for  $\LaTeX$  to be programmed to do extra stuff automatically at the end of the document. We're not going to do either of those yet, though.

A useful side-effect of marking the end of the document text is that you can store comments or temporary text underneath the `\end{document}` in the knowledge that  $\LaTeX$  will never try to typeset them.

This `\begin... \end` pair is an example of a common  $\LaTeX$  structure called an *environment*. Environments enclose text which is to be handled in a specific way. All environments start with `\begin{...}` and end with `\end{...}` (putting the name of the environment in the curly braces).

## EXERCISE 2

**Add a document environment**

1. Add the **document** environment to your file.
2. Leave a blank line between the Document Class Declaration and the `\begin{document}` (you'll see why later).
3. Save the file.

## 3.3 Titling

The first thing you put in the **document** environment is almost always your document title, the author's name, and the date (except in letters, which have a special set of commands for addressing which we'll look at later).

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

\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2004}
\maketitle

\end{document}
```

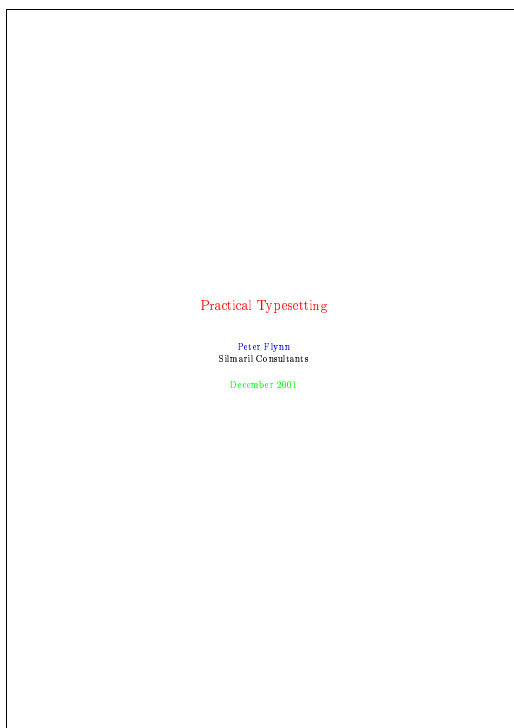
The `\title`, `\author`, and `\date` commands should be self-explanatory. The double backslash (`\\`) means a forced linebreak. You always finish the title block with the `\maketitle` command, otherwise the titling will never be typeset. The `\maketitle` command is reprogrammable so you can alter the appearance of titles on a consistent basis.

#### EXERCISE 3

##### Adding the title block

1. Add a `\title`, `\author`, `\date`, and `\maketitle` command to your file.
2. Use your own name, make up a title, and give a date.
3. Add two backslashes after your name and add your organisation or department name.
4. The order of the first three commands is not important, but the `\maketitle` command must come last.

When this file is typeset, you get something like this (I've cheated and done it in colour (§ 5.1.1) for fun — yours will be in black and white for the moment):





However, before we see how to get this displayed or printed, there are a few more elements to cover: abstracts, sectioning, the Table of Contents, and paragraphs. If you're really impatient, though, refer to Chapter 4 to see how to typeset and display it.

### 3.4 Abstracts and summaries

In reports and articles it is normal for the author (you) to provide an Abstract or Summary, in which you describe briefly what you have written about and explain its importance. Abstracts in articles are usually only a few paragraphs long; Summaries in reports can run to several pages, depending on the length and complexity of the report itself.

In both cases the Abstract or Summary is theoretically optional (that is,  $\LaTeX$  doesn't force you to have one), but almost always included. In practice, of course, you go back and type the Abstract or Summary *after* having written the rest of the document, but for the sake of the example we'll jump the gun and type it now.

Immediately after the `\maketitle` you can use the `abstract` environment, in which you simply type your Abstract or Summary, leaving a blank line between paragraphs (see §3.6 for this convention).

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

\title{Practical Typesetting}
\author{Peter Flynn\Silmaril Consultants}
\date{December 2004}
\maketitle

\begin{abstract}
This document 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 documents, the Abstract is often called a Management Summary, or Executive Summary, or Business Preview, or some similar phrase.  $\LaTeX$  lets you change the name associated with the `abstract` environment to any kind of title you want, using the `\renewcommand` command to give the `\abstractname` a new value:

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

Notice how the name of the command you are renewing (`\abstractname`) goes in the first set of curly braces, and the new value you want it to have goes in the

second set of curly braces. The environment you use is still called **abstract** (that is, you still type `\begin{abstract}...\end{abstract}`): what the `\abstractname` does is change the name that gets displayed and printed, not the name of the environment you store the text in.

## EXERCISE 4

**Using an Abstract or Summary**

1. Add the `\renewcommand` line given above to your Preamble.  
The Preamble is at the start of the document, in that gap *after* the `\documentclass` line but *before* the `\begin{document}` (remember I said we'd see what we left it blank for: see the panel 'The Preamble' on p.34).
2. Add an **abstract** environment after the `\maketitle` and type in a couple of short paragraphs of text.
3. Save the file (no, I'm not paranoid, just careful).

If you look carefully at the example Abstract above, you'll see I added an extra command to the Preamble (`\usepackage[latin1]{inputenc}`). We'll see later what this means (Brownie points for guessing it, though, if you read §2.6).

**The Preamble**

Modifications which you want to affect a whole document go at the very start of your  $\text{\LaTeX}$  file, immediately after the `\documentclass` line and before the `\begin{document}` line:

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

This position, between the Document Class Declaration and the beginning of the **document** environment, is called the *preamble*, and it is used for small or temporary modifications to the style and behaviour of the document. Major or permanent modifications should go in a `.sty` file and be invoked with a `\usepackage` command.

## 3.5 Sections

In the body of your document,  $\text{\LaTeX}$  provides seven levels of division or 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 unstructured text. Even novels are normally divided into chapters, although short stories are often made up solely 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 meaning in articles and letters.<sup>4</sup>

<sup>4</sup>It is arguable that chapters also have no place in reports, either, as these are conventionally divided into sections as the top-level division.  $\text{\LaTeX}$ , however, assumes your reports have chapters, but this is only the default, and can be changed very simply (see Chapter 9).

Depth	Division	Command	Notes
-1	Part	<code>\part</code>	Only in books and reports
0	Chapter	<code>\chapter</code>	Only in books and reports
1	Section	<code>\section</code>	Not in letters
2	Subsection	<code>\subsection</code>	Not in letters
3	Subsubsection	<code>\subsubsection</code>	Not in letters
4	Titled paragraph	<code>\paragraph</code>	Not in letters
5	Titled subparagraph	<code>\subparagraph</code>	Not in letters

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. You can turn section numbering off at a specific depth: details in §3.5.1.

```
\section{New recruitment policies}
...
\subsection{Effect on staff turnover}
...
\chapter{Business plan 2004--2006}
```

There are packages<sup>5</sup> to let you control the typeface, style, spacing, and appearance of section headings: it's much easier to use them than to try and reprogram the headings manually. Two of the most popular are the `ssection` and `sectsty` packages.

Headings also get put automatically into the Table of Contents, if you specify one (it's optional, and we haven't done that yet: see §3.7). But if you make manual styling changes to your heading, for example a very long title, or some special line-breaks or unusual font-play, this would appear in the Table of Contents as well, which you almost certainly don't want.  $\LaTeX$  allows you to give an optional extra version of the heading text which only gets used in the Table of Contents and any running heads, if they are in effect (§8.1.2). This goes in [square brackets] before the curly braces:

```
\section[Effect on staff turnover]{The effect of the revised
recruitment policies on staff turnover at divisional
headquarters}
```

#### EXERCISE 5

##### Start your document text

1. Add a `\chapter` command after your Abstract or Summary, giving the title of your first chapter.
2. If you're planning ahead, add a few more for chapters 2, 3, etc. Leave a few blank lines between them to make it easier to add paragraphs of text later.
3. I shouldn't need to tell you what to do after making significant changes to your document file.

### 3.5.1 Section numbering

All document divisions get numbered by default. Parts get Roman numerals (Part I, Part II, etc.); chapters and sections get decimal numbering, and Appendixes (which are just a special case of chapters, and share the same structure) are lettered (A, B, C, etc.).

<sup>5</sup>Details of how to use  $\LaTeX$  packages are in §5.1.

You can change the depth to which section numbering occurs, so you can turn it off selectively. In this document it is set to 3. If you only want parts, chapters, and sections numbered, not subsections or subsubsections etc., you can change the value of the `secnumdepth` counter using the `\setcounter` command, giving the depth value from the table in §3.5:

```
\setcounter{secnumdepth}{1}
```

A related counter is `tocdepth`, which specifies what depth to take the Table of Contents to. It can be reset in exactly the same way as `secnumdepth`. The current setting for this document is 2.

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 divisional commands from `\part*` to `\ subparagraph*` have this ‘starred’ version which can be used on special occasions for an unnumbered heading when the setting of `secnumdepth` would normally mean it would be numbered.

### 3.6 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. Now read this paragraph again.

The spacing between paragraphs is a separately definable quantity, a *dimension* or *length* called `\parskip`. This is normally zero (no space between paragraphs, because that’s how books are normally typeset), but you can easily set it to any size you want with the `\setlength` command in the Preamble:

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

This will set the space between paragraphs to 1cm. See §2.7.1 for details of the various size units  $\LaTeX$  can use. *Leaving multiple blank lines between paragraphs in your source document achieves nothing*: all extra blank lines get ignored by  $\LaTeX$  because the space between paragraphs is controlled only by the value of `\parskip`. To change the space between paragraphs, specify it with the command as shown above.

White-space in  $\LaTeX$  can also be made flexible (what  $\LaTeX$ : *A Document Preparation System*<sup>6</sup> calls ‘rubber’ lengths). This means that values such as `\parskip` can have a default dimension plus an amount of expansion minus an amount of contraction. This is useful on pages in complex documents where not every page will be an exact number of fixed-height lines long, so some give-and-take in vertical space is useful. You specify this in a `\setlength` command like this:

```
\setlength{\parskip}{1cm plus4mm minus3mm}
```

Paragraph indentation can also be set with the `\setlength` command, although you would always make it a fixed size, never a flexible one, otherwise you would have very ragged-looking paragraphs.

---

<sup>6</sup>Lamport (1994)

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

By default, the first paragraph after a heading follows the standard Anglo-American publishers' practice of *no* indentation. Subsequent paragraphs are indented by the value of `\parindent` (default 18pt).<sup>7</sup> You can change this in the same way as any other length.

In the printed copy of this document, the paragraph indentation is set to 10.0pt and the space between paragraphs is set to 0.0pt plus 1.0pt. These values do not apply in the Web (HTML) version because not all browsers are capable of that fine a level of control, and because users can apply their own stylesheets regardless of what this document proposes.

#### EXERCISE 6

##### Start typing!

1. Type some paragraphs of text. Leave a blank line between each. Don't bother about line-wrapping or formatting —  $\TeX$  will take care of all that.
2. If you're feeling adventurous, add a `\section` command with the title of a section within your first chapter, and continue typing paragraphs of text below that.
3. Add one or more `\setlength` commands to your Preamble if you want to experiment with changing paragraph spacing and indentation.

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

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

If you do this, though, and leave `\parskip` set to zero, your readers won't be able to tell easily where each paragraph begins! If you want to use the style of having no indentation with a space between paragraphs, use the `parskip` package, which does it for you (and makes adjustments to the spacing of lists and other structures which use paragraph spacing, so they don't get too far apart).

## 3.7 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, just add the command `\tableofcontents` at the point where you want it printed (usually after the `\maketitle` command and before the Abstract or Summary).

Entries for the ToC are recorded each time you process your document, and reproduced the *next* time you process it, so you need to re-run  $\TeX$  one extra time to ensure that all ToC page-number references are correctly calculated.

We've already seen in §3.5 how to use the optional argument to the sectioning commands to add text to the ToC which is slightly different from the one printed in the body of the document. It is also possible to add extra lines to the ToC, to force extra or unnumbered section headings to be included.

<sup>7</sup>Paragraph spacing and indentation are cultural settings. If you are typesetting in a language other than English, you should use the `babel` package, which alters many things, including the spacing and the naming of sections, to conform with the standards of different countries and languages.

## EXERCISE 7

**Inserting the table of contents**

1. Go back and add a `\tableofcontents` command after the `\maketitle` command in your document.
2. You guessed.

The commands `\listoffigures` and `\listoftables` work in exactly the same way as `\tableofcontents` to automatically list all your tables and figures. If you use them, they normally go after the `\tableofcontents` command.

The `\tableofcontents` command normally shows only numbered section headings, and only down to the level defined by the `tocdepth` counter (see §3.5.1), but you can add extra entries with the `\addcontentsline` command. For example if you use an unnumbered section heading command to start a preliminary piece of text like a Foreword or Preface, you can write:

```
\subsection*{Preface}  
\addcontentsline{toc}{subsection}{Preface}
```

This will format an unnumbered ToC entry for 'Preface' in the 'subsection' font style. You can use the same mechanism to add lines to the List of Figures or List of Tables by substituting `lof` or `lot` for `toc`.

## CHAPTER 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 plain-text file with a filetype of `.tex` and a name of your own choosing.

### EXERCISE 8

#### **Saving your file**

If you haven't already saved your file, do so now (some editors and interfaces let you typeset the document without saving it!).

Pick a sensible filename in a sensible directory. Names should be short enough to display and search for, but descriptive enough to make sense. See the panel 'Picking suitable filenames' on p.41 for more details.

## 4.1 Typesetting

Typesetting your document is usually done with by clicking on a button in a toolbar or an entry in a menu. Which one you click on depends on what output you want.

- The standard (default)  $\LaTeX$  program produces a device-independent (DVI) file which can be used with any  $\TeX$  viewer or printer driver on any make or model of computer (there are dozens of these available: at least one of each (viewer and printer driver) should have been installed with your distribution of  $\TeX$ ).
- The *pdflatex* program produces an Adobe Acrobat Portable Document Format (PDF) file which can be used with any suitable viewer, such as *GSview*, *PDFview*, the *Opera* browser, or Adobe's own *Acrobat Reader*.

Depending on which one you choose, you may have to [re]configure your editor so that it runs the right program. They can all do all of them, but they don't always come pre-set with buttons or menus for every possible option, because they can't guess which one you want.

### 4.1.1 Standard $\LaTeX$

There are two ways of running  $\LaTeX$ : from the toolbar or menu, or from the command line. Toolbars and menus are most common in graphical systems, and are the normal way to run  $\LaTeX$ . Command lines are used in non-graphical systems and in automated processes where  $\LaTeX$  is run unattended (so-called 'batch' or 'scripted' processing).

## EXERCISE 9

**Running  $\LaTeX$  from the toolbar or menu**

Run  $\LaTeX$  on your file according to which system you're using:

- In *WinEdt*, click the  $\LaTeX$  toolbar icon;
- In *Emacs*, click the  $\LaTeX$  File menu item.

Your editor may suggest you save your file if you haven't already done so. Do it.

Whichever way you run  $\LaTeX$ , it will process your file and display a log or record of what it's doing (see the example in Exercise 10: it looks the same no matter what system you use). This is to let you see where (if!) there are any errors or problems.

*Don't panic if you see errors:* it's very common for learners to mistype or mis-spell commands, forget curly braces, type a forward slash instead of a backslash, or use a special character by mistake. Errors are easily spotted (lines in the log window beginning with an exclamation mark) and easily corrected in your editor, and you can run  $\LaTeX$  again to check you have fixed everything. §4.2 describes some of the most common mistakes and suggests how to fix them.

When  $\LaTeX$  finds an error (a real one, not just a warning), it displays an error message and pauses. You must type one of the following letters to continue:

---

**Key Meaning**

- $\boxed{x}$  Stop immediately and exit the program.
- $\boxed{q}$  Carry on quietly as best you can and don't bother me with any more error messages.
- $\boxed{e}$  Stop the program but re-position the text in my editor at the point where you found the error.
- $\boxed{h}$  Try to give me more help.
- $\boxed{i}$  (followed by a correction) input the correction in place of the error and carry on.

The log will also tell you if there were problems with overlong or underrun lines (unusual hyphenations, for example), pages running short or long, and other typographical niceties (most of which you can ignore until later).

Unlike other systems, which try to hide unevennesses in the text — usually unsuccessfully — by interfering with the letter-spacing,  $\LaTeX$  takes the view that the author or editor should be able to contribute. While it is certainly possible to set  $\LaTeX$ 's parameters so that the spacing is sufficiently sloppy that you will almost never get a warning about badly-fitting lines or pages, you will almost certainly just be delaying matters until you start to get complaints from your readers or publishers.

In *WinEdt* the log appears in a separate window and you have to press the  $\boxed{\text{Enter}}$  key to dismiss it after checking it. In *Emacs*, the log appears in the bottom half of the edit window and can be dismissed with the  $\boxed{\text{Files One Window}}$  menu or by pressing  $\boxed{\text{Ctrl-X}} \boxed{1}$ .

If there were no errors, your file is ready for displaying or printing.



### Picking suitable filenames

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 with  $\TeX$ .

Make filenames as short or as long as you wish, but strictly avoid spaces. Stick to upper- and lower-case letters without accents (A–Z and a–z), the digits 0–9, the hyphen (-), and the full point or period (.), (similar to the conventions for a Web URI): it will let you refer to  $\TeX$  files over the Web more easily and make your files more portable.

## 4.1.2 pdf $\LaTeX$

If you have configured your editor to generate PDF files direct instead of DVI files, by using the *pdflatex* program instead of standard  $\LaTeX$ , then you click the pdf $\LaTeX$  icon in *WinEdt* or type the `pdflatex` command in a terminal (console) window (exactly the same method as shown above for standard  $\LaTeX$ ). *Emacs* does not have a default menu configured for *pdflatex*.

## 4.1.3 Running $\LaTeX$ from a command window

This is worth practising even if you normally use a graphical access system, so that you understand what it does.

### EXERCISE 10

#### Running $\LaTeX$ in a terminal or console window

- If you are using a non-graphical system, then by definition you are already using a command-line terminal or console.
- Under Linux or Unix (X Window systems such as KDE, Gnome, OpenWindows, CDE, etc.) you open a command (shell) window by clicking on the shell or screen icon in the control panel at the bottom of your screen.
- Under Microsoft Windows you open a command window by clicking on the **Start** Programs **MS-DOS** or **Start** Command Prompt menu item.

When the command window appears, type

```
cd documents
```

Substitute the name of your document directory (folder) for `documents` in the `cd` command, and your document filename for `mybook` in the `latex` command. Remember to press the **Enter** key at the end of each line.

You can run *pdflatex* in the same way by typing `pdflatex` instead of `latex`.

```

peter@bealtaine:~/doc - Konsole - Konsole
File Sessions Settings Help
$ cd /home/peter/doc
$ latex practype
This is TeX, Version 3.14159 (Web2C 7.3.1)
<practype.tex
LaTeX2ε <2000/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german, i
talian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/report.cls
Document Class: report 2000/05/19 v1.4b Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size11.clo)
(/usr/share/texmf/tex/latex/base/inputenc.sty
(/usr/share/texmf/tex/latex/base/latin1.def))
No file practype.aux.
[1] [1] <practype.aux>
Output written on practype.dvi (2 pages, 1396 bytes).
Transcript written on practype.log.
$

```

## 4.2 Error messages

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

Only a few common error messages are given here: those most likely to be encountered by beginners. If you find another error message, and it's not clear what you should do, ask for help.

Fortunately it's usually easy to find  $\LaTeX$  errors, as the layout of an error message is always the same. Error messages begin with an exclamation mark at the start of the line in the log, and give a description of the error, followed by a line starting with the line number in your document file where the error was spotted.

Newcomers should remember to check the list of special characters in (§2.4): a very large number of errors when you are learning  $\LaTeX$  are due to accidentally typing a special character when you didn't mean to. This disappears after a few days as you get used to them.

```

! Too many }'s.
1.6 \date December 2004}

```

In the example above, the reason  $\LaTeX$  thinks 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 (which it is!).<sup>1</sup>

```

! Undefined control sequence.
1.6 \dtae
    {December 2004}

```

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 fact: all  $\LaTeX$  knows is that it's undefined).

<sup>1</sup>In fact, there are other things which can follow the `\date` command apart from a date in curly braces, so  $\LaTeX$  cannot possibly guess that you've missed out the opening curly brace — until it finds a closing one!

```
Runaway argument?
{December 2004 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
      \par
1.8
```

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.

I'll repeat the advice from earlier: if you find an error message you can't understand, ask for help. See the section on online help (§5.3) for details.

```
Underfull \hbox (badness 1394) in paragraph at lines 28--30
[] []\LY1/brm/b/n/10 Bull, RJ: \LY1/brm/m/n/10 Ac-count-ing in Busi-
[94]
```

In this example of a warning,  $\LaTeX$  cannot stretch the line wide enough without making the spacing bigger than its currently permitted maximum. The badness (0–10,000) indicates how severe this is (here you can probably ignore it). It says what lines of your file it was typesetting when it found this, and the number in square brackets is the number of the page onto which the offending line was printed.

The codes separated by slashes are the typeface and font style and size used in the line. Ignore them for the moment: details are in step 11 in the procedure on p. 93.

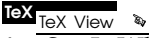

```
[101]
Overfull \hbox (9.11617pt too wide) in paragraph at lines 860--861
[]\LY1/brm/m/n/10 Windows, \LY1/brm/m/it/10 see \LY1/brm/m/n/10 X Win-
```

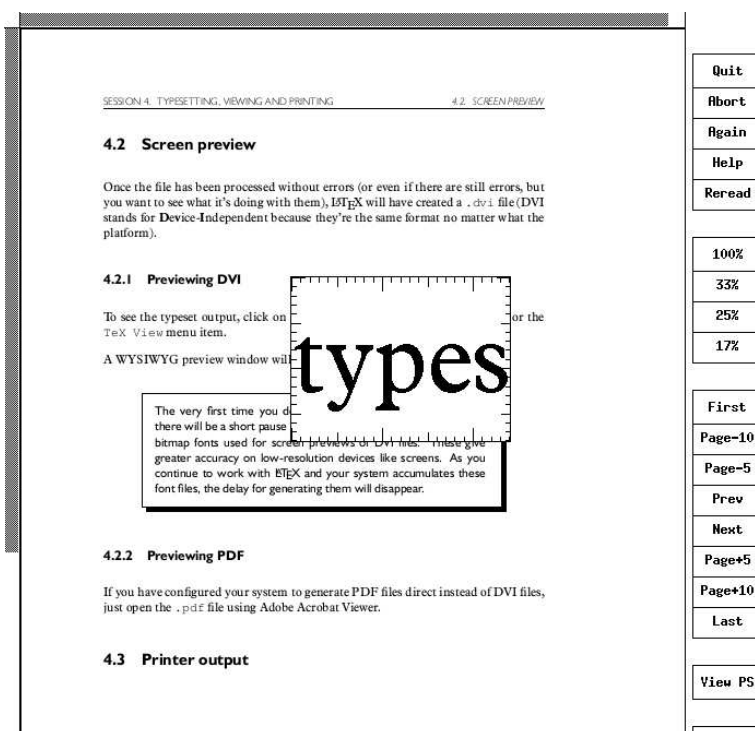
Last of all, the opposite warning: this line is too long by a shade over 9pt. The chosen hyphenation point which minimises the error is shown at the end of the line (*Win-*). Line numbers and page numbers are given as before. In this case, 9pt is too much to ignore, and a manual correction needs making, or the flexibility settings need changing (outside the scope of this booklet).

## 4.3 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 with the same name as your document but the filetype `.dvi`. If you're using *pdf $\LaTeX$* , a PDF file will have been created, and you can skip to §4.3.2.

### 4.3.1 Previewing DVI output

To see the typeset output, click on the  menu item in *Emacs* or the  menu item in *WinEdt*. A What You See Is What You Get (WYSIWYG) preview window will appear with your typeset display.



Most previewers have a wide range of scaling, zooming, and measuring functions, but remember this is a *picture* of your output: you cannot edit the image. To change it, you always edit your source text and reprocess.

With *xdvi* and its derivatives like *doiview*, you can leave the display window open, and after you've reprocessed your document through  $\text{\LaTeX}$ , moving your mouse back into the window will make the display update automatically (click your mouse if you have your windowing system set to need a click to focus).

The very first time you display your DVI output with a new installation of  $\text{\TeX}$ , there may be a short pause if the previewer needs to create the special bitmaps used for screen previews of some fonts. These give greater accuracy on low-resolution devices like screens. As you continue to work with  $\text{\TeX}$  and your system accumulates these font files, the pause for generating them will disappear.

### 4.3.2 Previewing with PDF

If you have configured your system to generate PDF files direct instead of DVI files, just open the `.pdf` file using any PDF viewer or browser.

It is possible to configure *WinEdt* to display a toolbar icon which will pop up *Acrobat Reader* or some other browser with the current PDF output file.

Adobe's *Acrobat Reader* cannot automatically update the view if you reprocess your document through *pdflatex*. You have to close the display with `Ctrl-W` and reload the file with `Alt-T`.

**Bitmap fonts**

Acrobat Reader is extremely poor at rendering Type3 bitmap fonts. If you are using these (mostly old  $\text{\TeX}$  installations who have not upgraded to Type1), you will see very fuzzy display at low magnifications. It will print perfectly, but Acrobat Reader's display is disappointing. The solution is to use a better viewer or to install and upgrade to the Type1 versions of the CM fonts (see § 8.3.3), or both, but there remain a few useful fonts which are still only available in Type3 format. If you need to use them, you probably need to warn your readers to expect a fuzzy display from Acrobat Reader (but good printout), and to change to a better reader if they can.

**4.3.3 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, a descendant of PostScript, is rapidly taking over, but PostScript itself is still extremely common, largely because it is very robust, and is usually an ASCII file, which makes it very portable and easy to generate. The drawback is the large size of PostScript files, especially if they contain bitmapped graphics.

The *dvips* program which comes with all  $\text{\TeX}$  systems is used to generate PostScript files directly from your DVI output. These .ps files can be viewed, printed, or sent to a platemaker or filmsetter.

An alternative to viewing the DVI file direct is therefore 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 `dvips` toolbar icon or menu entry and click on it.

It's very simple to do manually anyway: let's assume your  $\text{\TeX}$  file was called `mydoc.tex`, so processing it has created `mydoc.dvi`. Just type:

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

in a command window (see Exercise 10 on p. 41 for how to get one) 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 the PostScript interpreter *Ghostsript*, which should have been installed along with your whole  $\text{\TeX}$  system (if not, install both now: *GSview* is separately licensed and cannot be included in the  $\text{\TeX}$  Live distribution, so you have to download it yourself).

*GSview* can be set to watch the PostScript file and automatically update the display any time the file is changed, without you even having to click on the window.

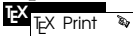
**4.4 Printer output**

$\text{\TeX}$  systems print on almost anything from the simplest dot-matrix printers to the biggest phototypesetters, including all the laser printers and a host of other devices in between. *How* you do it varies slightly according to how you do your typesetting and previewing:

**If you are using PDF** you can print directly from your PDF viewer (e.g. *PDFview*, *Acrobat Reader*, etc.). Be careful never to click on the 'Fit to page' option in your viewer, as it will change the size of your document so all your measurements become wrong.

**If you are using DVI** and you have a previewer which has a **print** function configured for your printer, you can use that. If not, create a PostScript file and use that (see below).

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 PostScript output directly from *dvips* to the printer without the need to open it in a previewer first. In *Emacs*, for example, this is what happens when you use the  menu item.

**Non-PostScript printers** You can create a PostScript file with *dvips* (see §4.3.3) and use *GSview* to print it (*GSview* can print PostScript files to almost any make or model of non-PostScript printer);

Or, if you want a non-PostScript/Ghostscript solution install or configure a T<sub>E</sub>X print driver for your printer (as supplied with your T<sub>E</sub>X 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 Epson, *dvihp* for Hewlett-Packard, *dviaw* for Apple LaserWriters, etc.). Configure the driver to print directly to the print queue, or pipe it to the print queue manually. On Linux with an HP printer, for example, this would be

```
dvihp mydoc | lpr
```

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

```
dvihp mydoc -o mydoc.hp  
print /b mydoc.hp
```

Read the documentation for the driver, as the options and defaults vary.

Both the *dvips* program and all the previewers that print tend to have facilities for printing selected pages, printing in reverse, scaling the page size, and printing only odd or even pages for two-sided work. If you are using PostScript there are programs for manipulating the output, for example to perform page imposition to get 4, 8, or 16 pages to a sheet for making booklets.

#### EXERCISE 11

##### Print it!

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

## CHAPTER V

# CTAN, packages, and online help

The Comprehensive T<sub>E</sub>X Archive Network (CTAN) is a repository or collection of Web and FTP servers worldwide which contain copies of almost every free piece of software related to T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X. CTAN is rooted at <http://www.ctan.org/> and there are several online indexes. There are complete T<sub>E</sub>X and L<sub>A</sub>T<sub>E</sub>X systems for all platforms, utilities for text and graphics processing, conversion programs into and out of L<sub>A</sub>T<sub>E</sub>X, extra typefaces, and (possibly the most important) the L<sub>A</sub>T<sub>E</sub>X *packages*.

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 (§5.3) until you have checked CTAN and FAQ first.

## 5.1 Packages

Add-on features for L<sub>A</sub>T<sub>E</sub>X are known as 'packages'. Dozens of these are pre-installed with L<sub>A</sub>T<sub>E</sub>X and can be used in your documents immediately. To find out what packages are available, you should use the CTAN search page<sup>1</sup> which includes a link to Graham Williams' comprehensive package catalogue.

When you try to typeset a document which requires a package which is not installed on your system, L<sub>A</sub>T<sub>E</sub>X will warn you with an error message and you can then download the package and install it using the instructions in §5.2. You can also download updates to packages you already have (ones that were installed along with your version of L<sub>A</sub>T<sub>E</sub>X as well as ones you added).

There is no theoretical limit to the number of packages you can have installed (apart from the size of your disk), but there is probably a physical limit to the number that can be used inside any one L<sub>A</sub>T<sub>E</sub>X document at the same time, 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, insert a `\usepackage` command in your document preamble with the package name in curly braces. For example, to use the color package, which lets you typeset in colours (I warned you this was coming!):

---

<sup>1</sup><http://www.ctan.org/search>

```

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

\title{Practical Typesetting}
\author{Peter Flynn\Silmaril Consultants}
\date{December 2004}
\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. However, if a package needs optional settings in square brackets, you *have* to give it its own `\usepackage` command, like `geometry` shown here.

```

\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 2004}
\maketitle

\end{document}

```

This is a simple way to do colours in titling on a once-off basis: if it's for a repeatable style we'll see in Chapter 9 how it can be automated and kept out of the author's way.

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.

#### EXERCISE 12

##### Add colour

Use the `color` package to add some colour to your document. Use the `geometry` package to change the margins.

Print your document if you have a colour printer (monochrome printers should print it in shades of grey).

### 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 directories full of `.dvi` files, one for every package installed. These can be previewed or printed like any other DVI file (see §4.3.1). If your installation procedure has not installed the documentation, it can all be downloaded from CTAN anyway.

Before using a package, you should read the documentation carefully, 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 create the documentation `.dvi` file for additional packages you install yourself.



## EXERCISE 13

**Read all about it**

Find and view (or print) the documentation on the geometry package you used in Exercise 12.

Investigate one of the other package documentation files in the directory.

## 5.2 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.

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

If the two files are not there, it means the package is part of a much larger bundle which you don't normally update unless you change version of  $\text{\LaTeX}$ . For example, there is no `color.dtx` and `color.ins` for the `color` package because it forms part of the `graphics` bundle. Such packages change very rarely, as they form part of the core of  $\text{\LaTeX}$  and are very stable. In general you should not try to update these packages in isolation.

Some rare or unusual packages are still supplied as a single `.sty` file intended for the now obsolete  $\text{\LaTeX}2.09$ . You can try to use these if you wish but they are not guaranteed to work, and have now almost all been replaced by  $\text{\LaTeX}2\epsilon$  versions. Always look for the `.dtx` and `.ins` pair of files first.

Download both files to a *temporary directory*. Keep something like `C:\tmp` for this in Windows; Linux systems already have a `/tmp` directory. There are four steps to installing a  $\text{\LaTeX}$  package:

### 1. Extract the files

Run  $\text{\LaTeX}$  on the `.ins` file. That is, open the file in your editor and process it as if it were a  $\text{\LaTeX}$  document, or if you prefer, type `latex` followed by the `.ins` filename in a command window in the temporary directory.

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 or print the names of the files created if there are a lot of them (read the log file if you want to see their names again).

### 2. Create the documentation

Run  $\text{\LaTeX}$  on the `.dtx` file twice. This will create a `.dvi` file of documentation explaining what the package is for and how to use it. Two passes through  $\text{\LaTeX}$  are needed in order to resolve any internal crossreferences in the text (a feature we'll come onto later). If you prefer to create PDF then run `pdflatex` instead. View or print this file in the usual manner (see §4.3 *et seq.*).

### 3. Install the files

While the documentation is printing, move or copy the files created in step 1 from your temporary directory to the correct place[s] in your  $\text{\TeX}$  installation directory tree. In an installation that conforms to the  $\text{\TeX}$  Directory Structure (TDS), this would be suitably-named subdirectories of `texmf-local/tex/latex/`. See the panel 'Replicating the TDS' on p.51 for how to create a parallel structure in your local directory, and the table on p.50 for where 'the right place' is.

Often there is just a `.sty` file to move but in the case of complex packages there may be more, and they may belong in different locations. For example, new  $\text{\BIBTeX}$  packages or font packages will typically have several files to install.

Type	Directory (under <code>texmf-local/</code> )	Comments
<code>.cls</code>	<code>tex/latex/base</code>	Document class file
<code>.sty</code>	<code>tex/latex/packageName</code>	Style file: the normal package content
<code>.bst</code>	<code>bibtex/bst</code>	BIB $\TeX$ style
<code>.mf</code>	<code>fonts/source/public/fontName</code>	METAFONT font outline
<code>.fd</code>	<code>tex/latex/mfnfss</code>	Font Definition files for METAFONT fonts only
<code>.fd</code>	<code>tex/latex/psnfss</code>	Font Definition files for PostScript Type 1 fonts only
<code>.pfb</code>	<code>/fonts/type1/foundry/fontName</code>	PostScript Type 1 font outline
<code>.afm</code>	<code>/fonts/afm/foundry/fontName</code>	Adobe font metrics for PostScript Type 1 fonts
<code>.tfm</code>	<code>/fonts/tfm/foundry/fontName</code>	$\TeX$ font metrics for METAFONT and PostScript Type 1 fonts
<code>.vf</code>	<code>/fonts/vf/foundry/fontName</code>	$\TeX$ virtual fonts
<code>.pdf, .dvi</code>	<code>/doc</code>	package documentation
others	<code>tex/latex/packageName</code>	other types of file unless instructed otherwise

For BIB $\TeX$  styles, the  $\LaTeX$  `.sty` files belong in `texmf-local/tex/latex/...` (called after the package) but the `.bst` file belongs in `texmf-local/bibtex/bst`.

Don't forget to move the `.dvi` or `.pdf` file of documentation into `texmf-local/doc`.

If there are configuration or other files, read the documentation to find out if there is a special or preferred location to move them to.

#### 4. Update your index

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

This last 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, such as old shared Unix systems or very old, small-scale PC systems, so there is no way for an installation program to guess where to put the files: *you* have to know this. There are many more systems where the owner, user, or installer has chosen *not* to follow the recommended TDS directory structure, or is unable to do so for security reasons (such as a shared system where she cannot write to a protected directory).

The reason for having the `texmf-local` directory (called `texmf.local` on some systems) is to provide a place for local modifications or personal updates if you are a user on a shared system (Unix, Linux, VMS, Windows NT/2000/XP, etc.) where you do not have write-access to the main  $\TeX$  installation directory tree. You can also have a personal `texmf` subdirectory in your own login directory. Your installation must be configured to look in these directories first, however, so that any updates to standard packages will be found there *before* the superseded copies in the main `texmf` tree. All modern  $\TeX$  installations should do this anyway, but if not, you can edit `texmf/web2c/texmf.cnf` yourself. There is an example in Appendix B.

## EXERCISE 14

**Install a package**

Download and install the `paralist` package (which implements inline lists).

**Replicating the TDS**

I find it useful to make the directory structure of `texmf-local` the same as that of `texmf`. Examine the subdirectories of `texmf/tex/latex/` for examples. For updates of packages which came with your  $\text{\TeX}$  distribution (as distinct from new ones you are adding yourself), you can then use the same subdirectory name and position in `texmf-local/...` as the original used in `texmf/...`

If you want to create the entire subdirectory structure ready for use, you can do it under Unix with the following command:

```
cd /usr/TeX/texmf; find . -type d -exec mkdir -p \
/usr/TeX/texmf-local/{ } \;
```

If you are using Microsoft Windows, you can download *Cygwin*, which provides you with the standard Unix tools in a shell window. The above command should also work on a Mac running OS X. In all cases, if your installation directory is not `/usr/TeX`, you need to substitute the actual paths to your `texmf` and `texmf-local` directories.

## 5.3 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 Frequently-Asked Questions (FAQ) document so that you avoid wasting online time asking about things for which there is already an easily-accessible answer. The FAQ is at <http://www.tex.ac.uk/faq/>.

The Usenet newsgroup `comp.text.tex` is the principal forum for questions and answers about  $\text{\TeX}$ . 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 URI into your browser's 'Location' or 'Address' window: `news:comp.text.tex` (if your browser doesn't support Usenet news properly, change it for one that does, like *Mozilla*<sup>2</sup>).

Another support resource is the mailing list `texhax@tug.org`. Again, feel free to ask questions, but again, try to answer the question yourself first (and say what you've tried in your message).

The  $\text{\TeX}$  Users Group, as well as most local user groups, maintains a web site (<http://tug.org>) with lots of information about various aspects of the  $\text{\TeX}$  system. See Appendix C for information on joining TUG.

If you need commercial levels of support, such as 24-hour phone contact, or macro-writing services, you can buy one of the several excellent commercial versions of  $\text{\TeX}$ , or contact a consultancy which deals with  $\text{\TeX}$  (details on the TUG Web site).

Remember: check the FAQ first!

<sup>2</sup><http://www.mozilla.org/>

## CHAPTER VI

# Other document structures

It is actually quite possible to write whole documents using nothing but section headings and paragraphs. As mentioned in §3.5, novels, for example, usually consist just of chapters divided into paragraphs. More commonly, however, you need other features as well, especially if the document is technical in nature or complex in structure.

It's worth pointing out that 'technical' doesn't necessarily mean 'computer technical' or 'engineering technical': it just means it contains a lot of *τηχνη* (*tekne*), the specialist material or artistry of its field. A literary analysis such as *La Textualisation de Madame Bovary*<sup>1</sup> (on the marginal notes in the manuscripts of Flaubert's *Madame Bovary*<sup>2</sup> manuscripts) is every bit as technical in the literary or linguistic field as the maintenance manual for an Airbus 300D is in the aircraft engineering field.

This chapter covers the most common features needed in writing structured documents: lists, tables, figures (including images), sidebars like boxes and panels, and verbatim text (computer program listings). In Chapter 7 we will cover footnotes, cross-references, citations, and other textual tools.

### 6.1 A brief note on structure

It's very easy to sit down at a keyboard with a traditional wordprocessor and just start typing. If it's a very short document, or something transient or relatively unimportant, then you just want to type it in and make it look 'right' by highlighting with the mouse and clicking on font styles and sizes.

In doing so, you may achieve the effect you wanted, but your actions have left no trace behind of *why* you made these changes. As I said, this is unimportant for trivial or short-term documents, but if you write longer or more complex documents, or if you often write documents to a regular pattern, then making them consistent by manual methods becomes a nightmare.  $\LaTeX$ 's facilities for automation are based on you providing this 'why' information.

If you can tick any of the features below about your documents, then you have already started thinking about structure.

- The document naturally divides into sections (parts, chapters, etc.).
- The document is long.
- There is lots of repetitive formatting in the document.
- The document is complex (intellectually or visually).
- There are lots of figures or tables (or examples, exercises, panels, sidebars, etc.).
- Accuracy is important in formatting the document.

---

<sup>1</sup>Mac Namara (2003)

<sup>2</sup>Flaubert (1857)

- A master copy is needed for future reference or reprinting.
- This is a formal or official document needing special care and attention.
- It's *my* thesis, book, leaflet, pamphlet, paper, article, etc. *That's* why I care.
- The document (or part of it) may need ongoing or occasional re-editing and republishing.

If you've got that far, you're over half-way done. Using a structural editor — even a simple outliner — can make a huge difference to the quality of your thinking because you are consciously organising your thoughts before setting them down. And it can make just as big a difference to your formatting as well: more consistent, better presented, easier for the reader to navigate through, and more likely to be read and understood — which is presumably why you are writing the document in the first place.

## 6.2 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 (when they are known as 'simple' or 'trivial' lists).

### 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 even be programmed to be hierarchical (1.a.viii, 2.3.6, etc.).

### Descriptive or labelled lists

(sometimes called 'discussion' lists), which are composed of subheadings or topic labels (usually unnumbered but typographically distinct), each followed by one or more indented paragraphs of discussion or explanation.

### Inline lists

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

There are actually two other types, segmented lists and reference lists, but these are much rarer. The structure of lists in  $\text{\LaTeX}$  is identical for each type, but with a different environment name. Lists are another example of this  $\text{\LaTeX}$  technique (environments), where a pair of matched commands surrounds some text which needs special treatment.

Within a list environment, list items are always identified by the command `\item` (followed by an item label in [square brackets] in the case of labelled lists). You don't type the bullet or the number or the formatting, it's all automated.

### 6.2.1 Itemized lists

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

```
\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}
```

- ⇒ Itemized lists usually have a bullet;
- ⇒ 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;
- ⇒ The bullet can be changed for any other symbol, for example from the `bbding` or `pifont` package.

See §9.6.1 for details of how to change the settings for list item bullets.

### 6.2.2 Enumerated lists

To create an enumerated list, use the **enumerate** environment:

```
\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.

See §6.2.6 for details of how to change the numbering schemes for each level.

### 6.2.3 Description lists

To create a description list, use the **description** environment:

```

\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 other 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 other lists;

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

### 6.2.4 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{\itshape Inline lists}, which are sequential in
nature, just like enumerated lists, but are
\begin{inparaenum}[\itshape a)]
\item formatted within their paragraph
\item usually 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, just like enumerated lists, but are *a*) formatted within their paragraph; and *b*) usually labelled with letters, like this example. The items are often Boolean, with the final item prefixed by 'and' or 'or'.

See Chapter 8 for details of font-changing commands.

#### EXERCISE 15

##### List practice

Add some lists to your document. Pick any two of the ones described here to practise with.

If you successfully installed `paralist` in Exercise 14 then you can use inline lists as described in § 6.2.4.

### 6.2.5 Reference lists and segmented lists

Reference lists are visually indistinguishable from numbered or lettered lists, but the numbering or lettering does *not* imply a sequence. The numbers or letters are just used as labels so that the items can be referred to from elsewhere in the text (as in ‘see item 501(c)3’). In this sense they are really a kind of sub-sectional division, and L<sup>A</sup>T<sub>E</sub>X’s `\paragraph` or `\subparagraph` commands (with appropriate renumbering) would probably be a far better solution than using a list. Label them and refer to them with `\label` and `\ref` as for any other cross-reference (see §7.4).

Segmented lists are a highly specialised structure and outside the scope of this document. For details of their usage, see the the chapter ‘Segmentation and Alignment’ in *Guidelines for the Text Encoding Initiative*<sup>3</sup>.

### 6.2.6 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.

Multiple embedded lists automatically change the bullet or numbering scheme so that the levels don’t get confused, and the spacing between levels is adjusted to become fractionally tighter for more deeply nested levels:

1. by default an outer enumerated list is numbered in Arabic numerals;
  - (a) an embedded enumerated list is lettered in lowercase;
    - i. a third level is numbered in lowercase Roman numerals;
      - A. the fourth level uses uppercase alphabetic letters.

But these are only defaults and can easily be changed by redefining the relevant set of values. You could also add a fifth and further levels, although I suspect that would mean your document structure needed some careful analysis, as lists embedded five deep will probably confuse your readers.

The values for lists come in pairs:<sup>4</sup> for each level there is a counter to count the items and a command to produce the label:

Level	Default	Counter	Label command
1	digit.	<code>enumi</code>	<code>\theenumi</code>
2	(letter)	<code>enumii</code>	<code>\theenumii</code>
3	roman.	<code>enumiii</code>	<code>\theenumiii</code>
4	LETTER.	<code>enumiv</code>	<code>\theenumiv</code>

Note that each counter and command ends with the Roman numeral value of its level (this is to overcome the rule that L<sup>A</sup>T<sub>E</sub>X commands can only be made of letters). To change the format of a numbered list item counter, just renew the meaning of its label:

```
\renewcommand{\theenumi}{\alpha{enumi}}
\renewcommand{\theenumii}{\roman{enumii}}
\renewcommand{\theenumiii}{\arabic{enumiii}}
```

<sup>3</sup>Burnard/Sperberg-McQueen (1995)

<sup>4</sup>In fact, any time you define a counter in L<sup>A</sup>T<sub>E</sub>X, you automatically get a command to reproduce its value. So if you defined a new counter *example* to use in a teaching book, by saying `\newcounter{example}`, that automatically makes available the command `\theexample` for use when you want to display the current value of *example*.



This would make the outermost list use uppercase letters, the second level use lowercase roman, and the third level use ordinary Arabic numerals. The fourth level would remain unaffected.

#### Lists: a caution to the unwary

Treat lists with care: people sometimes use tables for labelled 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 (usually to place the item label level with the description or explanation) *except* by using a table, hence they are misled into believing that their text is really a table when it's actually not.

#### EXERCISE 16

##### Nesting

Extend your use of lists by nesting one type inside a different one.

## 6.3 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 plain-text form, using a wordprocessor, using  $\LaTeX$ , using HTML or XML, using a DTP system, or some other text-handling package. Fortunately,  $\LaTeX$  provides a table model with a mixture of defaults and configurability to let it produce very high quality tables with a minimum of effort.

#### 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'). Sometimes you can get 'informal' tables, which simply occur between two paragraphs, without caption or number.

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

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

### 6.3.1 Floats

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 elsewhere 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, or by adjusting some of the parameters which control automatic floating.

Authors sometimes have too many floats occurring 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.

There is a `float` package which lets you create new classes of floating object (perhaps Examples or Exercises).

### 6.3.2 Formal tables

To create a Table, use the `table` environment containing a `\caption` command where you type the caption, and the `\label` command to give the Table a label by which you can refer to it.

```
\begin{table}
\caption{Project expenditure to year-end 2003}
\label{ye2003exp}
...
\end{table}
```

Numbering is automatic, but the `\label` command *must follow* the `\caption` command, not precede it. The numbering automatically includes the chapter number in document classes where this is appropriate (but this can of course be overridden). The `\caption` command has an optional argument to provide a short caption if the full caption would be too long for the List of Figures:

```
\caption[Something short]{Some very long caption that will
only look reasonable in the full figure.}
```

### 6.3.3 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, use the `tabular` environment. The `\begin{tabular}` command must be followed by a compulsory second argument in curly braces giving the alignment of the columns. These are specified for each column using one of single letters `l`, `c`, and `r` for left-aligned, centered, or right-aligned text, or the letter `p` followed by a width argument if you want a long entry to wrap to several lines (a miniature paragraph as a single cell).

$\TeX$ 's original tabular settings were designed for classical numerical tabulations, where each cell contains a single value. The `p` specification allows a cell to be a miniature paragraph set to a specific width. These are *not* multi-row entries, they are single cells which contain multiple lines of typesetting: the distinction is very important. Auto-adjusting cell sizes like you see in a Web browser (the 'Netscape' table model) are possible with the `tabularx` package, but these are often inelegant in print, however convenient they may be in a Web browser.

The `array` package provides for many other typographic variations such as left-aligned, right-aligned, and centred multi-line columns, and other packages provide decimal-aligned columns, the aforementioned row-spanning, multi-page, and rotated (landscape format) tables.

As an example, a tabular setting with three columns, the first one centered, the second left-aligned, and the third one right-aligned, would therefore be specified as `{c l r}`, as in the example below. Note the use of indentation to make the elements of the table clear for editing, and note also how the typeset formatting is unaffected by this (see Table 6.1).

Table 6.1: Project expenditure to year-end 2003

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
<b>Total</b>	<b>64,783</b>

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

```

\begin{table}
\caption{Project expenditure to year-end 2003}
\label{ye2003exp}
\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\\
&Total&64,783
\end{tabular}
\par\medskip\footnotesize
The Institute also contributes to (a) and (b).
\end{center}
\end{table}

```

You do not need to format the tabular data in your editor:  $\LaTeX$  does this for you when it typesets the table, using the column specifications you provided. Extra space is automatically added between columns, and can be adjusted by changing the `\tabcolsep` dimension. Takaaki Ota provides an excellent Tables mode for *Emacs* which provides a spreadsheet-like interface and can generate  $\LaTeX$  table source code (see Figure 6.1).

It is conventional to centre the tabular setting within the Table, using the **center** environment (note US spelling) or the **\centering** command. 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** command draws a rule across a range of columns (here, under column three only). If used, these commands *follow* the `\` of the row they apply to. There are some extra formatting commands after the tabular material in the example. These are explained in Chapter 8.

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. The exception to this is when the **\multicolumn** command is used to create cells which span multiple columns. There is also a package (**multirow**) to enable cells to span multiple rows, but both of these techniques are outside the scope of this document.

Figure 6.1: Tables mode for *Emacs*

### 6.3.4 Tabular techniques for alignment

As mentioned earlier, it's also perfectly possible to typeset tabular matter outside a formal Table, where you want to lay out an informal tabulation between paragraphs where a fully floating formal Table would be unnecessary (these are usually quite short: there are several of them in this document).

Tabular mode can also be used wherever you need to align material side by side, such as in designing letterheads, where you may want your company logo and address on one side and some other information on the other.

By default,  $\text{\LaTeX}$  typesets **tabular** environments inline to the surrounding text, so if you want your alignment displayed by itself, put it inside a positioning environment like **center**, **flushright**, or **flushleft**, or leave a blank line or `\par` before and after so it gets typeset separately.

There is much more to tabular setting for which there is no space here. Full details are in the manuals mentioned in the Foreword. One final note to remind you of the automated crossreferencing features: because the example table is labelled, it can be referenced from anywhere in the document as Table 6.1 just by using `\ref{ye2003exp}`, regardless of how much the surrounding document or structure is moved or edited.

#### EXERCISE 17

##### Create a tabulation

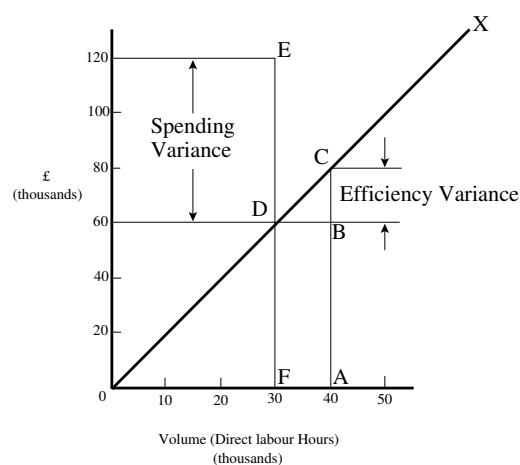
Create one of the following in your document:

- a formal Table with a caption showing the number of people in your class broken down by age and sex;
- an informal tabulation showing the price for three products;
- the logo 

YEAR
2000

 (hint: § 6.7.2)

Figure 6.2: Total variable overhead variance (after Bull (1972) p.191)



## 6.4 Figures

As explained in §6.3.1, Figures and Tables float to a vacant part of the page, as they are not part of the sequence of sentences making up your text, but illustrative objects that you refer to.

Figures can contain text, diagrams, pictures, or any other kind of illustration. To create a figure, use the **figure** environment: like Tables, they automatically get numbered, and must include a caption (with a label after the caption, if needed, exactly the same as for Tables)

```
\begin{figure}
\caption{Total variable overhead variance (after
\citeauthor[p.191]{bull})}
\label{workeff}
\begin{center}
\fbbox{\includegraphics[width=.5\columnwidth]{diagram}}
\end{center}
\end{figure}
```

You can see that the structure is very similar to the **table** environment, but in this case we have a graphic included. Details of this command (**\includegraphics**) are in the next section. Details of the bibliographic citation mechanism are in §7.4.2

The content of the Figure could of course also be textual, in the form of a list or a text diagram.  $\LaTeX$  has a simple drawing environment called **picture**, which lets you create a limited set of lines and curves, but for a diagram of any complexity, you should use a standard vector drawing program like *tkpaint*, *Adobe Illustrator*, or *Corel Draw*, for reasons we shall see in the next section.

## 6.5 Images

Images (graphics) can be included anywhere in a  $\LaTeX$  document, although in most cases of formal documents they will occur in Figures (see preceding section). To use **graphics**, you need to use the **graphicx** package in your preamble: `\usepackage{graphicx}`<sup>5</sup>

This enables the command **\includegraphics** which is used to insert an image in the document. The command is followed by the name of your graphics file *without*

<sup>5</sup>You may find a lot of old files which use a package called `epsf`. Don't. It's obsolete.

the *filetype*, for example: `\includegraphics{myhouse}` (we'll see soon why you don't include the *filetype*).

In most cases you should just make sure the image file is in the same folder (directory) as the document you use it in. This avoids a lot of messing around remembering where you put the files. If you have images you want to use in several different documents in different places on your disk, there is a way to tell  $\LaTeX$  where to look (see §6.5).

For standard  $\LaTeX$ , graphics files *must* be in Encapsulated PostScript (EPS) format: this has been the publishing industry standard for portable graphics for many years, and no other format will work portably in standard  $\LaTeX$ .<sup>6</sup>

All good graphics packages can save images as EPS, but be very careful because some packages, especially on Microsoft Windows platforms, use very poor quality drivers which create very poor quality EPS files. If in doubt, check with an expert. If you find an EPS graphic doesn't print, the chances are it's been badly made by the graphics software.

For *pdf $\LaTeX$* , graphics files can be in Joint Photographic Experts (JPG), Portable Network Graphic (PNG), or PDF format, *not* EPS. This means if you want to use both standard  $\LaTeX$  as well as *pdf $\LaTeX$* , you need to keep your graphics in two formats, EPS and one of the others. This is why you don't include the *filetype* in the filename you give with `\includegraphics`:  $\LaTeX$  will assume EPS and *pdf $\LaTeX$*  will look for JPG, PNG or PDF files matching the name.

It is in fact 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 `\includegraphics` command can take optional arguments within square brackets before the filename, e.g. `\includegraphics[width=3in]{myhouse}` to let you re-size the image to fit. You can change height or width and the other dimension will change to scale. If you specify both, the image will be distorted to fit.

For details of all the arguments, see the documentation on the `graphicx` package or a copy of the *The  $\LaTeX$  Companion*<sup>7</sup>. This package also includes commands to `atop`, `rotin`, and `scale` text. As mentioned before, always use a vector graphics package for creating drawings and diagrams, as these packages can save directly in EPS or PDF format, which means the drawing can be scaled to any size without loss of accuracy. Never, ever (except in the direst necessity) save any *diagram* as a bitmap (JPG, PNG, etc.) as these become blurred and jagged when scaled. Use PNG to save screenshots, to preserve the individual coloured dots (pixels), and use JPG for photographs. The Tagged Image File Format (TIFF), popular with graphic designers, should be avoided because far too many companies have designed and implemented non-standard, conflicting, proprietary extensions to the format, making it virtually useless for transfer between computers (except in faxes, where it's still used in a much stricter version).

<sup>6</sup>Some distributions of  $\TeX$  systems do allow other formats, such as Microsoft Bitmap (BMP) files, Hewlett-Packard's Printer Control Language (PCL) files, and others; but you cannot send such documents to other  $\LaTeX$  users and expect them to work if they don't have the same distribution installed as you have.

<sup>7</sup>Goossens/Mittelbach/Samarin (1993)

## EXERCISE 18

**Adding pictures**

Add `\usepackage{graphicx}` to the preamble of your document, and copy or download an image you want to include. Make sure it is a JPG, PNG, or PDF image if you use *pdf<sub>l</sub>atex*, or an EPS image if you use standard  $\text{\LaTeX}$ .

Add `\includegraphics` and the filename in curly braces (without the filetype), and process the document and preview or print it.

Make it into a figure following the example in § 6.4.

Be aware that some DVI viewers are not able to display all types of graphics, and some cannot display colour. For best results, use PDF or PostScript preview.

I said earlier that there was a way to tell  $\text{\LaTeX}$  where to look if you had stored images centrally for use in many different documents. The answer is in a command `\graphicspath` which you supply with an argument giving the name of an additional directory path you want searched when a file uses the `\includegraphics` command, for example:

```
\graphicspath{c:\mypict~1\camera}
\graphicspath{/var/lib/images}
\graphicspath{HardDisk:Documents:Reports:Pictures}
```

I've used the 'safe' (MS-DOS) form of the Windows MyPictures folder because it's A Bad Idea to use directory names containing spaces (see the panel 'Picking suitable filenames' on p.41). Using `\graphicspath` does make your file less portable, though, because file paths tend to be specific both to an operating system and to your computer, like the examples above.

## 6.6 Verbatim text

If you are documenting computer procedures, you probably need fixed-width type for examples of programming or data input or output. Even if you are writing about completely non-computer topics, you may often want to quote a URI or email address which needs to be typeset specially. It is particularly important in these two examples to avoid hyphenating them if they have to break over a line-end, because the hyphen might be taken by the user as a part of the address.

Standard  $\text{\LaTeX}$  includes two features for handling fixed-format text, and there are many more available in packages.

### 6.6.1 Inline verbatim

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. This is a very rare exception to the rule that arguments go in curly braces. I often use the plus sign for this, for example to show a  $\text{\LaTeX}$  command, I type `\verb+\includegraphics[width=3in]{myhouse}+` in order to display `\includegraphics[width=3in]{myhouse}`, but sometimes I use the *grave accent* (*backtick* or open-quote) or the vertical bar when the phrase already has a plus sign in it, like `\verb|\(y=a+2x^2\)|` when illustrating the  $\text{\LaTeX}$  equation `\(y=a+x^2\)`.

This command has the advantage that it turns off all special characters (see § 2.4) except the one you use as the delimiter, so you can easily quote sequences of characters in any computer syntax without problems. However,  $\text{\LaTeX}$  will never

break the argument of `\verb` at a line-end when formatting a paragraph, even if it contains spaces, so if it happens to be long, and falls towards the end of a line, it will stick out into the margin. See §2.7.2 for more information on line-ends and hyphenation.

The `url` package avoids this by providing the command `\url` which works in the same way as `\verb`, with the argument enclosed in a pair of characters, but performs a hyphenless break at punctuation characters, as in `http://www.ucc.ie/doc/ucc/siteowner.xml`. It was designed for Web URIs,<sup>8</sup> so it understands their syntax and will never break mid-way through an unpunctuated word, only at slashes and full points. Bear in mind, however, that spaces are forbidden in URIs, so using spaces in `\url` arguments will fail, as will using other non-URI-valid characters.

### 6.6.2 Display verbatim

For longer (multiline) 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 2004}
\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 provides a `Verbatim` environment (note the capital letter) which lets you draw a rule round the `verbatim` text, change the font size, and even have typographic effects inside the `Verbatim` environment. It can also be used in conjunction with the `fancybox` package (see below), and it can add reference line numbers (useful for chunks of data or programming), and it can include entire external files.

#### EXERCISE 19

##### Try some fixed-format text

Add your email address and home page URI using the `\verb` and `\url` commands. You'll need to `\usepackage{url}` for the latter.

If you know some programming, try a few lines enclosed in `verbatim` and `Verbatim` environments.

<sup>8</sup>The original term Uniform Resource Locator (URL) is strongly deprecated in the Web community in favour of the more accurate Uniform Resource Indicator (URI). For details see <http://www.w3.org/Addressing/>. Unfortunately the older term still persists, especially in  $\text{\LaTeX}$  and XML markup.



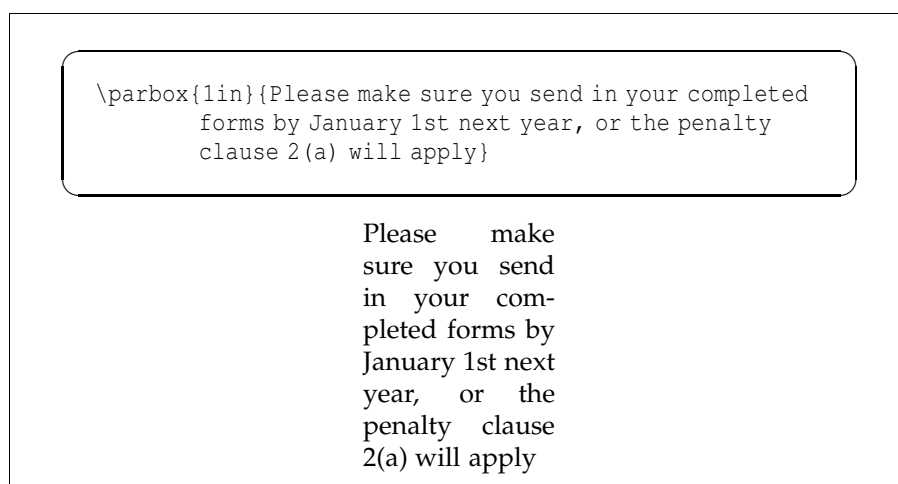
## 6.7 Boxes, sidebars, and panels

$\LaTeX$ , like most typesetting systems, works by setting text into boxes. The default box is the width of the current page, and works like an old compositor's galley (tray) from the days of metal type: it accumulates typeset text until it's a bit longer than the specified page height. At this stage  $\LaTeX$  works out how much of it really will fit on a page, snips it off and ships it out to the DVI or PDF file, and puts the rest back into the galley to accumulate towards the following page.

### 6.7.1 Boxes of text

Because of this 'box' model,  $\LaTeX$  can typeset any text into a box of any width wherever on the page you need it.

The simplest command for small amounts of text is `\parbox`. This command needs two arguments in curly braces: the first is the width you want the text set to, and the second is the text itself, for example:



The text is typeset to the required width, and the box is extended downwards for as long as is required to fit the text. Note that the baseline of a `\parbox` is set to the midpoint of the box; that is, if you include a `\parbox` in mid-sentence, the centre of the box will be lined up with the line of type currently being set.

You can specify that it should be the top or bottom by adding an optional `t` or `b` in square brackets before the width. For example, `\parbox[t]{1in}{...}` will produce a box with the baseline aligned with the top line of the text in the box.

Notice that when setting very narrow measures with type that is too large, the spacing may become uneven and there may be too much hyphenation. Either use `\raggedright` or reduce the type size, or (in extreme cases) reword the text or break each line by hand. It is rare for  $\LaTeX$  to need this: the example above was deliberately chosen to be obtuse as an illustration.

Where the contents is more extensive or more complicated, you can use the `minipage` environment. Within this you can use virtually everything that occurs in normal text (e.g. lists, paragraphs, tabulations, etc.) with the exception of floats like tables and figures. The `minipage` environment has an argument just like `\parbox` does, and it means the same: the width you want the text set to.

```

\begin{minipage}{3in}
Please make sure you send in your completed forms by January
1st next year, or the penalty clause 2(a) will apply.
\begin{itemize}
\item Incomplete forms will be returned to you unprocessed.
\item Forms must be accompanied by the correct fee.
\item There is no appeal. The adjudicators' decision is final.
\end{itemize}
\end{minipage}

```

Please make sure you send in your completed forms by January 1st next year, or the penalty clause 2(a) will apply.

- Incomplete forms will be returned to you unprocessed.
- Forms must be accompanied by the correct fee.
- There is no appeal. The adjudicators' decision is final.

Note that in `minipages` and `\parboxes`, the paragraph indentation is reset to zero. If you need to change it, set it inside the `minipage` or `\parbox` using the `\setlength` command (see §3.6).

There are two other ways of typesetting text to widths other than the normal text width: you can use a one-row, one-cell tabulation with the `p` column type specification, and you can use the `\vbox` command, which is outside the scope of this document.

### 6.7.2 Framed boxes

To put a frame round `some text`, use the `\fbox` command: `\fbox{some text}`. This works for a few words in mid-line, but the framed box and its contents won't break over the end of a line. To typeset multiline text in a box, put it in a `\parbox`, or use a `minipage` or `tabular` environment as described above, and enclose the whole thing in a `\fbox`.

```

\fbox{\begin{tabular}{p{1in}}
Multiline text in a box typeset using \textsf{tabular}
\end{tabular}}

```

Multiline text in a box typeset using tabular

Note the `\begin{tabular}` and `\begin{minipage}` still need the width specified in the normal way:

```

\fbbox{\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.<sup>a</sup>

---

<sup>a</sup>Like this.

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

```

\setlength{\fbboxsep}{1em}
\setlength{\fbboxrule}{2pt}

```

### 6.7.3 Sidebars and panels

The `fancybox` package lets you extend the principle of `\fbbox` with commands to surround text in square, oval (round-cornered), and drop-shadow boxes (e.g. `\ovalbox`, `\shadowbox`, etc.: see the documentation for details).

You can create panels of any size with these borders by using the `minipage` environment to typeset the text inside a special `Sbox` environment which `fancybox` defines. The `minipage` formats the text but the `Sbox` ‘captures’ it, allowing you to put the frame round it as it prints.

The printed version of this document has examples and there is a useful one shown in §9.5.

## CHAPTER VII

# Textual tools

Every text-handling system needs to support a repertoire of tools for doing things with text.  $\LaTeX$  implements many dozens, of which a small selection of the most frequently used is given here:

- quotations (sometimes called ‘block quotes’);
- footnotes and end-notes;
- marginal notes;
- cross-references, both normal ones and bibliographic citations;
- indexes and glossaries;
- typesetting multiple columns.

### 7.1 Quotations

Direct speech and short quotes within a sentence ‘like this’ are done with simple quotation marks as described in §2.5. Sometimes, however, you may want longer quotations set as a separate paragraph. Typically these are indented from the surrounding text, and often in a different font or size.  $\LaTeX$  has two environments for doing this:

**The quote environment** is for up to a line of text each per (short) quotation, with the whole thing indented from the previous paragraph but with no additional indentation on each quote;

```
\begin{quote}  
Do, Ronny, Do. \textit{Nancy Reagan}  
  
Da Do Ron Ron. \textit{The Crystals}  
\end{quote}
```

Do, Ronny, Do. *Nancy Reagan*

Da Do Ron Ron. *The Crystals*

**The quotation environment** is for longer passages (a paragraph or more) of a single quotation, where not only is the block of text indented, but each paragraph of it also has its own extra indentation on the first line.

```
\begin{quotation}\small
At the turn of the century William Davy, a Devonshire
parson, finding errors in the first edition of his
\citetitle{davy}, asked for a new edition to be
printed. His publisher refused and Davy purchased a
press, type, and paper. He harnessed his gardener to
the press and apprenticed his housemaid to the
typesetting. After twelve years' work, a new
edition of fourteen sets of twenty-six volumes was
issued---which surely indicates that, when typomania
is coupled with religious fervour, anything up to a
miracle may be achieved.\citequote[p.76]{ryder}
\end{quotation}
```

At the turn of the century William Davy, a Devonshire parson, finding errors in the first edition of his *A System of Divinity*<sup>a</sup>, asked for a new edition to be printed. His publisher refused and Davy purchased a press, type, and paper. He harnessed his gardener to the press and apprenticed his housemaid to the typesetting. After twelve years' work, a new edition of fourteen sets of twenty-six volumes was issued—which surely indicates that, when typomania is coupled with religious fervour, anything up to a miracle may be achieved. [Ryder, *Printing for Pleasure* (1976)], p.76

---

<sup>a</sup>Davy (1806)

Such quotations are often set in a smaller size of type (although this is not the default, hence the use of the `\small` command in the example). The inclusion of the bibliographic citation at the end is optional: here it is done with a non-standard command `\citequote` invented for this example (there is more about how to do things like this in Chapter 9).

## 7.2 Footnotes and end-notes

The command `\footnote`, followed by the text of the footnote in curly braces, will produce an auto-numbered footnote with a superior number where you put the command, and the note automatically printed at the foot of the page.<sup>1</sup> The number is reset to 1 at the start of each chapter (but you can override that and make them run continuously throughout the document, or restart at 1 on each page or section).

LaTeX automatically creates room for the footnote, and automatically reformats it if you change your document in such a way that the point of attachment and the footnote would move to the next (or preceding) page.

Because of the way LaTeX reads the whole footnote before doing anything with it, you can't use `\verb` (§6.6.1) alone in footnotes: either precede it with `\protect` or use [abuse?] the `\url` command instead, which you should be using for Web and email addresses in any case).

Footnotes inside minipages (see §6.7) produce lettered notes instead of numbered ones, and they get printed at the bottom of the minipage, *not* the bottom of the physical page (but this too can be changed).

There is a package to hold over your footnotes and make them print at the end of the chapter instead (`endnote`) or at the end of the whole document, and there is a package (`fnpara`) to print many short footnotes in several columns so they take up less space. It is also possible to have several separate series of footnotes active simultaneously, which is useful in critical editions or commentaries: a numbered series may be used

---

<sup>1</sup>Like this.

to refer to an original author's notes; a lettered series can be used for notes by a commentator or authority; and a third series is available for your own notes. It is also possible to format footnotes within footnotes.

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, where it might otherwise need to move to a different page.

## 7.3 Marginal notes

You can add marginal notes to your text instead of (or as well as) footnotes. You need to make sure that you have a wide-enough margin, of course: use the `geometry` package (see §5.1.1) to allocate enough space, otherwise the notes will be too cramped. There are several packages to help with formatting marginal notes, but the simplest way is to define it yourself. Add this new command to your preamble:

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

Then you can 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.

We're jumping ahead a bit here, as we haven't covered how to define your own commands yet. I won't even try to explain it here, although the careful reader can probably deduce some of it by inspection. See Chapter 9 for more information about making up your own commands.

## 7.4 Cross-references

This is one of the most powerful features of  $\LaTeX$ . You can label any point in a document with a name you make up, and then refer to it by that name from anywhere else in the document, and  $\LaTeX$  will always work out the cross-reference number for you, no matter how much you edit the text or move it around.

A similar method is used to cite documents in a bibliography or list of references, and there are packages to sort and format these in the correct manner for different journals.

### 7.4.1 Normal cross-references

You label a place in your document by using the command `\label` followed by a short name you make up, in curly braces:<sup>2</sup> we've already seen this done for labelling Figures and Tables.

```
\section{New Research}
\label{newstuff}
```

You can then refer to this point from anywhere in the same document with the command `\ref` followed by the name you used, e.g.

```
In section~\ref{newstuff} there is a list of recent projects.
```

<sup>2</sup>This section is labelled 'normalxref', for example.

If the label is in normal text, the reference will provide the current chapter or section number or both (depending on the current document class).<sup>3</sup> If the label was inside a Table or Figure, the reference provides the Table number or Figure number prefixed by the chapter number. A label in an enumerated list will provide a reference to the item number. If there is no apparent structure to this part of the document, the reference will be null. Labels must be unique (that is, each value must occur only *once* as a label within a single document), but you can have as many references to them as you like.

Note the use of the unbreakable space (~) between the `\ref` and the word before it. This prints a space but prevents the line ever breaking at that point, should it fall close to the end of a line.

The command `\pageref` followed by any of your label values will provide the page number where the label occurred, regardless of the document structure. This makes it possible to refer to something by page number as well as its `\ref` number, which is useful to readers in very long documents.

Unresolved references are printed as three question marks, and also cause a warning message at the end of the log file. There's never any harm in having `\labels` you don't refer to, but using `\ref` when you don't have a matching `\label` is an error.

## 7.4.2 Bibliographic references

The mechanism used for references to reading lists and bibliographies is almost identical to that used for normal cross-references. Although it is possible to type the details of each citation manually, there is a companion program to  $\text{\LaTeX}$  called  $\text{\BIBTeX}$ , which manages bibliographic references automatically, reduces the time needed to maintain and format them, and dramatically improves accuracy. Using  $\text{\BIBTeX}$  means you only ever have to type the bibliographic details of a work once. You can then cite it in any document you write, and it will get reformatted automatically to the style you specify.

### 7.4.2.1 Citing references

$\text{\BIBTeX}$  works exactly the same way as many other bibliographic databases: you keep details of every document you want to refer to in a separate file, using  $\text{\BIBTeX}$ 's own format (see example below). Many writers make a habit of adding the details of every book and article they read, so that when they write a document, these entries are always available for reference. You give each entry a short label, just like you do with normal cross-references (see §7.4.1), and it is this label you use to refer to in your own documents when you cite the work using the `\cite` command:

```
...as has clearly been shown by Fothergill~\cite{fg}.
```

By default, this creates a cross-reference number in square brackets [1] which is a common style in the Natural Sciences (see §7.4.2.5 for details of how to change this). There are dozens of alternative citation formats in extra packages, including the popular author/year format:

```
...as has clearly been shown by~\citeauthoryear{fg}.
```

```
...as has clearly been shown by Fothergill (1929).
```

<sup>3</sup>Thus I can refer here to `\ref{normalxref}` and get the value §7.4.1.

Note that in this case the author name is not typed, but automatically extracted by  $\text{\LaTeX}$ . There are lots of variants on this technique in many packages, allowing you to phrase your sentences with references in as natural a way as possible, and rely on  $\text{\LaTeX}$  to insert the right data (if you examine the source of this document you'll find I use some homebrew commands  $\text{\backslash authorof}$  and  $\text{\backslash titleof}$  for a similar purpose).

To print the bibliographic listing (usually called 'References' in articles and 'Bibliography' in books and reports), add these two lines towards the end of your document, or wherever you want it printed, substituting the name of your own  $\text{\LaTeX}$  file and the name of your chosen bibliography style:

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

- The  $\text{\backslash bibliography}$  command is followed by the filename of your  $\text{\LaTeX}$  file *without* the `.bib` extension.
- The  $\text{\backslash bibliographystyle}$  command is followed by the name of any of  $\text{\LaTeX}$ 's supported bibliography styles, of which there are many dozens available from CTAN.<sup>4</sup>

The styles `plain` and `alpha` are two common generic styles used for drafts: see below for others.

#### 7.4.2.2 Running *bibtex*

When you run the *bibtex* program, the details of every document you have cited will be extracted from your database, formatted according to the style you specify, and stored in a temporary bibliographic (`.bbl`) file with a label corresponding to the one you used in your citation, ready for  $\text{\LaTeX}$  to use. This is entirely automatic: all you do is cite your references in your  $\text{\LaTeX}$  document using the labels you gave the entries in your  $\text{\LaTeX}$  file, and run the *bibtex* program.

After processing your file with  $\text{\LaTeX}$ , run  $\text{\LaTeX}$  on it by clicking on the  $\text{\LaTeX}$  toolbar icon (*WinEdt*) or the  $\text{\LaTeX}$  `BIBTeX File` menu entry (*Emacs*) or by typing the command `bibtex` followed by the name of your document (without the `.tex` extension). When you run  $\text{\LaTeX}$  again it uses the `.bbl` file to add the references where you tell it to, and a subsequent run of  $\text{\LaTeX}$  will format the correct citation numbers (or author/year, or whatever format you are using).

```
$ latex mybook
$ bibtex mybook
$ latex mybook
$ latex mybook
```

Because of this three-stage process, you always get a warning message about an 'unresolved reference' the first time you add a new reference to a previously uncited work. This will disappear after subsequent runs of *bibtex* and  $\text{\LaTeX}$ .

In practice, authors tend to run  $\text{\LaTeX}$  from time to time during writing anyway, so they can preview the document. Just run  $\text{\LaTeX}$  after adding a new  $\text{\backslash cite}$  command, and subsequent runs of  $\text{\LaTeX}$  will incrementally incorporate all references without you having to worry about it. You only need to follow the full formal sequence ( $\text{\LaTeX}$ ,  $\text{\LaTeX}$ ,  $\text{\LaTeX}$ ,  $\text{\LaTeX}$ ) when you have finished writing and want to ensure that all references have been resolved.

<sup>4</sup>The style shown in the example here provides formatting according to the specifications for Transactions of the IEEE (revised).



### 7.4.2.3 BIB<sub>T</sub>E<sub>X</sub> format

The format for the BIB<sub>T</sub>E<sub>X</sub> file is specified in the BIB<sub>T</sub>E<sub>X</sub> documentation (see §5.1.2 for how to find and print it). You create a file with a name ending in `.bib`, and add your entries, for example:

```
@book{fg,
  title      = {{An Innkeeper's Diary}},
  author     = {John Fothergill},
  edition    = {3rd},
  publisher  = {Penguin},
  year       = 1929,
  address    = {London}
}
```

There is a prescribed set of fields for each of a dozen or so types of document: book, article (in a journal), article (in a collection), chapter (in a book), thesis, report, paper (in a Proceedings), etc. Each entry identifies the document type after the '@' sign, followed by the entry label that you make up, and then each field (in any order), using the format:

```
keyword = {value},
```

Most T<sub>E</sub>X-sensitive editors have a BIB<sub>T</sub>E<sub>X</sub> mode which understands these entries. *Emacs* automatically uses its `bibtex-mode` whenever you open a filename ending in `.bib`. When editing BIB<sub>T</sub>E<sub>X</sub> databases, the rules are simple:

- Omit the comma after the last field in the entry (only).
- Titles may have their case changed in some styles: to prevent this, enclose the title in double curly braces as in the example.
- Values which are purely numeric (e.g. years) may omit the curly braces.
- Fields can occur in any order but the format must otherwise be *very strictly* observed.
- Fields which are not used do not have to be included (so if your editor automatically inserts them blank, you can safely delete the unused lines).

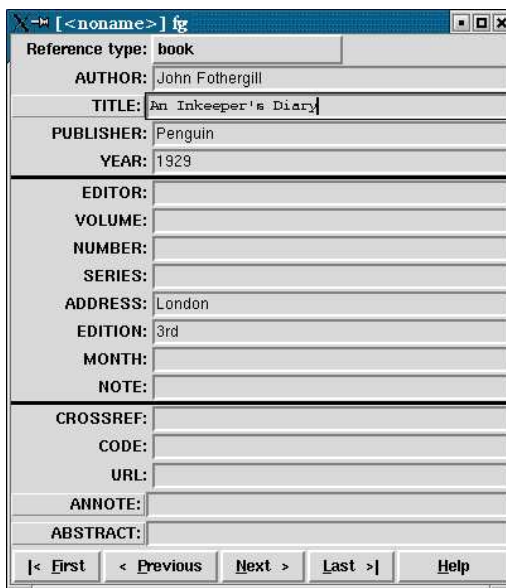
To help with this, there are several interfaces to creating and maintaining BIB<sub>T</sub>E<sub>X</sub> files, such as the *tkbibtex* program, which runs on most platforms (see Figure 7.1).

### 7.4.2.4 Changing the layout

To change the title printed over the reference listing, just change the value of `\refname` (articles) or `\bibname` (books and reports) by adding a line like this in your preamble:

```
\renewcommand{\bibname}{Reading List}
```

The formatting specifications (BIB<sub>T</sub>E<sub>X</sub> styles) are based on standard settings for journals and books from dozens of publishers: you just pick the one you want by name. The `texmf/bib/bst` subdirectory of your installation contains the ones installed by default, and you search on CTAN for others (look for `.bst` files). Most of the others are named after university press styles (e.g. *harvard*, *oxford*) or the publisher or journal which specified them (e.g. *elsevier*, *kluwer*, etc.).

Figure 7.1: tkBIB<sub>T</sub>E<sub>X</sub>, one of several graphical interfaces to BIB<sub>T</sub>E<sub>X</sub> databases

Some of them have an accompanying package file which you need to include with the normal `\usepackage` command in your preamble. In this case the format may be distributed as `.dtx` and `.ins` files and will need installing in the same way as any other package (see §5.2). Always read the documentation, because most of the formats are very specific to the journal they were designed for, and have fairly absolute requirements.

If you are writing for a specific publisher, you should remember that the rules or formats are laid down by the typographic designer of that journal or publisher: you cannot arbitrarily change the format just because you don't happen to like it: it's not your choice!

It is also possible to write your own BIB<sub>T</sub>E<sub>X</sub> (`.bst`) style files, although it uses a language of its own which really needs a computer science background to understand. However, this is rendered unnecessary in most cases: there is an extensive program (actually written in  $\text{\LaTeX}$ ) called *makebst*, which makes `.bst` files by asking you a (long) series of questions about exactly how you want your citations formatted. Just type `latex makebst` in a command window, but give it a dummy run first, because some of the questions are very detailed, so you need to have thought through how you want your citations to appear before you start.

#### 7.4.2.5 Other modes of citation

The method of citing a work by numeric reference is common in the Natural Sciences but is not used in the Humanities and Law. In these fields, citations are usually done with short references (author/short-title/year) in a numbered footnote. Sometimes they are actually called 'footnotes' to the exclusion of ordinary footnotes, although they are really citations which happen by convention to be *displayed* as footnotes: an important distinction rarely appreciated by authors until they come to need a normal footnote.

The bibliography at the back of the document is then printed *unnumbered* in alphabetic order of author, or perhaps chronologically if the time-frame is very large. This unnumbered format is why it is conventionally called 'References' rather than 'Bibliography': sufficient working citation has already been provided in the footnote, and the list at the back is for reference purposes only; whereas in the Natural Sciences,

the citation is just a number, or possibly an author and year only, so the full listing is called a Bibliography.

The jurabib package (originally intended for German law articles but now extended to other fields in the Humanities, and to other languages) has extensive features for doing this style of citation and is very strongly recommended.

## 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 `\makeindex` command in your preamble:

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

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

**Plain entry** Typing `\index{beer}` will create an entry for ‘beer’ with the current page number.

**Subindex entry** For an entry with a subentry use an exclamation mark to separate them: `\index{beer!lite}`. Subsubentries like `\index{beer!lite!American}` work to another level deep.

**Cross-references** ‘See’ entries are done with the vertical bar (one of the rare times it does *not* get interpreted as a math character): `\index{Microbrew|see{beer}}`

**Font changes** To change the style of an entry, use the @-sign followed by a font change command:

```
\index{beer!Rogue!Chocolate Stout@\textit{Chocolate Stout}}
```

This example indexes ‘*Chocolate Stout*’ and italicises it at the same time. Any of the standard `\text...` font-change commands work here: see the table on p. 84 for details. You can also change the font of the index number on its own, as for first-usage references, by using the vertical bar in a similar way to the ‘see’ entries above, but substituting a font-change command (*without* a backslash) such as `textbf` for bold-face text: `\index{beer!Rogue!Chocolate Stout|textbf}` (see the index).

**Out of sequence** The same method can be used as for font changes, just without the font command: `\index{Oregon Brewing Company@Rogue}` will add an entry for ‘Rogue’ in the ‘O’ section of the index, as if it was spelled ‘Oregon Brewing Company’.

When the document has been processed through LaTeX it will have created a `.idx` file, which you run through the *makeindex* program by typing (for example):

```
makeindex mythesis
```

Some editors may have a button or menu entry for this. The program will look for the `.idx` file and output a `.ind` file. This gets used by the command `\printindex`

which you put at the end of your document, where you want the index printed. 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: both the books by Lamport (1994) and by Goossens/Mittelbach/Samarin (1993) duck the issue, but there is some documentation on `glotex` on CTAN.

## 7.6 Multiple columns

Use the `multicol` package: the environment is called **multicols** (note the plural form) and it takes the number of columns as a separate argument in curly braces:

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

$\LaTeX$  has built-in support for two-column typesetting via the `twocolumn` option in the standard Document Class Declarations, but it is relatively inflexible in that you cannot change from full-width to double-column and back again on the same page, and the final page does not balance the column heights. However, it does feature special **figure\*** and **table\*** environ-

ments which typeset full-width figures and tables across a double-column setting.

The more extensive 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 still be used. It also balances the height of the final

page so that all columns are the same height, and you can control the width of the gutter by redefining the `\columnsep` command to a new dimension.

The package provides the **multicols** environment (note the extra 's') and you follow the begin-environment command with the number of columns needed as a second argument: `\begin{multicols}{5}`.

## CHAPTER VIII

# Fonts and layouts

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

As I hope you have seen, this is normally completely unnecessary in  $\text{\LaTeX}$ , which does most of the hard work for you automatically. However, there are occasions when you need to make manual typographic changes, and this chapter is about how to do them.

## 8.1 Changing layout

The design of the page can be a very subjective matter, and also rather a subtle one. Many organisations large and small pay considerable sums to designers to come up with page layouts to suit their purposes. Styles in page layouts change with the years, as do fashions in everything else, so what may have looked attractive in 1989 may look rather dated in 2003.

As with most aspects of typography, making the document readable involves making it consistent, so the reader is not interrupted or distracted too much by apparently random changes in margins, widths, or placement of objects. However, there are a number of different occasions where the layout usually *does* change, related to the frequency with which the format appears.

- The title page (and the half-title, copyright page, dedication, and other one-page preliminaries if you use them) is usually designed individually, as the information on it only occurs once in that format anywhere in the document.
- The table of contents and other related lists like figures and tables probably all need to share one design.
- The prelims like Foreword, Introduction, and Preface should likewise follow the same format between them.
- Chapter and Appendix start pages usually share a layout.
- Other (normal) pages have a single layout, but it may specify individual variations to handle tables, lists, figures, sidebars, exercises, footnotes, etc.

If you are going to design a whole document, it's probably a good idea to read a couple of books on layout design first, to get a feel for the conventions which contribute to making the reader comfortable reading.

While unusual or radical layouts have an important role in attention-grabbing or making a socio-political statement (*WIRED*<sup>1</sup> magazine is an obvious recent example), they are usually out of place in business reports, white papers, books, theses, and journals. In ephemera, on the other hand, as in advertising, they are probably critical.

---

<sup>1</sup>Anderson (1993–)

### 8.1.1 Spacing

We mentioned in §7.3 and elsewhere the existence of the `geometry` package which lets you change margins. It also lets you set the text-area height and width and a lot of other layout settings: read the documentation for details (see §5.1.2 for how to read package documentation).

```
\usepackage[left=2cm,top=1cm,bottom=2cm,right=3cm,
nohead,nofoot]{geometry}
```

The spacing around the individual textual components (headings, paragraphs, lists, footnotes, etc.) can also be changed on a document-wide basis, as we saw with paragraph spacing and indentation in §3.6.

Changing the spacing of section headings for the whole document can be done with the `sectsty` package, designed to let you adjust section-head spacing without having to know about the internal  $\LaTeX$  coding. The spacing for lists can be adjusted with the `mdwlist` package. In both cases the user with highly specific requirements such as a publisher's Compositor's Specification should read the relevant sections in the *The  $\LaTeX$  Companion*<sup>2</sup> or ask for expert help, as there are many internal settings which can also be changed to fine-tune your design, but which need some knowledge of  $\LaTeX$ 's internals.

All the above are for automating changes so that they occur every time in a consistent manner. You can also make manual changes whenever you need:

**Flexible vertical space** There are three preset commands `\smallskip`, `\medskip`, and `\bigskip`. These output flexible (dynamic, or 'rubber') space, approximately 3pt, 6pt, and 12pt high respectively, but they 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`).

**Fixed vertical space** For a fixed-height space which will *not* stretch or shrink, use the command `\vspace` followed by a length in curly braces, e.g. `\vspace{18pt}` (again, this has to be after a paragraph break). Bear in mind that extra space which ends up at a page-break when the document is formatted will get discarded entirely to make the bottom and top lines fall in the correct places. To force a space to remain and be taken into account even after a page break (rare), use the starred variant: `\vspace*`.

**Double spacing** Double-spacing normal lines of text is usually a bad idea, as it looks very ugly. It is still, unfortunately, a requirement in some universities for thesis submission, a historical relic from the days of typewriters. Nowadays,  $1\frac{1}{3}$  or  $1\frac{1}{2}$  line spacing is considered acceptable, according to your font size. If your institution still thinks they should have double line spacing, they are quite simply wrong. 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, use the `setspace` package, which has commands for double line-spacing and one-and-a-half line spacing, but be prepared for some very ugly output.

The space between lines is defined by the length of `\baselineskip` multiplied by the value of `\baselinestretch`. In general, *don't meddle with `\baselineskip` at all*, and with `\baselinestretch` only if you know what you are doing. (Both can, however, safely be used as reference values in commands like `\vspace*{\baselineskip}` to leave a whole line space.) The value of `\baselineskip` changes with the font size (see §8.2.4) but

<sup>2</sup>Goossens/Mittelbach/Samarin (1993)

is conventionally set to 1.2 times the current nominal 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. Quite separately, there are some perfectly genuine and acceptable reasons for wanting wide line spacing, for example when typesetting a proof of a critical or variorum edition, where editors and contributors are going to want to add notes manually by writing between the lines, or where the text is going to be overprinted by something else like Braille, or in advertising or display text for special effects.

**Horizontal space** There is a horizontal equivalent to the `\vspace` command: `\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), which we saw between single and double quotes in §2.5. It's also sometimes used between the full point after abbreviations and a following number, as in page references like p.199, where a word space would look too big, and setting it solid would look too tight.
- `\enspace` ( $1/2$ em). There is no direct equivalent predefined in  $\LaTeX$  for mid and thick spaces as used by metal typesetters, although it would be possible to define them. The en as a unit is often used as the width of a single digit in some fonts, as a convenience so that tables of figures are easy to line up;
- `\quad` (1em);
- `\qquad` (2em).

Beyond this, all horizontal space within paragraphs is automatically flexible, as this is what  $\LaTeX$  uses to achieve justification. Never be tempted to try and change the spacing between letters unless you have some professional training in typography. Some systems use letterspacing (incorrectly called 'tracking') as an aid to justification and it is almost always wrong to do so (and looks it). While it is possible to change letterspacing, it should only be done by a typographer, and then only very rarely, as the settings are very subtle and beyond the scope of this booklet.

## 8.1.2 Headers and footers

$\LaTeX$  has built-in settings to control the page style of its default page layouts. These are implemented with the `\pagestyle` command, which can take an argument of `plain` (page number centered at the bottom), `empty` (nothing at all), or `headings` (running heads). The value `myheadings` can be used if you want to reprogram the definitions of how `\markright` and `\markboth` are used, which control how chapter and section titles get into page headers. The command `\thispagestyle` (taking the same arguments) can be used to force a specific style for the current page only.

The `fancyhdr` package lets you redefine the left-hand, centre, and right-hand page headers and footers for both odd and even pages (twelve objects in all). These areas can contain a page number, fixed text, variable text (like the current chapter or section title, or the catch-words of a dictionary), or even a small image. They can also be used to do page backgrounds and frames, by making one of them the top corner of an invisible box which 'hangs' text or images down over the whole page.

The settings for the downloadable version of this document can be used as an example: for the whole story you have to read the documentation.

```

\pagestyle{fancy}\fancyhead{}
\renewcommand\headrulewidth{.1pt}
\fancyhead[L0,RE]{\footnotesize\sffamily\lite\leftmark}
\fancyhead[LE,RO]{\footnotesize\sffamily\lite\itshape\rightmark}
\fancyfoot[C]{}
\fancyfoot[LE,RO]{\setlength{\fboxsep}{2pt}\ovalbox{\footnotesize
\sffamily\thepage}}
\fancyfoot[L0,RE]{\footnotesize\sffamily\lite\@title}
\fancypagestyle{plain}{\fancyhf{}
\fancyfoot[R]{\setlength{\fboxsep}{2pt}\ovalbox{%
\footnotesize\sffamily\thepage}}
\fancyfoot[L]{\footnotesize\sffamily\lite\@title}
\renewcommand\headrulewidth{0pt}}

```

This is probably more complex than most documents, but it illustrates some of the most common techniques:

1. Settings are prefixed by making the `\pagestyle` ‘fancy’ and setting the `\fancyhead` to null.
2. The thickness of the rule at the top of the page can be changed (or set to 0pt to make it disappear).
3. The header and footer settings are specified with L, C, and R for left, centre, and right; and with O and E for Odd and Even numbered pages. In each setting, the typeface style, size, and font can be specified along with macros which implement various dynamic texts (here, the current chapter and section titles, which  $\LaTeX$  stores in `\rightmark` and `\leftmark`).
4. The ‘plain’ variant is used for chapter starts, and resets some of the parameters accordingly.

## 8.2 Using fonts

The default typeface in  $\LaTeX$  is Computer Modern (CM). This typeface was designed by Knuth for use with  $\TeX$  because it is a book face, and he designed  $\TeX$  originally for typesetting books. 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, rather than Times, because until recently the mathematical symbols for Times were a commercial product often unavailable to users of free software.

Computer Modern is based on a 19th-century book typeface from Monotype, which is why it looks a little like an old-fashioned school book. This paragraph is set in Computer Modern so you can see what it looks like. The typeface is written in METAFONT, a font-drawing program made by Knuth to accompany  $\TeX$  systems.

If you are reading this in the HTML version, the above paragraph is only a low-resolution copy because browsers don’t usually have the Computer Modern font available. All the rest of this document is set in Palatino, with Avant Garde for some of the headings and Courier for the fixed-width type.

In addition to CM, there are many other METAFONT fonts which can be downloaded from CTAN, including a large collection of historical, symbol, initial, and non-Latin fonts.  $\LaTeX$  also comes with the complete ‘Adobe 35’ collection of typefaces which are built into every laser printer, so you have the same base set as other DTP systems. There are some more fonts included in PostScript Type 1 format donated by the X Consortium which match those distributed free with the X Window system. Plus, of course, standard  $\LaTeX$  can use any of the thousands of Type 1 fonts, and *pdf $\LaTeX$*  can use any of the thousands of TrueType fonts as well.





Just to make it clear: standard  $\TeX$  uses METAFONT and PostScript Type 1 fonts only. To use TrueType fonts as well, you must use *pdf $\LaTeX$*  instead.

### 8.2.1 Changing the default font family

$\TeX$  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 use one of the packages listed in Table 8.1, 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. *However...* 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 inelegantly wide). This is the preferred setting if you want Times.<sup>3</sup>

**newcent** changes to New Century Schoolbook/Helvetica/Courier.

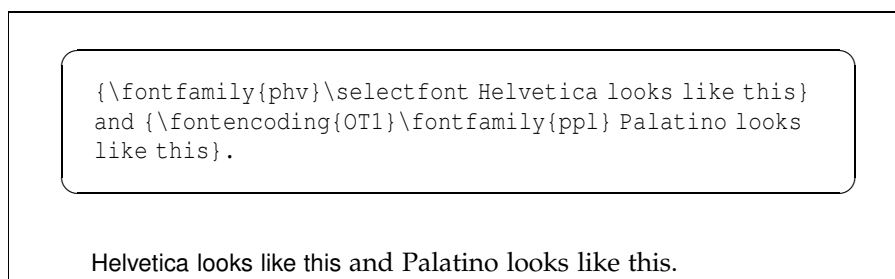
**palatino** changes to Palatino/Avant Garde/Courier.

**palatcm** changes the roman to Palatino only, but with CM mathematics

Where no package name is given in Table 8.1, it means the font is rarely used as a default by itself except in special cases like users’ own homebrew packages. To use the font you have to specify it manually, or make a little macro for yourself if you use it more than once, as shown below.

### 8.2.2 Changing the font family temporarily

To shift to another font family on a temporary basis, group the text within curly braces to limit the scope of the font change, and use the commands `\fontencoding` (if needed), `\fontfamily`, and `\selectfont` commands *immediately inside* the opening curly brace, e.g.



In this example, the `\fontencoding` command has been used to ensure that the typeface will work even if the sentence is used in the middle of something typeset in a different encoding (like this document).

<sup>3</sup>The `pslatex` package is also said to be outdated by some experts because it implemented rather long-windedly what can now be done in three commands. However, until these replace the current version, I recommend continuing to use `pslatex` when you want Times with Helvetica and Courier.

### Grouping

Notice the use of curly braces in a slightly different way 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.

In a normal document, of course, random typeface changes like this are very uncommon. You select your faces once at the start of the document, and stick with them. Most cases where people want to do unusual typeface changes involve things like special symbols on a repetitive basis, and  $\LaTeX$  provides much easier programmable ways to make these changes into shorthand commands (called macros: see Chapter 9). You could, for example, make a macro called `\product` which would let you typeset product names in a distinct typeface:

```
Andlinger has replaced \product{Splosh} with \product{SuperSplosh}
```

This is one of  $\LaTeX$ ’s most powerful features. It means that if you needed to change your `\product` command at some later stage to use a different font, you only have to change three characters in the macro (the font family abbreviation), and you don’t need to edit your document text at all!

### 8.2.3 Changing font style

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

Type style	Command	Example (using Computer Modern)
Upright (default)	<code>\upshape*</code>	The quick brown fox jumps over the lazy dog
Italic	<code>\itshape</code>	<i>The quick brown fox jumps over the lazy dog</i>
Slanted	<code>\slshape*</code>	<i>The quick brown fox jumps over the lazy dog</i>
Small Capitals	<code>\scshape*</code>	THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
Bold	<code>\bfseries*</code>	<b>The quick brown fox jumps over the lazy dog</b>
Bold Extended	<code>\bfseries†</code>	<b>The quick brown fox jumps over the lazy dog</b>
Sans-serif	<code>\sffamily</code>	The quick brown fox jumps over the lazy dog
Monospace	<code>\ttfamily</code>	The quick brown fox jumps over the lazy dog

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

† Some typefaces do not have both bold and bold extended: by default  $\LaTeX$  uses `\bfseries` for bold extended.

These ‘shape’, ‘series’, and ‘family’ commands 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}...
```

This gives you ***bold italic sans-serif type***, but beware of pushing your fonts beyond their limits unless you are a typographer. It is not normally meaningful to combine one shape or series class with another of the same class, such as trying to get slanted-italics. It’s an impossibility to combine one family with another (such as a seriffed sans-serif typeface!). Slanted plus italics, for example, doesn’t make sense, as italics are already slanted (although it is technically possible); and while some typefaces may well possess

italic small caps, they are not in common use. Sans-serif and monospace (typewriter) are different fonts, and often different typeface families entirely.<sup>4</sup>

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	<code>\textit{a few words}</code>	puts <i>a few words</i> into italics
Slanted	<code>\textsl{a few words}</code>	puts <i>a few words</i> into slanted type*
Small Capitals	<code>\textsc{a few words}</code>	puts A FEW WORDS into small caps
Bold	<code>\textbf{a few words}</code>	puts <b>a few words</b> into bold type
Sans-serif	<code>\textsf{a few words}</code>	puts a few words into sans-serif type
Monospace	<code>\texttt{a few words}</code>	puts a few words into typewriter type

\* If slanted is available separately from italics.

You can nest these inside one another too:

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

### 8.2.4 Font sizes

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

These sizes are also reflected in the size steps at which Computer Modern was designed. It often comes as a surprise to new users that many typefaces are not designed as a single font and just scaled up or down, but specially drawn at different sizes to make them more legible.

As an example, 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 so you can see there really is a significant difference. In general, you probably don’t want to go scaling fonts too much beyond their design size because they will start to look very odd.

The default sizes (and the commands that operate them) are based on the use of a 10pt font. Using the larger defaults (11pt and 12pt) for the body font will scale these sizes up approximately in proportion. The exact sizes used are listed in the macros in the Class Option files `size10.clo`, `size11.clo` and `size12.clo`. TeX’s default fonts above 10pt are in fact scaled by a factor of 1.2, as shown in the fourth column of the table below.

<sup>4</sup>Although if you’re a typographer wanting to experiment with typewriter typefaces with and without serifs, you can use METAFONT to do exactly this kind of thing. But that’s outside the scope of this document.

Command	Example	Nominal point size	Exact point size
<code>\tiny</code>	The quick brown fox jumps over the lazy dog	5	5
<code>\scriptsize</code>	The quick brown fox jumps over the lazy dog	7	7
<code>\footnotesize</code>	The quick brown fox jumps over the lazy dog	8	8
<code>\small</code>	The quick brown fox jumps over the lazy dog	9	9
<code>\normalsize</code>	The quick brown fox jumps over the lazy dog	10	10
<code>\large</code>	The quick brown fox jumps over the lazy	12	12
<code>\Large</code>	The quick brown fox jumps over t	14	14.40
<code>\LARGE</code>	The quick brown fox jumps o	18	17.28
<code>\huge</code>	The quick brown fox jum	20	20.74
<code>\Huge</code>	The quick brown fox	24	24.88

While this relieves the beginner of having to worry about the ‘right’ size for a given task, when you need a specific size there is the `\fontsize` command:

```
\fontsize{22}{28}\selectfont This is 22pt type 6pt leaded
```

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

Computer Modern fonts (the default) come fixed at the named size steps shown in the table, and if you try to use an odd size in between,  $\text{\TeX}$  will pick the closest step instead. If you really need to use CM at arbitrary sizes there is a package `type1cm` which lets you override the default steps. If you use PostScript (Type 1) fonts, the step sizes do not apply and the font scaling is infinitely variable.

### 8.2.5 Logical markup

All this playing around with fonts is very pretty but you normally only do it for a reason, even if that reason is just to be decorative. Italics, for example, are used for many things:

Cause	Example
Foreign words	<i>ex officio</i>
Scientific names	<i>Ranunculus ficaria</i>
Emphasis	<i>must not</i>
Titles of documents	<i>The <math>\text{\TeX}</math> Companion</i>
Product names	<i>Corel’s WordPerfect</i>
Variables in maths	$E = mc^2$
Subtitles or headings	<i>How to get started</i>
Decoration	<i>FREE UPGRADE!!!</i>

Humans usually have no problem telling the difference between these reasons, because they can read and understand the meaning and context. Computers cannot (yet), so it has become conventional to use descriptive names which make the distinction explicit, even though the appearance may be the same.

$\text{\TeX}$  has some of these built in, like `\emph`, which provides *emphasis*. This has a special feature because *when the surrounding text is already italic, emphasis automatically reverts to upright type*, which is a normal practice for typesetting.

```
This has a special feature because {\itshape when the surrounding
text is already italic, \emph{emphasis} automatically reverts to
upright type, which is a normal practice for typesetting.
```

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

But why would you bother? In a short document it's probably not important, but if you're writing a long report, or a formal document like an article, a book, or a thesis, it makes writing and editing hugely easier if you can control whole groups of special effects with a single command, such as italicising, indexing, or cross-referencing to a glossary. If a format needs changing, you only have to change the definition, and every occurrence automatically follows suit.

It also makes it possible to find and act on groups of meanings — such as making an index of scientific names or product names (as in this document) — 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. This is the importance of automation: it can save you time and money.

In Chapter 9 we will see how to make your own simple commands like this.

### 8.2.6 Colour

You can typeset anything in L<sup>A</sup>T<sub>E</sub>X in any colour you want using the `color` package. First, you need to add the command `\usepackage{color}` to your preamble (note the US spelling of color!). This makes available a default palette of primary colours: `red`, `green`, and `blue` for the RGB colour model used for emitted light (television screens), and `cyan`, `magenta`, and `yellow` to go with black for the CMYK colour model used for reflected light (printing).

For the occasional word or phrase in colour, use the command `\textcolor` with two arguments, the colour name and the text: `\textcolor{red}{like this}`. There is a `\color` command as well, for use within groups:

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

If you have the PostScript printer driver `dvips` installed, you also get a separate 64-colour palette of predefined names (representing approximately the colours in the big box of *Crayola* colouring pencils much favoured by artists and designers). This adds a new colour model called `named`, so if you want the *Crayola* colour `RubineRed`, you can create a name of your own for it to use in colour-changing commands:

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

`RubineRed` looks `like this` (`\textcolor{myred}{like this}`) if you're reading this in colour. To use these names with *pdf<sub>l</sub>atex*, you need to use the `pdffex` option to the `color` package. You can still look up the CMYK codes in the file `texmf/tex/latex/graphics/dvipsnam.def` and use `cmyk` as the colour model name, if you have software which requires it. In that case you would say:

```
\definecolor{myred}{cmyk}{0,1,0.13,0}
```

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 values expressed as decimals, using this `\definecolor` command. For example, an RGB 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. You can then use `\textcolor` with your new colour name: [the midblue looks like this if you're reading in colour](#).

The color package also provides a colour version of `\fbox` (see §6.7.2) called `\colorbox`:

```
\colorbox{midblue}{\color{magenta}Magenta on midblue}
```

[Magenta on midblue](#): you can see how careful you need to be with colours.

## 8.3 Installing new fonts

Different fonts come in a variety of packagings: the three most common used with T<sub>E</sub>X systems are PostScript fonts, TrueType fonts, and METAFONT fonts. How you install them and where they go depends on how you installed L<sup>A</sup>T<sub>E</sub>X: all I can deal with here are the standard locations within the TDS.

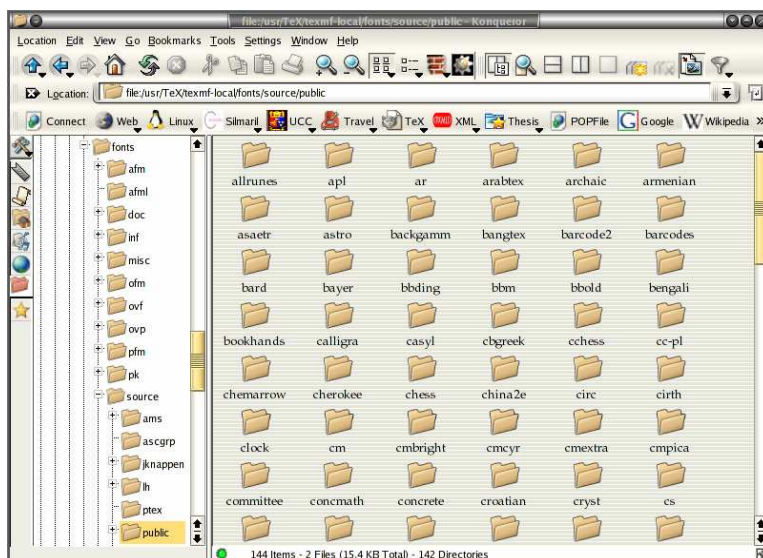
Most typefaces come supplied as one or more font files. PostScript fonts are supplied as a pair of files: an outline and a metric, as described in §8.3.1 and §8.3.2. A TrueType font combines these in a single file. For L<sup>A</sup>T<sub>E</sub>X use you also need a style (`.sty`) file and font definition (`.fd`), which can easily be created if they are not supplied.

The instructions here assume the use of the New Font Selection Scheme (NFSS) used in L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>. If you are running the obsolete L<sup>A</sup>T<sub>E</sub>X 2.09, upgrade it now.

### 8.3.1 Installing METAFONT fonts

This is the simplest installation. When you download METAFONT fonts from CTAN, you'll usually find a large number of Outline files (`.mf` files) and maybe some other types as well (see below).

1. Create a new subdirectory named after the typeface you're installing in `texmf-local/fonts/source/public/`:



2. Copy all the `.mf` (outline) files to this directory.
3. Copy the `.fd` (Font Definition) file[s] and the `.sty` (style) file to your `texmf/tex/latex/mfnfss` directory.
4. Run your  $\TeX$  indexer program (see step 4 in the procedure on p.50).

That's it. Unlike PostScript fonts, METAFONT fonts can be used to generate the font metric file (`.tfm` files) automatically on-the-fly the first time the typeface is used, so there should be nothing else to install. Some METAFONT fonts come with pre-generated `.tfm` files which you can install if your system is slow at generating them itself:

1. Create a new subdirectory within `texmf-local/fonts/tfm/public/` named the same as the one you created for the `.mf` files above.
2. Copy all the `.tfm` files into this subdirectory.
3. Run your  $\TeX$  indexer program (see step 4 in the procedure on p.50).

In some rare cases, pre-generated packed bitmap fonts (`.pk` files) are also available from CTAN (normally your previewer and print driver creates these automatically, but you can use the pre-generated ones if you have a very slow system). If you really want to install these, it's a similar procedure to the `.tfm` files:

4. Create a new subdirectory within `texmf-local/fonts/pk/modeless/` named the same as the one you created for the `.mf` and `.tfm` files above.
5. Copy all the `.nnpk` files into this subdirectory (*nnn* is a number reflecting the dot-density of the bitmap). On Microsoft systems the files may just end in `.pk` and be kept in subdirectories named after the dot-density, e.g. `dpi360`.
6. Run your  $\TeX$  indexer program (see step 4 in the procedure on p.50).

Now you can put a `\usepackage` command in your preamble with whatever name the `.sty` file was called, and read the documentation to see what commands it gives to use the font (refer to §5.2 and step 2 in the procedure on p.49).

If the font came *without* `.fd` or `.sty` files, you'll need to find someone who can make them for you (or follow the outline in §8.3.2, step 11 in the procedure on p.93).

### 8.3.2 Installing PostScript fonts

Lots of people will tell you that PostScript fonts and PostScript output are dead and that TrueType or OpenType fonts and PDF output are the way to go. While this is true for many cases, standard  $\TeX$  does not work with TrueType fonts and does not produce PDF directly. Only *pdflatex* does that, and there are still many printers whose typesetters and platemakers need PostScript rather than PDF. In addition, operating system support for scalable fonts is still very poor on Unix systems (including Linux), despite the advances in recent years, so in many cases it still makes sense to use  $\TeX$ 's built-in support for PostScript.

PostScript (Adobe Type 1) fonts come in two halves, just like METAFONT fonts: a Font Metric and an Outline. Unlike METAFONT fonts, however, you must install both: you cannot cause the the metrics to be generated from the outline as you can with METAFONT.

The two file types are `.afm` (Adobe Font Metric) and `.pfb` (PostScript Font Binary (PFB)) files. *You must have both for each font before you start.* If you only have the near-obsolete `.pfa` (PostScript Font ASCII) files, it may be possible to generate the `.pfb` files using the *t1binary* program from the *t1utils* package (see <http://gnuwin32.sourceforge.net/packages/t1utils.htm>) or the excellent *PFAedit* font editor (from <http://pfaedit.sourceforge.net>). There are unfortunately still some companies distributing Type 1 fonts in `.pfa` format (Mathematica is one reported recently).



The installation method I described in earlier editions has worked perfectly for me for years, but I have updated it here to use the facilities of the *updmap* program (which comes with your  $\TeX$  installation). This removes the need for one of the steps I gave before, which required editing the `psfonts.map` file, as this is now recreated by *updmap*. The procedure below is *not* the official way (that's *fontinst*), but it is the basis for a script I am working on called *Gutta-Percha*<sup>a</sup>, which automates the whole process.

<sup>a</sup>Yes, as in rubber.

I'll repeat this: before you start, make sure you have all the `.afm` and `.pfb` files for the typeface you want. In the example below, I'm going to use a single font from an imaginary typeface called Foo, so I have `foo.afm` and `foo.pfb` files.

### 1. Put the files in your temporary directory

This is `/tmp` on Linux, and should be `C:\tmp` or `C:\temp` or even `C:\Windows\temp` on Microsoft Windows.

### 2. Find out or decide on the short font name to use inside $\LaTeX$ .

This is *not* the full descriptive name (e.g. Baskerville Italic Bold Extended) but an encoded font name in the format `fnnsseec`, devised by Karl Berry, which stores the same information in no more than eight characters for compatibility with systems which cannot handle long filenames. The letters in the format above have the following meanings (see the *fontname* documentation for more details):

Letter	Meaning	Examples
f	foundry	b=Bitstream, m=Monotype, p=Adobe (PostScript)
nn	typeface name	ba=Baskerville, tm=Times, pl=Palatino
ss	series/shape	r=roman, bi=bold italic, etc.
ee	encoding	8a=default 8-bit ANSI, ly=Y&Y's $\TeX$ 'n'ANSI
c	[small]caps	(this is a literal 'c' character, used only if needed)

The `texmf/fontname` directory in your installation of  $\LaTeX$  has files for several foundries giving fully-formed names like these for common fonts (e.g. `ptmr8a` is [Adobe] PostScript Times Roman in an 8-bit ANSI encoding; `bgs1ly` is Bitstream Gill Sans Light in Y&Y's  $\TeX$ 'n'ANSI encoding [LY1]).<sup>5</sup> Read the documentation in *Fontname: Filenames for  $\TeX$  fonts*<sup>6</sup> to find out how to make up your own short names if the foundry and font you want is not shown in the `fontname` directory.

In this example we'll call our mythical example typeface 'zork' (standing for Zfonts Ordinary Bookface, because `k` is the letter used for Book fonts, `b` being already the code for bold) and we'll assume it comes in the two files `foo.afm` and `foo.pfb` that I mentioned above.

While the *fontname* directories have ready-made lists of these names for popular collections of typefaces, making them up requires some knowledge of typographic terms and a careful reading of the *fontname* documentation.

### 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'ANSI encoding which (to me) seems to be the only one that includes the glyphs I want where I want them. Your mileage may vary. This encoding is referred to as `LY1` within  $\LaTeX$  and the encoding file is in `texmf/dvips/base/texnansi.enc`. Encoding is needed because Adobe fonts store their characters in different places to the  $\TeX$  standard.

<sup>5</sup>Confusingly, Bitstream fonts (and others from similar sources) mostly have different names from the original fonts, so what they call Humanist 521 is actually Gill Sans. Until recently, US law only allowed the *names* of typefaces to be copyrighted, not the font designs themselves, leading to widespread piracy.

<sup>6</sup>Berry (June 2001)

Copy this encoding file to the temporary directory where you're doing all this stuff. If you're using the 8a or 8r encoding (or some other encoding), then copy that file instead (`8a.enc`, `8r.enc`).

#### 4. Convert .afm files to .tfm

The `afm2tfm` program is a standard utility in the `bin` directory of your  $\TeX$  installation. If it's not, update your installation.

In a command window, type:

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

This creates a special 'raw'  $\TeX$  Font Metric file (hence the special `r` prefix) that  $\LaTeX$  can use, with a list of all its properties encoded with LY1 (the `.vpl` or Virtual Property List file). 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. Small caps (optional)

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. Yes, it *does* overwrite the `rzorkly.tfm` created in the first command. Let it. And those are *two* of the 'greater-than' signs before the `zork.id` filename because we want to append to it, not overwrite it.

#### 6. Create the virtual font

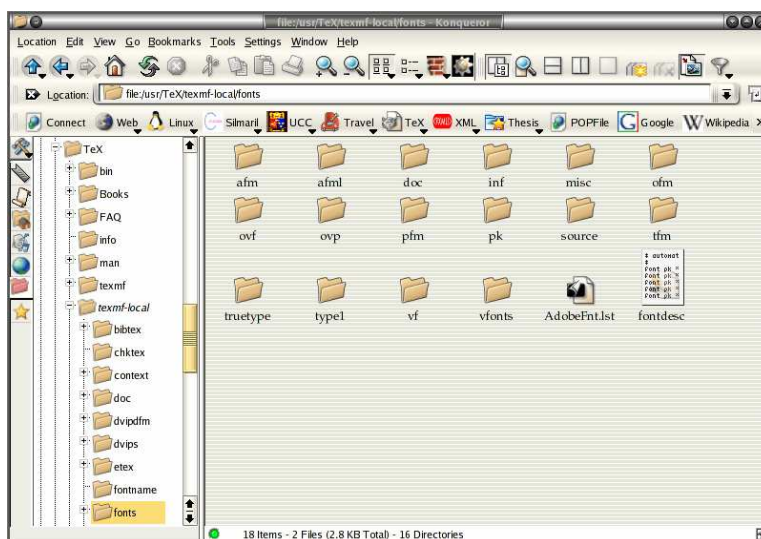
Turn the `.vpl` files into `.vf` and `.tfm` pairs.  $\LaTeX$  uses these to convert from Adobe's encoding to its own.

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

Again, the `vptovf` program is a standard part of your  $\TeX$  distribution.

#### 7. Make directories to hold the files

Under your `texmf-local` directory there should be a `fonts` directory, and in there there should be `afm`, `tfm`, `type1`, and `vf` directories. Create them if they do not already exist.



In each of these four, create a directory for the foundry, and within them create a directory for the typeface (using a human-readable typeface name, not the short Karl Berry fontname). In our example, this means:

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

Or if you're lazy like me:

```
(cd /usr/TeX/texmf-local/fonts;\
for d in afm tfm type1 vf;do mkdir -p $d/zfonts/ordinary;done)
```

For Microsoft Windows users, the path before `texmf-local` may look something like `C:\Program Files\TeXLive\`, depending on how and where you have installed your  $\TeX$  system.

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 make the intervening directories by hand first.

### 8. Copy the files to their rightful places

Copy the four groups of files to the four new directories:

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

You can of course do all this with a directory window and mouse if you find it easier.

### 9. Create a font map

The font map is what tells `dvips` which PFB file to use for which font. The configuration file for `dvips` is `texmf/dvips/config/config.ps` and it gets its

entries from the program *updmap* which reads map files for each typeface. The configuration file for *updmap* is `texmf-var/web2c/updmap.cfg`<sup>7</sup>, so it needs an entry for our new font, using the three-letter font family abbreviation (the first three letters of the Berry fontname (here ‘zor’):

```
Map zor.map
```

We also have to create this map file (`zor.map`) in a subdirectory of `texmf-local/dvips/config/` named after the foundry, so we need to create `texmf-local/dvips/config/zfonts` as well.

- (a) Open `/usr/TeX/texmf-var/web2c/updmap.cfg` in your editor.
- (b) At the bottom, add the line: `Map zor.map`
- (c) Save and close the file.

The font entries in our `zor.map` will be on a *single* line each, with no line-wrapping. Each entry gives the short name of the font, the long (Adobe) name, the PostScript encoding parameters (in quotes), and then two filenames prefixed by input redirects (less-than signs): the encoding file and the PostScript outline file.

- (a) First create the directory if it doesn’t already exist:

```
mkdir -p /usr/TeX/texmf-local/dvips/config/zfonts
```

- (b) Use your editor to open (create) the file `/usr/TeX/texmf-local/dvips/config/zfonts/zor.map`.
- (c) Insert the line:

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

- (d) Save and close the file.

You get the full font name (here, ‘Ordinary-Blackface’) from the `zork.id` which was created back in step 4 in the procedure on p. 90 when we ran *afm2tfm*. You must get this exactly right, because it’s the ‘official’ full name of the font, and PostScript files using this font need to match it.

#### 10. Create a style file

$\text{\TeX}$  needs a style file to implement the interface to the font. Call it after the typeface or something related; in this example we’ll call it `foozork.sty`. In it go some details of the name and date we did this, what version of  $\text{\TeX}$  it needs, and any other command necessary to operate the font, like the font encoding and whether it is to supersede the current default Roman font.

- (a) Use your editor to open (create) `foozork.sty` in `texmf-local/TeX/latex/psnfss`.
- (b) Insert the following lines:

<sup>7</sup>There is another one of these at `texmf/web2c/updmap.cfg`, but that contains the map references for the fonts which came with your distribution of  $\text{\TeX}$ , so you should not interfere with it.

```

% foozork - created from foo for Zork
\def\fileversion{1.0}
\def\filedate{2002/12/03}
\def\docdate{2002/12/03}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{foozork}[\filedate\space\fileversion\space
Zfonts Ordinary PSNFSS2e package]
\RequirePackage[LY1]{fontenc}
\renewcommand{\rmdefault}{zor}
\endinput

```

Note the following:

- The first argument to **\ProvidesPackage** *must* be the same as this style file name; and that the font family is referred to as `zor`, being the foundry letter plus the fontname abbreviation. This acts as a prefix for any/all font variants (bold, italic, etc.).
- If you are not using Y&Y encoding, omit the line referring to LY1 font encoding.
- If this is a typewriter font, make the renewed command **\rmdefault** into **\ttdefault**.
- If it's a sans-serif font, make it **\sfdefault** instead.
- Omit the command completely if you don't want the style file to supersede the current defaults but simply to make the font available. If you do this, you probably want to write a new command or two to use it, typically one for grouped use and one for argument use:

```

\newcommand{\zorkfamily}{\fontencoding{LY1}\fontfamily{zor}\selectfont}
\newcommand{\textzork}[1]{\zorkfamily#1}

```

(c) Save and close the file.

#### 11. Create the Font Definition file

The last file to create is the *font definition* (.fd) file. This is named following the pattern `eeefnn.fd`, using the same conventions as before, by prepending the (lowercase) encoding abbreviation to the foundry letter and fontname abbreviation, so our example would be `ly1zor.fd` for the LY1 encoding and the `zor` short font name.

- (a) Use your editor to open (create) `texmf-local/tex/latex/psnfss/ly1zor.fd`
- (b) Enter the following lines:

```

\ProvidesFile{ly1zor.fd}[2002/03/03 v0.1 manual font
definitions for LY1/zor.]

\DeclareFontFamily{LY1}{zor}{}

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

```

(c) Save and close the file.

FD files typically use one `\DeclareFontFamily` command which specifies the encoding and the short font name. Then as many pairs of `\DeclareFontShape` commands as you converted fonts (assuming you did both normal and small caps for each font: see step 5 in the procedure on p.90; if you didn't, then only one such command per font is needed here). The arguments to the `\DeclareFontShape` command to watch are the 3rd (weight/width), 4th (shape), and 5th (font outline name): 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 *The L<sup>A</sup>T<sub>E</sub>X Companion*<sup>8</sup> 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 for fontname. There is no `shape.map` in fontname because it's not part of font file names, it's purely a L<sup>A</sup>T<sub>E</sub>X creation, so here's what the same book says:

Character	Meaning
n	normal (upright)
it	italic
sl	slanted
sc	small caps
ui	upright italic
ol	outline

Add your own for other oddities, but be consistent: I use `cu` for cursive (scripts), for example, and `k` for blackletter faces (not to be confused with `k` as a *width* for 'book').

The default fontspec (5th) argument above `<->` means all sizes come from the same font outline (remember if this was a METAFONT font with different design sizes like CM it would be much more complex).

If the face has only a few variants, you can create any other entries for bold, italic, slanted, etc. with the relevant weight and width and shape values pointing at the relevant outline file.

If you want one font to substitute for a missing one (for example italics to substitute for slanted in a typeface which has no slanted variant of its own) give the `ssub` ('silent substitution') command in the fontspec: for example to make all references to `sl` (slanted) type use an existing italic font, make the 5th argument like this:

```
{<-> 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. Update the index and the map files

Run your T<sub>E</sub>X indexer program (see step 4 in the procedure on p.50) so that `updmap` can find the files it needs.

Then run `updmap` (just type `updmap`). This updates the maps and runs the T<sub>E</sub>X indexer program again automatically.

<sup>8</sup>Goossens/Mittelbach/Samarin (1993)

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

```
This is {\zorkfamily ZORK} or \textzork{ZORK}
```

### 8.3.3 Installing the Type 1 Computer Modern fonts

If your  $\text{\LaTeX}$  installation uses the METAFONT (bitmap) versions of the Computer Modern typeface, you may want to switch to the Type 1 (PostScript) version, especially if you are going to be using *pdf $\text{\LaTeX}$*  instead of standard  $\text{\LaTeX}$  because Acrobat Reader makes such a hames of displaying Type3 fonts. *GSview* and *pdfview* handles them correctly.

To do this, install one of the sets of CM PostScript fonts. There are several available:

- The fonts from BlueSky Research at <http://www.ctan.org/tex-archive/fonts/cm/ps-type1/bluesky/>
- Basil K. Malyshev's fonts at <http://www.ctan.org/tex-archive/fonts/cm/ps-type1/bakoma/>
- Vladimir Volovich's CM-Super at <http://www.ctan.org/tex-archive/fonts/ps-type1/cm-super/>
- Bogusław Jackowski's Latin Modern at <ftp://cam.ctan.org/tex-archive/fonts/ps-type1/lm.tar.gz>

The BaKoMa fonts include the American Mathematical Society (AMS) fonts for extended mathematics, but are more complex to install because they come with a special set of TFM files.

The BlueSky fonts are just PFB and AFM files, and are a drop-in replacement requiring no further changes, as they use the same TFM files as the METAFONT version. Follow the README file in the downloadable archive for installation instructions.

The Latin Modern and CM-Super fonts are new and I haven't tested them but they are well spoken of. Feedback on this is very welcome.

The  $\text{\TeX}$  Live distribution uses Type 1 versions of Computer Modern by default. There are more details in the FAQ at <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=uselmfonts>.

## CHAPTER IX

# Programmability (macros)

We've touched several times 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. It's also the one that needs most foreknowledge, which is why this chapter is in this position.

$\LaTeX$  is in fact itself just a collection of macros — rather a big collection — written in  $\TeX$ 's internal typesetting language. These *macros* are little program-like sets of instructions with a name which can be used as shorthand for an operation you wish to perform more than once.

Macros can be arbitrarily complex. Many of the ones used in the standard  $\LaTeX$  packages are several pages long, but as we will see, even short ones can very simply automate otherwise tedious chores and allow the author to concentrate on *writing*.

### 9.1 Simple replacement macros

In its simplest form, a  $\LaTeX$  macro can just be a straightforward text replacement of a phrase to avoid misspelling something each time you need it, e.g.

```
\newcommand{\ef}{European Foundation for the Improvement of Living  
and Working Conditions}
```

Put this in your preamble, and you can then use `\ef` in your document and it will typeset it as the full text. 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 §2.3.1). And when you want to force a space to be printed, use a backslash followed by a space, e.g.

```
The \ef\ is an institution of the Commission of the European Union.
```

As you can see from this example, the `\newcommand` command takes two arguments: *a*) the name you want to give the new command; and *b*) the expansion to be performed when you use it, so there are always two sets of curly braces after `\newcommand`.

### 9.2 Macros using information gathered previously

A more complex example is the macro `\maketitle` which is used in almost every formal document to format the title block. In the basic document classes (book, report, and article) it performs small variations on the layout of a centred block with the title followed by the author followed by the date, as we saw in §3.3.



If you inspect one of these document class files, such as `texmf/tex/latex/base/report.cls` you will see `\maketitle` defined (and several variants called `\@maketitle` for use in different circumstances). It uses the values for the title, author, and date which are assumed already to have been stored in the internal macros `\@title`, `\@author`, and `\@date` by the author using the matching `\title`, `\author`, and `\date` commands in the document.

This use of one command to store the information in another is a common way of gathering the information from the user. The use of macros containing the `@` character prevents their accidental misuse by the user: in fact to use them in your preamble we have to allow the `@` sign to become a ‘letter’ so it can be recognised in a command name, and remember to turn it off again afterwards (see item 1 below).

```
\makeatletter
\renewcommand{\maketitle}{%
  \begin{flushleft}%
    \sffamily
    {\Large\bfseries\color{red}\@title\par}%
    \medskip
    {\large\color{blue}\@author\par}%
    \medskip
    {\itshape\color{green}\@date\par}%
    \bigskip\hrule\vspace*{2pc}%
  \end{flushleft}%
}
\makeatother
```

Insert this in the sample file on p. 48 immediately before the `\begin{document}` and remove the `\color{...}` commands from the title, author, and date. Re-run the file through `TEX`, and you should get something like this:

### Practical Typesetting

Peter Flynn  
Silmaril Consultants  
December 2001

---

In this redefinition of `\maketitle`, we’ve done the following:

1. Enclosed the changes in `\makeatletter` and `\makeatother` to allow us to use the `@` sign in command names;<sup>1</sup>
2. Used `\renewcommand` and put `\maketitle` in curly braces after it;
3. Opened a pair of curly braces to hold the new definition. The closing curly brace is immediately before the `\makeatother`;
4. Inserted a `flushleft` environment so the whole title block is left-aligned;
5. Used `\sffamily` so the whole title block is in the defined sans-serif typeface;
6. For each of `\@title`, `\@author`, and `\@date`, we have used some font variation and colour, and enclosed each one in curly braces to restrict the changes just to each command. The closing `\par` makes sure that multiline title and authors and dates get typeset with the relevant line-spacing;
7. Added some flexible space between the lines, and around the `\hrule` (horizontal rule) at the end;

<sup>1</sup>If you move all this preamble into a style file of your own, you don’t need these commands: the use of `@` signs in command names is allowed in style and class files.

Note the % signs after any line ending in a curly brace, to make sure no intrusive white-space find its way into the output. These aren't needed after simple commands where there is no curly brace because excess white-space gets gobbled up there anyway.

### 9.3 Macros with arguments

But macros are not limited to text expansion. They can take arguments of their own, so you can define a command to do something with specific text you give it. This makes them much more powerful and generic, as you can write a macro to do something a certain way, and then use it hundreds of times with a different value each time.

We looked earlier (§8.2.5) at making new commands to put specific classes of words into certain fonts, such as product names into italics, keywords into bold, and so on. Here's an example for a command `\product`, which also indexes the product name and adds a trademark sign:

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

If I now type `\tmpproduct{Velcro}` then I get *Velcro*<sup>™</sup> typeset, and if you look in the index, you'll find this page referenced under '*Velcro*'. Let's examine what this does:

1. The macro is specified as having one argument (that's the [1] in the definition). This will be the product name you type in curly braces when you use `\product`. Macros can have up to nine arguments.
2. The expansion of the macro is contained in the second set of curly braces, spread over several lines (see item 5 for why).
3. It prints the value of the first argument (that's the #1) in italics, which is conventional for product names, and adds the `\texttrademark` command.
4. Finally, it creates an index entry using the same value (#1), making sure that it's italicised in the index (see the list on p.75 in §7.5 to remind yourself of how indexing something in a different font works).
5. Typing this macro over several lines makes it easier for humans to read. I could just as easily have typed

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

but it wouldn't have been as clear what I was doing.

One thing to notice is that to prevent unwanted spaces creeping into the output when  $\text{\LaTeX}$  reads the macro, I ended each line with a comment character (%).  $\text{\LaTeX}$  normally treats newlines as spaces when formatting (remember §2.4.1), so this stops the end of line being turned into an unwanted space when the macro is used.  $\text{\LaTeX}$  always ignores spaces at the *start* of macro lines anyway, so indenting lines for readability is fine.

In (§2.7.2) we mentioned the problem of frequent use of unbreakable text leading to poor justification or to hyphenation problems. A solution is to make a macro which puts the argument into an `\mbox` with the appropriate font change, but precedes it all with a conditional `\linebreak` which will make it more attractive to  $\text{\TeX}$  to start a new line.

```
\newcommand{\var}[1]{\linebreak[3]\mbox{\ttfamily#1}}
```

This only works effectively if you have a reasonably wide setting and paragraphs long enough for the differences in spacing elsewhere to get hidden. If you have to do this in narrow journal columns, you may have to adjust wording and spacing by hand occasionally.

## 9.4 Nested macros

Here's a slightly more complex example, where one macro calls another. It's common in normal text to refer to people by their forename and surname (in that order), for example Don Knuth, but to have them indexed as *surname, forename*. This pair of macros, `\person` and `\reindex`, automates that process to minimize typing and indexing.

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

1. The digit 1 in square brackets means that `\person` has one argument, so you put the whole name in a single set of curly braces, e.g. `\person{Don Knuth}`.
2. The first thing the macro does is output #1, which is the value of what you typed, just as it stands, so the whole name gets typeset exactly as you typed it.
3. But then it uses a special feature of Plain TeX macros (which use `\def` instead of  $\LaTeX$ 's `\newcommand`<sup>2</sup>): they too can have multiple arguments but you can separate them with other characters (here a space) to form a pattern which TeX will recognise when reading the arguments.

In this example (`\reindex`) it's expecting to see a string of characters (#1) followed by a space, followed by another string of characters (#2) followed by a dummy command (`\sentinel`). In effect this makes it a device for splitting a name into two halves on the space between them, so the two halves can be handled separately. The `\reindex` command can now read the two halves of the name separately.

4. The `\person` command invokes `\reindex` and follows it with the name you typed plus the dummy command `\sentinel` (which is just there to signal the end of the name). Because `\reindex` is expecting two arguments separated by a space and terminated by a `\sentinel`, it sees 'Don and Knuth' as two separate arguments.

It can therefore output them using `\index` in reverse order, which is exactly what we want.

A book or report with a large number of personal names to print and index could make significant use of this to allow them to be typed as `\person{Leslie Lamport}` and printed as Leslie Lamport, but have them indexed as 'Lamport, Leslie' with virtually no effort on the author's part at all.

---

<sup>2</sup>Don't try this at home alone, children! This one is safe enough, but you should strictly avoid `\def` for a couple of years. Stick to `\newcommand` for now.

## EXERCISE 20

**Other names**

Try to work out how to make this `\person` feature work with names like:

- Blanca Maria Bartosova de Paul
- Patricia Maria Soria de Miguel
- Arnaud de la Villèsbrunne
- Prince
- Pope John Paul II

Hints: the command `\space` produces a normal space, and one way around  $\TeX$ 's requirements on spaces after command names ending with a letter is to follow such commands with an empty set of curly braces `{ }`.

## 9.5 Macros and environments

As mentioned in §6.7.3, it is possible to define macros to capture text in an environment and reuse it afterwards. This avoids any features of the subsequent use affecting the formatting of the text.

One example of this uses the facilities of the `fancybox` package, which defines a variety of framed boxes to highlight your text, and a special environment `Sbox` which ‘captures’ your text for use in these boxes.

By putting the text (here in a `minipage` environment because we want to change the width) inside the `Sbox` environment, it is typeset into memory and stored in the macro `\theSbox`. It can then be used afterwards as the argument of the `\shadowbox` command (and in this example it has also been centred).

```

\begin{Sbox}
\begin{minipage}{3in}
This text is formatted to the specifications
of the minipage environment in which it occurs.

Having been typeset, it is held in the Sbox
until it is needed, which is after the end of
the minipage, where you can (for example) centre
it and put it in a special framed box.
\end{minipage}
\end{Sbox}
\begin{center}
\shadowbox{\theSbox}
\end{center}

```

This text is formatted to the specifications of the minipage environment in which it occurs.  
Having been typeset, it is held in the Sbox until it is needed, which is after the end of the minipage, where you can (for example) centre it and put it in a special framed box.

## 9.6 Reprogramming $\LaTeX$ 's internals

$\LaTeX$ 's internal macros can also be reprogrammed or even rewritten entirely, although doing this can require a considerable degree of expertise. Simple changes, however, are easily done.

Recall that L<sup>A</sup>T<sub>E</sub>X'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 (§4). 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 at all what you want. The zero is caused by it not being part of any chapter. But this numbering is controlled by macros, and you can redefine them. In this case it's a macro called `\thesection` which reproduces the current section number counter (see §6.2.6). It's redefined afresh in each document class file, using the command `\renewcommand` (in this case in `texmf/tex/latex/base/report.cls`):

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

You can see it invokes `\thechapter` (which is defined elsewhere to reproduce the value of the `chapter` counter), and it then prints a dot, followed by the Arabic value of the counter called `section` (that `\c@` notation is L<sup>A</sup>T<sub>E</sub>X's internal way of referring to counters). You can redefine this in your preamble to simply leave out the reference to chapters:

```
\renewcommand{\thesection}{\arabic{section}}
```

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 L<sup>A</sup>T<sub>E</sub>X's internal code (as you may have noticed in the example earlier). And I've also used the 'public' macro `\arabic` to output the value of `section` (L<sup>A</sup>T<sub>E</sub>X's internals use a 'private' set of control sequences containing @-signs, designed to protect them against being changed accidentally).

Now the introduction to your report will start with:

## 1 Introduction

What's important is that you *don't ever* need to 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.

### 9.6.1 Changing list item bullets

As mentioned earlier (§6.2.1), here's how to redefine a bullet for an itemized list, with a slight tweak:

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

Here we use the `bbding` package which has a large selection of 'dingbats' or little icons, and we make the label for top-level itemized lists print a right-pointing pencil (the names for the icons are in the package documentation: see §5.1.2 for how to get it).

In this case, we are using the `\raisebox` command within the redefinition because it turns out that the symbols in this font are positioned slightly too high for the typeface we're using. The `\raisebox` command takes two arguments: the first is a dimension, how much to raise the object by (and a negative value means 'lower': there is no `\lowerbox` command!); and the second is the text you want to affect. Here, we are shifting the symbol down by  $\frac{1}{4}$ ex (see §2.7.1 for a list of dimensions L<sup>A</sup>T<sub>E</sub>X can use).

There is a vast number of symbols available: see *A comprehensive list of symbols in T<sub>E</sub>X<sup>3</sup>* for a comprehensive list.

<sup>3</sup>Pakin (2002)

## CHAPTER X

# Compatibility with other systems

As we saw in Chapter 2,  $\LaTeX$  uses plain-text files, so they can be read and written by any standard application that can open text files. This helps preserve your information over time, as the plain-text format cannot be obsoleted or hijacked by any manufacturer or sectoral interest, and it will always be readable on any computer, from your handheld (yes,  $\LaTeX$  is available for some PDAs) to the biggest supercomputer.

However,  $\LaTeX$  is intended as the last stage of the editorial process: formatting for print or display. If you have a requirement to re-use the text in some other environment — a database perhaps, or on the Web or a CD-ROM or DVD, or in Braille or voice output — then it should be edited, stored, and maintained in something neutral like XML, and only converted to  $\LaTeX$  when a typeset copy is needed.

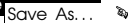
Although  $\LaTeX$  has many structured-document features in common with SGML and XML, it can still only be processed by the  $\LaTeX$  and *pdf<sub>l</sub>atex* programs. Because its macro features make it almost infinitely redefinable, processing it requires a program which can unravel arbitrarily complex macros, and  $\LaTeX$  and its siblings are the only programs which can do that effectively. Like other typesetters and formatters (Quark *XPress*, *PageMaker*, *FrameMaker*, Microsoft *Publisher*, *3B2* etc.),  $\LaTeX$  is largely a one-way street leading to typeset printing or display formatting.

Converting  $\LaTeX$  to some other format therefore means you will unavoidably lose some formatting, as  $\LaTeX$  has features that others systems simply don't possess, so they cannot be translated. Similarly, converting other formats to  $\LaTeX$  usually means editing back the stuff the other formats omit because they only store appearances, not structure.

### 10.1 Converting into $\LaTeX$

There are several systems which will save their text in  $\LaTeX$  format. The best known is probably the *LyX* editor, which is a wordprocessor-like interface to  $\LaTeX$  for Windows and Unix. Several maths packages like the *EuroMath* editor, and the *Mathematica* and *Maple* analysis packages, can also save material in  $\LaTeX$  format.

In general, most wordprocessors and DTP systems don't have the level of internal markup sophistication needed to create a  $\LaTeX$  file, or any other kind of structured document, because they only store what the text looks like, not why it's there or what role it fulfills. There are two ways out of this:

- Use the **File**  menu item to save the wordprocessor file as HTML, rationalise the HTML using Dave Raggett's *HTML Tidy*<sup>1</sup>, and convert the resulting XHTML to  $\LaTeX$  with any of the standard tools (see below).
- Use a specialist conversion tool like EBT's *DynaTag* (now available from Enigma, if you can persuade them to sell you a copy). It's expensive and they don't advertise it, but for bulk conversion of consistently-marked *Word* files into XML it beats everything else hands down. The *Word* files *must* be consistent, though,

---

<sup>1</sup><http://tidy.sourceforge.net/>

and must use named styles from a stylesheet, otherwise no system on earth is going to be able to guess what it means.

There is of course a third way, suitable for large volumes only: send it off to the Pacific Rim to be retyped into XML. There are hundreds of companies from India to Polynesia who do this at high speed and low cost. It sounds crazy when the document is already in electronic form, but it's a good example of the low quality of wordprocessor markup that this solution exists at all.

You will have noticed that all the solutions lead to one place: SGML or XML. As explained above and elsewhere, these formats are the only ones devised so far capable of storing sufficient information in machine-processable, publicly-accessible form for a document to be recreated in multiple output formats. Once your document is in XML or SGML, there is a large range of software available to turn it into other formats, including L<sup>A</sup>T<sub>E</sub>X. Processors in any of the common styling and processing languages like DSSSL, XSLT, *Omnimark*, *Metamorphosis*, *Balise*, etc. can easily be written to output L<sup>A</sup>T<sub>E</sub>X, and this approach is extremely common.

Much of this will be simplified when wordprocessors support native, arbitrary XML as a standard feature. Sun's *Star Office* and its Open Source sister, *OpenOffice*, have used XML as their native Save format for several years, and there is a project at the Organisation for the Advancement of Structured Information Systems (OASIS) for developing a common XML office file format based on those used by these two packages. *WordPerfect* has also had a native SGML (and now XML) editor for many years, which will work with any Document Type Definition (DTD) (but not a Schema). Microsoft has had a half-hearted 'Save As...XML' for a while, using an internal and largely undocumented Schema, but the 'Professional' versions of *Word* and *Excel* in *Office 11* (Office 2003) now have full support for arbitrary Schemas.

When these efforts coalesce into generalised support for arbitrary DTDs and Schemas, it will mean a wider choice of editing interfaces, and probably the ability to run XSLT conversion into L<sup>A</sup>T<sub>E</sub>X from within these editors, such as is done at the moment with *Emacs* or *XML Spy*.

## 10.2 Converting out of L<sup>A</sup>T<sub>E</sub>X

This is much harder to do comprehensively. As noted earlier, the L<sup>A</sup>T<sub>E</sub>X file format really requires the L<sup>A</sup>T<sub>E</sub>X program itself in order to process all the packages and macros, because there is no telling what complexities authors have added themselves (what a lot of this booklet is about!).

There is an excellent program on CTAN to do L<sup>A</sup>T<sub>E</sub>X-to-*Word* conversion, but it only handles a subset of the built-in commands of default L<sup>A</sup>T<sub>E</sub>X, not packages or your own macros.

One easy route into wordprocessing, however, is the reverse of the procedures suggested in the preceding section: convert L<sup>A</sup>T<sub>E</sub>X to HTML, which many wordprocessors read easily. The *HT<sub>E</sub>X2HTML* and *T<sub>E</sub>X4HT* programs do this quite well, given the limited formatting available through Web browsers.

If you have the full version of Adobe *Acrobat*, you can open a PDF file created by *pdf<sub>l</sub>atex*, select and copy all the text, and paste it into *Word* and some other wordprocessors, and retain some common formatting of headings, paragraphs, and lists. Both solutions still require the wordprocessor text to be edited into shape, but they preserve enough of the formatting to make it worthwhile for short documents. Otherwise, use the *pdf<sub>t</sub>otext* program to extract everything from the PDF file as plain (paragraph-formatted) text.

At worst, the *detex* program on CTAN will strip a L<sup>A</sup>T<sub>E</sub>X file of all markup and leave just the raw unformatted text, which can then be re-edited. There are also programs to extract the raw text from DVI and PostScript (PS) files.

### 10.3 Going beyond L<sup>A</sup>T<sub>E</sub>X

The reader will have deduced by now that while L<sup>A</sup>T<sub>E</sub>X is possibly the best programmable formatter around, the L<sup>A</sup>T<sub>E</sub>X file format is not generally usable with anything except the L<sup>A</sup>T<sub>E</sub>X program. L<sup>A</sup>T<sub>E</sub>X was originally written in the mid-1980s, about the same time as the Standard Generalized Markup Language (SGML), but the two projects were not related. However, T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X have proved such useful tools for formatting SGML and more recently XML that many users chose this route for their output, using conversions written in the languages already mentioned in §10.2.

Unfortunately, when the rise of the Web in the early 1990s popularised SGML using the HyperText Markup Language (HTML), browser writers deliberately chose to encourage authors to ignore the rules of SGML. Robust auto-converted formatting therefore became almost impossible except via the browsers' low-quality print routines.

It was not until 1997, when the Extensible Markup Language (XML) was devised, that it again became possible to provide the structural and descriptive power of SGML but without the complex and rarely-used options which had made standard SGML so difficult to program for.

XML is now becoming the principal system of markup. Because it is based on the international standard (SGML), it is not proprietary, so it has been implemented on most platforms, and there is lots of free software supporting it as well as many commercial packages. Like SGML, it is actually a meta-language to let you define your own markup, so it is much more flexible than HTML. Implementations of the companion Extensible Stylesheet Language (XSL) provide a direct route to PDF but at the expense of reinventing most of the wheels which L<sup>A</sup>T<sub>E</sub>X already possesses, so the sibling Extensible Stylesheet Language: Transformations (XSLT) can be used instead to translate to L<sup>A</sup>T<sub>E</sub>X source code. This is usually much faster than writing your own formatting from scratch in XSL, and it means that you can take full advantage of the packages and sophistication of L<sup>A</sup>T<sub>E</sub>X. A similar system is used for the Linux Documentation Project, which uses SGML transformed by the Document Style Semantics and Specification Language (DSSSL) to T<sub>E</sub>X

The source code of this booklet, available online at <http://www.ctan.org/tex-archive/info/beginlatex/src/includesXSLT> which does exactly this.



## APPENDIX A

# GNU Free Documentation License

*Version 1.2, November 2002*

Copyright (C) 2000,2001,2002 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### A.0 PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document ‘free’ in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ‘copyleft’, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### A.1 APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice

placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The ‘Document’, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as ‘you’. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A ‘Modified Version’ of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A ‘Secondary Section’ is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The ‘Invariant Sections’ are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The ‘Cover Texts’ are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A ‘Transparent’ copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not ‘Transparent’ is called ‘Opaque’.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The ‘Title Page’ means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, ‘Title Page’ means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section ‘Entitled XYZ’ means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as ‘Acknowledgements’, ‘Dedications’, ‘Endorsements’, or ‘History’.) To ‘Preserve the Title’ of such a section when you modify the Document means that it remains a section ‘Entitled XYZ’ according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only

as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## A.2 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non-commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## A.3 COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of

Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## A.4 MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum (§A.11) below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled 'History', Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled 'History' in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the 'History' section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled 'Acknowledgements' or 'Dedications', Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled 'Endorsements'. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled 'Endorsements' or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled 'Endorsements', provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## A.5 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 (§A.4) above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ‘History’ in the various original documents, forming one section Entitled ‘History’; likewise combine any sections Entitled ‘Acknowledgements’, and any sections Entitled ‘Dedications’. You must delete all sections Entitled ‘Endorsements’.

## A.6 COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of

this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## A.7 AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ‘aggregate’ if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## A.8 TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled ‘Acknowledgements’, ‘Dedications’, or ‘History’, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## A.9 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## A.10 FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ‘or any later version’ applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## A.11 ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the

document and put the following copyright and license notices just after the title page:

Copyright (c) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ‘GNU Free Documentation License’.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the ‘with... Texts.’ line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

## APPENDIX B

# Configuring T<sub>E</sub>X search paths

T<sub>E</sub>X systems run on a huge variety of platforms, and are typically made up of huge numbers of quite small files, so they cannot easily use each operating system's built-in methods of searching for a file when needed.

Instead, they use a technique borrowed from the Unix world, based on a simple index for each directory they need to look in. This is known as the ls-R database, from the Unix command (`ls -R`) which creates it, and the program which does it for T<sub>E</sub>X is sometimes actually called *mktexlsr*. This is the program referred to in step 4 in the procedure on p.50.

However, to know where to make these indexes, and thus where to search, T<sub>E</sub>X needs to be told about them. In a standard T<sub>E</sub>X installation this information is in `texmf/web2c/texmf.cnf`. The file is similar to a Unix script, but the only lines of significance for the search paths are the following (this is how they appear in the default Unix installation, omitting the comments):

```
TEXMFMAIN = /usr/TeX/texmf
TEXMFLOCAL = /usr/TeX/texmf-local
HOMETEXMF = $HOME/texmf
TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN}
SYSTEXMF = $TEXMF
VARTEXFONTS = /var/lib/texmf
TEXMFDDBS = $TEXMF;$VARTEXFONTS
```

As you can see, this defines where the main T<sub>E</sub>X/METAFONT directory is, where the local one is, and where the user's personal (home) one is. It then defines the order in which they are searched, and makes this the system-wide list. A temporary directory for bitmap fonts is set up, and added to the list, defining the places in which *texhash* or *mktexlsr* creates its databases.

In some installations, the local directory is set up in `/usr/local/share/texmf` or `/usr/share/texmf.local` or similar variations, so you would substitute this name for `/usr/TeX/texmf-local`. Under Microsoft Windows, the names will be full paths such as `C:\ProgramFiles\TeXLive\texmf`. On an Apple Mac, it might be `HardDisk:TeX:texmf`.

If you edit plain-text configuration files with anything other than a plain-text editor (e.g. a wordprocessor), or if you edit them with a plain-text editor which has been set to word-wrap long lines, make sure you turn line-wrapping *off* so that any long lines are preserved in their correct format.

## APPENDIX C

# T<sub>E</sub>X Users Group membership

The T<sub>E</sub>X Users Group was founded in 1980 for educational and scientific purposes: to provide an organization for those who have an interest in typography and font design, and are users of the T<sub>E</sub>X typesetting system invented by Donald Knuth. TUG is run by and for its members and represents the interests of T<sub>E</sub>X users worldwide.

### TUG membership benefits

Members of TUG help to support and promote the use of T<sub>E</sub>X, METAFONT, and related systems worldwide. All members receive *TUGboat*, the journal of the T<sub>E</sub>X Users Group, the T<sub>E</sub>X Live software distribution (a runnable T<sub>E</sub>X system), and the CTAN software distribution (containing most of the CTAN archive).

In addition, TUG members vote in TUG elections, and receive discounts on annual meeting fees, store purchases, and TUG-sponsored courses. TUG membership (less benefits) is tax-deductible, at least in the USA. See the TUG Web site for details.

### Becoming a TUG member

Please see the forms and information at <http://tug.org/join.html>. You can join online, or by filling out a paper form. The NTG (Dutch) and UKTUG (United Kingdom) T<sub>E</sub>X user groups have joint membership agreements with TUG whereby you can receive a discount for joining both user groups. To do this, please join via <http://www.ntg.nl/newmember.html> (the NTG membership page) or <http://uk.tug.org/Membership/> (the UKTUG page), respectively, and select the option for joint membership.

Each year's membership entitles you to the software and TUGboat produced for that year (even if it is produced in a subsequent calendar year, as is currently the case with TUGboat). You can order older issues of TUGboat and T<sub>E</sub>X memorabilia through the TUG store (<http://tug.org/store>).

The current TUG membership fee is \$65 (US) per year for individuals and \$35 for students and seniors. Add \$10 to the membership fee after May 31 to cover additional shipping and processing costs. The current rate for non-voting subscription memberships (for libraries, for example) is \$85. The current institutional rate is \$500, which includes up to seven individual memberships.

### Privacy

TUG uses your personal information only to mail you products, publications, notices, and (for voting members) official ballots. Also, if you give explicit agreement, we may incorporate it into a membership directory which will be made available only to TUG members.

TUG neither sells its membership list nor provides it to anyone outside of its own membership.

## References

1. **American Mathematical Society:** Short Math Guide for  $\LaTeX$ . Providence, RI: AMS, 2001 (URL: <http://www.ams.org/tex/short-math-guide.html>)
2. **Anderson, Chris,** editor: WIRED. San Francisco, CA: Condé Nast, 1993–, ISSN 1059–1028
3. **Beeton, Barbara,** editor: TUGboat. Portland, OR:  $\TeX$  Users Group, Since 1980, ISSN 0896–3207
4. **Berry, Karl:** Fontname: Filenames for  $\TeX$  fonts. Portland, OR, June 2001 – Technical report (URL: <http://www.ctan.org/tex-archive/info/fontname/>)
5. **Bull, RJ:** Accounting in Business. London: Butterworths, 1972, ISBN 0–406–70651–4
6. **Burnard, Lou/Sperberg-McQueen, Michael:** Guidelines for the Text Encoding Initiative. Oxford, 1995 – Technical report
7. **Davy, William:** A System of Divinity. Lustleigh, Devon: Published by the author, 1806
8. **Doob, Michael:** A Gentle Introduction to  $\TeX$ : A Manual for Self-Study. Portland, OR, 2002 – Technical report (URL: <http://www.ctan.org/tex-archive/info/gentle/>)
9. **Flaubert, Gustave:** Madame Bovary. Paris, 1857
10. **Flynn, Peter:** The HTML Handbook. London: International Thompson Computer Press, 1995, ISBN 1–85032–205–8
11. **Flynn, Peter:** Understanding SGML and XML Tools. Boston: Kluwer, 1998, ISBN 0–7923–8169–6
12. **Flynn, Peter:** The XML FAQ. Cork, Ireland, January 2003 – Technical report (URL: <http://www.ucc.ie/xml/>)
13. **Fothergill, John:** An Innkeeper’s Diary. 3rd edition. London: Penguin, 1929
14. **Free Software Foundation:** The GNU Free Documentation License. Boston, MA, 2003/02/10 23:42:49 – Technical report (URL: <http://www.fsf.org/copyleft/fdl.html>)
15. **Goossens, Michel/Mittelbach, Frank/Samarin, Alexander:** The  $\LaTeX$  Companion. Reading, MA: Addison-Wesley, 1993, ISBN 0–201–54199–8
16. **Goossens, Michel/Rahtz, Sebastian/Mittelbach, Frank:** The  $\LaTeX$  Graphics Companion. Reading, MA: Addison-Wesley, 1997, ISBN 0–201–85469–4
17. **Goossens, Michel et al.:** The  $\LaTeX$  Web Companion. Reading, MA: Addison-Wesley, 1999, ISBN 0–201–43311–7
18. **Heller, Robert:** New To  $\LaTeX$ ...Unlearning Bad Habits. 11 March 2003, Nr. MPG.18d82140d65ddc5898968c@news.earthlink.net (all pages) (URL: <news:comp.text.tex>)
19. **Knuth, Donald Ervin:** The Art of Computer Programming. Volume 1, 2nd edition. Reading, MA: Addison-Wesley, 1980, ISBN 0–201–89685–0
20. **Lamport, Leslie:**  $\LaTeX$ : A Document Preparation System. 2nd edition. Reading, MA: Addison-Wesley, 1994, ISBN 0–201–52983–1



21. **Mac Namara, Matthew:** *La Textualisation de Madame Bovary*. Amsterdam: Rodopi, 2003
22. **Oetiker, Tobias et al.:** The (Not So) Short Guide to  $\LaTeX 2\epsilon$ :  $\LaTeX 2\epsilon$  in 131 Minutes. 2001 – Technical report (URL: <http://www.ctan.org/tex-archive/info/lshort/>)
23. **Pakin, Scott:** A comprehensive list of symbols in  $\TeX$ . 2002 – Technical report (URL: <http://www.ctan.org/tex-archive/info/symbols/comprehensive/>)
24. **Reckdahl, Keith:** Using imported graphics in  $\LaTeX 2\epsilon$ . 1997 – Technical report (URL: <http://www.ctan.org/tex-archive/info/epslatex.pdf/>)
25. **Ryder, John:** *Printing for Pleasure*. London: Bodley Head, 1976, ISBN 0-370-10443-9
26.  **$\TeX$  Users Group:** Getting Started with  $\TeX$ ,  $\LaTeX$ , and friends. November 2003 – Technical report (URL: <http://www.tug.org/begin.html>)

## Index

The same fonts are used here as in the text of the booklet (see the Introduction) to distinguish between different meanings:

Notation	Meaning
CTAN	Acronyms (small caps in some typefaces)
<b>\command</b>	TeX control sequences (monospace font)
<i>term</i>	Defining instance of a specialist term (bold italics)
<i>product</i>	program or product name (italics)
<b>environment</b>	TeX environment (sans-serif bold)
package	TeX package (sans-serif; all available from CTAN)
options, <i>variables</i>	options and variables (oblique)

In the online version, these entries are all hyperlinked to their source: for clarity the stylesheet recommends retaining the traditional blue colour but removing the underlining which most browsers use to indicate a link.

Page numbers in **bold type** indicate the defining instance.

## Special characters

\ (.....)	28	DEC .....	11
\ ) .....	28	DSSSL .....	104
\ - .....	26	DTD .....	103
\ [ .....	28	DTP .....	11
\ # .....	22	DVI .....	39
\ \$ .....	22	EPS .....	62
\ % .....	22	FAQ .....	51
\ & .....	22	FTP .....	6
\ ] .....	28	GNU .....	19
\ ^ .....	22, 23	GUI .....	9
\ _ .....	22	HTML .....	104
\ { .....	22	JPG .....	62
\ } .....	22	NFSS .....	87
\ ~ .....	22, 23	OASIS .....	103
		PCL .....	62
		PDA .....	12
11pt .....	30	PDF .....	39
12pt .....	30	PFB .....	88
3B2 .....	102	PNG .....	62
		PS .....	103
<b>A</b>		RGB .....	86
a4paper .....	31	RTFM .....	14
abstract .....	33, 34	RTFML .....	2
<b>\abstractname</b> .....	33, 34	SGML .....	104
abstracts .....	33	TDS .....	49
accents .....	23	TIFF .....	62
<i>Acrobat</i> .....	103	URI .....	64
<i>Acrobat Reader</i> .....	13, 39, 44, 45	URL .....	64
<b>Acronyms, defined</b>		WYSIWYG .....	43
AMS .....	95	WYSIWYM .....	18
ASCII .....	16	XML .....	104
BMP .....	62	XSL .....	104
CLI .....	8	XSLT .....	104
CM .....	80	<b>\addcontentsline</b> .....	38
CMYK .....	86	<i>Adobe Illustrator</i> .....	61
CTAN .....	47	<i>afm2tfm</i> .....	90, 92

alpha	72	<code>\clearpage</code>	20
AMS	95	CLI	8, 12
Apple Mac	13	<code>\cline</code>	59
<i>apt-get</i>	14	CM	80, 80, 81, 82, 85
<code>\arabic</code>	101	cm (centimeters)	25
<i>arguments</i>	21	CMYK	86, 86
array	58	color	47–49, 86, 87
article	29	<code>\color</code>	86
ASCII	6, 16, 28, 45, 88	<code>\colorbox</code>	87
<i>asynchronous</i>	9	colour	86
<code>\author</code>	32, 97	columns	76
<code>\authorof</code>	72	<code>\columnsep</code>	76
<i>Autorun</i>	15	<code>\command</code>	8, 114
<b>B</b>			
b	65	<b>Commands</b>	
babel	27, 37	<code>\(</code>	28
<i>backslash</i>	19	<code>\)</code>	28
<code>\backslash</code>	22	<code>\-</code>	26
<i>backtick</i>	63	<code>\[</code>	28
<i>Balise</i>	103	<code>\#</code>	22
<code>\baselineskip</code>	78	<code>\\$</code>	22
<code>\baselinestretch</code>	78	<code>\%</code>	22
bbding	54, 101	<code>\&amp;</code>	22
<i>BBedit</i>	14	<code>\]</code>	28
beer	75	<code>\^</code>	22, 23
lite	75	<code>\_</code>	22
American	75	<code>\{</code>	22
<code>\begin</code>	31	<code>\}</code>	22
Berry, Karl	89, 91	<code>\~</code>	22, 23
<code>\bfseries</code>	83	<code>\abstractname</code>	33, 34
bibliographies	71	<code>\addcontentsline</code>	38
<code>\bibliography</code>	72	<code>\arabic</code>	101
<code>\bibliographystyle</code>	72	<code>\author</code>	32, 97
<code>\bibname</code>	73	<code>\authorof</code>	72
<i>bibtex</i>	72	<code>\backslash</code>	22
<code>\bigskip</code>	78	<code>\baselineskip</code>	78
BMP	62	<code>\baselinestretch</code>	78
book	29, 34	<code>\begin</code>	31
boxes	65	<code>\bfseries</code>	83
bp (big points)	25	<code>\bibliography</code>	72
braces	<i>see</i> curly braces	<code>\bibliographystyle</code>	72
<b>C</b>			
<code>\caption</code>	58	<code>\bibname</code>	73
cc (Ciceros)	25	<code>\bigskip</code>	78
CD-ROM	102	<code>\caption</code>	58
center	59, 60	<code>\centering</code>	59
<code>\centering</code>	59	<code>\chapter</code>	35
<code>\chapter</code>	35	<i>chapter</i>	101
<i>chapter</i>	101	<i>charmap</i>	23
<i>charmap</i>	23	Chocolate Stout	75
Chocolate Stout	75	<i>Chocolate Stout</i>	75
<i>Chocolate Stout</i>	75	<code>\cite</code>	71, 72
<code>\cite</code>	71, 72	<code>\citequote</code>	69
<code>\citequote</code>	69	<code>\clearpage</code>	20
		<code>\cline</code>	59
		<code>\color</code>	86
		<code>\colorbox</code>	87
		<code>\columnsep</code>	76
		<code>\command</code>	8, 114
		<code>\date</code>	32, 42, 43, 97
		<code>\DeclareFontFamily</code>	94
		<code>\DeclareFontShape</code>	94

<code>\def</code> .....	99	<code>\part</code> .....	35
<code>\definecolor</code> .....	86, 87	<code>\part*</code> .....	36
<code>\documentclass</code> .....	29, 34, 48	<code>\person</code> .....	99, 100
<code>\ef</code> .....	96	<code>\printindex</code> .....	75
<code>\emph</code> .....	85, 86	<code>\product</code> .....	83, 86, 98
<code>\end</code> .....	31	<code>\protect</code> .....	69
<code>\enspace</code> .....	79	<code>\ProvidesPackage</code> .....	93
<code>\EUR</code> .....	22	<code>\qqquad</code> .....	79
<code>\fancyhead</code> .....	80	<code>\quad</code> .....	25, 79
<code>\fbox</code> .....	66, 67, 87	<code>\raggedleft</code> .....	27
<code>\fontencoding</code> .....	82	<code>\raggedright</code> .....	27, 65
<code>\fontfamily</code> .....	82	<code>\raisebox</code> .....	101
<code>\fontsize</code> .....	85	<code>\ref</code> .....	56, 70, 71
<code>\footnote</code> .....	69	<code>\refname</code> .....	73
<code>\footnotesize</code> .....	85	<code>\reindex</code> .....	99
<code>\foreign</code> .....	86	<code>\renewcommand</code> ....	33, 34, 97, 101
<code>\glossary</code> .....	76	<code>\rightmark</code> .....	80
<code>\graphicspath</code> .....	63	<code>\rmdefault</code> .....	93
<code>\hline</code> .....	59	<code>\scriptsize</code> .....	85
<code>\hrule</code> .....	97	<code>\scshape</code> .....	83
<code>\hspace</code> .....	79	<code>\section</code> .....	35, 37
<code>\Huge</code> .....	85	<code>\selectfont</code> .....	82
<code>\huge</code> .....	85	<code>\sentinel</code> .....	99
<code>\hyphenation</code> .....	26	<code>\setcounter</code> .....	36
<code>\i</code> .....	24	<code>\setlength</code> .....	36, 37, 66
<code>\includegraphics</code> .....	61–63	<code>\sfdefault</code> .....	93
<code>\index</code> .....	75, 76, 99	<code>\sffamily</code> .....	83, 97
<code>\item</code> .....	53	<code>\shadowbox</code> .....	67, 100
<code>\itshape</code> .....	83	<code>\slshape</code> .....	83
<code>\label</code> .....	56, 58, 70, 71	<code>\small</code> .....	69, 85
<code>\LARGE</code> .....	85	<code>\smallskip</code> .....	78
<code>\Large</code> .....	85	<code>\space</code> .....	100
<code>\large</code> .....	85	<code>\subparagraph</code> .....	35, 56
<code>\leftmark</code> .....	80	<code>\subparagraph*</code> .....	36
<code>\linebreak</code> .....	98	<code>\subsection</code> .....	35
<code>\listoffigures</code> .....	38	<code>\subsubsection</code> .....	35
<code>\listoftables</code> .....	38	<code>\tableofcontents</code> .....	37, 38
<code>\makeatletter</code> .....	97	<code>\textcolor</code> .....	86, 87
<code>\makeatother</code> .....	97	<code>\textdegree</code> .....	28
<code>\makeglossary</code> .....	76	<code>\texteuro</code> .....	22
<code>\makeindex</code> .....	75	<code>\textit</code> .....	86
<code>\maketitle</code> 32–34, 37, 38, 43, 96, 97		<code>\textsterling</code> .....	22
<code>\markboth</code> .....	79	<code>\texttrademark</code> .....	98
<code>\markright</code> .....	79	<code>\thechapter</code> .....	101
<code>\mbox</code> .....	26, 98	<code>\theenumi</code> .....	56
<code>\medskip</code> .....	78	<code>\theenumii</code> .....	56
<code>\multicolumn</code> .....	59	<code>\theenumiii</code> .....	56
<code>\newcommand</code> .....	96, 99	<code>\theenumiv</code> .....	56
<code>\newpage</code> .....	8	<code>\theSbox</code> .....	100
<code>\normalsize</code> .....	85	<code>\thinspace</code> .....	23, 79
<code>\ovalbox</code> .....	67	<code>\thispagestyle</code> .....	79
<code>\pageref</code> .....	71	<code>\tiny</code> .....	85
<code>\pagestyle</code> .....	79, 80	<code>\title</code> .....	32, 97
<code>\par</code> .....	60, 78, 97	<code>\titleof</code> .....	72
<code>\paragraph</code> .....	35, 56	<code>\ttdefault</code> .....	93
<code>\parbox</code> .....	65, 66	<code>\ttfamily</code> .....	83

**\upshape** ..... 83  
**\url** ..... 64, 69  
**\usepackage** ..... 34, 47, 48, 74, 88  
**\vbox** ..... 66  
**\verb** ..... 63, 64, 69  
**\vspace** ..... 78, 79  
**\vspace\*** ..... 78  
*comment character* ..... 22  
*commutative* ..... 83  
*configure* ..... 50  
*Corel Draw* ..... 61  
*counter* ..... 8  
**Counters**  
    *chapter* ..... 101  
    *counter* ..... 8  
    *enumi* ..... 56  
    *enumii* ..... 56  
    *enumiii* ..... 56  
    *enumiv* ..... 56  
    *example* ..... 56  
    *secnumdepth* ..... 8, 36  
    *section* ..... 101  
    *tocdepth* ..... 36, 38  
    *variables* ..... 114  
*Crayola* ..... 11, 86  
cross-references ..... 70  
CTAN 7–9, 12, 13, 46, 47, 47, 48, 49, 51, 72,  
    73, 76, 80, 81, 87, 88, 103, 111, 114  
*curly braces* ..... 21  
*Cygwin* ..... 51

**D**

**\date** ..... 32, 42, 43, 97  
dd (Didot points) ..... 25  
DEC ..... 11  
**\DeclareFontFamily** ..... 94  
**\DeclareFontShape** ..... 94  
**\def** ..... 99  
**\definecolor** ..... 86, 87  
description ..... 54  
*detex* ..... 103  
dimension ..... 25  
*dimension* ..... 36  
*DocBook* ..... 8  
document ..... 31, 34  
document class ..... 29  
*document class* ..... 29  
**\documentclass** ..... 29, 34, 48  
double-spacing ..... 78  
draft ..... 30  
DSSSL ..... 103, 104  
DTD ..... 8, 103, 103  
DTP ..... 11, 12, 80  
DVD ..... 102  
DVI ..... 12, 39, 41, 43–45, 48, 63, 65, 103  
*dviaw* ..... 46

*dvieps* ..... 46  
*dvihp* ..... 46  
*dvipt* ..... 45, 46, 86, 91  
*dviptview* ..... 44  
*DynaTag* ..... 102

**E**

editors ..... 18  
**\ef** ..... 96  
*element* ..... 29  
*elsevier* ..... 73  
em (relative measure) ..... 25  
Emacs 7–9, 14–16, 18, 19, 22, 40, 41, 43, 46,  
    59, 60, 72, 73, 103  
**\emph** ..... 85, 86  
**\end** ..... 31  
endnote ..... 69  
**\enspace** ..... 79  
enumerate ..... 54  
*enumi* ..... 56  
*enumii* ..... 56  
*enumiii* ..... 56  
*enumiv* ..... 56  
environment ..... 53  
*environment* ..... 31  
environment ..... 8, 114

**Environments**

**abstract** ..... 33, 34  
**center** ..... 59, 60  
**description** ..... 54  
**document** ..... 31, 34  
**enumerate** ..... 54  
**environment** ..... 8, 114  
**equation** ..... 28  
**figure** ..... 61  
**figure\*** ..... 76  
**float** ..... 57  
**flushleft** ..... 60, 97  
**flushright** ..... 60  
**inparaenum** ..... 55  
**itemize** ..... 54  
**minipage** ..... 65–67, 100  
**multicols** ..... 76  
**multirow** ..... 59  
**picture** ..... 61  
**raggedleft** ..... 27  
**raggedright** ..... 27  
**Sbox** ..... 67, 100  
**table** ..... 58, 61  
**table\*** ..... 76  
**tabular** ..... 58, 60, 66  
**Verbatim** ..... 64  
**verbatim** ..... 64  
EPS ..... 62, 62, 63  
epsf ..... 61  
equation ..... 28  
Esser, Thomas ..... 13

- \EUR**.....22  
 €.....22  
 Euro symbol.....22  
*EuroMath*.....102  
 ex (relative measure).....25  
*example*.....56  
*Excel*.....103
- F**
- family*.....83  
 fancybox.....64, 67, 100  
 fancyhdr.....79  
**\fancyhead**.....80  
 fancyvrb.....64  
 FAQ.....10, 47, 51, 51  
**\fbox**.....66, 67, 87  
 \fboxrule.....67  
 \fboxsep.....67  
 figure.....61  
 figure\*.....76  
 figures.....61  
 filenames.....39  
 Fine, Jonathan.....9  
 float.....57  
 floats.....57, 61  
 flushleft.....60, 97  
 flushright.....60  
 fnpara.....69  
*font definition*.....93  
**\fontencoding**.....82  
**\fontfamily**.....82  
*fontinst*.....89  
*fontname*.....89  
 fontname.....94  
 fonts  
   METAFONT.....80  
   changing temporarily.....82  
   changing the default.....82  
   colour.....86  
   Computer Modern.....80  
   encoding.....89  
   examples.....81  
   families.....82  
   in general.....80  
   installing.....87  
   METAFONT.....80  
   PostScript.....80, 88  
   sizes.....30, 84  
   styles.....83  
   TrueType.....80  
   Type 1.....80  
**\fontsize**.....85  
**\footnote**.....69  
 footnotes.....69  
**\footnotesize**.....85  
**\foreign**.....86
- fpTeX.....7, 13  
*FrameMaker*.....102  
 FTP.....6, 47
- G**
- geometry.....48, 49, 70, 78  
*Ghostscript*.....45  
*GhostView*.....13  
 glossaries.....75  
**\glossary**.....76  
 GNU.....19  
 GNU.....8, 19  
 graphics.....61  
 graphics.....49  
**\graphicspath**.....63  
 graphicx.....61, 62  
*grave accent*.....63  
 grouping.....83  
*GSview*.....15  
*GSview*.....13, 39, 45, 46, 95  
 GUI.....9, 11  
*Gutta-Percha*.....89  
*gv*.....13
- H**
- hard space.....26  
*harvard*.....73  
*hash mark*.....22  
 help.....51  
 H&J.....see hyphenation, justification  
**\hline**.....59  
**\hrule**.....97  
**\hspace**.....79  
 HTML.....2, 8, 29, 104, 104  
*HTML Tidy*.....102  
**\Huge**.....85  
**\huge**.....85  
 hyphenation.....24  
**\hyphenation**.....26  
 hyphens  
   discretionary.....26  
   soft.....26
- I**
- \i**.....24  
 IBM.....8  
 images.....61  
 in (inches).....25  
**\includegraphics**.....61–63  
**\index**.....75, 76, 99  
 indexes.....75  
*Inline lists*.....55  
 inparaenum.....55  
 inputenc.....23, 24, 28  
*Instant Preview*.....9  
**\item**.....53  
 itemize.....54

<code>\itshape</code> .....	83	<code>\makeatletter</code> .....	97
<b>J</b>		<code>\makeatother</code> .....	97
Jackowski, Bogusław .....	95	<i>makebst</i> .....	74
<i>Java</i> .....	8	<code>\makeglossary</code> .....	76
JPG.....	62, 62, 63	<i>makeidx</i> .....	75
jurabib .....	75	<i>makeindex</i> .....	75
justification .....	24	<code>\makeindex</code> .....	75
<b>K</b>		<code>\maketitle</code> ....	32–34, 37, 38, 43, 96, 97
Kastrup, David .....	9	Malyshev, Basil K. ....	95
Kay, Michael .....	8	<i>Maple</i> .....	102
Kiffe, Tom .....	13	marginal notes.....	70
<i>klutwer</i> .....	73	margins.....	78
Knuth, Don .....	11, 99	<code>\markboth</code> .....	79
komascript .....	29, 30	<code>\markright</code> .....	79
<b>L</b>		<i>markup</i> .....	16
<code>\label</code> .....	56, 58, 70, 71	<i>marvosym</i> .....	22
Lamport, Leslie .....	11, 99	math characters.....	27
<code>\LARGE</code> .....	85	<i>Mathematica</i> .....	102
<code>\Large</code> .....	85	mathematics.....	7, 27
<code>\large</code> .....	85	Matthes, Eberhard .....	15
<i>L<small>A</small>T<small>E</small>X<small>2</small>H<small>T</small>M<small>L</small></i> .....	103	<code>\mbox</code> .....	26, 98
<code>\leftmark</code> .....	80	<i>mdwlist</i> .....	78
<i>length</i> .....	36	<code>\medskip</code> .....	78
<code>\length</code> .....	8	memoir .....	29, 30
<b>Lengths (Dimensions)</b>		<i>metacharacters</i> .....	22
<code>\fboxrule</code> .....	67	M <small>E</small> T <small>A</small> F <small>O</small> N <small>T</small> 12, 50, 80–82, 84, 87, 88, 94, 95,	
<code>\fboxsep</code> .....	67	110, 111, 118	
<code>\length</code> .....	8	<i>Metamorphosis</i> .....	103
<code>\parindent</code> .....	37	Microbrew .....	<i>see beer</i>
<code>\parskip</code> .....	8, 36, 37	Microsoft Windows .....	13
<code>\spaceskip</code> .....	26	MiK <small>T</small> E <small>X</small> .....	13
<code>\tabcolsep</code> .....	59	minipage .....	65–67, 100
letter.....	29	mirror .....	62
letterpaper .....	31	<i>mkfs</i> .....	14
letterspacing .....	79	<i>mktexlsr</i> .....	50, 110
<code>\linebreak</code> .....	98	mm (millimeters).....	25
Linux .....	13	<i>Mozilla</i> .....	51
<code>\listoffigures</code> .....	38	multicol.....	76
<code>\listoftables</code> .....	38	multicols .....	76
lists.....	53	<code>\multicolumn</code> .....	59
bulleted .....	54	multirow.....	59
description.....	54	<i>My Computer</i> .....	15
discussion.....	54	<b>N</b>	
enumerated .....	54	newcent .....	82
inline .....	55	<code>\newcommand</code> .....	96, 99
itemized .....	54	<code>\newpage</code> .....	8
numbered.....	54	N <small>F</small> S <small>S</small> .....	87
<b>M</b>		<code>\normalsize</code> .....	85
Mac OS X.....	13	<i>Notepad</i> .....	6
Macintosh.....	19	<b>O</b>	
macros.....	96	O <small>A</small> S <small>I</small> S.....	103
<i>macros</i> .....	96	<i>octothorpe</i> .....	22
<code>\makeatletter</code> .....	97	<i>Office 11</i> .....	103
<code>\makeatother</code> .....	97	<i>Omnimark</i> .....	103
<i>makebst</i> .....	74	oneside.....	30
<code>\makeglossary</code> .....	76		
<i>makeidx</i> .....	75		
<i>makeindex</i> .....	75		
<code>\makeindex</code> .....	75		
<code>\maketitle</code> ....	32–34, 37, 38, 43, 96, 97		
Malyshev, Basil K. ....	95		
<i>Maple</i> .....	102		
marginal notes.....	70		
margins.....	78		
<code>\markboth</code> .....	79		
<code>\markright</code> .....	79		
<i>markup</i> .....	16		
<i>marvosym</i> .....	22		
math characters.....	27		
<i>Mathematica</i> .....	102		
mathematics.....	7, 27		
Matthes, Eberhard .....	15		
<code>\mbox</code> .....	26, 98		
<i>mdwlist</i> .....	78		
<code>\medskip</code> .....	78		
memoir .....	29, 30		
<i>metacharacters</i> .....	22		
M <small>E</small> T <small>A</small> F <small>O</small> N <small>T</small> 12, 50, 80–82, 84, 87, 88, 94, 95,			
110, 111, 118			
<i>Metamorphosis</i> .....	103		
Microbrew .....	<i>see beer</i>		
Microsoft Windows .....	13		
MiK <small>T</small> E <small>X</small> .....	13		
minipage .....	65–67, 100		
mirror .....	62		
<i>mkfs</i> .....	14		
<i>mktexlsr</i> .....	50, 110		
mm (millimeters).....	25		
<i>Mozilla</i> .....	51		
multicol.....	76		
multicols .....	76		
<code>\multicolumn</code> .....	59		
multirow.....	59		
<i>My Computer</i> .....	15		
<b>N</b>			
newcent .....	82		
<code>\newcommand</code> .....	96, 99		
<code>\newpage</code> .....	8		
N <small>F</small> S <small>S</small> .....	87		
<code>\normalsize</code> .....	85		
<i>Notepad</i> .....	6		
<b>O</b>			
O <small>A</small> S <small>I</small> S.....	103		
<i>octothorpe</i> .....	22		
<i>Office 11</i> .....	103		
<i>Omnimark</i> .....	103		
oneside.....	30		

- OpenOffice* ..... 103
- Opera* ..... 39
- Options**
- 11pt ..... 30
  - 12pt ..... 30
  - a4paper ..... 31
  - alpha ..... 72
  - b ..... 65
  - draft ..... 30
  - letterpaper ..... 31
  - oneside ..... 30
  - options ..... 114
  - pdftex ..... 86
  - plain ..... 72
  - t ..... 65
  - titlepage ..... 30
  - twoside ..... 30
- options ..... *see* Class Options
- options ..... 114
- Rogue ..... 75
- OS X ..... 13
- Ota, Takaaki ..... 59
- \ovalbox** ..... 67
- oxford* ..... 73
- P**
- package ..... 8, 114
- Packages**
- array ..... 58
  - article ..... 29
  - babel ..... 27, 37
  - bbding ..... 54, 101
  - book ..... 29, 34
  - color ..... 47–49, 86, 87
  - endnote ..... 69
  - epsf ..... 61
  - fancybox ..... 64, 67, 100
  - fancyhdr ..... 79
  - fancyvrb ..... 64
  - fnpara ..... 69
  - fontname ..... 94
  - geometry ..... 48, 49, 70, 78
  - graphics ..... 49
  - graphicx ..... 61, 62
  - inputenc ..... 23, 24, 28
  - jurabib ..... 75
  - komascript ..... 29, 30
  - letter ..... 29
  - makeidx ..... 75
  - marvosym ..... 22
  - mdwlist ..... 78
  - memoir ..... 29, 30
  - multicol ..... 76
  - newcent ..... 82
  - package ..... 8, 114
  - palatcm ..... 82
  - palatino ..... 82
  - paralist ..... 51, 55
  - parskip ..... 37
  - pifont ..... 54
  - preview-latex ..... 9
  - pslatex ..... 82
  - quotation ..... 68
  - quote ..... 68
  - report ..... 29, 34
  - sectsty ..... 35, 78
  - setspace ..... 78
  - so ..... 26
  - ssection ..... 35
  - tabular ..... 66
  - tabularx ..... 58
  - textcomp ..... 22
  - times ..... 82
  - type1cm ..... 85
  - url ..... 64
- packages
- documentation ..... 48
  - downloading ..... 49
  - installing ..... 49
  - using ..... 47
- packages** ..... 47
- page size
- margins ..... 78
  - scaling ..... 46
- PageMaker* ..... 102
- \pageref** ..... 71
- \pagestyle** ..... 79, 80
- palatcm ..... 82
- palatino ..... 82
- panels ..... 65
- paper sizes ..... 30
- \par** ..... 60, 78, 97
- \paragraph** ..... 35, 56
- paralist ..... 51, 55
- \parbox** ..... 65, 66
- \parindent ..... 37
- parskip ..... 37
- \parskip ..... 8, 36, 37
- \part** ..... 35
- \part\*** ..... 36
- pc (picas) ..... 25
- PCL ..... 62
- PDA ..... 12, 102
- PDF. 2, 12–14, 18, 23, 39, 41, 43–45, 49, 62, 63, 65, 88, 103, 104
- pdflatex* 39, 41, 43, 44, 49, 62, 63, 80, 82, 86, 88, 95, 102, 103
- pdftex ..... 86
- pdftotext* ..... 103
- PDFview* ..... 39, 45
- pdfview* ..... 95
- \person** ..... 99, 100



<i>PFAedit</i> .....	88
<i>PFB</i> .....	88, 91, 95
<i>PFE</i> .....	14
<i>picas</i> .....	25
<i>picture</i> .....	61
<i>pifont</i> .....	54
<i>plain</i> .....	72
<i>plain-text</i> .....	16
<i>PNG</i> .....	62, 62, 63
<i>points</i> .....	25
<i>Popineau, François</i> .....	13
<i>PostScript</i> .. 2, 12, 13, 18, 45, 46, 63, 80, 82, 85–88, 92, 95, 106	
<i>£</i> .....	22
<i>preamble</i> .....	34
<i>preview</i> .....	43
<i>preview-latex</i> .....	9
<i>\printindex</i> .....	75
<i>printing</i> .....	39, 45
<i>reverse order</i> .....	46
<i>selected pages</i> .....	46
<i>product</i> .....	8, 114
<i>\product</i> .....	83, 86, 98
<b>Products</b>	
<i>3B2</i> .....	102
<i>Acrobat</i> .....	103
<i>Acrobat Reader</i> .....	13, 39, 44, 45
<i>Adobe Illustrator</i> .....	61
<i>afm2tfm</i> .....	90, 92
<i>apt-get</i> .....	14
<i>Autorun</i> .....	15
<i>Balise</i> .....	103
<i>BBedit</i> .....	14
<i>bibtex</i> .....	72
<i>charmap</i> .....	23
<i>Chocolate Stout</i> .....	75
<i>configure</i> .....	50
<i>Corel Draw</i> .....	61
<i>Crayola</i> .....	11, 86
<i>Cygwin</i> .....	51
<i>detex</i> .....	103
<i>DocBook</i> .....	8
<i>dvialw</i> .....	46
<i>dvieps</i> .....	46
<i>dvilhp</i> .....	46
<i>dvips</i> .....	45, 46, 86, 91
<i>dviview</i> .....	44
<i>DynaTag</i> .....	102
<i>elsevier</i> .....	73
<i>Emacs7–9, 14–16, 18, 19, 22, 40, 41, 43, 46, 59, 60, 72, 73, 103</i>	
<i>EuroMath</i> .....	102
<i>Excel</i> .....	103
<i>fontinst</i> .....	89
<i>fontname</i> .....	89
<i>FrameMaker</i> .....	102
<i>Ghostscript</i> .....	45
<i>GhostView</i> .....	13
<i>GSView</i> .....	15
<i>GSview</i> .....	13, 39, 45, 46, 95
<i>Gutta-Percha</i> .....	89
<i>gv</i> .....	13
<i>harvard</i> .....	73
<i>HTML Tidy</i> .....	102
<i>Instant Preview</i> .....	9
<i>Java</i> .....	8
<i>kluwer</i> .....	73
<i>L<sup>A</sup>T<sub>E</sub>X2HTML</i> .....	103
<i>makebst</i> .....	74
<i>makeindex</i> .....	75
<i>Maple</i> .....	102
<i>Mathematica</i> .....	102
<i>Metamorphosis</i> .....	103
<i>mkfs</i> .....	14
<i>mktexlsr</i> .....	50, 110
<i>Mozilla</i> .....	51
<i>My Computer</i> .....	15
<i>Notepad</i> .....	6
<i>Office 11</i> .....	103
<i>Omnimark</i> .....	103
<i>OpenOffice</i> .....	103
<i>Opera</i> .....	39
<i>oxford</i> .....	73
<i>PageMaker</i> .....	102
<i>pdflatex</i> 39, 41, 43, 44, 49, 62, 63, 80, 82, 86, 88, 95, 102, 103	
<i>pdftotext</i> .....	103
<i>PDFview</i> .....	39, 45
<i>pdfview</i> .....	95
<i>PFAedit</i> .....	88
<i>PFE</i> .....	14
<i>product</i> .....	8, 114
<i>Publisher</i> .....	102
<i>Saxon</i> .....	8
<i>ScanDisk</i> .....	15
<i>Scientific Word</i> .....	9
<i>Star Office</i> .....	103
<i>t1binary</i> .....	88
<i>t1utils</i> .....	88
<i>texhash</i> .....	50, 110
<i>Textures</i> .....	9
<i>tkbibtex</i> .....	73
<i>tkpaint</i> .....	61
<i>up2date</i> .....	14
<i>updmap</i> .....	89, 92, 94
<i>Velcro</i> .....	98
<i>vi</i> .....	14
<i>Windows Update</i> .....	15
<i>WinEdt</i> .. 7, 8, 13–16, 18, 19, 40, 41, 43, 44, 72	
<i>WinShell</i> .....	7, 8, 14, 15
<i>Word</i> .....	7, 102, 103

- WordPerfect* ..... 103  
*xdvi* ..... 44  
*Xemacs* ..... 15  
*xkeycaps* ..... 23  
*XML Spy* ..... 103  
*Xpdf* ..... 13  
*xpdf* ..... 13  
*XPress* ..... 102  
*Zaurus* ..... 12  
**\protect** ..... 69  
**\ProvidesPackage** ..... 93  
 PS ..... 103  
 pslatex ..... 82  
 pt (points) ..... 25  
*Publisher* ..... 102
- ## Q
- \qqquad** ..... 79  
**\quad** ..... 25, 79  
 quotation ..... 68  
 quotation marks ..... 22  
 quote ..... 68
- ## R
- raggedleft ..... 27  
**\raggedleft** ..... 27  
 raggedright ..... 27  
**\raggedright** ..... 27, 65  
 Raggett, Dave ..... 102  
**\raisebox** ..... 101  
**\ref** ..... 56, 70, 71  
 references ..... 71  
**\refname** ..... 73  
**\reindex** ..... 99  
**\renewcommand** ..... 33, 34, 97, 101  
 report ..... 29, 34  
 RGB ..... 86, 86  
**\rightmark** ..... 80  
**\rmdefault** ..... 93  
 rotate ..... 62  
 RTFM ..... 14, 94  
 RTFML ..... 2
- ## S
- Saxon* ..... 8  
*Sbox* ..... 67, 100  
 scale ..... 62  
*ScanDisk* ..... 15  
 Schenk, Christian ..... 13  
*Scientific Word* ..... 9  
**\scriptsize** ..... 85  
**\scshape** ..... 83  
*secnumdepth* ..... 8, 36  
**\section** ..... 35, 37  
*section* ..... 101  
 section numbering ..... 35
- sections ..... 29, 34  
 sectsty ..... 35, 78  
**\selectfont** ..... 82  
**\sentinel** ..... 99  
*series* ..... 83  
**\setcounter** ..... 36  
**\setlength** ..... 36, 37, 66  
 setspace ..... 78  
**\sfdefault** ..... 93  
**\sffamily** ..... 83, 97  
 SGML ..... 29, 102–104, 104  
**\shadowbox** ..... 67, 100  
*shape* ..... 83  
 sidebars ..... 65  
**\slshape** ..... 83  
**\small** ..... 69, 85  
**\smallskip** ..... 78  
 so ..... 26  
 sp (scaled points) ..... 25  
 space ..... *see* white-space  
**\space** ..... 100  
**\spaceskip** ..... 26  
 spacing ..... *see* white-space, *see*  
     double-spacing  
 special characters ..... 21  
 ssection ..... 35  
*Star Office* ..... 103  
**\subparagraph** ..... 35, 56  
**\subparagraph\*** ..... 36  
**\subsection** ..... 35  
**\subsubsection** ..... 35  
 summaries ..... 33  
*synchronous* ..... 9
- ## T
- † ..... 65  
*t1binary* ..... 88  
*t1utils* ..... 88  
**\tabcolsep** ..... 59  
 table ..... 58, 61  
 table of contents  
     adding manual entry ..... 38  
     automated entries ..... 37  
 table\* ..... 76  
**\tableofcontents** ..... 37, 38  
 tables ..... 57  
 tabular ..... 58, 60, 66  
 tabular ..... 66  
 tabularx ..... 58  
 TDS ..... 49, 50, 87  
*temporary directory* ..... 49  
*term* ..... 8  
 teTeX ..... 7, 13  
*texhash* ..... 50, 110  
 TeX Live ..... 7  
**\textcolor** ..... 86, 87  
 textcomp ..... 22

<code>\textdegree</code> .....	28	<code>\vbox</code> .....	66
<code>\texteuro</code> .....	22	<i>Velcro</i> .....	98
<code>\textit</code> .....	86	<code>\verb</code> .....	63, 64, 69
<code>\textsterling</code> .....	22	Verbatim .....	64
<code>\texttrademark</code> .....	98	verbatim .....	64
<i>Textures</i> .....	9	verbatim text .....	63
<code>\thechapter</code> .....	101	<i>vi</i> .....	14
<code>\theenumi</code> .....	56	viewing .....	39
<code>\theenumii</code> .....	56	VMS .....	19
<code>\theenumiii</code> .....	56	Volovich, Vladimir .....	95
<code>\theenumiv</code> .....	56	<code>\vspace</code> .....	78, 79
<code>\theSbox</code> .....	100	<code>\vspace*</code> .....	78
<code>\thinspace</code> .....	23, 79		
<code>\thispagestyle</code> .....	79	<b>W</b>	
TIFF .....	62	white-space .....	21
times .....	82	double-spacing .....	78
<code>\tiny</code> .....	85	hard .....	26
<code>\title</code> .....	32, 97	horizontal .....	79
<code>\titleof</code> .....	72	vertical	
titlepage .....	30	disappearing .....	78
titles .....	31	fixed .....	78
<i>tkbibtex</i> .....	73	flexible .....	78
<i>tkpaint</i> .....	61	<i>Windows Update</i> .....	15
<i>tocdepth</i> .....	36, 38	<i>WinEdt</i> 7, 8, 13–16, 18, 19, 40, 41, 43, 44, 72	
tools .....	68	<i>WinShell</i> .....	7, 8, 14, 15
tracking .....	<i>see</i> letterspacing	<i>Word</i> .....	7, 102, 103
<code>\ttdefault</code> .....	93	<i>WordPerfect</i> .....	103
<code>\ttfamily</code> .....	83	WYSIWYG .....	12, 18, 43
twoside .....	30	WYSIWYM .....	18
type\cm .....	85		
typesetting .....	39	<b>X</b>	
typographics .....	77	<i>xdvi</i> .....	44
<b>U</b>		<i>Xemacs</i> .....	15
units .....	25	<i>xkeycaps</i> .....	23
Unix .....	13	XML .. 7, 8, 10, 12, 19, 29, 64, 102–104, 104	
<i>up2date</i> .....	14	XML .....	7
<i>updmap</i> .....	89, 92, 94	<i>XML Spy</i> .....	103
<code>\upshape</code> .....	83	<i>Xpdf</i> .....	13
URI .....	41, 51, 63, 64, 64	<i>xpdf</i> .....	13
URL .....	64	<i>XPress</i> .....	102
url .....	64	XSL .....	104, 104
<code>\url</code> .....	64, 69	XSLT .....	2, 8, 103, 104
<code>\usepackage</code> .....	34, 47, 48, 74, 88		
<b>V</b>		<b>Z</b>	
<i>variables</i> .....	114	<i>Zaurus</i> .....	12