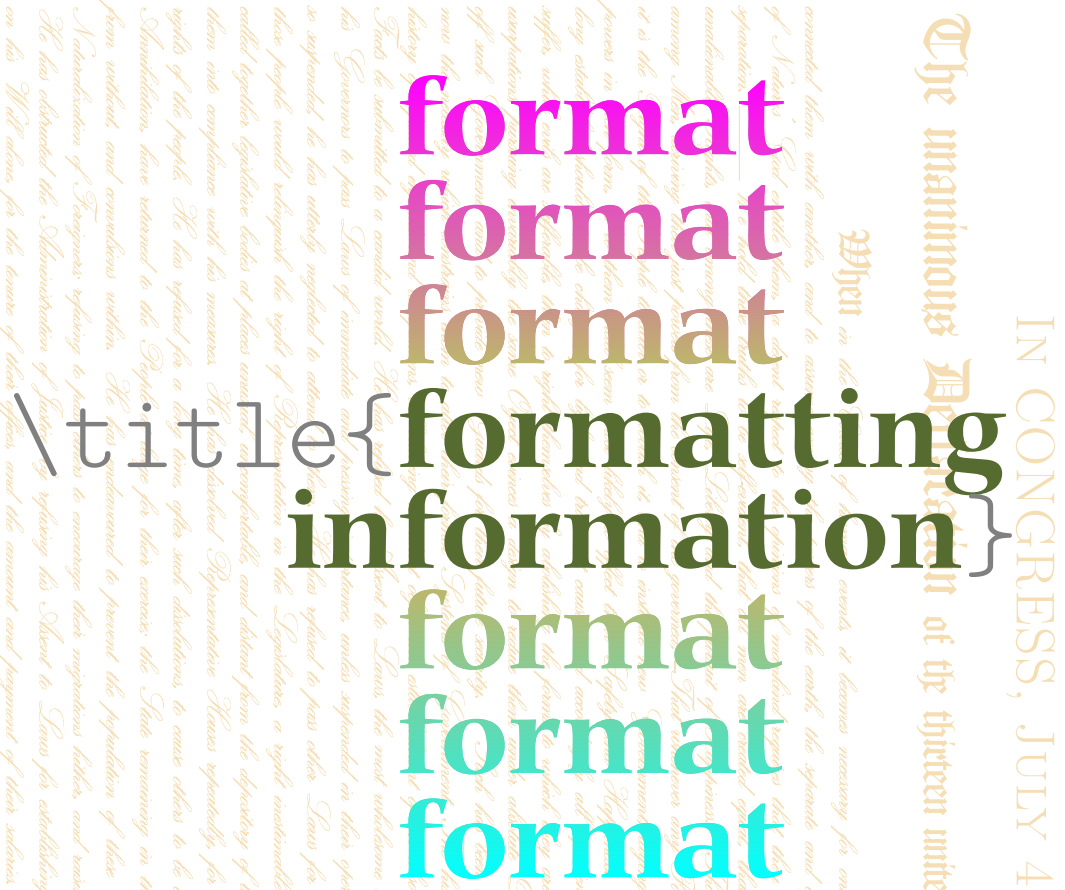


An introduction to typesetting with LaTeX

Peter Flynn

Formatting information

An introduction to typesetting with LaTeX

Acknowledgments

Earlier editions of 'Formatting Information' were prompted by the generous help I received from TEX users too numerous to mention individually. Shortly after TUGboat published the November 2003 edition, I was reminded by a spate of email of the fragility of documentation for any system which is constantly under development. While the core of LATEX is as stable as ever, there been revisions to packages, issues of new distributions, new tools, new interfaces, new books and online documents, corrections to my own errors, suggestions for rewording, and in one or two cases mild abuse for having omitted package X which the author felt to be indispensable.

The current edition is the result of a few years of allowing it to lie fallow, accumulating suggestions and finding errors, but taking on board the large number of changes which daily pass in front of all of us who read `comp.text.tex`, and the sometimes more obvious changes visible when one installs a new version of TEX.

I am grateful as always to the people who sent me corrections and suggestions for improvement. Please keep them coming: only this way can this book reflect what people want to learn. The same limitation still applies, however: no mathematics, as there are already a dozen or more excellent books on the market — as well as online documents — dealing with mathematical typesetting in TEX and LATEX in finer and better detail than I am capable of (and listed in *About this book* on p. xvii).

As I was finishing an earlier edition, I was asked to review an article for *The PracTEX Journal*, which grew out of the Practical TEX Conference in 2004. The author specifically took the writers of documentation to task for failing to explain things more clearly, and as I read more, I found myself agreeing, and resolving to clear up some specific problems areas as far as possible. I was delighted to see at the Practical TEX Conference, in 2006 and later, that more presenters, especially in the Humanities, have stepped up to Peter Flom's challenge.

It is very difficult for people who write technical documentation to remember how they struggled to learn what has now become to them 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. As computer systems become more sophisticated, they require less detailed knowledge from users, even while the takeup of computer usage rises. The result is a generation of users who know what they want, but who are wholly incapable of knowing when they've got it; who have only ever seen one way of doing something, and believe that if the result looks pretty, it means it must be right. As technical writers, we need to explain *why*, not just *how*, so if I have obscured something by making

unreasonable assumptions about *your* knowledge, please let me know so that I can correct it.

The author

Peter Flynn has been using structured text systems since the early 1980s and is author of *The HTML Handbook* and *Understanding SGML and XML Tools*, and editor of *The XML FAQ* (all typeset with LaTeX). Peter was Ireland's first webmaster, and also runs the text management consultancy Silmaril. When not TeXing or XMLing, he likes to cook, surf, read science fiction, and listen to Early and Baroque music. In his increasingly fractal spare time he is finishing his PhD on the usability of editors for structured text.

Technical note

This book is written and maintained in XML using the DocBook DTD with a customization layer for typographic markup. XSLT is used to generate HTML (for the Web and plain-text versions) and LaTeX (for PDF). The November 2003 edition was published in *TUGboat*.

Contents

List of Figures

List of Exercises

List of Tables

Foreword

This document originally accompanied a two-day introductory training course. 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 use the TUG web site or the CTAN repositories, do not buy the books and manuals, do not use the newsgroups and mailing lists, and do not download or read the free documentation. Instead, they try to get by using the time-honoured training technique known as 'sitting by Nellie', which involves looking over a colleague's shoulder in the office, lab, library, pub, or classroom, and absorbing all of 'Nellie"s bad habits along with the good ones. And they use guesswork or imagination for the rest.

People do this for many reasons: shortage of time, lack of information (no-one ever told them there was free documentation), dislike of reading manuals, or even just laziness (my own excuse). But chiefest of reasons is that so much of the existing documentation is written for people who are already experts at reading documentation. Most beginners don't want extensive reasoning over the available choices: they want simple, direct, prescriptive instruction. If you want one of *these*, do *this*.

In the summer of 2001 I presented a short proposal on the marketing of LaTeX to the annual TUG conference held at the University of Delaware, and I showed an example of a draft brochure[1] 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 it appears in almost every other manual and book on LaTeX but it is essential to the beginner and therefore bears repetition. Some of it appears in other forms elsewhere, and is included here because I felt it needed explaining. And some of it appears nowhere else but this document. I took the opportunity to revise the structure of the training course in parallel with the book, and to include a more comprehensive index. It is by no means perfect, and I would be grateful for comments and corrections to be sent to me at the address given under the credits. As I also noted earlier, it can be used as a one-day course if the users already have some experience of writing formal documents for publication or assessment (e.g. books, reports, essays, theses, articles, etc.).

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 the

[1] `http://latex.silmaril.ie/brochure/`

need to use some less-common packages which the new user is unlikely to have installed, meant that this document itself was not really a simple piece of LaTeX, no matter how simply it may describe the process itself.

However, as the careful reader may already have seen, the master source of the document is not maintained in LaTeX but in XML. Installations of TeX are becoming more comprehensive, which means that modern systems are likely to include all the fonts and packages needed, so what I called last time 'a future task' is now creeping up fast: to rewrite the transformation to that it can be guaranteed to process with all the current full LaTeX installations.

If you are just starting with LaTeX, at an early opportunity you should buy or borrow a copy of *LaTeX: A Document Preparation System* which is Leslie Lamport's original manual. More advanced users should get the *The LaTeX Companion*, the *The LaTeX Graphics Companion* and the *The LaTeX Web Companion*. Mathematical users might want to start with the *Short Math Guide for LaTeX*. Details are in p. 198.

Online resources

There are hundreds if not thousands of web pages about how to use LaTeX. The online version of this book is just one, but a couple of others I recommend are:

- Marc van Dongen's LaTeX and Friends (Author's publication, Cork, Ireland), 2011, at `http://csweb.ucc.ie/~dongen/LaTeX-and-Friends.pdf`
  Peter Flynn's The Very Short Guide to LaTeX (Author's publication, Cork, Ireland), 2011, at `http://latex.silmaril.ie/veryshortguide/veryshortguide.pdf`

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 over 30 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 has 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. Playing around with fonts and formatting is highly attractive to new computer users, and great fun for a while, but it is completely counter-productive for the serious author or editor who needs to concentrate on actual *writing* — ask any journalist or professional writer. 'Best-guess' estimates by experts in the field of usability engineering are that average computer users spend up to 50% of their time fiddling with the formatting rather than thinking or writing — and this is with the so-called 'office productivity packages' that major manufacturers peddle to their clients!

A few years ago a new LaTeX user expressed concern on the `comp.text.tex` newsgroup about 'learning to write in LaTeX'. Some excellent advice[2] was posted in response to this query, which I reproduce with permission below (the bold text is my own emphasis):

> 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 *Crayola*s -- 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)]

[2] `news:comp.text.tex/MPG.18d82140d65ddc5898968c@news.earthlink.net`

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' LaTeX systems can use any OpenType, TrueType, Adobe (PostScript) Type1 or Type3 (METAFONT) font. This is more than any other known typesetting system. LaTeX's default font is Computer Modern (based on Monotype Series 8: see the table on p. 133), not Times Roman, and some people get very upset because Computer Modern looks different to Times (I'm not making this up: it's just a typeface, guys, get over it).

MYTH: 'LaTeX isn't WYSIWYG' Simply not true. TeX's DVI and PDF is generally better quality WYSIWYG than any wordprocessor and most DTP systems. What people mean is that LaTeX's typographic display is asynchronous with the editor window. This is only true for the default CLI implementations. See About this book on p. xx for details of synchronous versions.

MYTH: 'LaTeX is obsolete' Quite the opposite: it's under constant development, with new features being added weekly. Check `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.

Learning to write well can be hard, but authors shouldn't have to make things even harder for themselves by using manually-driven systems which break their concentration every few seconds for some footling adjustment to the appearance, simply because the software is incapable of doing it right by itself.

Donald Knuth originally wrote TeX to typeset mathematics for the second edition of his master-work *The Art of Computer Programming*, and it remains pretty much the only typesetting program to include fully-automated mathematical formatting done the way mathematicians do it. But he also brought out a booklet called *Mathematical Writing* which shows how important it is to think about what you write, and how the computer should be able to help, not hinder, the author while writing.

TeX is of course much more than math: it's a programmable typesetting system which can be used for almost any formatting task, and the LaTeX document preparation system which is built on it has made it usable by almost anyone. Professor Knuth generously placed the entire TeX system in the public domain, which meant it is free for anyone to use, but which also meant that for many years there was little commercial publicity which

would have got TEX noticed outside the technical field, because there were few commercial versions.

Nowadays, however, there are many companies selling TEX software or services,[3] dozens of publishers accepting LATEX documents for publication, and hundreds of thousands of users using LATEX for millions of documents.[4]

There is occasionally some confusion among newcomers between the two programs, TEX and LATEX, and the other versions available, so I'd like to clear this up:

TEX A typesetting program, originally written by Don Knuth at Stanford in 1978–9. 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 A user interface for TEX, designed by Leslie Lamport at 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 underlying language. LATEX is the recommended system for all users except professional typographic programmers and computer scientists who want to study the internals of TEX.

ConTEXt (not 'Contest') A system similar to LATEX, but with its own set of commands, and a much greater emphasis on producing high-function PDF output. Documentation is less accessible than for LATEX, but the author, Hans Hagen provides excellent support at Pragma/ADE[5].

XETEX A recent reimplementation of TEX by Jonathan Kew which merges Unicode and modern font technologies. It is in common use in graphical environments such as TEXshop (Mac OS X), *Kile* (GNU/Linux), and *WinEDT* (Windows). Details are at the SIL web site[6].

pdfTEX and pdfLATEX An extended version of the TEX program that creates PDF directly instead of via DVI files and PostScript, written by Hàn Thế Thành. It can also enhance the result of TEX/LATEX typesetting with the help micro-typographic extensions, native font embedding, and

[3] See, for example, the list of TEX vendors in Table 1, and the list of consultants published by TUG.

[4] A guesstimate. With free software it's virtually impossible to tell how many people are using it.

[5] `http://www.pragma-ade.nl/`

[6] `http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=xetex`

More mythology

MYTH: 'LaTeX is a Unix system' People are also heard saying it's 'a Windows system', 'a Mac system', etc., etc. ad nauseam. TeX systems run on almost every computer in use, from the biggest supercomputers right down to handhelds (PDAs like the Sharp Zaurus, the Nokia N800, and most Apple and Android devices). That includes Windows and Linux PCs, Macs, and all other mini, mainframe, and 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 "too difficult" ' This has been heard from physicists who can split atoms; from mathematicians who can explain why π 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'. It's complete nonsense: most people can 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" Untrue. 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 web-based typesetting.

PDF support for hyperlinking. It can also produce DVI files, so it is currently (2011) the default TeX engine in most distributions.

TeXinfo Texinfo is the official documentation format of the GNU project.[7] It was invented by Richard Stallman and Bob Chassell. It uses a single source file to produce output in a number of formats, both online and printed (DVI, HTML, INFO, PDF, XML, etc.). TeXinfo documents can be processed with any TeX engine.

Both TeX and LaTeX have been constantly updated since their inception. Knuth has now frozen changes to the TeX engine so that users and developers can have a virtually bug-free, rock-stable platform to work with.[8] Typographic programming development continues with the New Typesetting System (NTS), planned as a successor to TeX. The LaTeX3 project has taken over development of LaTeX, and the current version is LaTeX 2_ε, which is what we are concentrating on here. Details of all developments can be had from the TUG web site at `http://www.tug.org`

[7] GNU's Not Unix (GNU) is a project to create a completely free computing system — 'free' meaning both free from encumbrances and restrictions as well as free of charge.

[8] Knuth still fixes bugs, although the chances of finding a bug in TeX these days approaches zero.

About this book

This book originally accompanied a 2-day course on using the LaTeX typesetting system. It was extensively revised and updated for publication, so that it could be used for self-study as well as in the classroom. For those with sufficient prior knowledge of computing and authoring, it has also successfully as the basis for a 1-day intensive introductory course. It is aimed at users of Linux, Apple Macintosh, or Microsoft Windows systems, but it can be used with LaTeX on any platform, including other Unix workstations, mainframes, and even some Personal Digital Assistant (PDA)s.

Who needs this book?

The course was originally designed for computer-literate professionals in business, academic, and nonprofit organisations. You may be in a similar position, but you may also come from another background entirely; you may be a hobbyist, a school or college student, a home computer user or a volunteer worker, or you might just be interested in high-quality automated typesetting. However, it's likely that you have one or more of the following or similar objectives:

- producing consistent, typeset-quality formatting
- formatting long *or* complex *or* highly-structured *or* repetitive *or* automatically-generated documents
- saving time and effort by automating common tasks
- gaining independence from expensive and restrictive proprietary hardware, software, or file formats
- creating robust, durable documents which will survive changes in technology

Skills needed

LaTeX is a very easy system to learn, and requires no specialist knowledge to get started, although it's useful if you understand something about formatting and readability. However, you do need to be completely familiar with using your own computer, which means knowing the following topics thoroughly. Note that none of these is specialist; they're fundamental, standard computer skills that everyone should know:

- how to point and click with your mouse to select text and pick from a menu (or to use keyboard shortcuts to do the same)

1.1.4–1.1.6, 1.3.1, 1.4.1–1.4.2,

- ☐ how to create, open, save, close, rename, copy, move, and delete files and folders (directories) 2.3
- ☐ where to find all 95 of the printable ASCII characters on your keyboard and what they mean, and how to type accents and symbols, if you need them 3.2.1.2
- ☐ how to use a good plain-text editor[9] 2.1.3
- ☐ how to use your Web browser and/or file transfer program to download and save files from the Internet 7.3.1.6
- ☐ how to uncompress and unwrap (unzip or detar) compressed files 2.3.7
- ☐ how to install software, both manually and using automated installers
- ☐ how to read and follow instructions 1.7

If you don't know how to do these things yet, it's important to go and learn them first. Trying to become familiar with basic computer skills *at the same time* as learning LaTeX is not going to be as effective as doing them in the right order.

It is important to understand that these are *not* specialist skills — they are standard for anyone who uses a computer, and they form a fundamental part of the basic knowledge of computers. With the exception of software installation, they are included in the European Computer Driving Licence (ECDL) course: the relevant sections of the ECDL syllabus are noted in the margin above.

Objectives of this book

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

- ☐ use your editor to create and maintain your documents
- ☐ use LaTeX markup to identify your document structure and formatting requirements
- ☐ typeset LaTeX documents, correct simple formatting errors, and display or print the results
- ☐ identify and use additional LaTeX packages (using the Internet for downloading where necessary and installing them)
- ☐ recognise the limitations of procedural markup systems and choose appropriate generic markup methods where appropriate

[9] *Not* a wordprocessor like *OpenOffice*, Lotus *Notes*, Corel *WordPerfect*, or Microsoft *Word*, and *not* a 'dumb' editor like Apple *TextEdit* or Microsoft *Notepad*.

Synopsis

The original course covered the following topics as separate sessions, and I have kept to this structure in the book as chapters:

1. Where to get and how to install LaTeX (using the TeX Users Group's TeX Collection DVD): you can skip this chapter if LaTeX is already correctly installed on your computer and you are already familiar with using a plaintext editor suitable for LaTeX work;
2. How to create LaTeX documents (with a Quick-Start Guide for the impatient);
3. Basic document structures (the Document Class Declaration and its layout options; the `document` environment with sections and paragraphs);
4. Typesetting, viewing, and printing;
5. Using packages and CTAN to adapt formatting to your needs;
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.).

I have made a few changes 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.

Where's the math?

Please understand that this 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 which cover these topics and more in great detail. Among the more popular are:

- *Getting Started with TEX, LATEX, and friends*, where all beginners should start;
- *The (Not So) Short Guide to LATEX 2ε: LATEX 2ε in 131 Minutes* is a good beginner's tutorial;
- *A Gentle Introduction to TEX: A Manual for Self-Study* is a classic tutorial on Plain TEX (not LATEX);
- *Using imported graphics in LATEX 2ε* shows you how to do (almost) anything with graphics: side-by-side, rotated, etc.;
- *Short Math Guide for LATEX* gets you started with the American Math Society's extensions;
- *A comprehensive list of symbols in TEX* shows over 2,500 symbols available.

This list was taken from the CTAN search page. There are also lots of books published about TEX and LATEX: the most important of these for users of this document are listed in the last paragraph of the *Foreword* on p. x.

Availability of LATEX systems

The standard implementations of TEX and related systems are in the TEX Collection, published annually on DVD by the TUG. These are all derived from Knuth's master versions, and adapted for all major platforms (Unix and Unix-like systems such as GNU/Linux and Apple Mac OS X; and Microsoft *Windows*). The DVD is sent free to all TUG members and can be obtained from your local user group. You can also download the ISO image file from CTAN to burn your own copy.

Commercial implementations are listed in *About this book* on p. xx.

Systems included on the TEX Collection DVD

ProTEXt (Windows) This is the popular *MiKTEX* implementation plus the *TEXMakerX* editor.

MacTEX (OS X) This is TEX Live plus the *TEXshop* editor (the Mac's built-in *Preview* is used for the WYSIWYG display).

TEX Live (Unix and GNU/Linux) Generic TEX Live for systems without a built-in package distribution. Users of Red Hat and Debian/Ubuntu systems should use their package manager instead (*yum* and RPM, or *apt* and *Synaptic*) to install the complete RPM or DEB package versions provided by their supplier's repository.

Because the TEX program (the internal 'engine' which does the actual typesetting) is independent of other software, it doesn't have its own editor like a wordprocessor does. Instead, you get to choose whichever editor you prefer: there are lots available, and you can switch between them to find one you like.

Graphical interfaces

Most users run LATEX with a graphical editor which has a toolbar and menus like other windowing applications. All the common formatting features of LATEX plus writing tools like spellchecking, indexing, and bibliographic citation are run the same way. Text-only interfaces are available for use on servers and automated production systems (see *About this book* on p. xix).

The Windows and Mac systems described in *About this book* on p. xviii come with a recommended editor (*TEXMakerX* and *TEXshop* respectively), but you can install any other suitable editor you prefer (see § 1.3). The Unix and GNU/Linux distribution does not install any editor because these systems usually have their own software repositories with suitable editors already available for installation, such as *Emacs*, *vi*, *TEXMaker*, or *Kile*.

Command-line interfaces

While you would use a graphical interface to *set up* an automated system like a web server or e-commerce environment, it is useless where LATEX is running in the background, unattended, where there is no human to click on buttons. In fact, the TEX typesetting engine is a Command-Line Interface (CLI) program, which can be used from a console or 'Command' window. You can type the command `latex` followed by the name of your document file (see Figure 4.1 in § 4.1.1 for an example).

Commands like these let you run LATEX in an automated environment like a Common Gateway Interface (CGI) script on a web server or a batch file on a document system. All the popular distributions for all systems, both free and commercial, include this CLI interface as standard.

WYSIWYG displays

LATEX usually displays your typeset results in a separate window, updated automatically every time the document is refreshed, because the typesetting is kept separate from the editing. This is called 'asynchronous' display. Some systems, however, can format the typesetting while you type each character, like a wordprocessor, although at the expense of some flexibility. These are called 'synchronous' displays.

Asynchronous typographic displays The WYSIWYG display is updated when the document is reprocessed, rather than *while* you are still typing, as it would with a wordprocessor. To update the display, just click on

the button which reformats the document. You are probably already familiar with this idea if you have used a spreadsheet, where the ReCalc button (F9) does the same thing.

TEX systems typeset the whole document at one go, including all indexing, cross-references, tables of contents, bibliographic citations, and the placement of figures and tables. TEX also formats whole paragraphs at a time, rather than line-by-line as wordprocessors do, in order to get the quality of spacing, hyphenation, and justification right. This approach makes it much faster than a wordprocessor in dealing with typical complex documents, as it can be done without holding the whole document in memory.

Synchronous typographic displays The WYSIWYG display is the editing window, and it updates while you type, like a wordprocessor. Some popular examples are *LyX* (all platforms), *Textures* (Mac), *BaKoMa TEX* (Windows), and *Scientific Word* (Windows) (see Table 1).

With a synchronous display you get Instant Textual Gratification™, but like a wordprocessor, your level of control is restricted to that of the system you use, which cannot provide access to everything that LATEX can do. For complete control of complex material you may still need to use separate editing and display windows as for asynchronous implementations.

Near-synchronous displays There are a few systems for very-close-to-synchronous WYSIWYG display. These include Jonathan Fine's *Instant Preview* and the TEX daemon, and David Kastrup's preview-latex package for embedding typographic fragments from the typeset display back into the editor window.

What You See Is What You Get (WYSIWYG) refers to the accuracy of the typographical display. Most modern ones are pretty good, given the fact that your screen is probably only a fraction of the accuracy of your printer (96 dots to the inch on your screen as opposed to 600 to the inch on your printer, or 1200 or more in photo-quality).

Commercial distributions

Although the TEX Collection is available free of charge, there are several excellent commercial implementations of TEX and LATEX, with enhanced support and additional features. If these are of benefit to you, I urge you to support them and buy their products. In most cases their companies, founders, and staff have been good friends of the TEX and LATEX communities for many years.[10]

[10] Y&Y, Inc, who produced a TEX distribution for many years, have ceased trading. Some of their add-on fonts are now being distributed by the TEX Users Group (see Appendix B),

Table 1: Popular commercial implementations of TEX systems

Product	Platform	Company	URI
PCTEX	MS-Windows	Personal TEX, Inc	http://www.pctex.com/
BaKoMa TEX	MS-Windows	Basil K Malyshev	http://www.bakoma-tex.com/
TrueTEX	MS-Windows	True TEX	http://truetex.com/
Textures	Apple Mac	Blue Sky Research	http://www.bluesky.com/
Scientific Word	MS-Windows	Mackichan Software	http://www.mackichan.com/
VTEX	MS-Windows, Linux, OS/2	MicroPress, Inc	http://www.micropress-inc.com/

Symbols and conventions

There are several typographic conventions about how you represent computer-related material in print, which are shown in Table 2. It is common to show typed commands, keywords, examples of input, and related text in a fixed-width (monospace) font, like an old typewriter, because that's how programs used to be printed, and how most programmers prefer to edit them. Special values, like numeric quantities represented by a name or symbol, are in italics, like they would be if they were mathematics. Terms or references to products, programs, packages, and other components of LATEX are given their own typographic form. Finally there are some symbols like keyboard keys and menus, which are shown graphically.

Production note

This document is written and maintained in XML, using a customized version of the *DocBook* DTD. Conversions were made to HTML and LATEX 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/src/`. If you want to try processing it yourself you will need *Java* (from Sun, IBM, or a number of others) and *Saxon* (from `http://saxon.sourceforge.net/`), in addition to a full installation of LATEX.

or have been replaced by Open Source implementations, and there is a mailing list at the TUG web site for the support of former Y&Y users.

Table 2: Typographic notations used in this document

Notation	Meaning
`\command`	LaTeX commands (control sequences) that you type which perform an action like `\clearpage`, or identify your text like `\footnote`
`\length`	Control sequences which store a dimension (a measurement in units), like `\textwidth`
counter	Values used for counting (whole numbers only), like *section*
term	The defining instance of a new term or acronym (indexed)
`environment`	A LaTeX formatting or identification environment, like `quotation`
package	A LaTeX add-on package (available from CTAN), like footmisc
product	A program or product name
`typewriter type`	Literal examples
mybook or *value*	Mnemonic examples of things you must type, where you have to supply real-life values of your own, like `\author{`*your name*`}`+\author{your name}+ means you must replace *your name* with your own real name.
X	A specific key on your keyboard
Ctrl–X	Two keys pressed together, not separately
Esc q	Two keys pressed one after another
Submit	An on-screen button to click
Menu Submenu *Item*	A drop-down menu with items;

1 Installing TEX and LATEX

TEX Live installs very easily on all modern desktop, laptop, and server platforms. You can get a copy of the DVD from your local TEX user group, or you can download the installation from CTAN in `http://www.ctan.org/tex-archive/systems/`

This course is based on using one of the following distributions of TEX from the TEX Collection DVD:

TEX Live for GNU/Linux, Unix, Mac OS X, and Microsoft Windows (this supersedes Thomas Esser's teTEX distribution which is no longer supported);

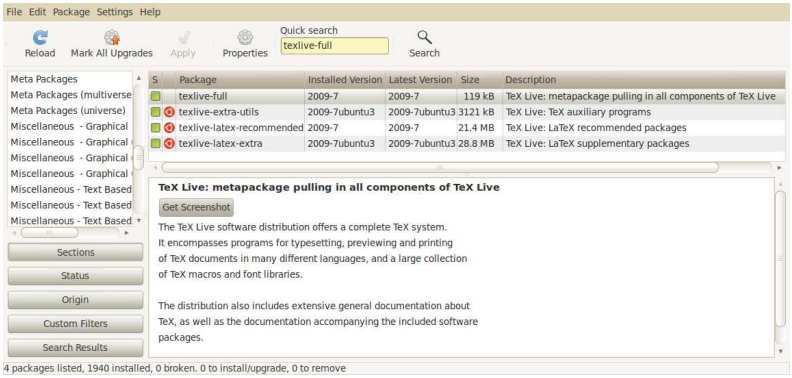
proTEXt for Microsoft Windows (by Thomas Feuerstack) which includes Christian Schenk's *MiKTEX*, the *TEXMakerX* editor, *GhostScript*, and *GSView*;

MacTEX for Apple Macs running OS X. *MacTEX* includes the *TEXshop* editor.

Other implementations of TEX can be downloaded from CTAN, but the above are the standard systems. LATEX is included with all distributions of TEX.

The TEX Collection DVD is issued annually by TUG in conjunction with many of the local TEX user groups around the world (see `http://www.tug.org/lugs.html` for addresses), and edited by Sebastian Rahtz, Karl Berry, Manfred Lotz, and the authors of the software mentioned above.

Figure 1.1: Installing TEX Live from *Synaptic*, the *Ubuntu* package manager



These people give an enormous amount of their personal time and energy to building and distributing these systems, and they deserve the thanks and support of the user community for all they do.

There is also a selection of commercial distributions you can buy, as described in p. i in §: they all process LATEX identically, but there are some differences in size, speed, packaging, installation, support, and extra software provided.

One final thing before we start: always check to see if there is a more recent version of the installation program online. See the list on p. 16 in § 1.4 for more details.

1.1 Installing the software

1.1.1 Unix and GNU/Linux

Users of modern GNU/Linux systems don't even need the TEX Collection DVD, as TEX installation packages are available online, built into the package manager for your system.

If your system has a graphical package manager (e.g. Ubuntu's *Software Center* or *Synaptic*, see Figure 1.1), run it and install `texlive-full`, `ghostscript`, `gv`, `okular`, `kile` (or `texmaker` or `emacs`), and `jabref`. Some of them may already be installed on your system. Go and have a cup of coffee while they automatically install all the necessary components.

The new *Ubuntu Software Centre* only allows one package to be installed at a time, so you may prefer to use a typed command (other Linux users may prefer this approach anyway). The command name varies from distribution to distribution; but two common ones are *yum* and *apt*:

```
sudo apt-get install texlive-full ghostscript gv okular kile texmaker emacs jabref
```
or
```
sudo yum install texlive-full ghostscript gv okular kile texmaker emacs jabref
```

Unfortunately, the *Okular* PDF/DVI viewer is new and incomplete: it prematurely replaced two older and much better-designed viewers, *kdvi* and *kpdf*; fortunately, at the time of writing, it is still possible to download those from different repositories and install them manually.

Unix and GNU/Linux installers I strongly recommend this method for all GNU/Linux users. You should only install from the TEX Collection DVD if you are using an older, hand-built, or commercial Unix system which has no package manager. Installing from the TEX Collection DVD for Unix requires root privileges and a good understanding of Unix systems management, and is beyond the scope of this book.

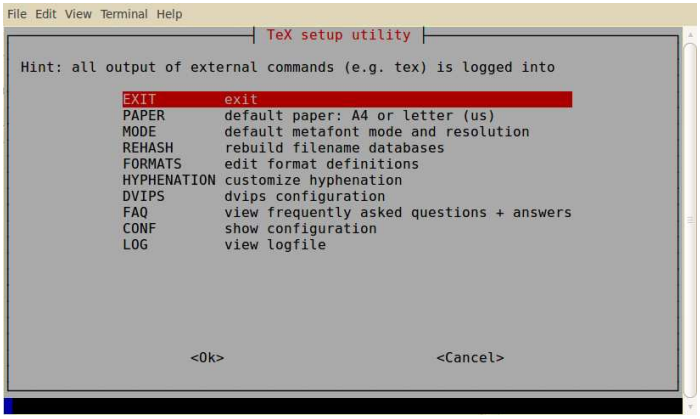
There is a minor drawback: the Linux repository versions of TEX are usually a year out of date, because of the volunteer effort required to put them together. For new users this should not be a concern, as updates do not affect core facilities. If there are very recent packages you badly need, you can install them separately.

If you do decide to switch to the TEX Live DVD, make sure you completely uninstall the *texlive-full* package first, otherwise your system will get hopelessly confused.

After installation, run *texconfig* (see Figure 1.2) in a terminal window to adjust your local settings. This is a console utility, so type `texconfig` to adjust your personal settings, or `sudo texconfig` to adjust them system-wide. In the utility, use the arrow keys to go up and down the options, and the TAB key to jump to (and switch between) the OK, Cancel, and other 'buttons' at the foot of the screen. The spacebar or the Enter key selects a menu item or button. Most settings are correct as installed, but you might want to change one of the following:

- the first option, DEST, lets you specify whether you normally want to print straight onto the printer, or 'print' into a file (to attach to email or upload somewhere)
- the default paper size (the PAPER option), if the installed size is not your most common one (A4 or Letter)
- the printer resolution (the MODE option), where you can adjust your printer settings; this allows you to fine-tune it for, say, a typesetter that you want to send output to instead of your own printer
- in the DVIPS option you can adjust your printer OFFSET (left and top margins), which is useful for older, less accurate printers

Figure 1.2: Running the post-installation program *texconfig*



You may also need the REHASH option later on. It is used to update TEX's fast-find database (see step 4 in the procedure on p. 82) after adding new or updated packages.

If your printer is a conventional home or office ink-jet or laser printer, and is not shown, the LaserJet5 setting (600dpi) is probably a good bet. While still in the utility, you can test the margin settings in another window by running the `testpage.tex` document through LATEX (by typing `latex testpage` and responding to the questions about paper size and double-sided printing). Print the resulting `.dvi` file with the command

```
dvips -f testpage | lpr
```

and adjust the margins in *texconfig* if necessary. These adjustments are not needed with PDF output.

1.1.2 Apple Mac OS X

Double-click the `MacTeX-yyyy-DVD.mpkg` package in the `mactex` folder of the TEX Collection DVD (replace the *yyyy* with the year of your distribution as shown on the DVD sleeve). Install the package in the normal Mac way by dragging the package icon onto the hard disk icon.

When it's finished, open your Applications folder in the Finder and go to the TEX subfolder and drag *TEXshop* out onto your Dock. *TEXshop* is the editor front-end to *MacTEX* which you use for writing your documents.

If you are going to use Adobe *Acrobat Reader* instead of *Preview*, make sure you clear the font cache and set the resolution to the system default (112dpi) otherwise you may get very weird displays.

Figure 1.3: The TEX Collection installation program

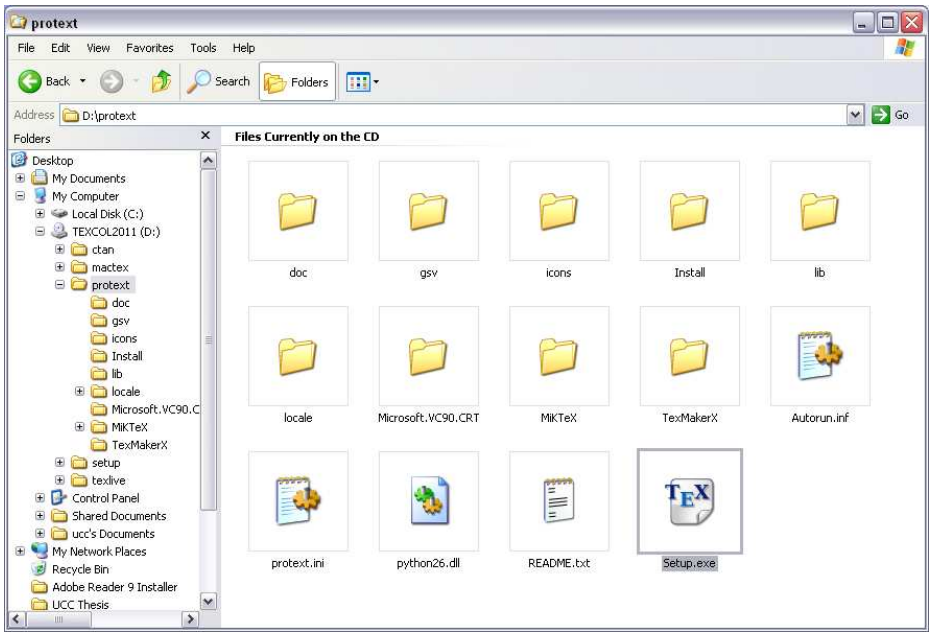
1.1.3 Microsoft Windows

If your system has auto-run enabled, inserting the TEX Collection DVD should start the setup program automatically (see Figure 1.3). Click on the Open proTeXt button to start.

If you have auto-run turned off, insert the DVD, go to the `protext` folder, and double-click on the `Setup.exe` program as shown in Figure 1.4.

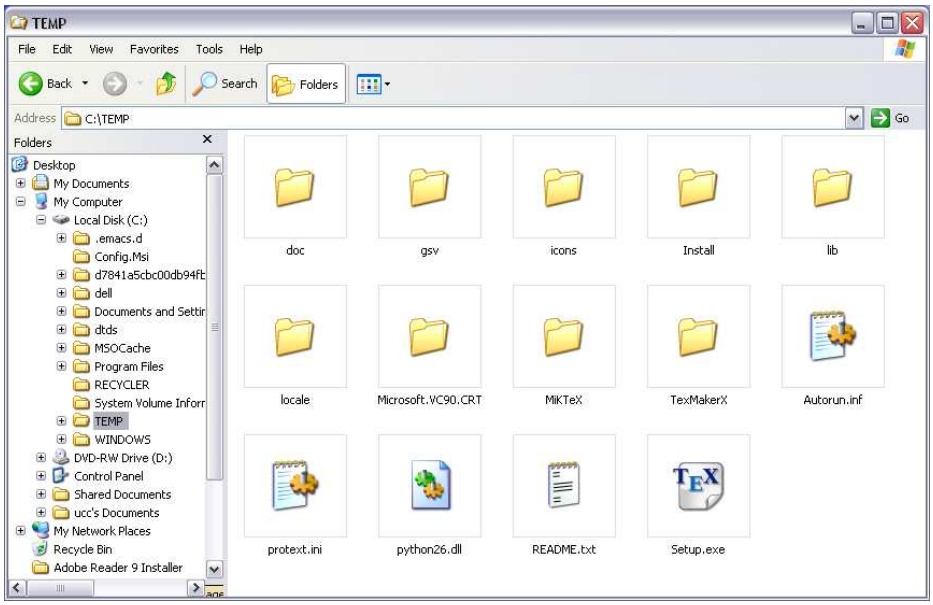
If you have downloaded the ProTEXt installation from the TEX Users Group web site, unzip it into a temporary folder and run the `Setup.exe` program from there (see Figure 1.5).

Figure 1.4: The ProTEXt setup program on the TEX Collection DVD



1. Make sure you have Adobe *Acrobat Reader* or a similar PDF reader installed
2. In the ProTEXt setup window, select your language and click on the *MiKTEX* Install button (see Figure 1.6)
3. Click in the box to accept the MiKTEX licence and click Next to continue (Figure 1.7)
4. Make sure the Complete MiKTEX option is selected and click Next to continue (Figure 1.8)
5. Choose a private installation or one that everyone who uses your computer can use, and click Next to continue (Figure 1.9)
6. Accept the installation folder that MikTEX suggests (unless you are an expert or have a special disk setup) and click Next to continue (Figure 1.10)
7. In the Options screen (Figure 1.11), select your paper size (A4 or US Letter), and whether or not you want extra packages to be downloaded and installed automatically (Yes or No) — on a laptop

Figure 1.5: The ProTEXt setup program in the unzipped download folder



where a network connection is not always present, choose 'Ask first' instead, then click Next to continue

8. Finally, accept the settings as shown in Figure 1.12 (or change them by clicking Back), and then click Start to start the installation process

9. During installation, MiKTEX will list the files it is installing and show a progress bar (Figure 1.13)

10. When it is all done, click Finish (Figure 1.14)

Two more things to do: install the *TEXmakerX* editor, and add a personal TEX folder for extra downloads such as additional fonts.

1. Go back to the proTEXt setup window and click on the *TEXmakerX* Install button (see Figure 1.6). The *TEXmakerX* installation program will start: click Next to continue (Figure 1.15)

2. Select the language to use during installation and click OK to continue (Figure 1.16)

3. Click Next in the following screen to continue (Figure 1.17)

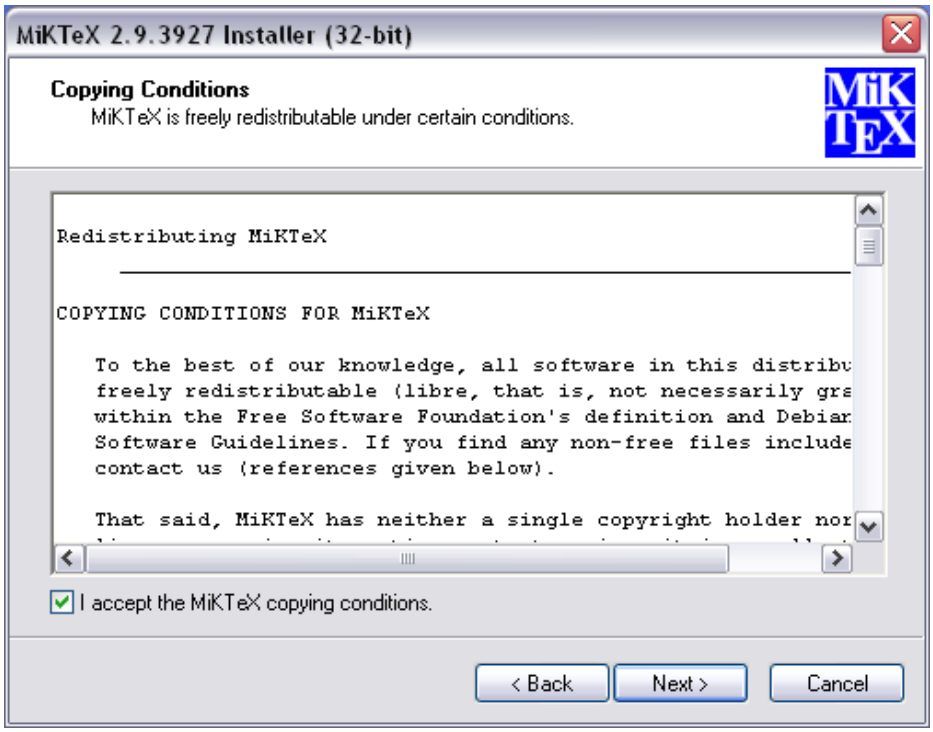
Figure 1.6: Installation of proTEXt

4. Accept the installation folder that TEXmakerX suggests (unless you are an expert or have a special disk setup) and click Next to continue (Figure 1.18)
5. Also accept the suggested location for the TEXmakerX shortcut and click Next to continue (Figure 1.19)
6. Finally, accept the settings as shown in Figure 1.20 (or change them by clicking Back), and then click Install to start the installation process
7. When it is all done, click Finish (Figure 1.21)

1.2 Adding your personal TEX directory

TEX Live and MiKTEX (but not MacTEX) will automatically download and install package updates and additional packages for you, by recognising

Figure 1.7: Accepting the MiKTEX licence



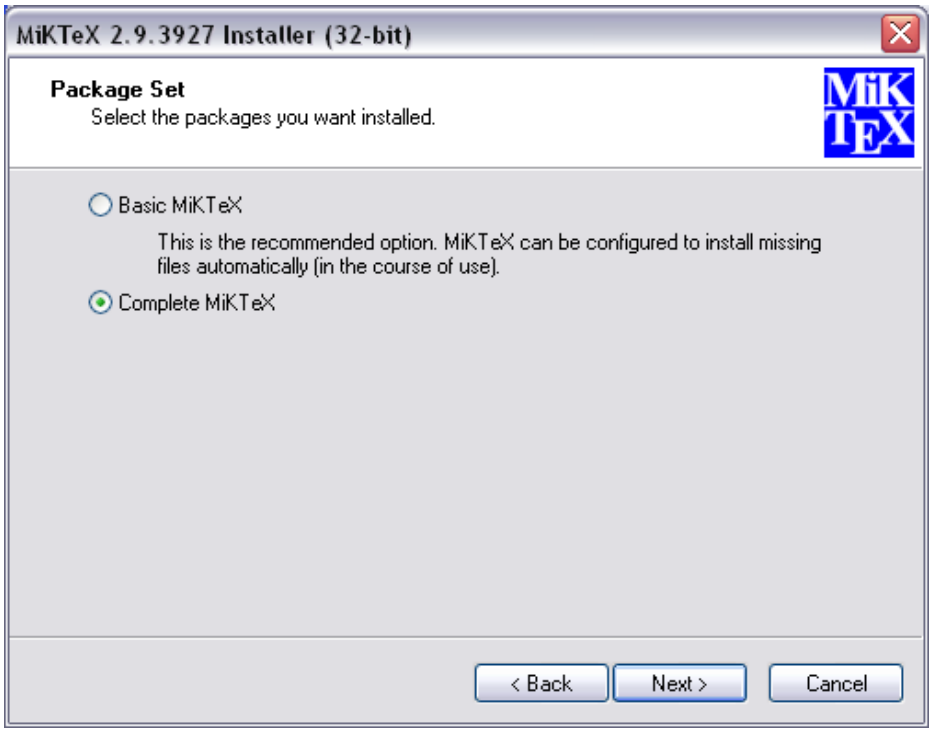
when you include a package in your document that isn't yet installed. This is rare if you pick the 'full' installation option, as you then have pretty much all the packages already installed, but there are always new packages coming out.

However, there are also times when you may want to add a new or unusual package by hand — perhaps a private package from your company or institution that LATEX would not know about, or even one you are writing yourself. To do this, you need a personal package folder (known as a Personal TEX Tree or Personal TEX Directory). You therefore need a place to put the files, and that's what this section is about.

LATEX will automatically check there first for packages, so anything you put in your personal TEX directory will supersede any file of the same name in your main TEX installation, which is why it's important for updates and special or private classes, packages, styles, and fonts.

The folder is called `texmf`. For Unix-based systems, including GNU/Linux and Mac OS X, that's all you have to create for now. For MiKTEX, you also

Figure 1.8: Selecting the complete installation



have to tell the system that you want the folder checking (see § 1.2.3: this is important, as otherwise MikTEX won't use it).

1.2.1 Unix and GNU/Linux

Create `~/texmf`.

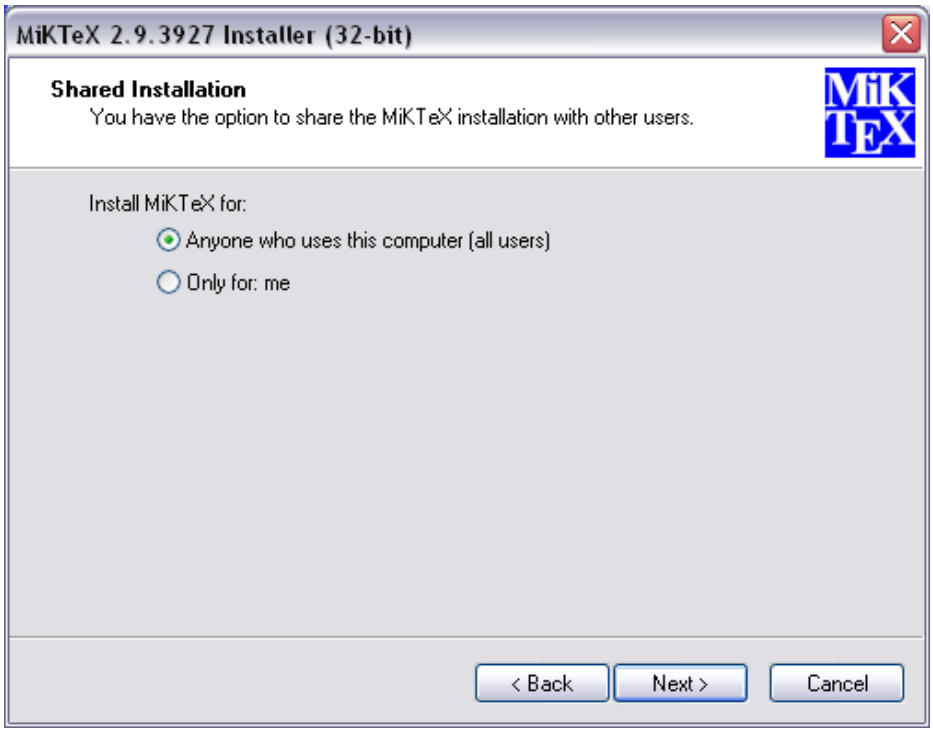
1.2.2 Apple Mac OS X

Create `~/Library/texmf`.

1.2.3 Microsoft Windows

1. Open *My Computer* (just called *Computer* in *Windows 7*), and create a new subfolder called `texmf` in the `C:` drive (XP) or `Computer\System\Users\\argument{yo` (Win7), as in Figure 1.22. Make sure it is called `texmf` (all lowercase) and nothing else.

Figure 1.9: Private or shared use



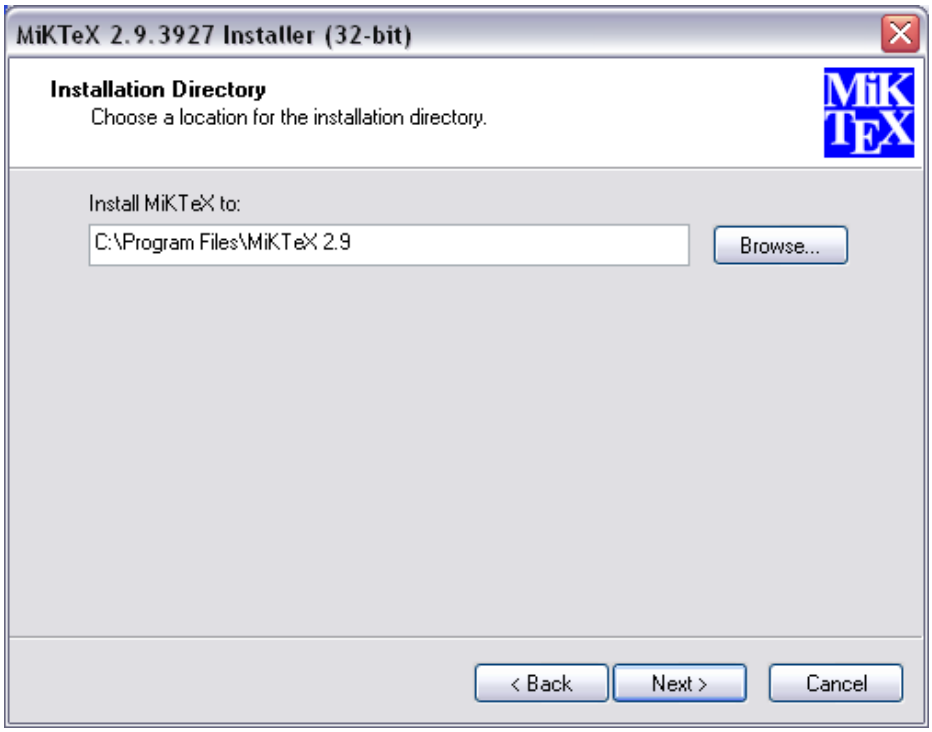
2. Click the Start or Windows button and run the MiKTEX Options (maintenance) program (it should be shown among your recent programs).

3. Click the Roots tab and the Add button, and navigate in the window to the place where you created the `texmf` folder above (Figure 1.23)

4. Lastly, click on the General tab and tell MiKTEX to update it along with its other folders by clicking the Refresh FNDB button (Figure 1.24)

   You *must* click on the Refresh FNDB button any time you make changes to the contents of your personal TEX directory (the `texmf` folder), otherwise MiKTEX will not be able to find the files.

Figure 1.10: The MiKTEX installation folder



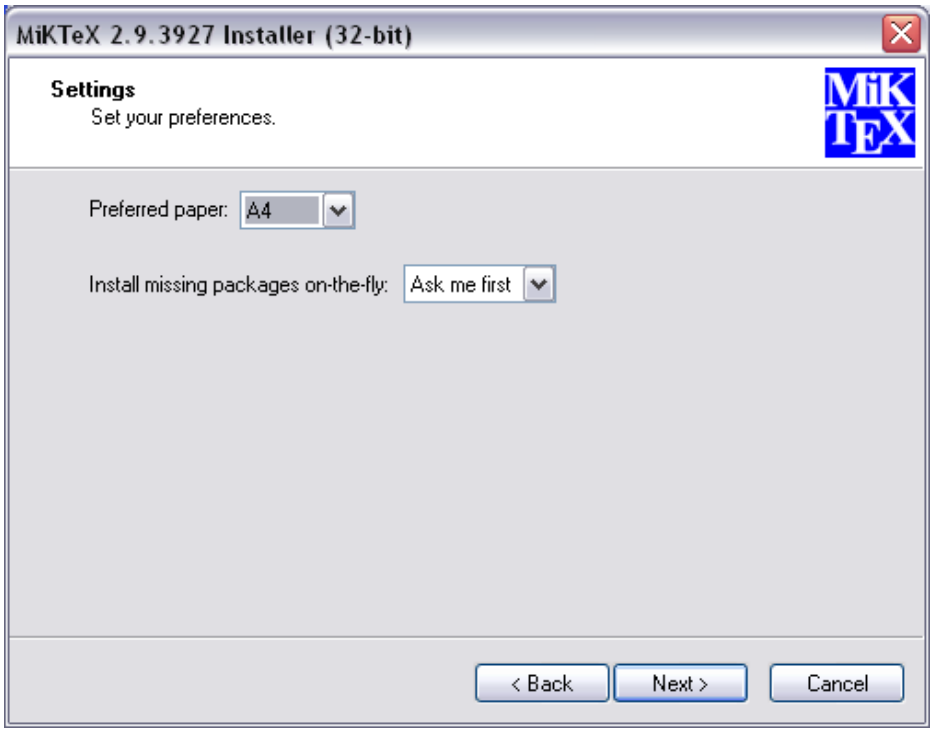
1.3 Picking an Editor

One of the best features of TEX-based systems like LATEX is that they don't force you to use any particular editor or viewer: you can pick one that you're comfortable with.

One of the worst features (for a beginner) is not understanding this: many new users have never seen flexible or adaptable software before, and may be unfamiliar with the idea that you don't have to do what you're told: you can pick and choose.

Nevertheless, I'm solving this by edict for beginners here: *unless you already have a pet editor or viewer*, just use the one shown below. In the case of *ProTEXt* (Windows) and *MacTEX* (Macs) the editor is the one that comes with the distribution (*TEXMakerX* and *TEXshop* respectively). Unix and GNU/Linux users need to choose and install an editor separately.

Figure 1.11: Paper size and package installation options



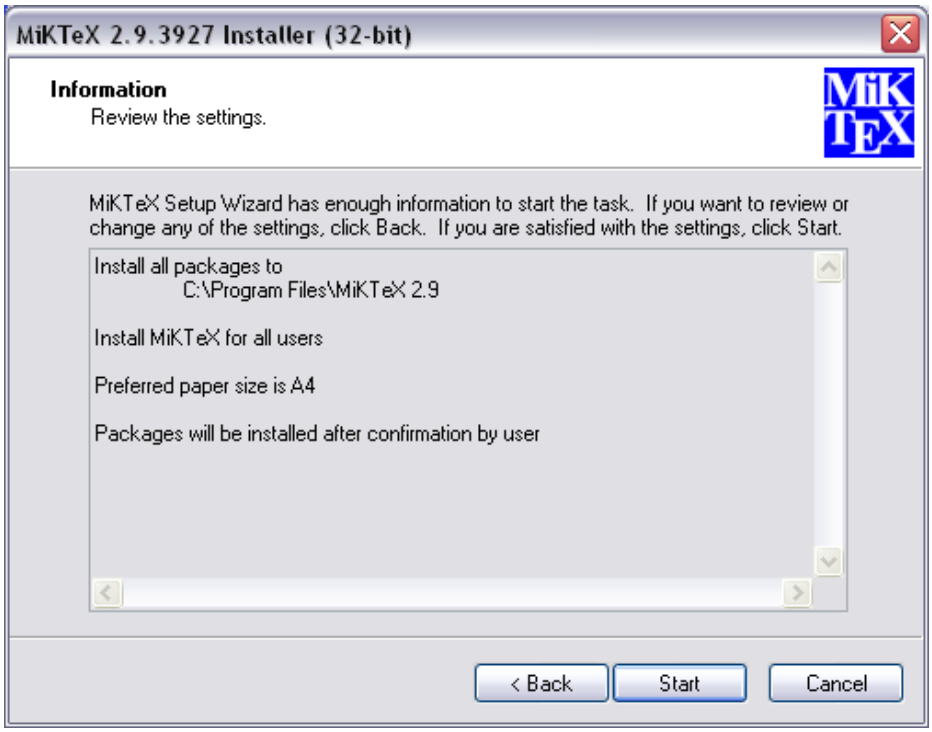
System	Package	Engine	Editor	Viewer
Microsoft *Windows*	*ProTEXt*	*MiKTEX*	*TEXMakerX*	Adobe *Acrobat Reader*
Apple Mac OS X	*MacTEX*	*gwTEX*	*TEXshop*	*Preview*
Unix and GNU/Linux	*TEX Live*	*TEX Live*	*Kile*, *TEXMaker*, or *Emacs*	*Okular* or *kpdf*/*kdvi*

However, there are many other editors, if you want to try them out and pick one you are comfortable with. In particular, *Emacs* is also available for Windows (*NTEmacs*) and for Mac OS X (*Aquamacs*); *TEXMaker* is also available for Macs; and the LyX near-WYSIWYG front-end to LaTeX is available for all platforms.

1.4 Installation problems

It's always annoying when a program that's supposed to install painlessly causes trouble, and none the more so when everyone else seems to have

Figure 1.12: Review your installation settings

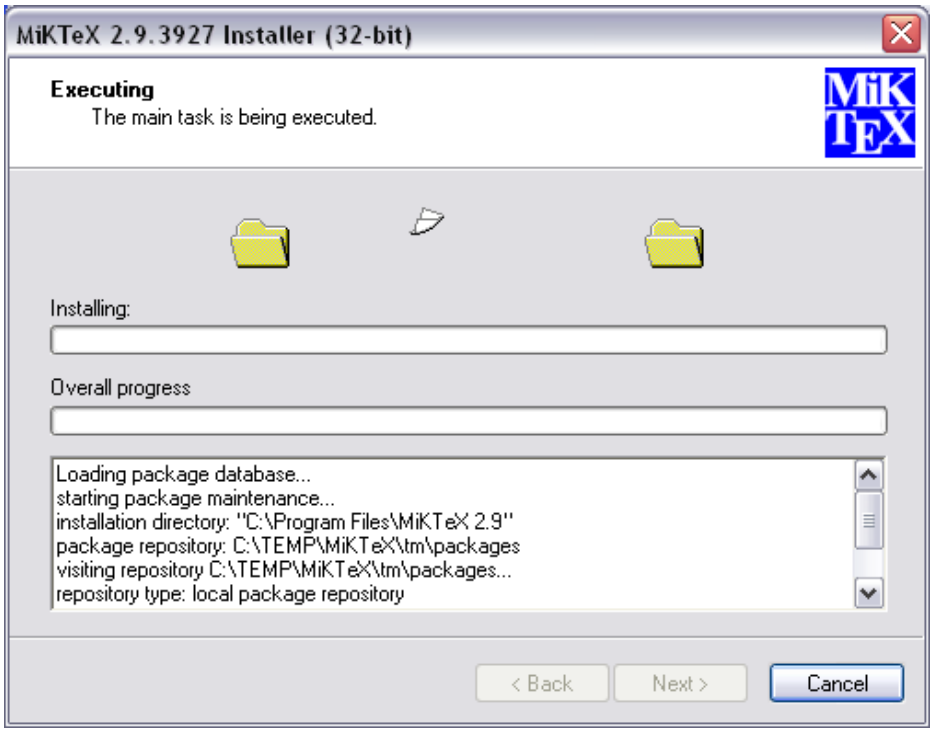


been able to install it without problems. I've installed TEX hundreds of times and very rarely had any difficulties, but these are a few of the occasions when I did.

Bad hard disks If you are using Microsoft Windows, you should run a scan and defragmentation of your hard disk[s] before you start. It should take under an hour on a modern machine unless you have a very large disk, but it may need overnight on an older machine. Clean your DVD drive if it has been in heavy use. TEX is made up of a very large number of very small files, so there is a lot of disk activity during an installation. Microsoft Windows runs very slowly when installing a lot of small files, so be patient.

On any system, if you are installing a new hard disk for your typesetting work, you have the chance to reformat it beforehand. Pick the smallest granularity (cluster size) possible, usually 1024 bytes (1Kb). This minimises the space needed for systems with a very large

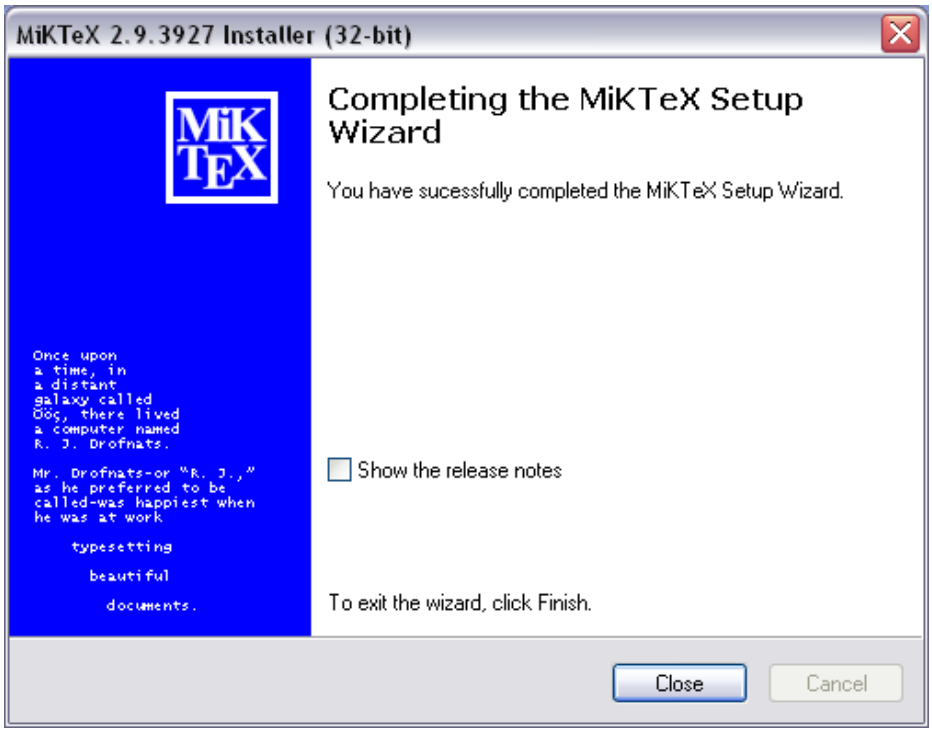
Figure 1.13: MiKTEX installing



number of very small files like TEX has, and may help improve the speed and reliability of the system.

Windows Registry errors This only affects Microsoft Windows users. The Registry is where Microsoft wants software companies to store details of all the programs you install. Unfortunately the Registry is grossly abused by marketing departments to try and foist undesirable links on you, the user. You will see this with many commercial programs, where a particular type of file you've been able to double-click on for years suddenly runs a different program. Some programs install obsolete or broken copies of program libraries (DLL files), overwriting ones which were working perfectly. Worse, the viruses, trojans, and worms which typically infect unprotected Windows systems can leave unwanted links to web pages, or change some of the ways in which Windows operates. The overall effect can be that the whole machine slows down, or that files which are expected to do one thing do

Figure 1.14:



another. The best solution is a thorough Registry clean-out, using one of the many free or commercial programs available for the purpose.

Use the latest versions Before installing, check the CTAN web site (subdirectories of `http://tug.ctan.org/tex-archive/systems/` named after the systems `protext`, `mactex`, etc. for an updated copy of the installation program. This is called `MacTeX-`*yyyy*`-DVD.mpkg` for Macs, and `Setup.exe` for Microsoft Windows. Just occasionally a bug slips through onto the production DVD, and although it's always fixed and notified on `comp.text.tex`, that's a high-volume newsgroup and even the sharpest eyes may miss an announcement.

Unix and GNU/Linux users will always get the latest repository copy from their system's package manager, but this may not be the absolute latest copy of TEX (see § ?? for why). If you are installing on Unix manually from the TEX Collection DVD instead, check on CTAN for an updated version of the file `install-tl.sh`.

Figure 1.15: Starting the TeXmakerX installation

Figure 1.16: Selecting the language to use during TeXmakerX installation

Figure 1.17: The TEXmakerX installation introductory screen

Stick to the defaults Unless you're a computer scientist or a software engineer, I *very strongly* suggest you never change or fiddle with the default directories for installation. I know some of them look odd, but they're that way for a purpose, especially when it comes to avoiding folder names with spaces in them, like the notorious `C:\Program Files`. Although most modern systems cope happily with spaces in filenames and directory names when using a graphical user interface, they are *always* A Bad Idea, especially for programs which can be run from scripts (TEX is one). Spaces and other non-alphanumeric characters should therefore be avoided like the plague (they are forbidden in web addresses [URIs] for the same very good reason: the people who designed them knew the pitfalls). It may look snazzier to put the installation in `My Cute $tuff`, but please *don't*: you'll just make it harder to find, harder to fix problems, and more embarrassing if you have to explain it to someone else trying to help you.

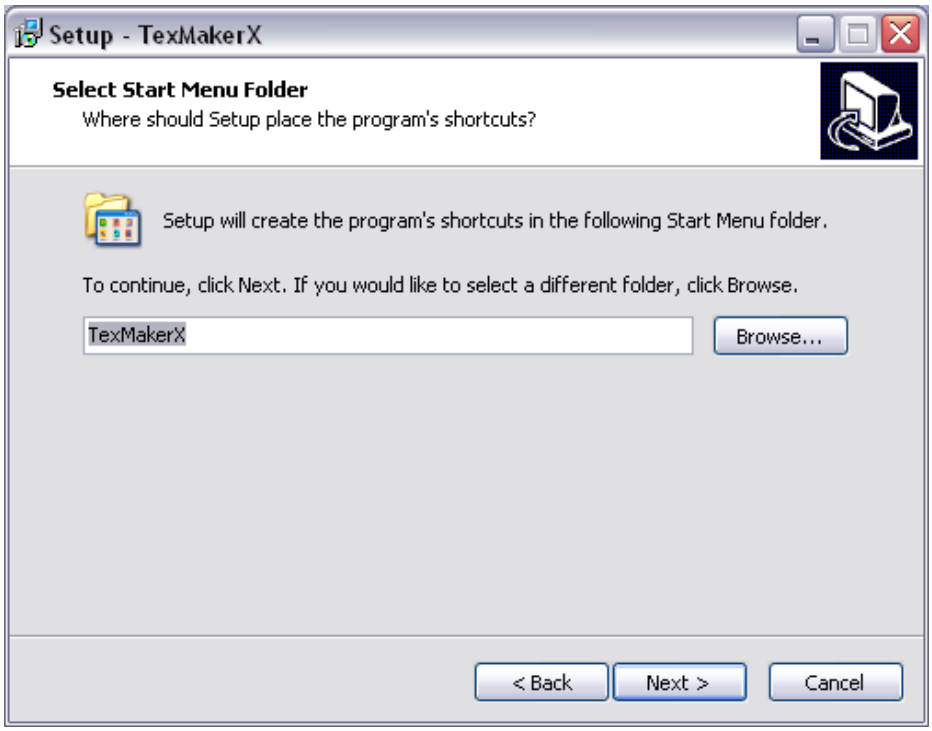
Figure 1.18: The TEXmakerX installation folder

64-bit Windows The *MiKTEX* distribution for Windows is a 32-bit program but it should install correctly on 64-bit Windows 7 systems. For safety, close down all other programs before starting the installation.

Locked systems If you want to install *proTEXt* on a computer in a lab or other group environment where the disk storage is locked down, and where the Administrator is unwilling or unavailable to install it for you, there are a couple of choices:

- Install it on a USB stick that you can unplug and carry with you. That way your LATEX installation is always with you. If you use it on another computer where the USB device mounts as a different disk letter, you will need to configure it so that it can 'see' where it is in the directory system.
- If you cannot install it at all, because the Windows Registry is also locked, and the Administrator is unwilling or unable to install it for you, you may be able to install it in a virtual container (eg

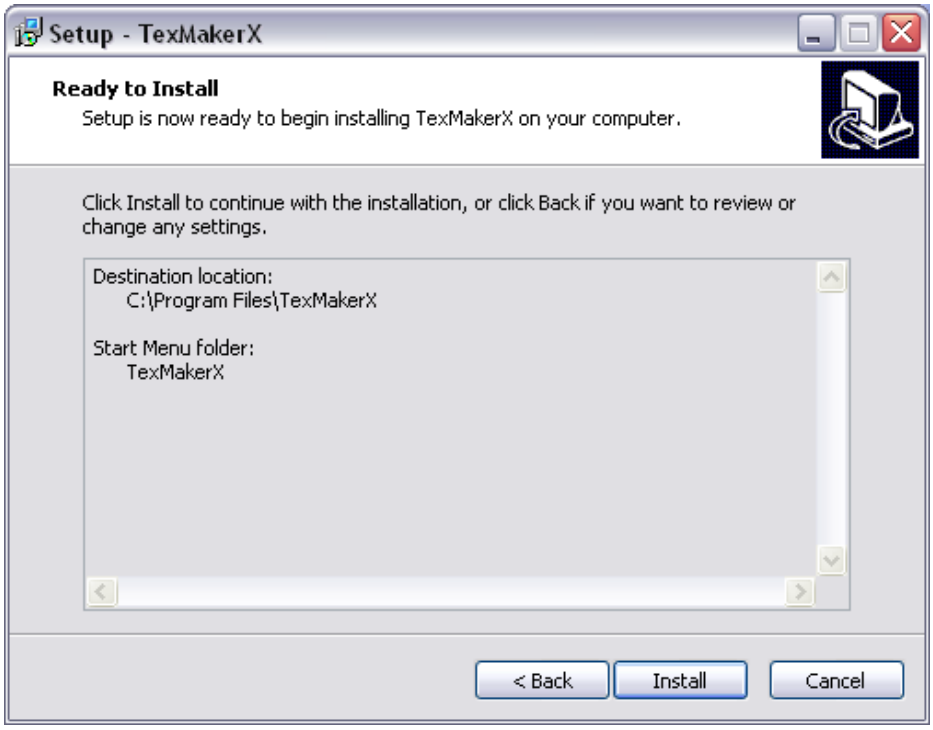
Figure 1.19: The TEXmakerX shortcut



Windows XP as a virtual image inside Windows 7). It will be slow, and it may be missing some facilities like alternate character sets, but it will execute.

Bear in mind that shared systems in large companies, universities, and similar organisations do usually prohibit software being installed by the user (you) because of security issues over viruses, support, maintenance, and other factors. If you feel your institution needs a network installation of LATEX, ask your Administrator or IT Centre to contact the TEX Users Group or any local use group (see Appendix B), who may be able to help.

Figure 1.20:



MiKTEX and TEXnicCenter

If you plan on using the **TEXnicCenter** editor instead of **TEXMakerX**, you must make sure when you install **MiKTEX**, that you make a careful note of the folder you install **MiKTEX** into (usually something like `C:\Program Files\MiKTeX 2.9\...`) because you will need that later, when you run **TEXnicCenter** for the first time after installation. It should be something like `C:\Program Files\MiKTeX 2.9\miktex\bin` (or a later version number, if time has moved on since the time or writing). You need to know this, because **TEXnicCenter** won't guess it for you.

Figure 1.21:

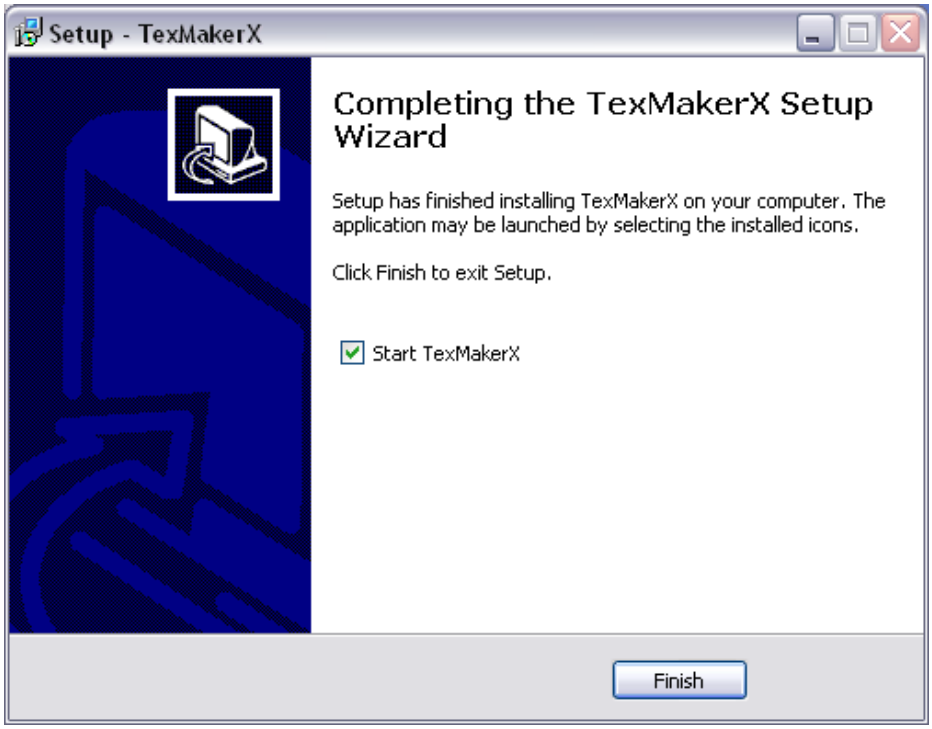

Setup - TexMakerX
Completing the TexMakerX Setup Wizard
Setup has finished installing TexMakerX on your computer. The application may be launched by selecting the installed icons.
Click Finish to exit Setup.
Start TexMakerX
Finish

Figure 1.22: Creating a new `texmf` folder

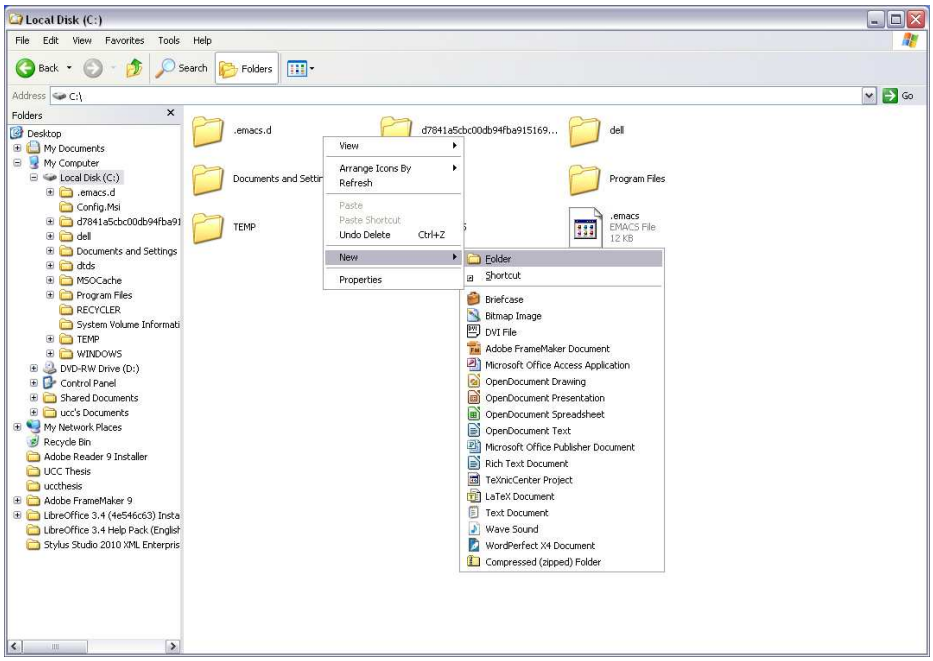


Figure 1.23: Adding your Personal TEX Directory to MiKTEX

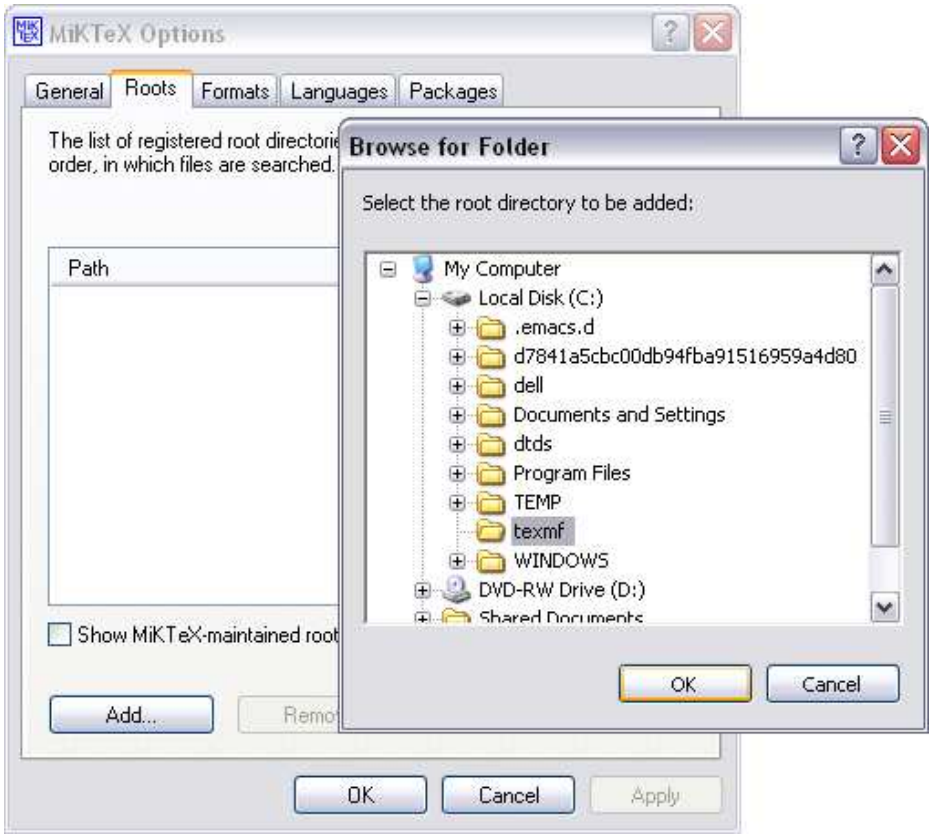
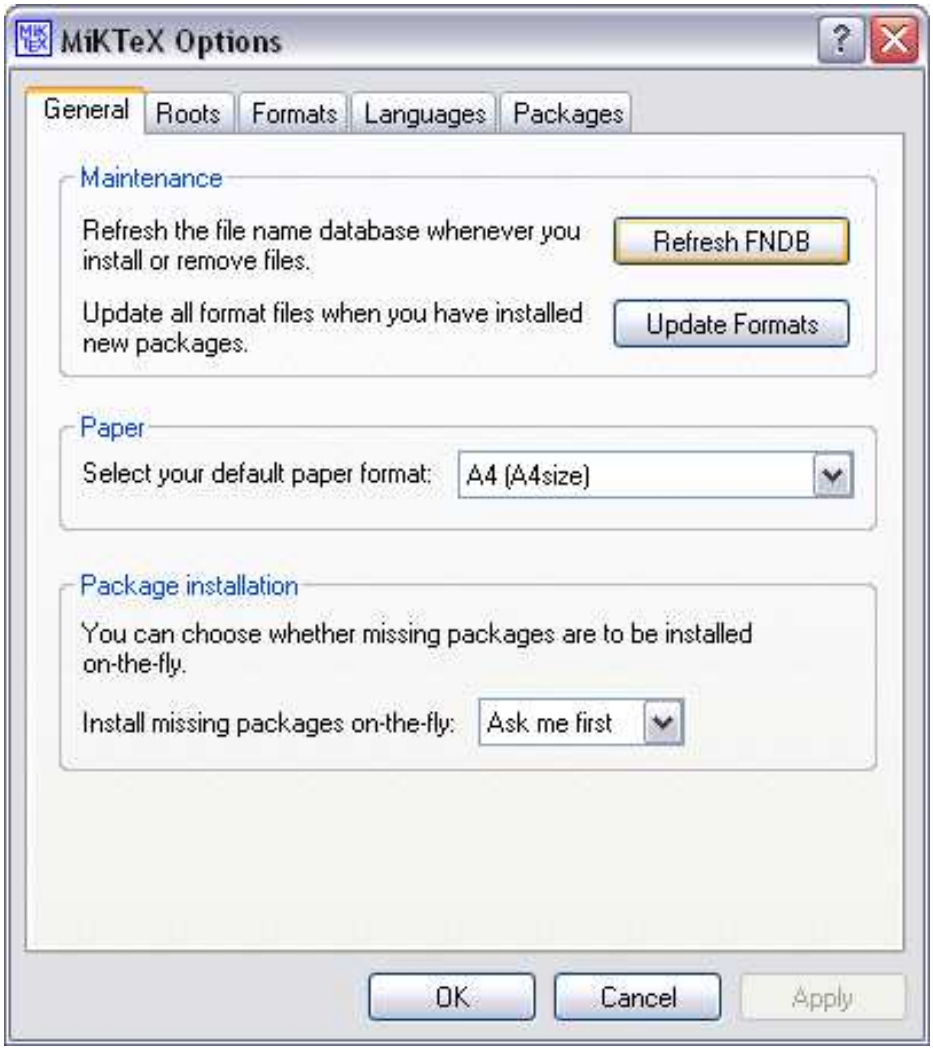


Figure 1.24: Updating MiKTEX's FileName DataBase (FNDB)



2 Using your editor to create documents

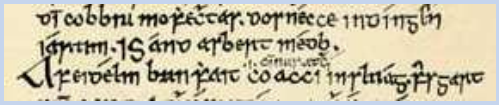
LaTeX documents are all *plain-text* files. This means printable characters only (in whatever writing system is native to your language and culture), no hidden internal binary gubbins like fonts or formatting (except for spaces and linebreaks).

The big advantage of this is that the files can be copied, downloaded, or uploaded to any 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 hiding or transporting virus infections in the way that binary (non-plaintext) wordprocessor files can be. Everything you can see is in the file and everything in the file is there for you to see: there is nothing hidden or secret and there are no manufacturers' proprietary 'gotchas' like suddenly going out of date with a new version, and refusing to open.

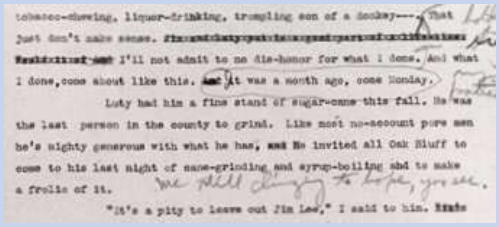
So, you may ask, if LaTeX files are all plaintext, how does LaTeX know how to format them? The answer is that it uses *markup*: a system of labels which tells LaTeX what's what. LaTeX and its packages recognise the labels and know how to format them, so you don't usually need to add formatting by hand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

Markup

This is a term from printing, and originally meant the notes on how to lay the document out, or the instructions which a proofreader might add during correction. It now also means instructions or descriptions added to a computer document to act as guidelines for identification or formatting. Markup has been around for ages.



Anon, 'Táin bó Cúailnge' c. 1100



Marjorie Kinnan Rawlings, 'Varmints' 1936

`.h1 Interest Rates`	Runoff	c.1970
`\section{Interest Rates}`	LaTeX	1984
`<sec><ttl>Interest Rates</ttl>...`	SGML (AAP)	1985
`<H2>Interest Rates</H2>`	HTML	1991
`<sect1><title>Interest Rates</title>...`	DocBook	1995

2.1 Markup

In a LaTeX document, you type your text along with markup which identifies the relevant 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 (packages) for typesetting.

LaTeX markup is all in (American) English, with a few abbreviations for long words to minimise typing, although most people use a menu or toolbar button, so actually typing it is rare except for the beginner.

You do not need to format any of your text *in your editor*, because LaTeX does the formatting all by itself when it typesets. You can of course regularise or neaten its appearance *in your editor* for your own ease of

editing (for example, keeping each item in a list on a separate line), but this is not required.

You will often hear LaTeX markup referred to as 'commands' or sometimes 'control sequences' (the proper TeXnical term for them). For all practical purposes these terms all mean the same thing.

The the panel 'Markup' on p. 28 shows some examples of markup. The similarity between the computer forms is striking, and not coincidental.

2.2 Quick start for the impatient

If you already know all this stuff about editors and plain-text files and running programs, and 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 know this stuff yet, then still do this section now, and treat it as part of the learning experience.

1. **Install LaTeX**
   See Chapter 1.

2. **Create a new, empty document**
   Start up your *editor* (➠ § 1.3) and open a New Document.

   Delete any template material it inserts, so that your new document is completely empty.

3. **Copy the example**
   Copy and paste the text from Figure 2.1 on p.30. Make sure you get all of it, and don't change anything.

4. **Save the document**
   Save the document as `quickstart.tex`

5. **Typeset the document**
   Click on the Run, Build, Typeset, or PDFLaTeX menu or toolbar icon for your system as indicated by the black cursor in Figure 2.2 on p.31.

6. **Preview the typesetting**
   Click on the Preview toolbar item (next to the Typeset icon) if this does not open automatically for you.

7. **Print it**
   Click on the Print toolbar icon in the viewer.

If you encounter any errors, it means you *do* need to read the rest of this chapter after all!

Figure 2.1: Quick-start example document text

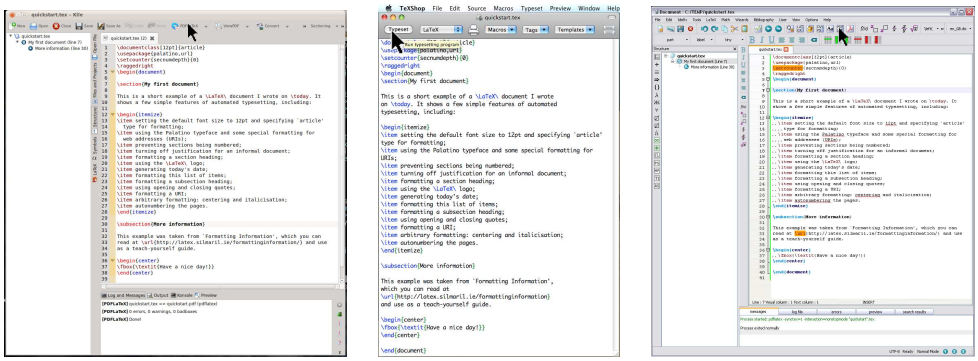
```
\documentclass[12pt]{article}
\usepackage{palatino,url}
\setcounter{secnumdepth}{0}
\raggedright
\begin{document}

\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 and specifying ‘article’
    type for formatting;
  \item using the Palatino typeface and some special formatting for
    web addresses (URIs);
  \item preventing sections being numbered;
  \item turning off justification for an informal document;
  \item formatting a section heading;
  \item using the \LaTeX\ logo;
  \item generating today’s date;
  \item formatting this list of items;
  \item formatting a subsection heading;
  \item using opening and closing quotes;
  \item formatting a URI;
  \item arbitrary formatting: centering and italicisation;
  \item autonumbering the pages.
\end{itemize}

\subsection{More information}

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

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

\end{document}
```

2.3 LaTeX commands

LaTeX commands all begin with a *backslash* (\) and consist of lowercase letters only.

Do not confuse the backslash with the forward slash (/). They are two different characters. The forward slash is used on the Web and on Unix-based systems (including Macs and GNU/Linux) to separate directory names and filenames. The backslash is used for the same purpose in the Microsoft

Figure 2.2: What to click on to typeset a document



a) *Kile* (Unix and GNU/Linux) *b*) *TeXshop* (Mac) *c*) *TeXMakerX* (Windows)

Windows file system, but you must *always* use the forward slash in folder and file names in LaTeX, even if you use Microsoft Windows.

2.3.1 Simple commands

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

```
\tableofcontents
```

This example is an instruction to LaTeX to insert the Table of Contents at that point. You would usually use this in a book or report (or perhaps a very long article) somewhere close to the beginning, normally after the title page but before the Preface or Introduction. You don't have to do anything else. Provided that you have used the sectioning commands described in § 3.5, all the formatting and numbering for the Table of Contents is completely automatic.

```
\sffamily\bfseries\Large
```

These are some more simple command examples: the first switches to the current sans-serif typeface, the second invokes bold type, and the third makes the type the Large size (see § 8.2 for the exact details). In this case, the

font changes affect *all* text after this point: to restrict it, there is a technique called 'grouping' (see the panel 'Grouping' on p. 138).

Simple one-word commands like `\tableofcontents` must be separated from any following *text* with *white-space*. This means a normal space, or a newline (linebreak) or a TAB character. For example either of these two forms will work:

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

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

If you forget the white-space, as in the following example, LaTeX will treat everything up to the next space as a command, and end up with what it thinks is a command called `\tableofcontentsThanks`. There's no such command, of course, so LaTeX will complain by displaying an error message (see § 4.2.3.2).

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

LaTeX swallows any white-space which follows a command ending in a letter. It does this automatically, so you don'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 in curly braces (see below) *must* be followed by white-space before normal text starts again, to keep it separate from the text.

2.3.2 Commands with arguments

Many LaTeX commands are followed by one or more *arguments*, meaning information to be acted upon. Here are two examples, a chapter title (see § 3.5) and a cross-reference label (see § 7.4.1):

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

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 and they mean different things.

With commands that take arguments you do *not* need to use extra white-space after the command, because the argument in curly braces will keep it separate from any normal text which comes after it. The following example is therefore exactly equivalent to the earlier one above.

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

2.3.3 White-space in LaTeX

In LaTeX documents, all *multiple* spaces and TAB characters are treated as if they were a *single* space during typesetting. All multiple newlines (linebreaks) are treated as if they were a single newline. LaTeX does its own spacing and alignment using the commands you give it and the layout in the stylesheet, 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 examples in the preceding section (although unusual!):

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

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

That is, it will get typeset exactly the same. Try it.

Why would you want odd spacing (or none)? The answer is usually never, but a lot of LaTeX is created by computer programs from other systems such as web scripts, XML documents, or databases, and it makes life easier if you don't have to worry about the odd space or two creeping in here and there in normal text: it simply won't have any effect. It also means that you don't have to worry about extra linebreaks between sections or paragraphs, or tabbing in lists of items if you want to use them to make your text easier to edit.

Table 2.1: Special characters in LaTeX

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

2.4 Special characters

There are ten keyboard characters which have special meanings to LaTeX, and *cannot* be used on their own except for the purposes shown in Table 2.1.

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

2.4.1 Using the special characters

We have already seen (the first paragraph of § 2.3) how to use the backslash to start a command, and curly braces to delimit an argument. The remaining special characters are:

$ This has a special mathematical meaning in TeX, so if you want to print $35.99 you must type `\$35.99`

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

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

If you want to print 45% you must type `45\%`.

As with all comments in documents, whether LaTeX or just a word-processor, don't forget to remove them before sending the original document source to someone else!

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

^ The caret sign in mathematics lets you type `\(E=mc^2\)` to get $E = mc^2$. If you need the circumflex accent on a letter like â, just type it normally or use the symbolic notation `\^a` (see § 2.6).

& The ampersand is used in LaTeX tables to separate the columns (see § 6.3). If you want to print AT&T you must type `AT\&T`.

_ The underscore in mathematics lets you type `\(r_2\)` for r_2. If you want to underline text (extremely rare in typesetting) see the last paragraph of § 8.2.3. If you need an underscore character as part of a program variable name like `SUB_TOTAL`, use the `\verb` command (see § 6.6).

~ The tilde in LaTeX prints as a normal space, but prevents a linebreak ever occurring at that point. It's often used between a person's initials and their surname, e.g. `Prof D.E.~Knuth` where a linebreak would make it harder to read.

If you want a *hash mark* (the *octothorpe* or American 'pound' [weight] or 'number' sign) you must 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`.

While we're on the subject of money, the official sans-serif Euro sign € needs the marvosym package and is done with the `\EUR` command (see § 5.1 for details of how to use LaTeX packages). Not every font has a Euro character, though, and the default € is based on a letter C. However, a slightly unusual but interesting serif Euro sign € is in the textcomp package using the `\texteuro` command when using the Computer Modern typeface (see § 8.2.2 for details of switching typefaces in mid-text).

2.5 Quotation marks

If you are using UTF-8 as your input encoding and file format, and the T1 font encodings, you can use your operating system's normal open-quote and close-quote characters.

```
He said, ‘‘I’m just going out.’’
```

He said, “I’m just going out.”

Do *not* use the unidirectional typewriter keyboard ' (apostrophe) or " (quotes) key for opening quotes. Correct typographic quotes in LaTeX are got with the ` key (grave-accent or ‘backtick’) for the opening quote, and the ' (apostrophe) key for the closing quote, doubled if you want double quotes:

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

If you are using *Emacs* as your editor, the " key is specially programmed in LaTeX-mode to think for itself and produce correct `` and '' characters automatically. This is one occasion when you *can* use the " key for an open-quote, because it will be interpreted correctly.

Browser fonts

If you are reading this in a browser, or if you have typeset the document yourself using different fonts, it may not show you real quotes (some old 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, and switch to a browser with better font-handling.

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):

2.6 Accents

For accented letters in western European and other Latin-alphabet languages just use the accented keys on your keyboard — if you have them. If you don’t have any, or can’t find the right combination of keystrokes to generate them, see the panel ‘If you don’t have accented letters on your keyboard’ on p. 38 and Table 2.2.

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

He said, ‘Her answer was “never” ’, and she meant it.

You must also tell LaTeX what character repertoire (‘input encoding’) you are using, and which set of fonts (‘font encoding’) to find the characters in. You specify these by using the inputenc and fontenc packages in your preamble with the relevant options. For example, to tell LaTeX you will be using ISO Latin–1 (most western European) characters and the original TeX encoding, use:

```
\usepackage[latin1]{inputenc}
\usepackage[OT1]{fontenc}
```

However, with most modern systems, Unicode compatibility will let you use almost any letter or symbol from any writing system encoded in UTF8 (the multibyte 8–bit encoding), for which LaTeX has extensible support. I therefore recommend the following preamble for all documents:

```
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
```

For language-specific hyphenation and cultural adaptation (including the correct language headings for all the parts of your document) you will need the babel package (see § 2.7.6), and for non-Latin typefaces you will also need the relevant font packages and typefaces (see § 8.2).

If you don’t have accented letter keys on your keyboard, or you can’t find the codes to type, or if you need additional accents or symbols which are not in any of the keyboard tables, you can use the symbolic notation in Table 2.2. In fact this can be used to put any accent over any letter: if you particularly want a g̃, for example, you can have one with the command `\~g` (and Welsh users can get a ŵ with `\^w`).

If you use this symbolic method *only*, you do not need to use the inputenc package described above.

If you don't have accented letters on your keyboard

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 GNU/Linux systems the letter é is usually got with AltGr–' e depending on the keyboard setup that was installed. Refer to the xkeycaps utility for a table of key codes and combinations (install it with your system's package manager or get it from `http://www.jwz.org/xkeycaps/`).
- Under Microsoft Windows the letter é is got with Ctrl–' e or by holding down the Alt key and typing 0 1 3 0 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).

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: it does have the advantage in portability that the LaTeX file remains plain ASCII, which will work on all machines everywhere, regardless of their internal encoding, and even with very old TeX installations.[1]

Irish and Turkish dotless-ı can be done with the special command `\i`, so an í (which is normally typed with í) may requires `\'\i` if you need to type it in the 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 LaTeX control sequences which end in a letter (see §2.3.1) always absorb any following space. So what you might see as *Rí Teamrac* would have 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-ȷ and to uppercase Í.

Note that recent versions of LaTeX can compensate for this when used with the utf8x option of the inputenc package, and the T1 option of the fontenc package (as shown above). In this case you can just type `f\'is` to get 'fís'.

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

Table 2.2: Built-in LaTeX accents

Accent	Example	Characters to type
Acute (fada)	é	\'e
Grave	è	\`e
Circumflex	ê	\^e
Umlaut or diresis	ë	\"e
Tilde	ñ	\~n
Macron	o	\=o
Bar-under	o	\b o
Dot-over (séımú)	ṁ	\.m
Dot-under	ṣ	\d s
Breve	ŭ	\u u
Há£ek (caron)	ǔ	\v u
Long umlaut	®	\H o
Tie-after	o͡o	\t oo
Cedilla	ç, Ç	\c c, \c C
O-E ligature	œ, ×	\oe, \OE
A-E ligature	,	\ae, \AE
A-ring	å, Å	\aa, \AA
O-slash	ø,	\o, \O
Soft-l	ł, Ł	\l, \L
Ess-zet (scharfes-S)	ÿ	\ss

2.7 Dimensions, hyphenation, justification, and breaking

LaTeX's internal measurement system is extremely 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 — 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 on printers, even at 600dpi, fine oblique lines or curves can still sometimes be seen to stagger the dots.

At the same time, many dimensions in LaTeX'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.

Table 2.3: Units in LaTeX

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 (65,536 to the pt)
dd	Didot (European standard) points (67.54 to the inch)
cc	Ciceros (European pica ems, 12dd)
	Printers' relative measures
em	ems of the current point size (historically the width of a letter 'M' but see 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

TeX uses a very sophisticated justification algorithm to achieve a smooth, even texture to normal paragraph text by justifying a whole paragraph at a time, quite unlike the line-by-line approach used in wordprocessors and DTP systems.

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

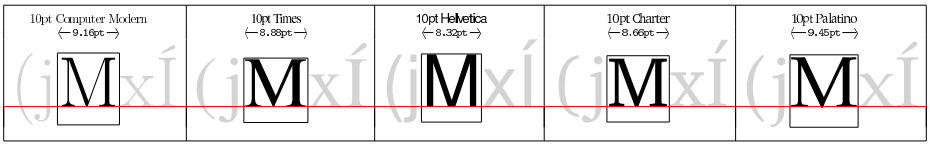
2.7.1 Specifying size units

Most people in printing and publishing in English-speaking cultures habitually use points, picas and ems; and their designers may also use cm and mm, especially when dealing with clients. Many older English-language speakers still use inches. In other cultures, [Didot] points and [Cicero] picas are also used professionally, but cm and mm are standard: inches are used only when communicating with North American cultures.

You can specify lengths in LaTeX in any of these units, plus some others (see Table 2.3).

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

Figure 2.3: An M of type of different faces boxed at 1em



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

the depth of the metal body on which foundry type was cast in the days of metal typesetting, *not* the printed height of the letters themselves. Thus the letter-size of 10pt type in one typeface can be radically different from 10pt type in another (look at Figure 2.3, where the widths are given for 10pt type). An em is the height of the type-body in a specific size, so 1em of 10pt type is 10pt and 1em of 24pt type is 24pt. A special name is given to the 12pt em, a 'pica' em, and a pica has become a fixed measure in its own right. An old name for a 1em space is a 'quad', and LaTeX has a command `\quad` for leaving exactly that much horizontal space.

To highlight the differences between typefaces at the same size, Figure 2.3 shows five capital Ms in different faces, surrounded by a box exactly 1em of those sizes wide, and showing the actual width of each M when set in 10pt type. Because of the different ways in which typefaces are designed, none of them is exactly 10pt wide.

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

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 in the document, use the **\hyphenation** command in your preamble (see the panel 'The Preamble' on p. 56) with one or more words as patterns in its argument, separated by spaces. This will even let you break 'helicopter' correctly. In this command you use normal hyphens in the pattern, 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 without advice, as it can change the appearance of your document quite significantly.

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

2.7.3 Unbreakable text

This is the opposite of discretionary hyphenation. To force LaTeX to treat a word as unbreakable, use the **\mbox** command:
`\mbox{pneumonoultramicroscopicsilicovolcanoconiosis}`.
This may have undesirable results, however, if you subsequently change margins or the width of the text: pneumonoultramicroscopicsilicovolcanoconiosis. . .

To tie two words together with an unbreakable space (hard space), use a tilde (~) instead of the space (see the list on p. 35 in § 2.4.1). 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`.

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 sthetic. *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). *Never* use a text-mode hyphen for either of these purposes.

Spacing after full points (periods)

Note that a full point (period) after a lowercase letter is treated in LaTeX as the end of a sentence, and it automatically puts a little more space before the next word. You do not (and should not) type extra space yourself.

However, after abbreviations in mid-sentence like 'Prof.', it's not the end of a sentence, so we can use backslash-space to force LaTeX to treat it as 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 at a line-end.

2.7.5 Justification

The default mode for typesetting in LaTeX 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. The soul package can be used if you need letter-spacing, but this is best left to the expert.

There are two commands **\raggedright** and **\raggedleft** which typeset with only one margin aligned. Ragged-right has the text ranged (aligned) on the left, and ragged-left has it aligned on the right. They can be used inside a *group* (➠ p. 138) to confine their action to a part of your text, or put in the Preamble if you want the whole document done that way.

These modes also exist as *environments* (➠ § 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}
```

Ragged setting turns off hyphenation. There is a package ragged2e which retains hyphenation in ragged setting, useful when you have a lot of long words.

To centre text, use the **\centering** command in a *group* (➠ p. 138), or use the `center` environment.

Be careful when centering headings or other display-size material, and add manual linebreaks where needed (\\) to make the breaks at sensible pauses in the meaning.

2.7.6 Languages

LaTeX can typeset in the native manner for several dozen languages. This affects hyphenation, word-spacing, indentation, and the names of the parts of documents displayed in headings (but not the commands used to produce them).

Most distributions of LaTeX 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[english,german,frenchb]{babel}
...
\selectlanguage{german}
```

Make sure that the base language of the document comes *last* in the list. 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'. The babel package also sets the hyphenation patterns *provided your version of LaTeX has them precompiled* (see the start of your log files for a list). For other languages you need to set the hyphenation separately (outside the scope of this book).

2.8 Mathematics

As explained in the *Preface* on p. xii, TeX was originally written to automate the typesetting of books containing mathematics. The careful reader will already have noticed that mathematics is typeset differently from normal text, which is why it has to be treated 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
|	Vertical bar
<	Less-than
>	Greater-than

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

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 inline (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 $-30°$ you need to type `\(-30\)°`.[2]

To typeset a math expression as 'displayed math' (centered between paragraphs), enclose it in the commands **\[** and **\]**.[3]

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

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

[2] 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.

[3] 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

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

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.

3 Basic document structures

If the quick-start exercise in §2.2 was enough to show you how a LaTeX document works, then this is where you get the rest of the basic information. If you skipped Chapter 2 then be prepared to go back to some of the sections in it, because I'll be referring to things you might not have come across yet.

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 consistent finish with minimum effort.

What's what

All you need to do in LaTeX is to say what's what by labelling things as what they are. If you want a list, you say so. It's not a bunch of paragraphs prefixed with bullets, it's a list.

One of the small mind-shifts needed to work with a markup system like LaTeX is that you need to be explicit. Unlike a wordprocessor, where the WYSIWYG display is the only way of communicating your intentions, with LaTeX you can actually tell it what to do.

Consistency helps make documents easier to read and understand, as well as making them more visually attractive. Consistency is also what publishers look for. They have a house style, and often a reputation to keep, so they rightly insist that if you do something a certain way once, you should do it the same way each time.

'Elements' are the component parts of a document: all the pieces which make up the whole. Almost everyone who reads books, newspapers, magazines, reports, articles, and other classes of documents will be familiar with the common structure of parts, chapters, sections, subsections, subsubsections, titles, subtitles, paragraphs, lists, tables, figures, and so on, even if they don't consciously think about it.

3.1 The Document Class Declaration

How to tell LaTeX what type of document this is In order to set things up correctly, LaTeX needs to know up front what type of document you are going to be writing. There are probably lots of different types of document you deal with: in LaTeX they are called 'classes' of documents — 'class' is just the computing science word for 'type'.

To tell LaTeX what class of document you are going to create, the first line of your file *must* identify it.[1] To start a report, for example, you would type the `\documentclass` command like this as the first line of your document:

```
\documentclass{report}
```

There are four built-in classes provided, and many others that you can download (some may already be installed for you):

report for business, technical, legal, academic, or scientific reports; theses[2], dissertations

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

book for books, booklets, and whole journals;

letter for letters.[3]

1 Readers familiar with SGML, HTML, and XML will recognize the concept as similar to the Document Type Declaration (it's still called a 'type' there, not a 'class').

2 Theses and dissertations require an Abstract, which is provided in the report class but not in the book class.

3 The built-in letter class is rather idiosyncratic: there are much better ones you can use which you will find in the memoir package and the komascript bundle.

These default classes are quite broad, so they are easily customised by adding *packages*, which are the style and layout plug-ins that LaTeX uses to let you automate formatting.

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 § 3.3) and adding the relevant packages.

The built-in classes are intended as starting-points, especially for drafts, and for compatibility when exchanging documents with other LaTeX users, as they come built into every installation of LaTeX and are therefore guaranteed to format identically everywhere. They are *not* intended as final-format publication-quality layouts and should never be used as such. For most other purposes, especially for publication, you use LaTeX packages to extend these classes to do what you need:

- The memoir package and the komascript bundle contain more sophisticated replacements for all the built-in classes, as well as additional ones;
- Many academic and scientific publishers provide their own special class files for articles and books (on their Web sites for download);
- Conference organisers may also provide class files for authors to write papers for presentations;
- 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 classes on a central server and configure LaTeX installations to look there first for packages, fonts, etc..

Books and journals are not usually printed on office-size paper. Although for draft purposes LaTeX's layouts fit on the standard A4 or Letter stationery in your printer, it makes them look odd: the margins are too wide, or the positioning is unusual, or the font size is too small, because the finished job will normally be trimmed to a completely different size entirely — try trimming the margins of the PDF version of this book to make it 185mm by 235mm (the same as *The LaTeX Companion* series) and you'll be amazed at how it changes the appearance!

These document classes are therefore adequate for drafts or for sending to a colleague to edit, but they are not designed for final-format publishing. For this you need a style file (package or class file) designed by the publisher to fit their series of publications (quite often based on the default classes, but looking very different). Many publishers provide these on their web sites for authors to download. Some are also available in the CTAN repository,

along with packages and classes for other predefined formats such as university theses.

3.1.1 Document class options

The default layouts are designed to fit as drafts on US Letter size paper.[4] To create documents with the correct proportions for 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 two most common options are a4paper and letterpaper. However, many European distributions of TEX now come preset for A4 instead of Letter.[5]

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 get scaled to match);

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. This option also sets graphics to print as an outline rectangle containing the name of the image, so that it prints quickly.

[4] 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 variant cutting close to the old Foolscap ($8\frac{1}{4}'' \times 13\frac{1}{4}''$). ISO standard 'A', 'B', and 'C' paper sizes, used everywhere else, are still virtually unknown in most parts of North America.

[5] Note that the standard built-in document classes (book, article, report, or letter) only use the paper size to adjust the margins: they do not embed the paper size name in any PostScript or PDF output file. If you are using *pdfLATEX*, or intend creating PostScript output, and you want to change the default paper size, you must specify it *both* in the Document Class option *and* as an option to the geometry package (see Exercise 2), in order to ensure that the paper size name gets embedded correctly in the output, otherwise printers may select the wrong paper tray, or reject the job.

If you were using LaTeX for a report to be in 12pt type on Letter paper, but printed one-sided in draft mode, you would use:

```
\documentclass[12pt,letterpaper,oneside,draft]{report}
```

The 10pt, 11pt, and 12pt settings cover between them probably 99% of all common document typesetting. There are extra options for other body type sizes in the extsizes bundle of document classes (extarticle, extbook, extreport, etc). In addition there are the 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.

Exercise 3.1: Create a new document

1. Use your editor to create a new document.
2. Type in a 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 (you would put your text where the dots are):

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

\begin{document}
...
\end{document}
```

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, like making an index.

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 (they don't even need to be preceded by the % comment character).

```
...
\end{document}
Don't forget to get the extra chapter from Jim!
```

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

Exercise 3.2: Adding the 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 the document title, the author's name, and the date (except in letters, which have a special set of commands for addressing). The title, author, and date are all examples of *metadata* or *metainformation* (information *about* information).

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

\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2009}
\maketitle

\end{document}
```

The `\title`, `\author`, and `\date` commands are (I hope) self-explanatory. You put the title, author name, and date in curly braces after the relevant command. The title and author are compulsory; if you omit the `\date` command, LaTeX uses today's date by default.

You *must* always finish the metadata with the `\maketitle` command, which tells LaTeX that it's complete and it can typeset the titling information at this point. If you omit `\maketitle`, the titling will never be typeset. This command is reprogrammable so you can alter the appearance of titles (like I did for the printed version of this document in Figure 3.1). It also means publishers can create new commands like `\datesubmitted` in their own document classes, in the knowledge that anything like that done before the `\maketitle` command will be honoured.

When this file is typeset, you get something like Figure 3.1 (I've cheated and done it in *colour* (§ 5.1.1) for fun — yours will be in black and white for the moment):

Exercise 3.3: Adding the metadata

1. Add the `\title`, `\author`, `\date`, and `\maketitle` commands to your file.
2. Use your own name, make up a title, and give a date.

The order of the first three commands is not important, but the `\maketitle` command must come last.

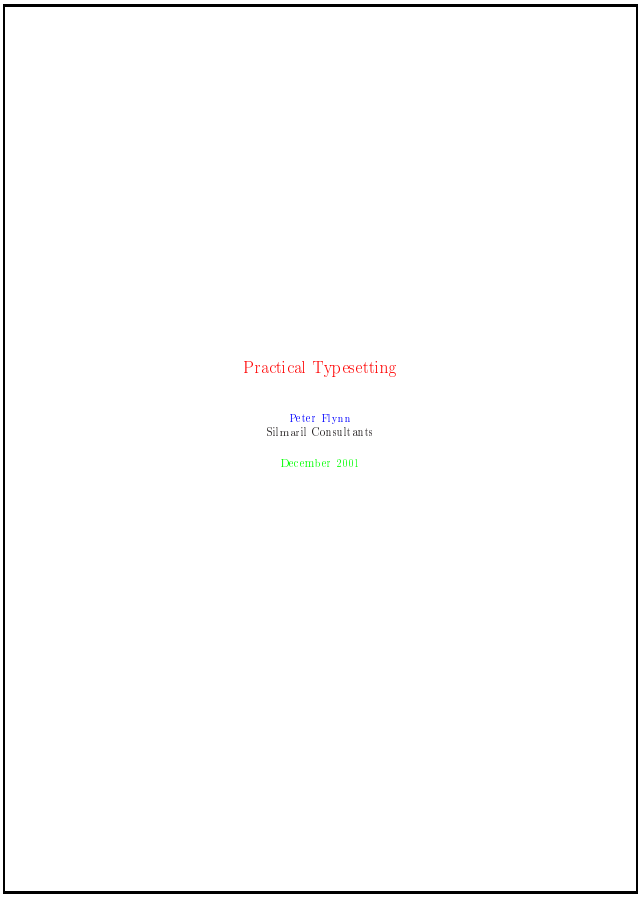
The double backslash (\\) is the LaTeX command for a premature (forced) linebreak. LaTeX normally decides by itself where to break lines, and it's usually right, but sometimes you need to cut a line short, like here, and start a new one. I could have left it out and just used a comma, so my name and my company would all appear on the one line, but I just decided that I wanted my company name on a separate line. In some publishers' document classes, they provide a special `\affiliation` command to put your company or institution name in instead. The double backslash is also used in the `\author` command for separating multiple authors.

The document isn't really ready for printing like this, but if you're really impatient, look at 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 to provide an Summary or Abstract, 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 or theses can run to several

Figure 3.1: Titling information



pages, depending on the length and complexity of the document or the readership it's aimed at.

In both cases the Abstract or Summary is optional (that is, LaTeX doesn't force you to have one), but it's rare to omit it because readers want and expect it. 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.

```
\documentclass[11pt,a4paper,oneside]{report}
\usepackage[utf8x]{inputenc}
\usepackage[T1]{fontenc}
\renewcommand{\abstractname}{Summary}
\begin{document}

\title{Practical Typesetting}
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2009}
\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
æsthetic) does not suffer.
\end{abstract}

\end{document}
```

After the **\maketitle** you use the abstract environment, in which you simply type your Abstract or Summary, leaving a blank line between paragraphs if there's more than one (see § 3.6 for this convention).

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 in your Preamble to give the command **\abstractname** a new value:

```
\renewcommand{\abstractname}{Key Points}
```

This does not change the name of the environment, only its printed title: you still use \begin{abstract} and \end{abstract}.

Notice how the name of the command you are renewing (**\abstractname**, in this case) 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 (this is an example of a command with *two* arguments).

If you look carefully at the example document, you'll see I sneakily added a few extra commands to the Preamble. We'll see later what these mean (Brownie points for working it out, though, if you read § 2.6).

The Preamble

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

```
\documentclass[a4paper]{report}
\renewcommand{\abstractname}{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 modifications to the style and behaviour of this document. Major or permanent modifications that you use all the time should go in a class file or package of your own making.

Exercise 3.4: Using an Abstract or Summary

1. Add the **\renewcommand** as shown 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. 56).
2. Add an `abstract` environment after the **\maketitle** and type in a paragraph or two of text.
3. Save the file (no, I'm not paranoid, just careful).

3.5 Sections

In the body of your document, 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. But even novels are normally divided into chapters, although short stories are often made up solely of paragraphs.

Chapters are only available in the book and report document classes, because they don't have any meaning in articles and letters. Parts are also undefined in letters.[6]

Depth	Division	Command	Notes
−1	Part	`\part`	Not in letters
0	Chapter	`\chapter`	Books and reports
1	Section	`\section`	Not in letters
2	Subsection	`\subsection`	Not in letters
3	Subsubsection	`\subsubsection`	Not in letters
4	Titled paragraph	`\paragraph`	Not in letters
5	Titled subparagraph	`\subparagraph`	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 2010--2020}
```

There are packages 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 sections and sectsty.

Headings also get put automatically into the Table of Contents, if you specify one (it's optional). 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 (see § 8.1.2). This optional alternative heading goes in [square brackets] before the curly braces:

[6] It is arguable that chapters also have no place in reports, either, as these are conventionally divided into sections as the top-level division. LaTeX, however, assumes your reports have chapters, but this is only the default, and can be changed very simply (see § 9.6).

```
\section[Effect on staff turnover]{An analysis of the effects
of the revised corporate recruitment policies on staff
turnover at divisional headquarters}
```

Exercise 3.5: 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 **\chapter** commands for subsequent chapters. Leave a few blank lines between them to make it easier to add paragraphs of text later.
3. By now 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 automatically. Parts get Roman numerals (Part I, Part II, etc.); chapters and sections get decimal numbering like this document, and Appendixes (which are just a special case of chapters, and share the same structure) are lettered (A, B, C, etc.). You can easily change this default if you want some special scheme.

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 the **\setcounter** command, giving the depth value from the table on p. 57:

```
\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.

```
\setcounter{tocdepth}{3}
```

To get an one-time (special case) *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 in isolated circumstances 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***) necessarily mean you get a blank line in the typeset output. Now read this paragraph again and again until that sinks in.

The spacing between paragraphs is an independently 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*.

White-space in LaTeX can also be made flexible (what Leslie Lamport 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 may 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.

```
\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).[7] 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 9.99756pt 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 3.6: Start typing!

1. Type some paragraphs of text. Leave a blank line between each. Don't bother about line-wrapping or formatting — LaTeX 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).

[7] 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.

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 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 LaTeX one extra time to ensure that all ToC page-number references are correctly resolved.

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.

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.

Exercise 3.7: Inserting the table of contents

1. Go back and add a `\tableofcontents` command after the `\end{abstract}` command in your document.
2. You guessed.

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' 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`.

There is also a command `\addtocontents` which lets you add any LaTeX commands to the ToC file. For example, to add a horizontal rule and a

6pt gap, you could say `\addtocontents{toc}{\par\hrule\vspace{6pt}}` at the place where you want it to occur. You should probably only use this command once you know what you are doing.

4 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.

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.

Exercise 4.1: 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' above for more details.

4.1 Typesetting

Typesetting your document is usually done 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 — there are two formats available:

- The standard (default) LaTeX program produces a device-independent (DVI) file which can be used with any TeX previewer or printer driver on any make or model of computer. There are dozens of these available: at least one of each (previewer and printer driver) should have been installed with your distribution of TeX.
- The *pdfLaTeX* program produces an Adobe Acrobat PDF file which can be used with any suitable previewer, such as *Okular*, *Foxit*, *GSview*, *xpdf*, *kpdf*, 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 Running 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 ('batch' or 'scripted' processing).

Whichever way you run LaTeX, it will process your file and display a log or record of what it's doing (see Exercise 3: it looks the same no matter what system you use). This is to let you see where (if!) there are any errors or problems. The log may appear in a subwindow or a new window, depending on your editor.[1]

[1] Some recent versions of *Emacs* hide the log if there were no errors, and display it only if something went wrong.

Exercise 4.2: Running LaTeX from the toolbar or menu

Run LaTeX on your file. According to which system you're using this will either be the LaTeX toolbar icon or the TeX → TeX File menu item.

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

If LaTeX reports any errors — easily identifiable as lines in the log beginning with an exclamation mark (!) — *don't panic!* Turn to §4.2, identify what went wrong, and fix it in your input file. Then re-run LaTeX. If there were no errors, your file is ready for displaying or printing.

It is worth practising running LaTeX from a command window even if you normally use a Graphical User Interface (GUI) (one with windows and a mouse), so that you understand what it does. See Figure 4.1 for an example.

Exercise 4.3: Running LaTeX in a terminal or console window

- ☐ Under graphical Unix-based systems (Linux and Mac) 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** followed by the name of the folder where you saved your sample document, then press the Enter or Return key:

```
cd Documents
latex quickstart
```

Then type **latex** or **pdflatex** followed by the name you gave the sample document, and press the Enter or Return key.

4.1.2 Standard LaTeX and pdfLaTeX

Your editor can be set up to run the original ('standard') LaTeX and generate DVI files, or to run pdfLaTeX and generate PDF files. Both produce identical output, and differ only in the graphics file formats they can handle (see §6.5), and in some typographic advances like microjustification.

Figure 4.1: Command-line usage

```
File  Edit  View  Terminal  Help
peter@pentacle:~$ cd Documents
peter@pentacle:~/Documents$ latex quickstart
This is pdfTeX, Version 3.1415926-1.40.10 (TeX Live 2009/Debian)
 restricted \write18 enabled.
entering extended mode
(./quickstart.tex
LaTeX2e <2009/09/24>
Babel <v3.8l> and hyphenation patterns for english, usenglishmax, dumylang, noh
yphenation, farsi, arabic, croatian, bulgarian, ukrainian, russian, czech, slov
ak, danish, dutch, finnish, french, basque, ngerman, german, german-x-2009-06-1
9, ngerman-x-2009-06-19, ibycus, monogreek, greek, ancientgreek, hungarian, san
skrit, italian, latin, latvian, lithuanian, mongolian2a, mongolian, bokmal, nyn
orsk, romanian, irish, coptic, serbian, turkish, welsh, esperanto, uppersorbian
, estonian, indonesian, interlingua, icelandic, kurmanji, slovenian, polish, po
rtuguese, spanish, galician, catalan, swedish, ukenglish, pinyin, loaded.
(/usr/share/texmf-texlive/tex/latex/base/article.cls
Document Class: article 2007/10/19 v1.4h Standard LaTeX document class
(/usr/share/texmf-texlive/tex/latex/base/size12.clo))
(/usr/share/texmf-texlive/tex/latex/psnfss/palatino.sty)
(/usr/share/texmf-texlive/tex/latex/ltxmisc/url.sty)
No file quickstart.aux.
(/usr/share/texmf-texlive/tex/latex/psnfss/ot1ppl.fd)
(/usr/share/texmf-texlive/tex/latex/psnfss/omsppl.fd)
(/usr/share/texmf-texlive/tex/latex/psnfss/ot1pcr.fd) [1] (./quickstart.aux) )
Output written on quickstart.dvi (1 page, 1900 bytes).
Transcript written on quickstart.log.
peter@pentacle:~/Documents$
```

Apple Mac versions of LaTeX come preset to produce PDF files. *Emacs* does not have a default menu configured for *pdfLaTeX* but if you have already run standard LaTeX on the file, you can type the `pdflatex` command in the `*TeX-Shell*` pane.

4.2 Errors and warnings

LaTeX describes what it's typesetting while it does it, and if it encounters something it doesn't understand or can't do, it will display a message saying what's wrong. It may also display warnings for less serious conditions.

Don't panic if you see error messages: it's very common for beginners 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 and easily corrected in your editor, and you can then run LaTeX again to check you have fixed everything. Some of the most common errors are described in § 4.2 with an explanation of how to fix them.

Some editors show hotlinks in the LaTeX log window where you can click on an error message and the cursor will jump to the line in your document where the error was spotted.

4.2.1 Error messages

The format of an error message is always the same. Error messages begin with an exclamation mark at the start of the line, and give a description of the error, followed by another line starting with the number, which refers to the line-number in your document file which LaTeX was processing when the error was spotted. Here's an example, showing that the user mistyped the `\tableofcontents` command:

```
! Undefined control sequence.
l.6 \tableofcotnetns
```

When LaTeX finds an error like this, it displays the error message and pauses. You must type one of the following letters to continue:

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

a. This only works if you're using an editor which LaTeX can communicate with.
b. This is only a temporary fix to get the file processed. You still have to make that correction in the editor.

Some systems (*Emacs* is one example) run LaTeX with a 'non-stop' switch turned on, so it will always process through to the end of the file, regardless of errors, or until a limit is reached.

4.2.2 Warnings

Warnings don't begin with an exclamation mark: they are just comments by LaTeX about things you might want to look into, such as overlong or underrun lines (often caused by 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.

4.2.3 Examples

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

Most error messages are self-explanatory, but be aware that the place where LaTeX spots and reports an error may be later in the file than the place where it actually occurred. For example if you forget to close a curly brace which encloses, say, italics, LaTeX won't report this until something else occurs which can't happen until the curly brace is encountered (eg the end of the document!) Some errors can only be righted by humans who can read and understand what the document is supposed to mean or look like.

Newcomers — remember to check the list of *special characters* (➠ § 2.4): many 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 hours as you get used to them.

4.2.3.1 Too many }'s

```
! Too many }'s.
l.6 \date December 2004}
```

The reason LaTeX thinks there are too many }'s here is that the opening curly brace is missing after the `\date` control sequence and before the word `December`, so the closing curly brace is seen as one too many (which it is!).

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!

4.2.3.2 Undefined control sequence

```
! Undefined control sequence.
l.6 \dtae
         {December 2004}
```

In this example, LaTeX is complaining that it has no such command ('control sequence') as `\dtae`. Obviously it's been mistyped, but only a human can detect that fact: all LaTeX knows is that `\dtae` is not a command it knows about — it's undefined.

Mistypings are the commonest source of error. If your editor has drop-down menus to insert common commands and environments, use them!

4.2.3.3 Runaway argument

```
Runaway argument?
{December 2004 \maketitle
! Paragraph ended before \date was complete.
<to be read again>
                   \par
l.8
```

In this error, the closing curly brace has been omitted from the date. It's the opposite of the error in § 4.2.3.1, and it results 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.

4.2.3.4 Capacity exceeded

```
! TeX capacity exceeded, sorry [parameter stack size=5000].
```

This is rather more serious: it means TeX has completely run out of memory. This will happen if you try to push the system too far, like getting it to read lines which are quite excessively long, or macros which are too complex to fit in memory (or possibly just badly-written). I had it happen once with an author who had written a single paragraph over 37 pages long. I suggested this was perhaps a style that was unfair on his readers...

4.2.3.5 Underfull hbox

```
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]
```

This is a warning that LaTeX cannot stretch the line wide enough to fit, 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 a badness of 1394). 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. 153 if you're curious.

4.2.3.6 Overfull hbox

```
[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-
```

And 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 (over 3mm or more than $\frac{1}{8}''$), and a manual correction needs making (such as a change to the hyphenation), or the flexibility settings need changing (outside the scope of this book).

4.2.3.7 Missing package

```
! LaTeX Error: File `paralisy.sty' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: sty)

Enter file name:
```

When you use the `\usepackage` command to request LaTeX to use a certain package, it will look for a file with the specified name and the filetype `.sty`. In this case the user has mistyped the name of the paralist package, so it's easy to fix. However, if you get the name right, but the package is not installed on your machine, you will need to download and install it before continuing (see Chapter 5).

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 *pdfLaTeX*, a PDF file will have been created, and you can skip to § 4.3.3.

4.3.1 Previewing DVI output

To see the typeset output, click on the dvi Preview toolbar icon or use the TeX TeX View menu item. A WYSIWYG preview window will appear with your typeset display (see Figure 4.2).

Bitmap preview fonts in DVI viewers

All modern distributions of LaTeX use Type 1 Postscript fonts which scale themselves to the right size, but a few older fonts, especially specialist ones, still come in METAFONT format, and need to be recreated the first time they are used at a given size. The first time you display your DVI file using such fonts, there may be a short pause while the previewer creates the bitmaps used for screen previews of these fonts. As you continue to work with them and your system accumulates these font files, the pause for generating them will disappear. If you use pdfLaTeX, this pause happens at the end of processing, rather than when you display the document.

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 the file.

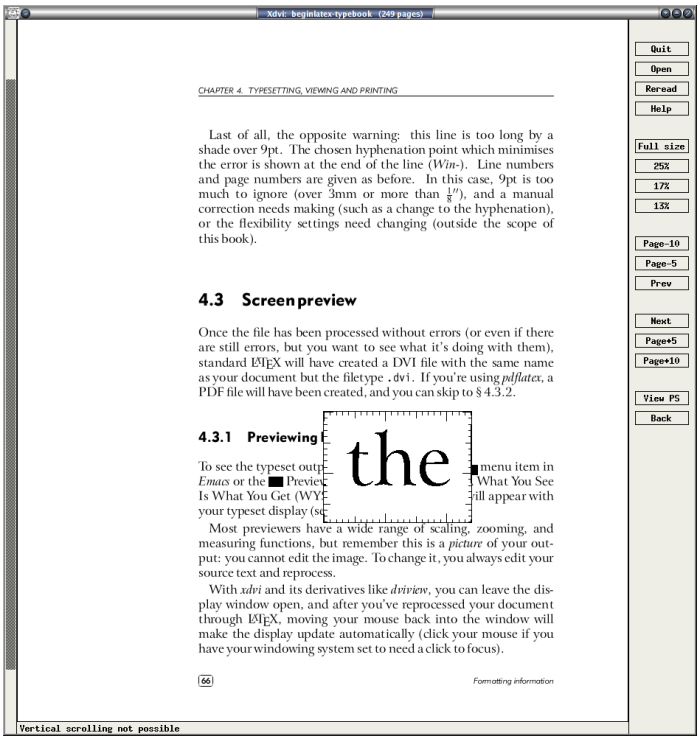
With *xdvi* and its derivatives like *dviview*, you can leave the display window open, and after you've reprocessed your document through LaTeX, moving your mouse back into the window will make the display update automatically (click your mouse if your windowing system needs a click to focus).

Figure 4.2 shows *xdvi* displaying a page. With a standard three-button mouse you get three levels of micro-zoom to let you inspect fine details.

4.3.2 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 nearly two decades, and all printers and publishers are accustomed to using it. PDF is a descendant of PostScript, and 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 (it is actually a programming language in

Figure 4.2: DVI preview



its own right). The drawback is the large size of PostScript files, especially if they contain bitmapped graphics.

The *dvips* program which comes with all TeX distributions is used to generate PostScript files directly from your DVI output. These `.ps` files can be viewed, printed, sent to a platemaker or filmsetter, put online for downloading, or converted to PDF or other formats.

DVI viewers cannot render some PostScript graphical manipulations like rotating and deforming, so an alternative to viewing the DVI file direct is to generate a PostScript file and use a PostScript viewer. You may have to 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 also very simple to do manually: let's assume your LaTeX file was called `mydoc.tex`, so processing it has created `mydoc.dvi`. Just type:

```
dvips -o quickstart.ps quickstart
```

in a command window (see Exercise 3 for how to use one) and *dvips* will create `quickstart.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 *Ghostscript*, which should have been installed automatically along with your TEX system (if not, install both now: *GSview* is separately licenced and cannot legally be included in some older TEX distributions, so you may 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.

4.3.3 Previewing with PDF

The Portable Document Format (PDF) is a derivative of PostScript. Whereas PostScript is a programming language in itself, PDF is in effect the *result* of processing a document through PostScript: it's a binary file format, extremely compact, and well-supported on all platforms.

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

Most editors are configured to display a toolbar icon which will pop up *Acrobat Reader* or some other viewer with the current PDF output file.

Adobe's *Acrobat Reader* cannot automatically update the view if you reprocess your document, in the way that *xdvi* and *GSview* can. You have to close the display with Ctrl–W and reload the file with Alt–F 1.

Bitmap preview fonts in Acrobat Reader

Acrobat Reader is poor at rendering Type 3 (bitmap) fonts. If you are using these (see § 4), you will see a very fuzzy display at low magnifications. It will print perfectly, but Acrobat Reader's display is disappointing. The solution is to use a better previewer or to upgrade to the Type 1 versions of the fonts if possible, or both. If you need to use Type 3 fonts in PDFs, 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.

4.4 Printer output

TEX systems print on almost anything from portable and domestic dot-matrix printers through normal office ink-jet and laser printers up to

the biggest commercial phototypesetters, including a host of other devices in between (numerically-controlled stencil-cutters, knitting machines, and ink-jet cake-decorators, to name but a few).

On most modern systems, printing happens in the normal manner through your system's printer drivers: you just click on the Print icon in your preview. However, TEX's ancillary programs are capable of creating highly-optimised printfiles for almost any printer or typesetter, allowing you to send printout to printers that are not connected to your computer.

The rest of this section deals with how to print on older systems without a print management converter, where the procedure may vary slightly according to how you do your typesetting and previewing.

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 *GSview* instead.

If you are using PDF you can print directly from your PDF viewer. Be careful about using Adobe Acrobat Reader's 'Shrink to fit' option, as it will change the size of your document so all your measurements will be different. Turn it off.

Non-PostScript printers You can create a PostScript file with *dvips* (see §4.3.2) and use *GSview* to print it (*GSview* can print PostScript files to almost any make or model of non-PostScript printer).

If you have a real PostScript printer or you are using a system with built-in PostScript printing support (such as Linux or Mac), you can create and send PostScript output directly from your editor 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 TEX TEX Print menu item.

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 (*pstops*), for example to perform page imposition to get 4, 8, or 16 pages to a sheet for making booklets (*psnup*).

Exercise 4.4: Print it!

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

If you need a non-PostScript/*Ghostscript* solution, install a separate TEX print driver for your printer. Some may be supplied with your TEX installation, and there are dozens more 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, *dvialw* 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 quickstart | lpr
```

Microsoft Windows has no easy way to bypass the print spool, but you can do it from an MS-DOS command window with (using a HP printer as an example):

```
dvihp quickstart -o quickstart.hp
copy /b quickstart.hp LPT1:
```

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

5 CTAN, packages, and online help

The Comprehensive TEX Archive Network (CTAN) is a repository of Web documents and files from HyperText Transfer Protocol (HTTP) and File Transfer Protocol (FTP) servers worldwide which contain copies of almost every piece of free software related to TEX and LATEX.

CTAN is based on three main servers, and there are several online indexes available. There are complete TEX and LATEX systems for all platforms, utilities for text and graphics processing, conversion programs into and out of LATEX, printer drivers, extra typefaces, and (possibly the most important) the LATEX packages. The three main servers are:

- TEX Users Group: `http://www.ctan.org/`
- UK TEX Users Group: `http://www.tex.ac.uk/`

Always try CTAN first

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 the FAQ (§ 5.3.1).

- Deutschsprachige Anwendervereinigung TEX e.V. (DANTE, the German-speaking TEX Users Group); `http://dante.ctan.org/`

5.1 Packages

Add-on features for LATEX are known as *packages*. Dozens of these are pre-installed with LATEX and can be used in your documents immediately. They are all be stored in subdirectories of `texmf/tex/latex` (or its equivalent) named after each package. To find out what other packages are available and what they do, you should use the CTAN search page[1] which includes a link to Graham Williams' comprehensive package catalogue.

A package is a file or collection of files containing extra LATEX commands and programming which add new styling features or modify those already existing. Installed package files all end with `.sty` (they used to be called 'style files') and there may be ancillary files as well.

When you try to typeset a document which requires a package which is not installed on your system, LATEX will warn you with an error message that it is missing (see § 4.2.3.7), and you can then download the package and install it using the instructions in § 5.2.

Auto-install of missing packages MIKTEX has a feature that spots if you try to use a package that isn't installed, and offers to download and install it for you there and then. This is now being ported to other distributions of LATEX, so check the documentation on the DVD to see if it is working in your installation. This avoids you having to do manual installation except for a few packages that are very old or do not conform to the TDS standard.

You can also download updates to packages you already have, both the ones that were installed along with your version of LATEX as well as ones you have added.

There is no limit to the number of packages you can have installed on your computer (apart from disk space!), but there is probably a physical limit to the number that can be used inside any one LATEX 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 (the style file for this document uses over 30).

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, as we have already seen in earlier chapters. For example, to use the xcolor package, which lets you typeset in colours (I warned you this was coming!), you would type:

[1] `http://www.ctan.org/search`

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

Some packages allow optional settings in square brackets. If you use these, you must give the package its own separate **\usepackage** command, like geometry and xcolor shown below:

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

\title{\color{Crimson}Practical Typesetting}
\author{\color{DarkSlateBlue}Peter Flynn\\Silmaril Consultants}
\date{\color{ForestGreen}January 2011}
\maketitle

\end{document}
```

(Incidentally, this is a very crude and cumbersome way to do colours in titling. It's fine for a one-time document, but it will interfere with running heads if you use them; and if it's for a repeatable style we'll see in Chapter 9 how it can be automated as part of the **\maketitle** command 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 geometry and xcolor do. Read the documentation for the package concerned to find out what can be done.

5.1.2 Package documentation

To find out what commands a package provides (and thus how to use it), you need to read the documentation. The simplest way is to use your command window and type `texdoc` followed by the package name. This will bring up the documentation in your PDF or DVI viewer.

Exercise 5.1: Add colour

Use the xcolor package to add some colour to your document. Stick with primary colours for the moment.

Use the geometry package to change the margins.

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

If that doesn't find it, in the `texmf/doc` subdirectory of your installation there should be directories full of `.dvi` and `.pdf` files, one for every package installed. These can be previewed or printed like any other DVI or PDF file (see §4.3.1). If your installation procedure has not installed the documentation, the files can all be downloaded from CTAN.

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 generate the documentation for additional packages you install yourself.

Exercise 5.2: Read all about it

Find and view (or print) the documentation on the geometry package you used in Exercise 1.

Investigate some 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) — and you're not using a system with the *tlmgr* auto-installer — you can use the indexes on any CTAN server to find the package you need and the directory where it can be downloaded from.

5.2.1 Downloading packages

What you need to look for is almost always *two files*, one ending in `.dtx` and the other in `.ins`. The first is a DOCTEX file, which combines the

package programs and their documentation in a single file. The second is the installation routine (much smaller). You *must always* download *both* files. In most cases, CTAN provides a 'Download ZIP File' link, which gets you all the files you need for the package.

If the two files or the package are not there, it means one of two things:

- *Either* the package is part of a much larger bundle which you shouldn't normally update unless you change version of LaTeX;[2]
- *or* it's one of a few rare or unusual packages still supplied as a single `.sty` or `.cls` file originally written for the now obsolete LaTeX 2.09, or by an author who has a moral or philiosophical objection to using DocTeX.[3]

Download both files (or the ZIP file) to a *temporary directory*. If you use Windows, create a folder like `C:\tmp` or `C:\temp` for this; Mac and Linux systems already have a `/tmp` directory. If you downloaded the ZIP file, unzip it there.

5.2.2 Installing a package

There are four steps to installing a LaTeX package:

1. **Extract the class or package files**
   Run LaTeX on the `.ins` file. That is, open the file in your editor and process it as if it were a LaTeX document (which it is), or if you prefer, type `latex` followed by the `.ins` filename in a command window in your 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 LaTeX or *pdfLaTeX* on the `.dtx` file *twice*. This will create a `.dvi` or `.pdf` file of documentation explaining what the package is for and how to use it. Two passes through LaTeX are needed in order to resolve any internal crossreferences in the text (a feature we'll come onto later). View or print this file in the usual manner (see § 4.3).

[2] For example, there is no `xcolor.dtx` and `xcolor.ins` for the xcolor package because it forms part of the graphics bundle, which is installed on all LaTeX systems anyway. Such packages change very rarely, as they form part of the core of LaTeX and are very stable. You should never try to update these packages in isolation.

[3] Almost all of these have been updated to work with LaTeX 2$_\varepsilon$, so they should be installed as in step 3 in the procedure on p. 82.

3. **Install the files**
   Move the files created in step 1 from your temporary directory to the right place[s] in your personal TEX directory tree (see below) — *always* your 'personal' TEX directory tree, *a*) to prevent your new package accidentally overwriting master files in the main TEX directories; and *b*) to avoid your newly-installed files being overwritten when you next update your version of TEX. Never, *never* put files into the TEX master installation tree. If you are updating a shared system, however, you can put files into the *local* TEX directory tree.

   'The right place' sometimes causes confusion, especially if your TEX installation is old or does not conform to the TEX Directory Structure. For a TDS-conformant system, 'the right place' is your personal TEX directory tree unless you are a systems manager updating a shared machine, in which case it's the local TEX directory tree. Your personal TEX directory tree is in your home directory (folder):

   - Unix and GNU/Linux systems: `~/texmf/`
   - Apple Mac systems: `~/Library/texmf`
   - Windows MIKTEX systems: `Computer/username/texmf`

   Create this directory now if it does not already exist.

   Windows MIKTEX users (only) must use the MIKTEX Administration program to add the new folder to the search tree. Each time you update files in there, you must run the FNDB updater in the MIKTEX Administration program.

   'Suitably-named' means sensible and meaningful (and probably short). For a package like paralist, for example, I'd call the directory `~/texmf/tex/latex/paralist`.

   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 BIBTEX packages or font packages will typically have several files to install. This is why it is important to create a subdirectory for the package rather than dump the files into `misc` along with other unrelated stuff.

   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. **Shared systems and MIKTEX: update your index**
   If you are updating a shared system and putting the files into the local TEX directory tree, or if you are using Windows MIKTEX, you must afterwards run your TEX indexer program to update the package database. This program comes with every modern version of TEX and is

Table 5.1: Where to put files from packages

Type	Directory in your personal `texmf/`	Description
`.cls`	`tex/latex/`*packagename*	Document class file
`.sty`	`tex/latex/`*packagename*	Style file: the normal package content
`.bst`	`bibtex/bst/`*packagename*	BibTeX style
`.mf`	`fonts/source/public/`*typeface*	METAFONT outline
`.fd`	`tex/latex/mfnfss`	Font Definition files for METAFONT fonts
`.fd`	`tex/latex/psnfss`	Font Definition files for PostScript Type 1 fonts
`.pfb`	`fonts/type1/`*foundry*`/`*typeface*	PostScript Type 1 outline
`.afm`	`fonts/afm/`*foundry*`/`*typeface*	Adobe Font Metrics for Type 1 fonts
`.tfm`	`fonts/tfm/`*foundry*`/`*typeface*	TeX Font Metrics for METAFONT and Type 1 fonts
`.vf`	`fonts/vf/`*foundry*`/`*typeface*	TeX virtual fonts
`.dvi`	`doc/`*packagename*	package documentation
`.pdf`	`doc/`*packagename*	package documentation
others	`tex/latex/`*packagename*	other types of file unless instructed otherwise

variously called *texhash*, *mktexlsr*, or even *configure*, or it might just be a mouse click on a button or menu in your configuration system (like MIKTeX's). Read the documentation that came with your installation to find out which it is.

On Linux, GNU/Unix, and Apple Mac systems *do not* create an `ls-R` database in your `~/texmf` or `~/Library/texmf` directory or run the indexer. These systems search your personal TeX directory automatically.

On MIKTeX and shared systems, run your TeX indexer program after making changes

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

Exercise 5.3: Install a package

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

The *tlmgr* auto-updater is widely used in TEX Live systems *except* where TEX has been installed from Debian-based Unix *system* packages. On Windows and Apple Mac, and on Unix systems where TEX Live has been installed from the TUG DVD or download, *tlmgr* is the normal way to update packages. The manual process described above is *only* for those cases where *tlmgr* cannot be used.

This includes the thousands of installations which do not conform to the TDS, such as old shared Unix systems and some Microsoft Windows systems, so there is no way for an installation program to guess where to put the files: *you* have to know this yourself. There are also systems where the owner, user, or installer has chosen *not* to follow the recommended TDS directory structure, or is unable to do so for political or security reasons (such as a shared system where she cannot write to a protected directory).

The reason for having the local `texmf` directory (called `texmf-local` or `texmf.local` on some systems) is to provide a place for local modifications on a shared or managed system (such as a server) which will override but otherwise not interfere with the main TEX installation directory tree. Your installation should already be configured to look in the personal and local directories first, so that any updates to standard packages will be found there *before* the copies in the main `texmf` tree. All modern TEX installations do this, but if not, you can edit `texmf/web2c/texmf.cnf` (or on a shared system, ask your systems manager or support person to do so). There is an example in Appendix A.

5.2.3 Replicating the TDS

The TEX Directory Structure (TDS) is documented at `http://www.tug.org/tds/`. I find it useful to make the subdirectory structure of your local `texmf` directory the same as that of the main installation `texmf` directory. Examine the subdirectories of `texmf/tex/latex/` for examples. For updates of packages which came with your LATEX distribution (as distinct from new ones you are adding yourself), you can then use the same subdirectory name and position in your local `texmf/...` as the original used in the main `texmf/...`, and LATEX will then always use the updated version.

If you want to create the entire subdirectory structure ready for use, you can do it under Unix with the following commands (this example uses the

Ubuntu/Debian directory `/usr/local/share/texmf` rather than the Red Hat `/usr/share/texmf-local`):

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

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 also works on a Mac running OS X if you use `/usr/local/texlive/`*yyyy*`/texmf-dist` in the first command (replacing *yyyy* with the year of the distribution) and `~/Library/texmf/{}` in the second. In other cases, if your directories are not as given in the example, you need to substitute the actual paths to your main `texmf` and local `texmf` directories.

5.3 Online help

The indexes and documentation files on CTAN are the primary online resource for self-help on specific packages, and you should read these carefully before asking questions about packages.

5.3.1 The FAQ

For general queries you should 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 managed by the UK TEX Users Group and can be found at `http://www.tex.ac.uk/faq/`.

5.3.2 The TEXhax mailing list

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).

5.3.3 Web sites

The TEX Users Group, as well as most local user groups, maintains a web site (`http://www.tug.org`) with lots of information about various aspects of the TEX system. See Appendix B for information on joining TUG.

5.3.4 News

The Usenet newsgroup `comp.text.tex` is the principal forum for other questions and answers about LaTeX. Feel free to ask questions, but please do not ask frequently-asked questions: read the FAQ instead. The people who answer the questions do so voluntarily, unpaid, and in their own time, so please 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*[4]), or download one of the many free newsreaders.[5]

5.3.5 Google LaTeX list

There is a Google Groups mailing list for LaTeX users at `http://groups.google.com/group/latexusersgroup?hl=en`

5.3.6 Commercial support

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 TeX, or contact a consultancy which deals with TeX (details on the TUG Web site).

[4] `http://www.mozilla.org/`

[5] Note that this means newsreaders for the Usenet News (NNTP) service. It does *not* mean syndication readers for RSS, which are a different thing entirely — these are unfortunately also sometimes referred to as 'newsreaders'.

6 Other document structures

It is perfectly 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. However, it's more common to 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 $\tau\eta\chi\nu\epsilon$ (*tekne*), the specialist material or artistry of its field. A literary analysis such as *La Textualisation de* Madame Bovary (on the marginal notes in the manuscripts of Gustave Flaubert's novel) is every bit as technical in the literary or linguistic field as the maintenance manual for the Airbus 380 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 little think about 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. This is usually 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 your documents have any of the features below, then you have probably 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.
- ☐ 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.

Table 6.1: Types of lists

Random or arbitrary lists (sometimes called 'itemized' or 'bulleted' lists) where the order of items is 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 (sometimes called 'numbered' 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 mutually inclusive or exclusive, with the final item prefixed by 'and' or 'or' respectively.

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, shown in Table 6.1.

There are actually two other types, segmented lists and reference lists, but these are much rarer, and outside the scope of this document.

The structure of lists in LaTeX is identical for each type, but with a different environment name. Lists are another example of this 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.

The default list bullet is round and solid[1] (•) which is also available with the command `\textbullet` if you load the textcomp 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:

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

In standard LaTeX document classes, the vertical spacing between items, and above and below the lists as a whole, is more than between paragraphs. If you want tightly-packed lists, use the mdwlist package, which provides ‘starred’ versions (`itemize*`, `enumerate*`, etc.).

[1] If your browser font doesn't show it, don't worry: some don't. LaTeX will.

```
\begin{enumerate}

\item Enumerated lists use numbering on each
item (can also be letters or roman numerals);

\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 (can also be letters or roman numerals);
2. Long items use 'hanging indentation', just the same as for itemized lists;
3. The numbering system can be changed for any level.

6.2.3 Description lists

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

All three of these types of lists can have multiple paragraphs per item: just type the additional paragraphs in the normal way, with a blank line between each. So long as they are still contained within the enclosing environment, they will automatically be indented to follow underneath their item.

6.2.4 Inline lists

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

See Chapter 8 for details of the font-changing commands used in the optional argument to `inparaenum`.

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

```
\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.

elsewhere in the text (as in 'see item 501(c)3'). In this sense they are really a kind of sub-sectional division, and LaTeX'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 chapter 'Segmentation and Alignment' (17) in Lou Burnard and Michael Sperberg-McQueen's *TEI Guidelines*.

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.

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

Inline lists, which are sequential in nature, just like enumerated lists, but are *a*) formatted within their paragraph; *b*) usually labelled with letters; and *c*) usually have the final item prefixed with ‘and’ or ‘or’ , like this example.

Exercise 6.1: List practice

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

If you successfully installed paralist in Exercise 3 then you can use inline lists as described in § 6.2.4.

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.

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.

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: for each level there is a counter to count the items and a command to produce the label:[2]

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

Note that each counter and command ends with the Roman numeral value of its level (this is to overcome the rule that LaTeX commands can only be made of letters — digits wouldn't work here). To change the format of a numbered list item counter, just renew the meaning of its label:

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

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.

Exercise 6.2: 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.

[2] In fact, any time you define a counter in LaTeX, 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*.

6.3.1 Floats

Tables and Figures are what printers and publishers 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 many floats occurring in rapid succession, which raises the problem of 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.

Lists and Tables: 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.

Terminology

LaTeX, in common with standard typesetting 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.

But this is a skill few authors have, and it's one point at which professional typographic advice or manual intervention may be needed.

There is a float package which lets you create new classes of floating object (perhaps Examples or Exercises), and it also implements a method of forcing a float not to float (that is, to appear where it occurs in the text, even if that breaks the page).

6.3.2 Formal tables

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

```
\begin{table}
\caption{Project expenditure to year-end 2012}
\label{ye2012exp}
...
\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 Tables* (➠ § 3.7):

```
\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 § 6.4 on Figures, where they are more common.

To typeset tabular matter, use the `tabular` environment. You follow the `\begin{tabular}` command with 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.

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 p column specifications are *not* multi-row (row-spanned) entries, they are single cells which contain multiple lines of typesetting: the distinction is very important. Auto-adjusting columns are possible with the tabularx and tabulary packages, offering different approaches to dynamic table formatting.

The array package provides for many other typographic variations such as left-aligned, right-aligned, and centred multi-line columns, and the prefixing of a column specification with formatting to be applied to all cells in that column, saving you having to enter it in every cell. The colortbl package lets you colour rows, columns, and cells; and the dcolumn package provides decimal-aligned columns. Multi-column (column-spanning) is built into LaTeX tables; for multi-row (row-spanning) cells you need the multirow package. Multi-page and rotated (landscape format) tables can be done with the longtable, rotating, and landscape packages.

As an example, the tabular setting illustrated has three columns, the first one centered, the second left-aligned, and the third one right-aligned, and would therefore be specified as {clr}. Note the use of indentation to make the elements of the table clearer for editing, and note also how the typeset formatting is unaffected by this (see Table 6.2).

```
\begin{table}
  \caption{Project expenditure to year-end 2012}
  \label{ye2012exp}
  \begin{center}
    \begin{tabular}{clr}
      &Item&\EUR\ 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, etc.&1,500\\\cline{3-3}
        &Total&64,783
    \end{tabular}
    \par\medskip\footnotesize
The Institute also contributes to (a) and (b).
  \end{center}
\end{table}
```

If you copy and paste this into your example document, you will need to add the marvosym package which enables the CEC-conformant € symbol **\EUR**.

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

Table 6.2: Project expenditure to year-end 2012

	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, etc.	**1,500**$_0$
	Total	**64,783**

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

provided. 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). *Winedt* and *TeXMakerX* both have a similar graphical table-editing mode. If your tabular data comes from outside LaTeX, Nicola Talbot's excellent datatool package allows the import of data from (e.g.) spreadsheet `.csv` files.

Extra space is automatically added on both sides of all columns, and can be adjusted by changing the value of the `\tabcolsep` dimension before you begin the tabular environment.

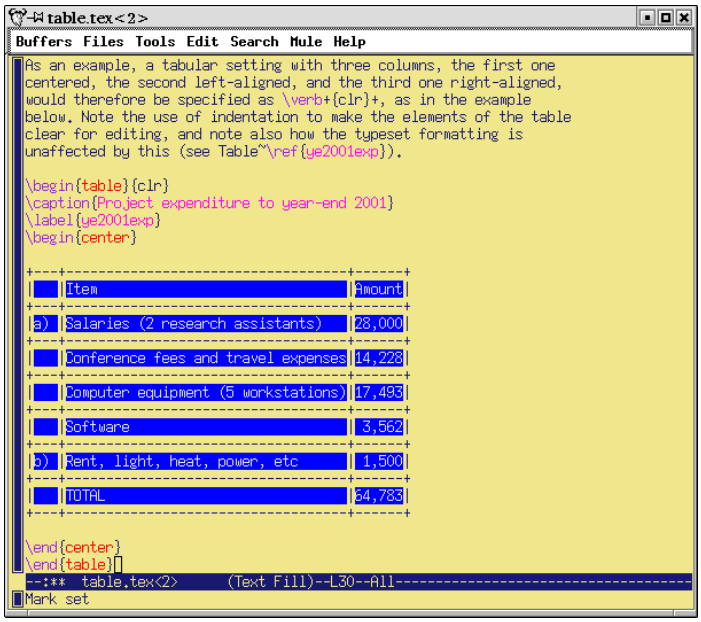
To change the line-spacing in a tabular setting, you can redefine the `\arraystretch` command with `\renewcommand`. `\arraystretch` is a *multiplier*, preset to 1, so setting it to 1.5 would make the lines of your tabular settings one and a half times more than normal. You can increase the line-spacing after an individual row by following the double backslash with a dimension in square brackets, for example `\\[6pt]` (negative values decrease the line-spacing).

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 marked 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 — the argument needs a range). 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

Figure 6.1: Tables mode for *Emacs*



`\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.

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, LaTeX typesets `tabular` environments *inline* to the surrounding text (i.e. within the paragraph), 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: full details are in the manuals mentioned in the the last paragraph of the *Foreword* on p. x. One final note to remind you of the automated crossreferencing features: because the example table is labelled, it can be referenced from anywhere in this document as Table 6.2 just by using `\ref{ye2012exp}`, regardless of how much the surrounding document or structure is moved or edited.

Exercise 6.3: 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)

6.4 Figures

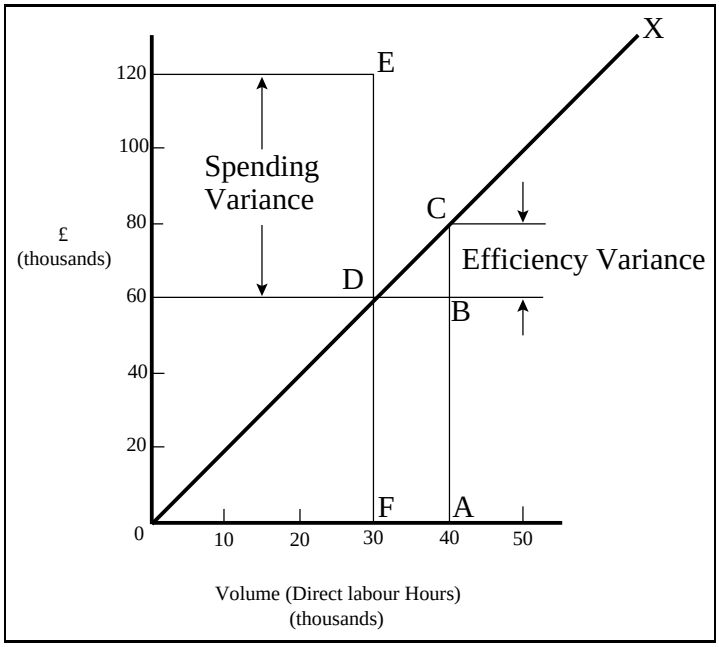
As explained in § 6.3.1, Figures and Tables float to a vacant part of the page, as they are not part of the sequence of sentences making up your text, but illustrative objects that you refer to.

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}
\includegraphics[width=.75\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

Figure 6.2: Total variable overhead variance (after Bull)



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 (see § 6.5.1).

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}`[3]

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 the filetype*, for example: `\includegraphics{myhouse}` (we'll see in a minute 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

[3] You may find a lot of old files which use a package called epsf. Don't use it: it's obsolete.

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 § 6.5.2).

- For standard LaTeX with *dvips*, graphics files *must* be in Encapsulated PostScript (EPS) format: this was the publishing industry standard for portable graphics for many years, and no other format will work portably in standard LaTeX.[4]

  All good *graphics* packages can save images as EPS, but be very careful with other software such as statistics, engineering, mathematical, and numerical analysis packages, because some of them, especially on Microsoft Windows platforms, use a very poor quality driver, which in turn creates 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 creating software. Downloading Adobe's own PostScript driver from their Web site and using that instead may improve things, but the only real solution is to use software that creates decent output.

- For *pdfLaTeX*, graphics files can be in Portable Network Graphic (PNG), Joint Photographic Experts Group (JPG), or PDF format, and *not* EPS. This means if you want to use both standard LaTeX as well as *pdfLaTeX*, you need to keep your graphics in two formats, EPS and one of the others. This is why you should avoid including the filetype in the filename you give with `\includegraphics`: standard LaTeX will assume EPS, and *pdfLaTeX* will look for PNG, PDF, or JPG automatically, in that order.[5]

For these reasons, if you create vector EPS graphics, and convert them to PDF format, *do not* keep additional JPG or PNG copies of the same image in the same directory, because they risk being used by *pdfLaTeX* instead of the PDF file, resulting in a lower-quality image.

The `\includegraphics` command can take optional arguments within square brackets before the filename to specify either the height or width, and the other dimension will automatically change to scale. If you specify both, the image will be distorted to fit. You can scale an image by a factor instead

[4] Some commercial distributions of TeX systems allow other formats to be used, such as GIF, Microsoft Bitmap (BMP), or Hewlett-Packard's Printer Control Language (PCL) files, and others, by using additional conversion software provided by the supplier; 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. If you use standard LaTeX, stick to EPS.

[5] Strictly speaking the exact order (from the newest definition in `pdftex.def`) is `.png`, `.pdf`, `.jpg`, `.mps`, `.jpeg`, `.jbig2`, `.jb2`, `.PNG`, `.PDF`, `.JPG`, `.JPEG`, `.JBIG2`, and `.JB2`. Thanks to Enrico Gregorio and Philipp Stephani on `comp.text.tex` for locating this.

```
\begin{center}
\includegraphics[width=3cm]{twithcat}
\end{center}
```

of specifying height or width; clip it to specified coordinates; and rotate it in either direction. Multiple optional arguments are separated with commas.

For details of all the arguments, see the documentation on the graphicx package or a copy of the *The LaTeX Companion*. This package also includes commands to ɹoʇɐʇoɹ, ɹoɹɹıɯ, and **scale** text.

It is in fact possible to tell LaTeX to generate the right file format by itself during processing, but this requires an external command-line graphics converter like *ImageMagick*, and as it gets done afresh each time, it slows things down rather a lot.

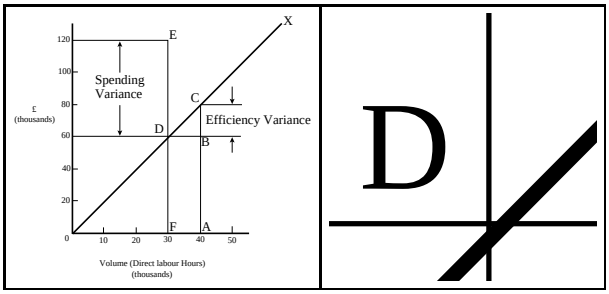
EPS files, especially bitmaps, can be very large indeed, because they are stored in ASCII format. Staszek Wawrykiewicz has drawn my attention to a useful MS-DOS program to overcome this, called *cep* ('Compressed Encapsulated Postscript') available from CTAN in the `support/pstools` directory, which can compress EPS files to a fraction of their original size. The original file can be replaced by the new smaller version and still used directly with `\includegraphics`.

One final warning about using EPS files with `\includegraphics`: never try to specify an absolute path (one beginning with a slash, or one starting with `../`). The *dvips* driver will not accept these because they pose a security risk to PostScript documents. Unlike PDF, PostScript is a programming language, capable of opening files, and the last thing you want is to create a document able to mess with your filesystem (or someone else's).

6.5.1 Making images

There are two types of image: bitmaps and vectors.

Figure 6.3: The diagram from Figure 6.2 shrunk and enlarged



Bitmaps Bitmap images are made of coloured dots, so if you enlarge them, they go jagged at the edges, and if you shrink them, they go blurry. Bitmaps are fine for photographs, where every dot is a different colour, and no-one will notice if you don't shrink or enlarge too much. Bitmaps for diagrams and drawings, however, are almost always the wrong choice, and often disastrously bad.

Vectors Vector drawings are made from instructions (eg 'draw this from here to here, using a line this thick'). They can be enlarged or shrunk as much as you like, and never lose accuracy, because they *get redrawn* automatically at any size. You can't do photographs as vectors, but it's the only acceptable method for drawings or diagrams.

Vector graphic packages are also better suited for saving your image directly in EPS or PDF format (both of which use vectors internally). All the major graphics-generating packages in all disciplines output vector formats: *AutoCAD*, *ChemDraw*, *MathCAD*, *Maple*, *Mathematica*, *ArcInfo*, and so on. EPS is the universally-accepted format for creating vector graphics for publication, with PDF a close second. Most of the major graphics (drawing) packages can also save as EPS, such as *PhotoShop*, *PaintShop Pro*, Adobe *Illustrator*, Corel *Draw*, and *GIMP*. There are also some free vector plotting and diagramming packages available like *InkScape*, *tkPaint*, and *GNUplot* which do the same. Never, ever (except in the direst necessity) create any *diagram* as a bitmap.

Bitmap formats like JPG and PNG are ideal for photographs, as they are also able to compress the data substantially without too much loss of quality. However, compressed formats are bad for screenshots, if you are documenting computer tasks, because too much compression makes them blurry. The popular Graphics Interchange Format (GIF) is good for screenshots, but is not supported by TEX: use PNG instead, with the compression turned down to minimum. Avoid uncompressible formats

like BMP as they produce enormous and unmanageable files. The Tagged Image File Format (TIFF), popular with graphic designers, should also 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 different types of computers (except in faxes, where it's still used in a much stricter version).

Exercise 6.4: 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 pdfLaTeX, or an EPS image if you use standard 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 previewers are not able to display all types of graphics, and some cannot display colour. For best results, use PDF or PostScript preview.

6.5.2 Graphics storage

I mentioned earlier that there was a way to tell 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 one or more names of additional directories you want searched when a file uses the **\includegraphics** command, for example:

```
\graphicspath{{c:/mypict~1/camera}}
\graphicspath{{/var/lib/images}{/home/peter/Pictures}}
```

Put the path in an additional set of curly braces (this lets you add more paths later: each in their own subset of curly braces). I've used the 'safe' (MS-DOS) form of the Windows `MyPictures` folder in the example because it's A Bad Idea to use directory names containing spaces (see the panel 'Picking suitable filenames' on p. 63). 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.

If you use *dvips* to print with or to generate PostScript files, be aware that some versions will not by default handle EPS files which are outside the current directory, and issue the error message that it is 'unable to find' the

image. This is because PostScript is a programming language, and it would theoretically be possible for a maliciously-made image to contain code which might compromise your system. The decision to restrict operation in this way has been widely criticised, but it seems unlikely to be changed. If you are certain that your EPS files are kosher, use the `R0` option in your command, e.g. `\dvips -R0 dvifile`

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[6] 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. In the case of web and email addresses, it is also very important to use a typeface which distinguishes well between 1 (digit one), l (lowercase ell) and I (uppercase eye), and between 0 (zero) and O (uppercase oh). Monospaced typewriter type usually makes this clear.

Standard LaTeX includes two features for handling fixed-format text, inline and display verbatim, 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 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 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 — including TeX — without problems. However, LaTeX will never break the argument of `\verb`

[6] The original term Uniform Resource Locator (URL) is now deprecated 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 LaTeX and XML markup.

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`, but uses the standard curly braces to enclose the string of characters. The big difference between `\verb` and `\url` is that the latter performs a hyphenless break at punctuation characters, as in `http://latex.silmaril.ie/formattinginformation/index.html#microformats`. It was designed for Web URIs, so it understands their syntax and will never break mid-way through an unpunctuated word, only at slashes and full points (and never at embedded hyphens). Bear in mind, however, that spaces and non-ASCII characters are forbidden in URIs, so using spaces in a `\url` argument 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 § 6.7.3), and it can add reference line numbers (useful for chunks of data or programming), and it can even include entire external files.

Exercise 6.5: 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.

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 lines of 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 (box) 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. 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, as in the example shown.

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.Like this small para-graph. You can specify that the top or bottom should align with any surrounding text 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 words may not fit nicely and the spacing may become uneven or 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

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

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

is rare for LaTeX to need this: the example above was deliberately chosen to be obtuse as an illustration of excessive spacing.

Where the contents is more extensive or more complicated, you can use the `minipage` environment.

Within a `minipage` 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.

Note that in both `minipage`s and **`\parbox`**es, the paragraph indentation (`\parindent`) 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 `tabular` environment with the `p` column type specification, or you can use the **`\vbox`** command, which is Plain TeX, and 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`**.

```
\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.

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

```
\setlength{\fboxsep}{1ex}
\setlength{\fboxrule}{1pt}
```

As we saw before, setting justified text in narrow measures will produce poor spacing: either use the **`\raggedright`** command, or change the font size, or add explicit extra hyphenation points.

Note the `\begin{tabular}` and `\begin{minipage}` commands still need the width specifying: in the case of the `\begin{tabular}` by the use

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

This multiline text is more flexible than a tabular setting:

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

[a] Like this.

```
\fbox{\begin{tabular}{p{1.5in}}
Multiline text in a box typeset using \textsf{tabular}
\end{tabular}}
```

Multiline text in a box typeset using tabular

of the `p` column type with its width specification, and in the case of `\begin{minipage}` by the second argument.

6.7.3 Sidebars and panels

The fancybox package lets you extend the principle of `\fbox` 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 delay putting the frame around until it is complete.

The printed version of this document uses this extensively and there is a worked example shown in § 9.5.

7 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:

- offset quotations (sometimes called 'block quotes');
- footnotes and end-notes;
- marginal notes;
- cross-references, both normal ones and bibliographic citations;
- indexes and glossaries;
- typesetting in 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. LaTeX has two environments for doing this.

Such quotations are often set in a smaller size of type, although this is not the default, but you can use one of the size commands like `\small` (see

```
\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*

§ 8.2.4) as shown in the second example on p. 115. The inclusion of the bibliographic citation at the end is optional: here it is done with a non-standard command `\citequote` which I invented for this example (there is more about how to do things like this in Chapter 9).

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;

The `quotation` environment is for longer passages (a paragraph or more) of a single quotation, where the whole block of text indented, and each paragraph of it also has its own indentation on the first line.

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 raised small number where you put the command, and the numbered text automatically printed at the foot of the page.[1] 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 even 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) inside footnotes on its own: either use the `\VerbatimFootnotes` command from the fancyvrb package, or precede

[1] Like this.

```
\begin{quotation}\small\noindent
At the turn of the century William Davy, a Devonshire
parson, finding errors in the first edition of his
\titleof{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*, 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]

\footnote with **\protect**, or use (abuse?) the **\url** command instead (which you should be using for Web and email addresses in any case).

Footnotes in titling commands (**\title**, **\author**, etc) produce the symbols *, †, ‡, §, ¶, ||, **, ††, and ‡‡ for the values 1–9 (and an error message for the tenth such footnote). Footnotes inside a `minipage` environment (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 to print many short footnotes in a single footnoted paragraph so they take up less space (fnpara). 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 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.

If your footnotes are few and far between, you may want to use the sequence of footnote symbols above instead of numbers. You can do this by redefining the output of the footnote counter to be the `\fnsymbol` command:

```
\renewcommand{\thefootnote{\fnsymbol{footnote}}
```

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

Like this.

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}}
```

Some text where you need it.

Then you can use `\marginal{Some text}` 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 attentive reader can probably deduce some of it by inspection. See Chapter 9 for more information about making up your own commands.

7.4 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.

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

In § 7.4.1 there is a list of recent projects.

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:[2] 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.

(The `\S` command produces a section sign (§) and the `\P` command produces a paragraph sign (¶).)

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).[3] If the label was inside a Table or Figure, the reference provides the Table number or Figure number prefixed by the chapter number (in Tables and Figures the `\label` command *must* come *after* the `\caption` command). 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.

2 This section is labelled '`normalxref`', for example.

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

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.

> **Note**
>
> LaTeX records the label values each time the document is processed, but updated values only get used the next time the document is processed. You always need to process the document an extra time before printing or viewing, if you have changed or added references, to make sure the values are correctly resolved.

Unresolved references are printed as two question marks, and also cause a warning message at the end of the log file. There's never any harm in having `\label`s 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 LaTeX called BibTeX, which manages bibliographic references automatically. This reduces the time needed to maintain and format them, and dramatically improves accuracy. Using 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 formatted automatically to the style you specify.

> **BibLaTeX**
>
> 
>
> There is a new bibliographic referencing system for LaTeX called biblatex, which is being developed to overcome some of the complexities of BibTeX's programming language ad provide a wide range of formatting styles. Keep an eye on the newsgroups and mailing lists.

7.4.2.1 Citing references

BibTeX works exactly the same way as other bibliographic databases: you keep details of every document you want to refer to in a separate file, using 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 or ID, just like you do with normal cross-references (see § 7.4.1), and you use this label in your 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 common author (year) and (author, year) formats.

Note that you never have to type the author's name or the year, just the label, because the correct author and year are automatically extracted by BibTeX. There are lots of variants on the standard `\cite` command in the larger packages such as apacite and natbib, allowing you to phrase your sentences with references in as natural a way as possible, and rely on BibTeX to insert the right data. The natbib package is particularly useful as it implements several forms of citation and referencing; originally intended for the natural sciences (whence its name), it is now used in many fields (see § 8).

If you need to include internal location information in a citation, such as a chapter or section number, or a page number or range, put this in an optional argument *before* the label, like this: `\cite[pp.47-52]fg`.

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

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

- The `\bibliography` command is followed by the filename of your BibTeX file *without* the `.bib` extension.
- The `\bibliographystyle` command is followed by the name of any of LaTeX's supported bibliography styles, of which there are many dozens available from CTAN.

The styles plain and alpha are two common generic styles used for drafts, and are built into LaTeX, so they don't need any package adding. The

example above uses Transactions of the Institute of Electrical and Electronics Engineers (IEEETR).

The larger bibliographic style packages (e.g. harvard) have one package name but many different `\bibliographystyle` formats, such as kluwer, agsm, apsr, dcu, etc.. Details are in the package documentation.

Harvard style

The Harvard style, which is very popular, is broken when used with pdflatex if you also have LaTeX2HTML installed (it turns on PDF hyperlinking and then fails on an undefined starred command). I strongly recommend using the implementation of Harvard done with the packages natbib and har2nat instead.

If you need to produce multiple bibliographies, perhaps on a per-chapter basis, or grouped separately for some reason, the bibunits or chapterbib packages are very useful.

7.4.2.2 Running bibtex

After running LaTeX, you run the *bibtex* program, either from the command line or via a toolbar button or menu option in your editor. Most integrated editors (e.g. *Kile*, *WinEdt*, *TeXMakerX*, *TeXnicCenter*) will notice when you use BibTeX, and automatically run *bibtex* for you.

The *bibtex* program extracts the details of every reference you have cited in your LaTeX document from your `.bib` database, formats them according to the style you have specified, and stores them in a bibliographic listing (`.bbl`) file. The next time you run LaTeX, it uses that file with the labels you used in your citations to add the references to your document. The sequence (done automatically by editors) is therefore:

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

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

In practice, authors tend to run LaTeX from time to time during writing anyway, so they can preview the document. Just run BibTeX after adding a new `\cite` command, and subsequent runs of LaTeX will incrementally

incorporate all references without you having to worry about it. You only need to follow the full formal sequence (LaTeX, BibTeX, LaTeX, and LaTeX one more time) when you have finished writing and want to ensure that all references have been resolved and any other crossreferences that may have got moved by the addition of newly cited material.

7.4.2.3 BibTeX format

The format for the BibTeX file is specified in the BibTeX 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}
}
```

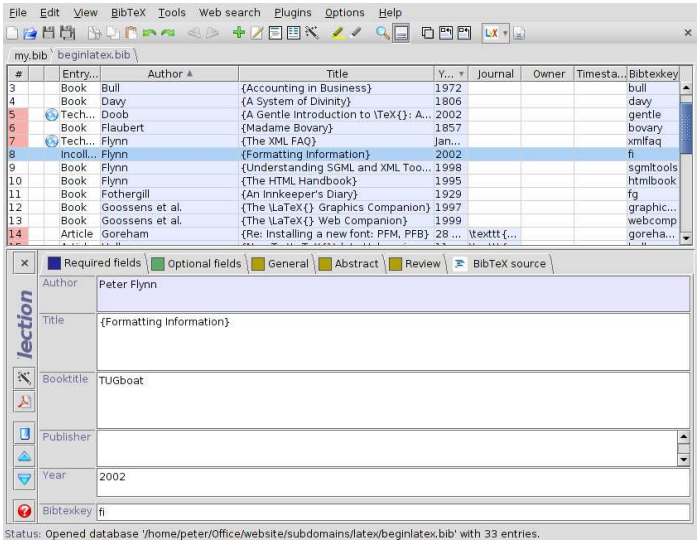
There is a prescribed set of fields for each of a dozen or so types of document: book, article (in a journal), article (in a collection), chapter (in a book), thesis, report, paper (in a Proceedings), etc. Each entry identifies the document type after the '`@`' sign, followed by the entry label that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

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

Most TeX-sensitive editors have a BibTeX mode which understands these entries. *Emacs* automatically uses its `bibtex-mode` whenever you open a file-name ending in `.bib`. When editing BibTeX databases, the rules are simple:

- Omit the comma after the last field in the entry (*only* — eg after `{London}` in the example).
- 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 strictly observed.

Figure 7.1: JabRef, one of several graphical interfaces to BibTeX databases



- Fields which are not used do not have to be included (so if your editor automatically inserts them as blank or prefixed by `OPT` [optional], you can safely delete them as unused lines).

To help with this, there are several graphical interfaces to creating and maintaining BibTeX files, such as *JabRef* (see Figure 7.1), *tkbibtex*, or *pybliographic*.

These interfaces work like those used with wordprocessors, such as *ProCite*, *Reference Manager*, and *EndNote*, and the Reference Information System (RIS) file format can be used to exchange bibliographic data between many of them.

More importantly, there are several online systems such as *Mendeley* and *Zotero* which can capture bibliographic data directly from the web sites of libraries, publishers' pages, and online databases, and save it in BibTeX or RIS format.

7.4.2.4 Changing the layout

To change the title printed at the start of your 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 (BibTeX styles) are based on standard settings for journals and books from dozens of publishers: you just pick the one you want by name. The main `texmf/bibtex/bst` subdirectory of your installation contains the ones installed by default, and you can search on CTAN for others (look for `.bst` files). Many of them are named after university press or institutional styles (e.g. *harvard*, *ieeetr*, *oxford*) or the publisher or journal which specified them (e.g. *elsevier*, *kluwer*, etc.).

Some of them have an accompanying package (`.sty`) 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 may have fairly absolute requirements.

Publishers' 'house styles'

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 BibTeX (`.bst`) style files, although it uses a language of its own which really needs a computer science background to understand (one of the reasons that BibLaTeX was started). However, this is rendered unnecessary in most cases: there is an extensive program (actually written in 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 Law or the Humanities. In these fields, citations are usually done with author–year in parentheses, or short references (author/short-title/year) in a numbered footnote. In some disciplines the references themselves are actually called 'footnotes', to the exclusion of ordinary footnotes, although they are really references which happen by convention to be *displayed* as footnotes: an important distinction rarely appreciated by authors in these fields until they come to need a normal footnote.

For these disciplines, the bibliography at the back of the document is 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': a sufficient working (short) reference has already been provided in the citation or footnote; 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.

Jens Berger's 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 strongly recommended, but the author has now stepped down from maintaining it, although there is a large and active body of people using it who can help with support.

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 to initialise the index:

```
\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, in 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. 140 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 name alone (*without* a backslash or curly braces) such as `textbf` for bold-face text (see the index):

```
\index{beer!Rogue!Chocolate Stout|textbf}
```

Out of sequence The same method can be used as for font changes, but using the alternate index word instead of the font command name, so `\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
```

Editors usually have a button or menu entry for this, or may automate in the same way as they do BibTeX, as part of the 'build' procedure. Whichever way you run it, the program will look for a `.idx` file with the same name as your document, and output a `.ind` file with the sorted index in it. 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 with a space between letters of the alphabet. The Unix manual page[4] for the *makeindex* program has details of how to add letter headings to each alphabet group.

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 and by Mittelbach et al. duck the issue, but there is some documentation on `glotex` on CTAN.

[4] On GNU/Linux and Mac systems, just type the command `man makeindex`; the page is also available in many reference sites on the web: search for that command.

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 second 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*` environments 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 — if possible: it's not always achievable — and you can control the width of the gutter by setting the `\columnsep` length to a new dimension.

Multi-column work needs some skill in typographic layout, though: the narrowness of the columns makes typesetting less likely to fit smoothly because it's hard to hyphenate and justify well when there is little space to manœuvre in.

8 Fonts and layouts

This is the chapter that most users want first, because they come to structured documents from a wordprocessing environment where the *only* way to convey different types 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 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 1991 may look rather dated in 2011.

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, the half-title, copyright page, dedication, and other one-page preliminaries (if you use them) are usually designed individually, as the information on them only occurs once in that format anywhere in the document.
- The table of contents and other related lists like figures and tables all need to share one design.
- Longer 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 within the page there will be 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 in making a socio-political statement (*WIRED* 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.

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}
```

Bear in mind when using the geometry package that you only need to specify some of *either* the margins *or* the text height/width. Because it knows the paper size already, if you give it the text width and the left-hand margin, for example, it can work out the right-hand margin. A new feature introduced in 2010 is the **\newgeometry** command, which lets you reset the margin settings in mid-document. This isn't something you want to do very often, but until now it had been very difficult to do.

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. There are a lot of packages available to do various aspects of this, far too many to list here: search CTAN[1] to find what you need.

Changing the spacing of section headings for the whole document can be done with the sectsty or section packages, designed to let you adjust section-head spacing without having to know about the internal LaTeX coding, which is quite complex.

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* 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 commands `\smallskip`, `\medskip`, and `\bigskip`. These output flexible (dynamic, or 'rubber') space, approximately 3pt, 6pt, and 12pt high respectively, and 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 vertical space to remain and be taken into account even after a page break (very rare), use the starred variant `\vspace*`, e.g. `\vspace*{19pt}`.

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, $1^1/_3$ 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 probably wrong, and just don't understand that the world has moved on since the

[1] `http://ctan.org/search.html`

typewriter. Show them this paragraph and explain that they need to enter the 21st century and adapt to the features of computer 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 aware that it will not look pretty.

The space between lines is defined by the value of the length variable `\baselineskip` multiplied by the value of the `\baselinestretch` command. 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 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 normal 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 `\hspace{1in}` will insert a 1″ space like this in mid-paragraph. There are also some predefined (shorter) spaces available:

- `\thinspace` ($\frac{1}{6}$ em), which we saw between single and double quotes in the last paragraph of § 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` ($\frac{1}{2}$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 used as the width of a single digit in many 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 inter-letter spacing (incorrectly called 'tracking') as an aid to justification and *it is almost always wrong to do so* (and looks it). While it *is* of course possible to change letterspacing in LaTeX (with the soul package), 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 book.[2]

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 one of the following arguments.

plain for a page number centered at the bottom — this is the default;

empty for nothing at all, not even a page number — use this when you are doing one-page documents like posters or handouts, where a page number is not meaningful;

headings for running heads based on the current chapter and section — this is common for articles, books, and reports, so that every page is identifiable even if printed or copied separately;

myheadings which lets you use your own [re]programmed definitions of how to use the `\markright` and `\markboth` commands, 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.

However, the easiest way to get specialist running heads is to use the fancyhdr package, which lets you redefine the left-hand, centre, and right-hand page headers and footers for both odd-numbered and even-numbered 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 typeset version of this document can be used as an example: for the whole story you have to read the documentation.

[2] This does not apply for the German technique in blackletter type of using letter-spacing instead of (non-existent) italics. The defaults in the soul package were designed to cater for this.

```
\pagestyle{fancy}\fancyhead{}
\renewcommand\headrulewidth{.1pt}
\fancyhead[LO,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[LO,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 common requirements:

1. Settings are prefixed by making the **\pagestyle** 'fancy' and setting the **\fancyhead** to null to zap any predefined values.

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 the Times you find in wordprocessors, 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 was designed using METAFONT, the font-drawing program made by Knuth to accompany TEX systems, but it is now also available in Type 1 and TrueType formats.

Web browser fonts

If you are reading this in a web browser, the above paragraph is only a low-resolution copy because browsers don't usually have the Computer Modern font available.

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 'Adobe 35' typefaces which are built into laser printers and other DTP systems, and some more fonts donated by the X Consortium. Plus, of course, standard LATEX can use any of the thousands of Type 1 fonts available, and *pdfLATEX* can use any of the thousands of TrueType fonts as well.

XETEX

Jonathan Kew's XETEX (see the Preface on p. xiii) attempts to provide transparent access to any font of any format on your computer, making conversion and separate font installation unnecessary.

In the following lists, if there is a package available, its name is given in parentheses after the name of the typeface. The font-family name is shown on the right-hand side. If a non-standard font-encoding is needed, its name is shown before the font-family name.

Latin-alphabet typefaces (METAFONT)

Computer Modern Roman	`cmr`
The quick brown fox jumps over the lazy dog	
Computer Modern Sans	`cmss`
The quick brown fox jumps over the lazy dog	
Computer Modern Typewriter	`cmtt`
The quick brown fox jumps over the lazy dog	
Pandora (pandora)	`panr`
The quick brown fox jumps over the lazy dog	
Pandora Sans	`pss`
The quick brown fox jumps over the lazy dog	
Pandora Typewriter	`pntt`

The quick brown fox jumps over the lazy dog

Universal	uni
The quick brown fox jumps over the lazy dog	
Concrete (ccr)	ccr
The quick brown fox jumps over the lazy dog	
Éireannach (eiad)	eiad
Níl aon cineáil mar do chineáil féin	
Rustic	rust
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG	
Uncial	uncl
The quick brown fox jumps over the lazy dog	
Dürer	cdr
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG	
Fraktur	U yfrak
Fuchs, Du haſt die Gans geſtohlen, gib ſie wieder her!	
Gothic	U ygoth
Ꝟt it pleſe any man ſpirituel or temporel	
Schwäbische	U yswab
Fuchs, Du haſt die Gans geſtohlen, gib ſie wieder her!	

Latin-alphabet typefaces (PostScript Type 1) from Adobe

Avant Garde (avant)	pag
The quick brown fox jumps over the lazy dog	
Bookman[3] (bookman)	pbk
The quick brown fox jumps over the lazy dog	
Courier (courier)	pcr
The quick brown fox jumps over the lazy dog	
Helvetica (helvet)	phv
The quick brown fox jumps over the lazy dog	
New Century Schoolbook[4] (newcent)	pnc
The quick brown fox jumps over the lazy dog	
Palatino[5] (palatino)	ppl
The quick brown fox jumps over the lazy dog	
Symbol	U psy
Τηε θυιχκ βροων φοξ φυμπσ οωερ τηε λαζψ δογ	
Times New Roman[6] (times)	ptm
The quick brown fox jumps over the lazy dog	
Zapf Chancery (zapfchan)	pzc
The quick brown fox jumps over the lazy dog	
Zapf Dingbats (pifont)	U pzd
✶❋✵ ❑◆❉❋✷ ✲❒❑◗■ ✵❑❙ ❊◆❍❒▲ ❑❖✵❒ ▼❋✵ ●✵❚❙ ✵❑✺	

3 Uses Avant Garde as the sans-serif and Courier for monospace.
4 Uses Helvetica as the sans-serif font and Courier for monospace.
5 Uses Avant Garde as the sans-serif and Courier for monospace.
6 Uses Helvetica as the sans-serif font and Courier for monospace. Mathematical symbols for Times are available both free and commercially.

As mentioned in §4.4, the 'Adobe 35' fonts can be used with any printer, not just a laser printer or typesetter. The *Ghostscript* interpreter and the *GSview* viewer come with a large set of printer drivers, so you just create PostScript output and print from *GSview*.

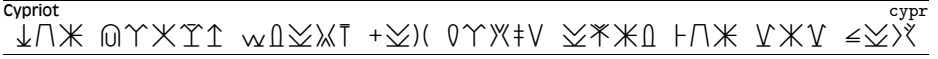
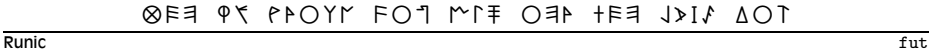
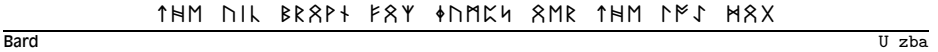
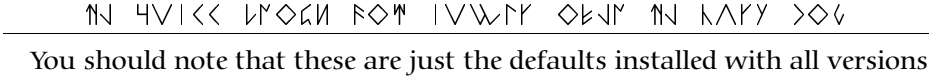
Incidentally, the 35 refers to the total number of fonts for the 10 typefaces, including their bold, italic, and bold-italic variants.

Postscript Type 1 fonts have been the mainstay of the graphic arts industries for many years, as they allow much better definition of variance ('hinting') than most other formats. However, the font format remains proprietary to Adobe, even though they have released it for public use, which means they could change it without warning. A new format called 'OpenType' is designed to overcome this, and some versions of TEX are already able to use OpenType fonts.

Latin-alphabet fonts (PostScript Type 1) from the X Consortium

Charter (charter)	bch
The quick brown fox jumps over the lazy dog	
Nimbus Roman	unm
The quick brown fox jumps over the lazy dog	
Nimbus Sans	unms
The quick brown fox jumps over the lazy dog	
URW Antiqua	uaq
The quick brown fox jumps over the lazy dog	
URW Grotesk	ugq
The quick brown fox jumps over the lazy dog	
Utopia[7] (utopia)	put
The quick brown fox jumps over the lazy dog	

Non-Latin-alphabet typefaces (METAFONT)

Cypriot	cypr
	
Etruscan	etr
T he qik bro n fo x m ps o ver the lazy do g	
Linear 'B'	T1 linb
	
Phoenician	phnc
	
Runic	fut
	
Bard	U zba
	

You should note that these are just the defaults installed with all versions of LATEX. There are hundreds more listed in Palle Jørgensen's comprehensive

7 Removed from recent distributions as it is not free.

What types of font?

Just to make it clear: standard LaTeX uses only METAFONT and PostScript Type 1 fonts. pdfLaTeX can use TrueType fonts as well, but needs special configuring to do so. XeLaTeX can use almost any type of font you have installed.

LaTeX Font Catalog published by the Danish TeX Users Group at `http://www.tug.dk/FontCatalogue/`.

8.2.1 Changing the default font family

LaTeX expects to work with three font families as defaults:

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

The start-up default for LaTeX equates the `rm` default with the `cmr` font-family (Computer Modern Roman), `sf` with `cmss`, and `tt` with `cmtt`. If you use one of the packages listed in the tables on pp. 133–135, it will replace the defaults of the same type: for example, the bookman package makes the default `rm` font-family Bookman (`pbk`), but leaves the sans-serif (`sf`) and monospace (`tt`) families untouched. Equally, the package helvet changes the default sans-serif family to Helvetica[8] but leaves the serif (Roman) and monospace families 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, but they are for text work only, as they leave mathematics in Computer Modern:

times changes to Times/Helvetica/Courier.

[8] The Helvetica typeface family has a notoriously large x-height, making it hard to match with other typefaces at the same nominal size. The helvet package therefore has a scaled option that lets you reduce the optical size slightly so that the font sits more easily with others: `\usepackage[scaled=0.86]helvet`, for example.

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.[9]

newcent changes to New Century Schoolbook/Helvetica/Courier.

palatino changes to Palatino/Avant Garde/Courier.

If you use mathematics, there are two fairly complete implementations of non-CM typefaces in the mathptmx (Times) and mathpazo (Palatino) packages. The whole business of math fonts is perpetually under revision in any case, as mathematicians are perpetually inventing new symbols, which take a while to appear in typefaces. The American Mathematical Society (AMS) and other organisations are involved with a project called Styx, which is expected eventually to define the complete suite of mathematical characters in a rational and extensible manner — but don't hold your breath.

Where no package name is given in the tables on pp. 133–135, it means the font is rarely used as a default by itself except in special cases like users' own homebrew packages. To use such a font you have to specify it manually, or make a little macro for yourself if you use it more than once.

8.2.2 Changing the font-family temporarily

To shift to another font family on a temporary basis, use the commands `\fontencoding` (if needed), `\fontfamily`, and `\selectfont`, and *enclose the commands **and** the text in curly braces*.

> **Grouping**
>
> This is a different way of using curly braces to how we have used them before: see the panel 'Grouping' on p. 138 — It limits the effect of a simple command (see § 2.3.1) to the material inside the braces.

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).[10]

In a normal document, of course, random typeface changes like this are rather rare. You select your typeface[s] once at the start of the document, and stick with them.

[9] The pslatex package is also said to be outdated by some experts because it implements 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 narrow Courier.

[10] Test for the observant reader: in what typeface will the colon (:) in the example be set?

```
{\fontfamily{phv}\selectfont
  Helvetica looks like this}:
{\fontencoding{OT1}\fontfamily{bch}\selectfont
  Charter looks like this}.
```

Helvetica looks like this: Charter looks like this.

Grouping

Note carefully this use of curly braces to restrict the scope of a change rather than delimit the argument to a command. This is called a **group**, and it makes the effect any changes made inside the braces local, so that they do not interfere with the text following. Any changes to fonts or other values made within the curly braces cease when the closing curly brace is processed. If you use a paragraphic formatting command like **`\centering`**, **`\flushleft`**, or **`\flushright`** in a group, you must end the text with a **`\par`** to finish the paragraph, otherwise the formatting will not take effect. If you use an environment like `center`, `flushleft`, or `flushright`, you do not need the **`\par`** because environments are inherently paragraphic and will do it for you.

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, Inc., 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 name), and you don't need to edit your document text at all! What's more, a macro could do other things at the same time, like add an entry to an index of products.

Table 8.1: Typeface styles, families, shapes, and series

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

However, vastly more common are changes to type *style*, while staying with the same font-family.

8.2.3 Changing font style

Within each typeface or font family there are usually several different 'looks' to the type design. LaTeX distinguishes between *font family*, *font shape*, and *font series* (see Table 8.1).

> **Commands without arguments**
>
> Remember back in § 2.3.1 seeing simple commands with no arguments? The commands above affect all following text, so to restrict their effect, group them in curly braces along with the text you want affected.

These 'shape', 'series', and 'family' commands are *commutative*, so you can combine a shape with a series and/or a family, without the need to use `\selectfont`:

```
This gives you {\bfseries\itshape\sffamily bold italic
sans-serif type}, but beware
```

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.[11]

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

Type style	Command	Example
Italic	`\textit{text}`	puts *text* into italics
Slanted	`\textsl{text}`	puts $text$ into slanted type*
Small Capitals	`\textsc{text}`	puts TEXT into small caps
Bold	`\textbf{text}`	puts **text** into bold type
Sans-serif	`\textsf{text}`	puts text into sans-serif type
Monospace	`\texttt{text}`	puts `text` 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}}...
```

Underlining isn't a font, and it is extremely rare in typography except for special purposes. If you think you need it, use the ulem package with the `normalem` option, and the **`\uline`** command.

8.2.4 Font sizes

LaTeX 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.

[11] 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 way outside the scope of this document.

As an example, here's 12pt Computer Modern, and here's 5pt Computer Modern scaled up to 12pt, and here's 17pt Computer Modern scaled down to 12pt 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, which is the normal size for most texts. Using the larger defaults (11pt and 12pt) for the body font will use 11pt and 12pt designs, with other sizes (eg headings) resized to match. 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.

Command	Example	Nominal point size	Exact point size
`\tiny`	The quick brown fox jumps over the lazy dog	5	5
`\scriptsize`	The quick brown fox jumps over the laz	7	7
`\footnotesize`	The quick brown fox jumps over the l	8	8
`\small`	The quick brown fox jumps over th	9	9
`\normalsize`	The quick brown fox jumps over	10	10
`\large`	The quick brown fox jumps	12	12
`\Large`	The quick brown fox ju	14	14.40
`\LARGE`	The quick brown fo	18	17.28
`\huge`	The quick brown	20	20.74
`\Huge`	The quick bro	24	24.88

Commands without arguments (again)

The commands above affect all following text, so to restrict their effect, group them in curly braces along with the text you want affected.
The inter-line spacing gets reset in proportion, but only gets applied at the end of the pararaph, so if you have a whole paragraph or list item to be set in Large type, there needs to be a **`\par`** before the closing curly brace, otherwise the default line-spacing will be used.

While these 'shorthand' commands relieve the beginner of having to worry about the 'right' size for a given task, when you need a specific size you can use the fix-cm package to override the step sizes, and use the **`\fontsize`** command:

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

'Leading' comes from the old metal-type practice of adding a lead strip between lines to increase the spacing.

Ordinal superscripts

Don't use them in normal text. Superscripted ordinals (like 21st) are a historical relic of Victorian typography. They are unnecessary and are never used in modern professional typesetting. They were re-introduced by Microsoft Word apparently because some Americans like their wordprocessing to look old-fashioned.

If you want to try and mimic low-quality wordprocessing, or if you are trying to make a typographic fac-simile of Victorian typsetting, use the **`\textsuperscript`** command from the textcomp package. Do not use math mode for text superscripts.

The **`\fontsize`** command 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 the lines).

Step-sized fonts

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, LaTeX will pick the closest step instead. Many other fonts follow this pattern: if you need to use fonts at arbitrary other sizes there is a package fix-cm which lets you override the default steps.

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	Effect
Foreign words	*ex officio*
Scientific names	*Ranunculus ficaria*
Emphasis	*must not*
Titles of documents	*The LaTeX Companion*
Product names	Corel's *WordPerfect*
Variables in maths	$E = mc^2$
Subtitles or headings	*How to get started*
Use of a letter as a word	Who knocked the *L* out of London?
Decoration	*FREE UPGRADE!!!*

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.

LaTeX 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 the 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 the
```

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.

Warning from the past

Beware of this 'vaine conceipt of simple men, which judge things by ther effects, and not by ther causes'. (Edmund Spenser, 1633)

It's hugely more efficient to have control of the cause than the effect.

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

8.2.6 Colour

You can typeset anything in LaTeX in any colour you want using the xcolor package. First, you need to add the command `\usepackage{xcolor}` 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 (computer and television screens), and cyan, magenta, yellow, and 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, xcolor provides a 64-colour palette of predefined *color names*. These represent approximately the colours in the big box of 64 *Crayola* colouring pencils much favoured by artists and designers. To get this palette, use `\usepackage[dvipsnames]xcolor`.

Now if you want the *Crayola* colour Crimson, you can use it as a colour name:

```
\color{Crimson}
\textcolor{Crimson}{some red text}
```

The biggest selection of predefined colours is available with the svgnames option: this defines all the colours defined by the Scalable Vector Graphics (SVG) standard for web graphics (so it's good for compatibility).

As some of the predefined colour names are quite long, you can create a short name of your own for colours you use frequently, using the `\definecolor` command:

```
\definecolor{mb}{named}{MidnightBlue}
```

The `\definecolor` command needs three arguments: your shorthand name, the name of the colour model, and the colour specification. In the case of the `named` model, the last argument is one of the colour names specified by the option you loaded the package with. To use these names with *pdfLaTeX*, you may need to use the pdftex option to the xcolor package, depending on the version of TeX you have installed.

Using the `\definecolor` command, you can define any colour you want by giving it a name, specifying which colour model, and providing the Red-Green-Blue (RGB) or Cyan-Magenta-Yellow-Black (CMYK) colour values *expressed as decimal fractions, separated by commas*. For example, an RGB shade given as (37,125,224) in decimal (#250FE0 in hexadecimal as used on the Web) can be given as:

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

(To get the fractional value, divide the integer decimal value by 255, the maximum for each of the hues in the Red-Green-Blue colour model.) You can then use `\textcolor` with your new colour name: midblue looks like this if you're reading in colour.

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

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

However, combining colours is an art and a skill: using conflicting colours like Magenta on midblue is as good a warning as any to learn about colour models and palettes before trying to use them!

8.3 Installing new fonts

Different fonts come in a variety of packagings: the three most common used with TeX systems are PostScript fonts, TrueType fonts, and METAFONT fonts. How you install them and where they go depends on

Directories (Folders)

In the examples below, all the folders (directories) are assumed to be in your Local TEX Directory unless otherwise explicitly given. See Chapter 1 for how to find this out.

how you installed LATEX: all I can deal with here are the standard locations within the TDS.

Typefaces come supplied as one or more font 'outline' files and a number of ancillary files:

METAFONT **typefaces** have a number of `.mf` source (outline) files, possibly also some `.fd` (font definition) files and a `.sty` (style) file. The `.tfm` (TEX font metric) files are not needed, as they can be generated from the outlines.

PostScript typefaces come as a pair of files: a `.pfb` (PostScript font binary) or `.pfa` (PostScript font ASCII) outline, and an `.afm` (Adobe font metric) file. There may also be `.inf` and other files but these are not needed for use with TEX systems.

TrueType typefaces are a single `.ttf` file, which combines outline and metrics in one.

The instructions here assume the use of the New Font Selection Scheme (NFSS) used in LATEX 2ε. If you are running the obsolete LATEX 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 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 `fonts/source/public/`:
2. Copy all the `.mf` files to this directory.
3. Copy the `.fd` (Font Definition) file[s] to your `tex/latex/mfnfss` directory.
4. Copy the `.sty` (style) file to a subdirectory (create it) of `tex/latex` named after the typeface.

5. Run your TEX indexer program (see step 4 in the procedure on p. 82).

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.

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 the last paragraph of § 5.2.1 and step 2 in the procedure on p. 81).

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. 153).

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 may be true for some cases, standard LATEX 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 use PostScript equipment rather than PDF. In addition, operating system support for scalable fonts is still very poor on Unix systems (including GNU/Linux), despite the advances in recent years, and many rebranded ('knock-off' or pirated) TrueType fonts supplied with other systems are of very poor quality, so in many cases it still makes sense to use TEX's built-in support for PostScript. When XETEX becomes the default processor, it will no longer be necessary to install different types of font files separately, as XETEX is able to use all fonts natively from your system's font folder.

Two files are needed for each font: the `.afm` Adobe Font Metric (AFM) and the `.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 (PFA) files, it may be possible to generate the `.pfb` files using the *t1binary* program from the *t1utils* suite (see `http://gnuwin32.sourceforge.net/packages/t1utils.htm`) or the excellent *FontForge* font editor (from `http://fontforge.sourceforge.net`). There are unfortunately still some companies distributing Type 1 fonts in `.pfa` format (Mathematica is one reported recently).

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

2. **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 on your computer for more details — lists of the codes used are in the files `supplier.map`, `weight.map`, `width.map`, `variant.map`, and the various `.map` files for each foundry):

Letter	Meaning	Examples
`f`	foundry	`b`=Bitstream, `m`=Monotype, `p`=Adobe
`nn`	typeface	`ba`=Baskerville, `tm`=Times, `pl`=Palatino
`ss`	series/shape	`r`=roman, `b`=bold, `i`=italic, etc.
`ee`	encoding	`8r`=revised TeXBase1, `ly`=Y&Y's TeX'n'ANSI
`v`	variant	smallcaps, outline, script, etc.

   The `fonts/fontname` directory in your main (*not* local) installation directory of TeX has files for several foundries giving fully-formed names like these for common fonts (e.g. `ptmr8r` is [Adobe] **P**ostScript **Tim**es **R**oman in an **8**–bit revised TeX encoding; `bgslly` is **B**itstream **G**ill **S**ans **L**ight in Y&Y's TeX'n'ANSI encoding [**LY**1]).[12] Read the documentation in *Fontname: Filenames for TeX fonts* to find out how to make up your own short names if the foundry and font you want is not shown in the lists in the `fonts/map/fontname` directory.

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

   While the `font/map/fontname` directories have ready-made maps 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**
   Encoding is needed because Adobe fonts store their characters in

[12] Confusingly, Bitstream fonts (and others from similar sources) mostly have different names from the original fonts, to avoid copyright issues, 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.

different places to the TEX standard. This is what tripped me up the first few times until someone pointed me at Y&Y's[13]TEX'n'ANSI encoding which at the time was the only one that included the glyphs I want where I expected them to be.[14] Now, however, I recommend using the 8r encoding for all PostScript fonts. The encoding vector file `8r.enc` should be in your main (*not* local) TEX installation directory in `fonts/enc/dvips/base`.

To avoid having to type the long path each time below, just copy this file to the temporary directory where you're doing all this stuff.

4. **Convert `.afm` files to `.tfm`**
   The *afm2tfm* and *vptovf* programs are standard TEX utilities in the `bin` directory of your *main*TEX 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 8r (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.

[13] Y&Y, Inc has ceased trading and their TEX distribution is not longer available, although there is email support at `http://lists.ucc.ie/lists/archives/yandytex.html`, and their encoding files continue to be used.

[14] The only one I had problems with is 'Å', which for some weird reason isn't catered for in this encoding.

6. **Create the virtual font**
   Turn the `.vpl` files into `.vf` and `.tfm` pairs. LaTeX needs these to convert from Adobe's encoding to its own.

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

7. **Make directories to hold the files**
   Under your local directory there should be a `fonts` directory, and in it there should be `afm`, `tfm`, `type1`, and `vf` subdirectories. 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). On my computer, this means:

```
cd /usr/local/share/texmf/fonts mkdir -p afm/zfonts/zork
mkdir -p tfm/zfonts/zork mkdir -p type1/zfonts/zork
mkdir -p vf/zfonts/zork cd /tmp
```

   Or if you're lazy:

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

   For Microsoft Windows users, you may have to do this in the *MikTeX* directories.

   The `mkdir` command exists in Windows as well, but 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 \textsl{\uline{localdir}}/fonts/afm/zfonts/zork/
cp *.tfm \textsl{\uline{localdir}}/fonts/tfm/zfonts/zork/
cp *.pfb \textsl{\uline{localdir}}/fonts/type1/zfonts/zork/
cp *.vf \textsl{\uline{localdir}}/fonts/vf/zfonts/zork/
```

where *localdir* is the prefix of your local TEX installation directory. 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.[15] There should be a personal configuration file for your use of LATEX called `10local.cfg`, in your login folder in a subfolder called `.texmf-config/updmap.d`. All it needs is an entry for our new font, using the three-letter font family abbreviation (the first three letters of the Karl Berry fontname (here '`zor`'):

```
Map zor.map
```

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

(a) Open `10local.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 \textsl{\uline{localdir}}/dvips/config/zfonts
```

(b) Use your editor to open (create) the file `dvips/config/zfonts/zor.map`.

(c) Insert the line:

[15] The configuration file for *dvips* is `config.ps` and there are others for other drivers (e.g. PDF): they get their entries from the program *updmap* which reads map files for each typeface. The configuration file for *updmap* is `updmap.cfg`, and that in turn is updated by the program *updmap-sys*. But hopefully you won't need to know that.

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

(d) Save and close the file.

You get the full font name (here, 'Zork-Blackface') from the `zork.id` which was created back in step 4 in the procedure on p. 149 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**
LaTeX 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 LaTeX 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 your local `tex/latex/psnfss` directory.

(b) Insert the following lines:

```
% foozork - created from foo for Zork
\def\fileversion{1.0}
\def\filedate{2010/12/03}
\def\docdate{2010/12/03}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{foozork}[\filedate\space\fileversion\space
 Zfonts Ordinary PSNFSS2e package]
\RequirePackage[T1]{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.).
- The `T1` option to the fontenc package is the name used for the interface to the 8r 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 that 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{T1}\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 `eefnn.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 `t1zor.fd` for the T1 (8r) encoding and the `zor` short font name.

(a) Use your editor to open (create) `tex/latex/psnfss/t1zor.fd`

(b) Enter the following lines:

```
\ProvidesFile{t1zor.fd}[2010/03/03 v0.1 manual
              font definitions for LY1/zor.]

\DeclareFontFamily{T1}{zor}{}

\DeclareFontShape{T1}{zor}{k}{n}{<-> zorkly}{}
\DeclareFontShape{T1}{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. 149; 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 414–15 of the *The LaTeX Companion* and should also be in your copies of `fonts/map/fontname/weight.map` and `fonts/map/fontname/width.map`. The rules for combining weight and width need care: read the documentation for the fontname package. There is no `shape.map` in fontname because it's not part of font file names, it's purely a LaTeX 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 `<->` in the 5th argument in the `\DeclareFontShape` command means that all sizes are to 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 TEX indexer program (see step 4 in the procedure on p. 82) so that *updmap* can find the files it needs.

    ```
    texhash
    ```

    (or whatever your installation calls it). Then run *updmap* to add the map file:

    ```
    updmap --enable Map=zor.map
    ```

    This updates the maps for *dvips*, *pdfLATEX*, and others. Finally, run the TEX indexer program again so that it can find the map in its new location.

    ```
    texhash
    ```

Now you can `\usepackage{foozork}` in your LATEX file to make it the default font. To use the font incidentally instead of as the default, you can use the commands you added at the end of step 10 in the procedure on p. 152:

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

8.3.3 The LATEX font catalogue

The LATEX Font Catalog is a web site created and maintained by Palle Jørgensen at `http://www.tug.dk/FontCatalogue/`. It lists over 200 typefaces for use with LATEX, many of them available nowhere else, with samples and links to the directories on CTAN where you can download them. You can ~~waste~~ spend many fascinating hours downloading and installing them and trying them out in your documents.

For newcomers, installing a new typeface can appear challenging, when described as I have done for Postscript fonts. But the typefaces in the

LaTeX Font Catalog are prebuilt for LaTeX, so all you have to do is download the `.zip` file, unzip it into your personal TeX directory, and move the subdirectories into the right places. A worked example is the best way to describe this.

Let's suppose we want to install the sans-serif Kurier typeface. This is nothing to do with the Courier typewriter face, but was designed in pre-computing times by Ma[a]gorzata Budyta, and digitized and extended by Janusz M Nowacki (thanks to the GUST web site[16] for this information).

1. If we click on the name in the sans-serif page of the Catalog[17], we can see a sample paragraph, and we can click on the link at the bottom of the page[18] to go to the CTAN directory where the typeface is stored.

2. Here there is a brief `README` file, and links to the individual font subdirectories, but most importantly there is a link at the top of the page to the `.zip` file containing it all. Download this and unzip it straight into your personal TeX directory (see § 1.2 for what this is and where to ceate it).

3. Now open your personal TeX directory in your directory browser (Windows: *My Computer* or just *Computer*; Mac: *Finder*; Linux: *nautilus* or *dolphin*). You will see that inside the `kurier` directory there are subdirectories called `doc`, `fonts`, and `tex` (see Figure 8.1).

4. Drag and drop each of those subdirectories into your Personal TeX directory (`texmf`). If directories with the same names already exist, your system will probably ask if you want the new ones merged with the existing ones: the answer is yes, so click OK. In some typefaces there may be more subdirectories than shown here for Kurier: do the same with them all.

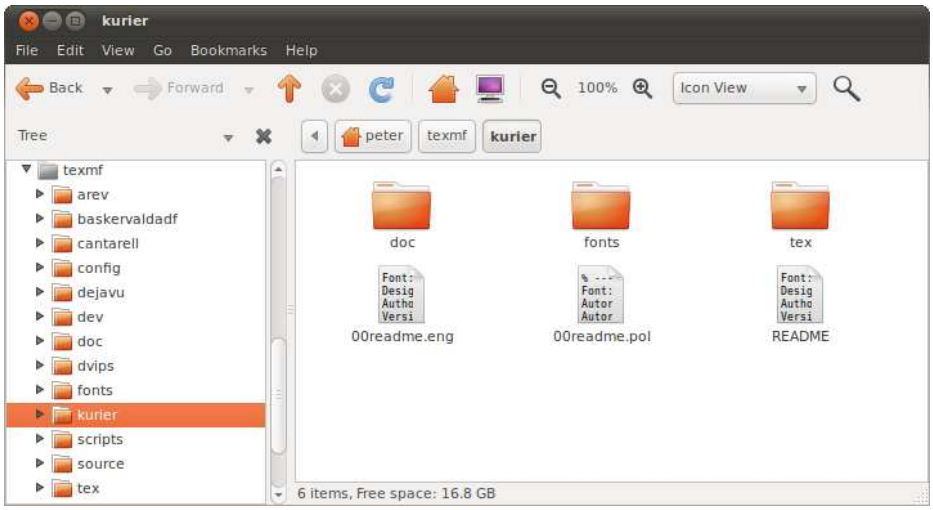
5. The next step is to find the `.map` file that LaTeX needs to set up the link between its short (KB) font names and the long ones used by PDF and Postscript output. This will be in the `fonts/map/dvips/`*`fontname`* subdirectory (see Figure 8.2).

   In this case there are many, but the one we want is just called `kurier.map` (the others are there in case you only wanted to install a single font, not the whole typeface).

   Carefully note down the directory path to this file (in my case, `~/texmf/fonts/map/dvips/kurier/kurier.map`).

16 `http://www.gust.org.pl/projects/e-foundry/kurier-iwona`
17 `http://www.tug.dk/FontCatalogue/sansseriffonts.html`
18 `http://www.ctan.org/tex-archive/fonts/kurier/`

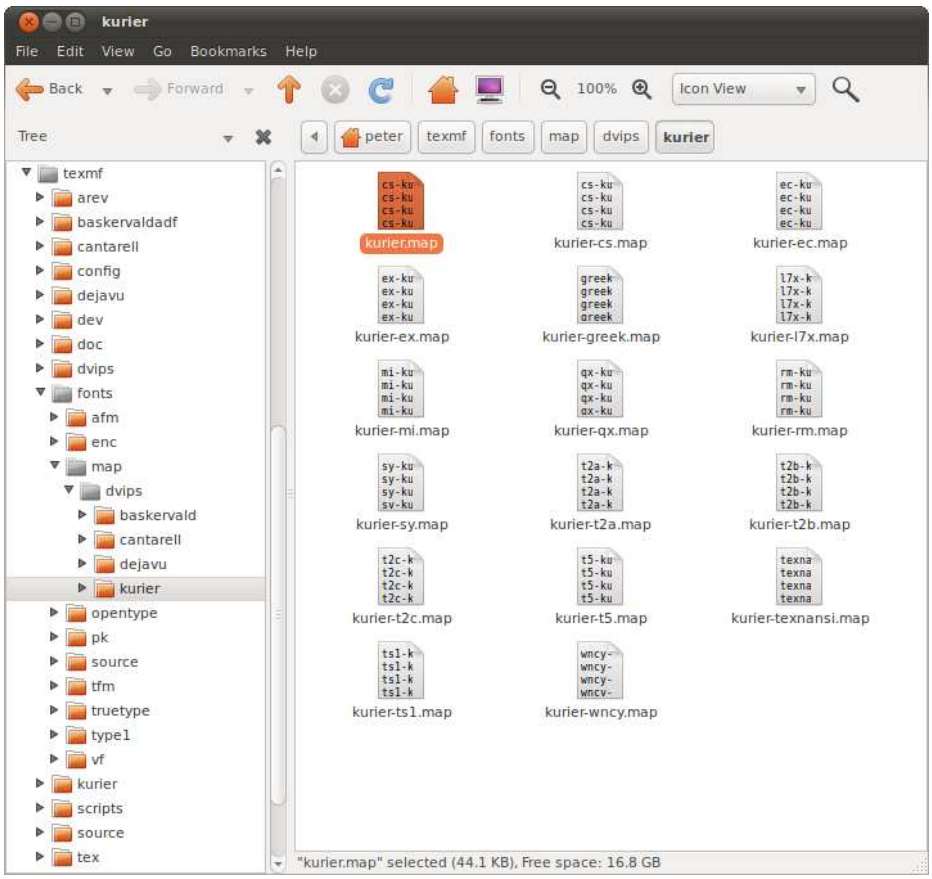
Figure 8.1: Layout of a font zip file downloaded from CTAN



6. The last step is to run the font map update program *updmap* to enable use of the map file. You need to do this in a terminal or command window, by typing the command . This reloads all your font maps, so it takes a few minutes to run.

Once that's done, you can `\includepackagekurier` in your documents and start using the typeface.

Figure 8.2: Location of the map file for a typeface downloaded from CTAN



9 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 immediately before the `\begin{document}` in the sample file in the first listing, and remove the `\color{...}` commands from the title, author, and date. Re-run the file through LaTeX, and you should get something like this:

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;[1]
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;

[1] 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** to put product names into italics, keywords into bold, and so on. Here's an example for a command **\tmproduct**, which also indexes the product name and adds a trademark sign:

```
\newcommand{\tmproduct}[1]{%
        \textit{#1}\texttrademark%
        \index{#1@\textit{#1}}%
}
```

If I now type `\tmproduct{Velcro}` then I get *Velcro*™ 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. 124 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 LaTeX reads the macro, I ended each line with a comment character (%). LaTeX normally treats newlines as spaces when formatting (remember the list on p. 34 in §2.4.1), so this stops the end of line being turned into an unwanted space when the macro is used. LaTeX usually 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 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 Donald 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`[2]): 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.

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).

[2] 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 9.1: 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 LaTeX's requirements on spaces after command names ending with a letter is to follow such commands with an empty set of curly braces **{}**.

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 LaTeX's default document structure for the Report document class uses Chapters as the main unit of text, whereas in reality most reports are divided into Sections, not Chapters (§ 6). The result of this is that if you start off your report with `\section{Introduction}`, it will print as

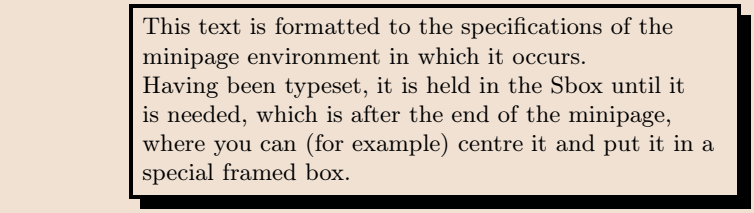
0.1 Introduction

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 the last paragraph of § 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 LaTeX's internal way of referring to counters). You can redefine this in your preamble to simply leave out the reference to chapters:

```
\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)
align it and put it in a special framed box.
\end{minipage}
\end{Sbox}
\begin{center}
\shadowbox{\TheSbox}
\end{center}
```




```
\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 LaTeX'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* (LaTeX'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* (➠ § 4), 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).

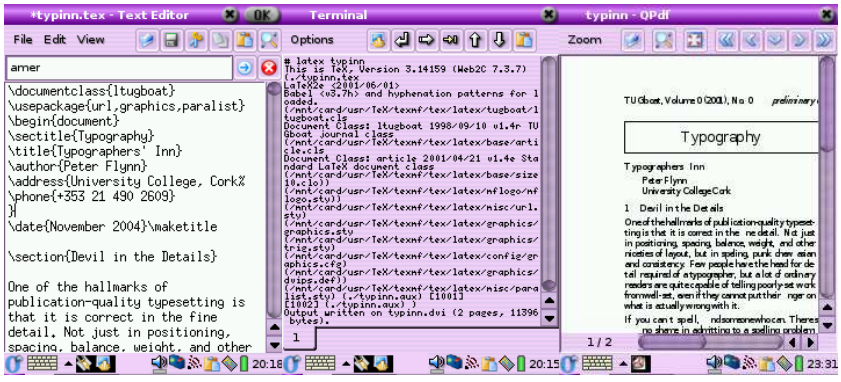
In this case, we are using the `\raisebox` command within the redefinition because it turns out that the symbols in this font are positioned slightly too high for the typeface we're using. The `\raisebox` command takes two arguments: the first is a dimension, how much to raise the object by (and a negative value means 'lower': there is no need for a `\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 LaTeX can use).

There is a vast number of symbols available: see *A comprehensive list of symbols in TeX* for a comprehensive list.

10 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, see Figure 10.1) to the biggest supercomputer.

Figure 10.1: LaTeX editing and processing on the Sharp Zaurus 5500 PDA



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 probably be edited, stored, and maintained in something neutral like the Extensible Markup Language (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 *pdfLaTeX* 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 — although there are several ways to minimise this loss. Similarly, converting other formats into LaTeX often means editing back the stuff the other formats omit because they only store appearances, not structure.

However, there are at least two excellent systems for converting LaTeX directly to HyperText Markup Language (HTML) so you can publish it on the web, as we shall see in § 10.2.

10.1 Converting into LaTeX

There are several systems which will save their text in LaTeX format. The best known is probably LyX, which is a wordprocessor-like interface to LaTeX for Windows and Unix. Both *AbiWord* (Linux and Windows) and *Kword* (Linux) have a very good **Save As**... LaTeX output, and *OpenOffice* (all platforms) has a LaTeX plugin, so they can be used to open Microsoft *Word* documents and convert to LaTeX. Several maths packages like the *EuroMath* editor, and the *Mathematica* and *Maple* analysis packages, can also save material in LaTeX format.

In general, most other wordprocessors and DTP systems either don't have the level of internal markup sophistication needed to create a LaTeX file, or they lack a suitable filter to enable them to output what they do have. Often they are incapable of outputting any 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 Save As... menu item to save the wordprocessor file as HTML, rationalise the HTML using Dave Raggett's *HTML Tidy*[1], and convert the resulting Extensible HyperText Markup Language (XHTML) to LaTeX with any of the standard XML tools (see below).

- Use a specialist conversion tool like EBT's *DynaTag* (supposedly available from Enigma, if you can persuade them they have a copy to sell you; or you may still be able to get it from Red Bridge Interactive[2] in Providence, RI). It's expensive and they don't advertise it, but for bulk conversion of consistently-marked *Word* files into usable XML it beats everything else hands down. The *Word* files *must* be consistent, though, 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 scanned or retyped into XML or LaTeX. There are hundreds of companies from India to Polynesia who do this at high speed and low cost with very high accuracy. It sounds crazy when the document is already in electronic form, but it's a good example of the problem of low quality of wordprocessor markup that this solution exists at all.

You will have noticed that most of the solutions lead to one place: XML. As explained above and elsewhere, this format is the only one so far devised capable of storing sufficient information in machine-processable, publicly-accessible form to enable your document to be recreated in multiple output formats. Once your document is in XML, there is a large range of software available to turn it into other formats, including LaTeX. Processors in any of the common XML processing languages like the Extensible Stylesheet Language [Transformations] (XSLT) or *Omnimark* can easily be written to output LaTeX, and this approach is extremely common.

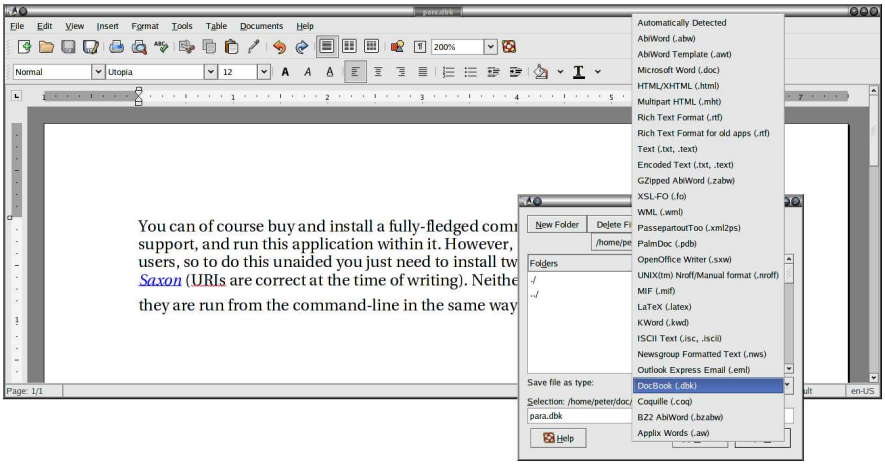
Much of this would be simplified if wordprocessors supported native, arbitrary XML/XSLT as a standard feature, because LaTeX output would become much simpler to produce, but this seems unlikely.

However, the native format for both *OpenOffice* and *Word* is now XML. Both `.docx` and `.odf` files are actually Zip files containing the XML document together with stylesheets, images, and other ancillary files. This means that for a robust transformation into LaTeX, you just need to write an XSLT stylesheet to do the job — non-trivial, as the XML formats used are extremely complex, but certainly possible.

Among the conversion programs for related formats on CTAN is Ujwal Sathyam and Paul DuBois's *rtf2latex2e*, which converts Rich Text Format (RTF) files (output by many wordprocessors) to LaTeX 2$_\varepsilon$. The package

1 `http://tidy.sourceforge.net/`
2 `http://www.rbii.com/`

Figure 10.2: Sample paragraph in *AbiWord* converted to XML



description says it has support for figures and tables, equations are read as figures, and it can the handle the latest RTF versions from Microsoft Word 97/98/2000, StarOffice, and other wordprocessors. It runs on Macs, Linux, other Unix systems, and Windows.

10.1.1 Getting LaTeX out of XML

Assuming you can get your document out of its wordprocessor format into XML by some method, here is a very brief example of how to turn it into LaTeX.

You can of course buy and install a fully-fledged commercial XML editor with XSLT support, and run this application within it. However, this is beyond the reach of many users, so to do this unaided you just need to install three pieces of software: *Java*[3], *Saxon*[4] and the DocBook 4.2 DTD[5] (URIs are correct at the time of writing). None of these has a visual interface: they are run from the command-line in the same way as is possible with LaTeX.

As an example, let's take the above paragraph, as typed or imported into *AbiWord* (see Figure 10.2). This is stored as a single paragraph with highlighting on the product names (italics), and the names are also links to their Internet sources, just as they are in this document. This is a convenient way to store two pieces of information in the same place.

3 `http://java.com/download/`
4 `http://saxon.sourceforge.net/`
5 `http://www.docbook.org/xml/4.2/index.html`

AbiWord can export in DocBook format, which is an XML vocabulary for describing technical (computer) documents — it's what I use for this book. *AbiWord* can also export LaTeX, but we're going make our own version, working from the XML (Brownie points for the reader who can guess why I'm not just accepting the LaTeX conversion output).

Although *AbiWord*'s default is to output an XML `book` document type, we'll convert it to a LaTeX article document class. Notice that *AbiWord* has correctly output the expected section and title markup empty, even though it's not used. Here's the XML output (I've changed the linebreaks to keep it within the bounds of the page size of the printed edition):

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.2//EN"
  "http://www.oasis-open.org/docbook/xml/4.2/docbookx.dtd">
<book>
<!-- ============================================= -->
<!-- This DocBook file was created by AbiWord.     -->
<!-- AbiWord is a free, Open Source word processor. -->
<!-- You may obtain more information about AbiWord
     at www.abisource.com                          -->
<!-- ============================================= -->
<chapter>
  <title></title>
  <section role="unnumbered">
    <title></title>
    <para>You can of course buy and install a fully-fledged
      commercial XML editor with XSLT support, and run this
      application within it. However, this is beyond the
      reach of many users, so to do this unaided you just
      need to install three pieces of software: <ulink
      url="http://java.com/download/"><emphasis>Java</emphasis></ulink>,
      <ulink
      url="http://saxon.sourceforge.net"><emphasis>Saxon</emphasis></ulink>,
      and the <ulink
      url="http://www.docbook.org/xml/4.2/index.html">DocBook
      4.2 DTD</ulink> (URIs are correct at the time of writing).
      None of these has a visual interface: they are run from
      the command-line in the same way as is possible with
      L<superscript>A</superscript>T<subscript>E</subscript>X.</para>
  </section>
</chapter>
</book>
```

The XSLT language lets us create templates for each type of element in an XML document. In our example, there are only three which need handling, as we did not create chapter or section titles (DocBook requires them to be present, but they don't have to be used).

- `para`, for the paragraph[s];
- `ulink`, for the URIs;
- `emphasis`, for the italicisation.

I'm going to cheat over the superscripting and subscripting of the letters in the LaTeX logo, and use my editor to replace the whole thing with the `\LaTeX` command. In the other three cases, we already know how LaTeX deals with these, so we can write our templates (see Figure 10.3).

Figure 10.3: XSLT script to convert the paragraph

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">

  <xsl:output method="text"/>

  <xsl:template match="/">
    <xsl:text>\documentclass{article}\usepackage{url}</xsl:text>
    <xsl:apply-templates/>
  </xsl:template>

  <xsl:template match="book">
    <xsl:text>\begin{document}</xsl:text>
    <xsl:apply-templates/>
    <xsl:text>\end{document}</xsl:text>
  </xsl:template>

  <xsl:template match="para">
    <xsl:apply-templates/>
    <xsl:text>&#x000A;</xsl:text>
  </xsl:template>

  <xsl:template match="ulink">
    <xsl:apply-templates/>
    <xsl:text>\footnote{\url{</xsl:text>
    <xsl:value-of select="@url"/>
    <xsl:text>}}</xsl:text>
  </xsl:template>

  <xsl:template match="emphasis">
    <xsl:text>\emph{</xsl:text>
    <xsl:apply-templates/>
    <xsl:text>}</xsl:text>
  </xsl:template>

</xsl:stylesheet>
```

If you run this through *Saxon*, which is an XSLT processor, you can output a LaTeX file which you can process (see Figure 10.4) and view (see Figure 10.5).

Writing XSLT is not hard, but requires a little learning. The output method here is `text`, which is LaTeX's file format (XSLT can also output HTML and other formats of XML).

1. The first template matches '/', which is the document root (before the `book` start-tag). At this stage we output the text which will start

Figure 10.4: Running and XML file through the Saxon XSLT processor

```
$ java -jar saxon9.jar -o para.ltx para.dbk para.xsl
$ pdflatex para.ltx
This is pdfTeX, Version 3.1415926-1.40.10 (TeX Live 2009/Debian)
 restricted \write18 enabled.
entering extended mode
(./para.ltx
LaTeX2e <2009/09/24>
Babel <v3.8l> and hyphenation patterns for english, usenglishmax,
dumylang, nohyphenation, farsi, arabic, croatian, bulgarian,
ukrainian, russian, czech, slovak, danish, dutch, finnish, french,
basque, ngerman, german, german-x-2009-06-19, ngerman-x-2009-06-19,
ibycus, monogreek, greek, ancientgreek, hungarian, sanskrit, italian,
latin, latvian, lithuanian, mongolian2a, mongolian, bokmal, nynorsk,
romanian, irish, coptic, serbian, turkish, welsh, esperanto,
uppersorbian, estonian, indonesian, interlingua, icelandic, kurmanji,
slovenian, polish, portuguese, spanish, galician, catalan, swedish,
ukenglish, pinyin, loaded.
(/usr/share/texmf-texlive/tex/latex/base/article.cls
Document Class: article 2007/10/19 v1.4h Standard LaTeX document class
(/usr/share/texmf-texlive/tex/latex/base/size10.clo))
(/usr/share/texmf-texlive/tex/latex/ltxmisc/url.sty) (./para.aux)
[1{/home/peter/.texmf-var/fonts/map/pdftex/updmap/pdftex.map}] (./para.aux))
</usr/share/texmf-texlive/fonts/type1/public/amsfonts/cm/cmr10.pfb>
</usr/share/texmf-texlive/fonts/type1/public/amsfonts/cm/cmr6.pfb>
</usr/share/texmf-texlive/fonts/type1/public/amsfonts/cm/cmr7.pfb>
</usr/share/texmf-texlive/fonts/type1/public/amsfonts/cm/cmti10.pfb>
</usr/share/texmf-texlive/fonts/type1/public/amsfonts/cm/cmtt8.pfb>
Output written on para.pdf (1 page, 54289 bytes).
Transcript written on para.log.
$
```

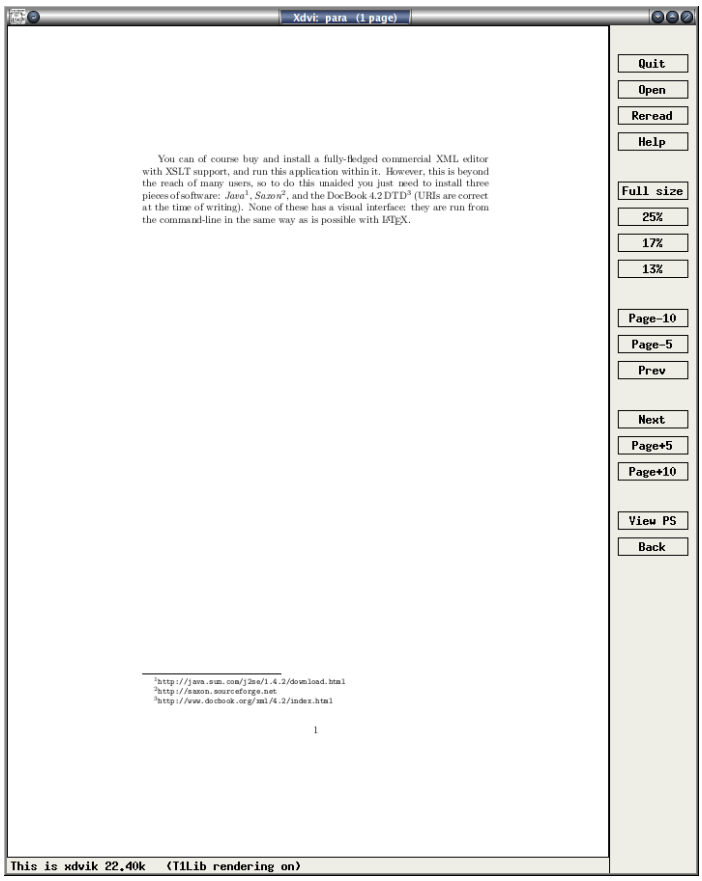
the LaTeX document, `\documentclass{article}` and `\usepackage{url}`. The `apply-templates` instructions tells the processor to carry on processing, looking for more matches. XML comments get ignored, and any elements which don't match a template simply have their contents passed through until the next match occurs, or until plain text is encountered (and output).[6]

2. The `book` template outputs the `\begin{document}` and the `\end{document}` commands, and between them the `apply-templates` makes it carry on processing.

3. The `para` template just outputs its content, but follows it with a linebreak (using the hexadecimal character code `x000A` (see the ASCII chart in Table C.1).

[6] Strictly speaking it isn't output at this stage: XML processors build a 'tree' (a hierarchy) of elements in memory, and they only get 'serialised' at the end of processing, into a stream of characters written to a file.

Figure 10.5: Displaying the typeset paragraph




```
\documentclass{article}\usepackage{url}\begin{document}
You can of course buy and install a fully-fledged commercial XML
editor with XSLT support, and run this application within it. However,
this is beyond the reach of many users, so to do this unaided you just
need to install three pieces of software:
\emph{Java}\footnote{\url{http://java.sun.com/j2se/1.4.2/download.html}},
\emph{Saxon}\footnote{\url{http://saxon.sourceforge.net}}, and the
DocBook 4.2
DTD\footnote{\url{http://www.docbook.org/xml/4.2/index.html}} (URIs
are correct at the time of writing). None of these has a visual
interface: they are run from the command-line in the same way as is
possible with \LaTeX.
\end{document}
```

4. The `ulink` template outputs its content but follows it with a footnote using the `\url` command to output the value of the `url` attribute.

5. The `emphasis` template surrounds its content with `\emph{` and `}`.

This is a relatively trivial example, but it serves to show that it's not hard to output LaTeX from XML. In fact there is a set of templates already written to produce LaTeX from a DocBook file at `http://www.dpawson.co.uk/docbook/tools.html#d4e2905`

10.2 Converting out of LaTeX

This is much harder to do comprehensively. As noted earlier, the LaTeX file format really requires the LaTeX 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 book is about!).

Many authors and editors rely on custom-designed or homebrew converters, often written in the standard shell scripting languages (Unix shells, Perl, Python, Tcl, etc.). Although some of the packages presented here are also written in the same languages, they have some advantages and restrictions compared with private conversions:

- Conversion done with the standard utilities (eg awk, tr, sed, grep, detex, etc.) can be faster for *ad hoc* translations, but it is easier to obtain consistency and a more sophisticated final product using *LaTeX2HTML* or *TeX4ht* — see below — or one of the other systems available.
- Embedding additional non-standard control sequences in LaTeX source code may make it harder to edit and maintain, and will definitely make it harder to port to another system.
- Both the above methods (and others) provide a fast and reasonable reliable way to get documents authored in LaTeX onto the Web in an acceptable — if not optimal — format.
- *LaTeX2HTML* was written to solve the problem of getting LaTeX-with-mathematics onto the Web, in the days before MathML and math-capable browsers. *TeX4ht* was written to turn LaTeX documents into Web hypertext — mathematics or not.

10.2.1 Conversion to Word

There are several programs on CTAN to do LaTeX-to-*Word* and similar conversions, but they do not all handle everything LaTeX can throw at them, and some only handle a subset of the built-in commands of default LaTeX.

Two in particular, however, have a good reputation, although I haven't used either of them (I tend to stay as far away from *Word* as possible):

- *latex2rtf* by Wilfried Hennings, Fernando Dorner, and Andreas Granzer translates LaTeX into RTF — the opposite of the *rtf2latex2e* mentioned earlier. RTF can be read by most wordprocessors, and this program preserves layout and formatting for most LaTeX documents using standard built-in commands.
- Kirill A Chikrii's *TeX2Word* for Microsoft Windows is a converter plug-in for *Word* to let it open TeX and LaTeX documents. The author's company claims that 'virtually any existing TeX/LaTeX package can be supported by *TeX2Word*' because it is customisable.

One easy route into wordprocessing, however, is the reverse of the procedures suggested in the preceding section: convert LaTeX to HTML, which many wordprocessors read easily. The following sections cover two packages for this. Once it's in HTML, you could run it through *Tidy* to make it XHTML, add some embedded styling using Cascading Style Sheets (CSS), and rename the file to end in `.doc`, which can fool *Word* into opening it natively.

10.2.2 LaTeX2HTML

As its name suggests, this is a system to convert LaTeX structured documents to HTML. Its main task is to reproduce the document structure as a set of interconnected HTML files. Despite using Perl, *LaTeX2HTML* relies very heavily on standard Unix facilities like the *NetPBM* graphics package and the pipe syntax. Microsoft Windows is not well suited to this kind of composite processing, although all the required facilities are available for download in various forms and should in theory allow the package to run — but reports of problems are common.

- The sectional structure is preserved, and navigational links are generated for the standard Next, Previous, and Up directions.
- Links are also used for the cross-references, citations, footnotes, ToC, and lists of figures and tables.
- Conversion is direct for common elements like lists, quotes, paragraph-breaks, type-styles, etc., where there is an obvious HTML equivalent.
- Heavily formatted objects such as math and diagrams are converted to images.
- There is no support for homebrew macros.

There is, however, support for arbitrary hypertext links, symbolic cross-references between 'evolving remote documents', conditional text, and the inclusion of raw HTML. These are extensions to LaTeX, implemented as new commands and environments.

LaTeX2HTML outputs a directory named after the input filename, and all the output files are put in that directory, so the output is self-contained and can be uploaded to a server as it stands.

10.2.3 TeX4ht

TeX4ht operates differently from *LaTeX2HTML*: it uses the TeX program to process the file, and handles conversion in a set of postprocessors for the common LaTeX packages. It can also output to XML, including Text Encoding Initiative (TEI) and DocBook, and the OpenOffice and WordXML formats, and it can create TeXinfo format manuals.

By default, documents retain the single-file structure implied by the original, but there is again a set of additional configuration directives to make use of the features of hypertext and navigation, and to split files for ease of use. This is a most powerful system, and probably the most flexible way to do the job.

10.2.4 Extraction from PostScript and PDF

If you have the full version of Adobe *Acrobat* (or one of several other commercial PDF products), you can open a PDF file created by *pdfLaTeX*, 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 *pdftotext* program to extract everything from the PDF file as plain (paragraph-formatted) text.

10.2.5 Last resort: strip the markup

At worst, the *detex* program on CTAN will strip a LaTeX 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 LaTeX

The reader will have deduced by now that while LaTeX is possibly the best programmable typesetting system around, the LaTeX file format is not generally usable with anything except the LaTeX program. LaTeX was originally written in the mid-1980s, about the same time as the Standard

Generalized Markup Language (SGML), but the two projects were not connected. However, TEX and LATEX 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 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 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 LATEX already possesses, so the sibling XSLT can be used instead to translate to LATEX source code, as shown in the example in § 10.1.1. 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 LATEX. A similar system is used for the Linux Documentation Project, which uses SGML transformed by the Document Style Semantics and Specification Language (DSSSL) to TEX

The source code of this book, available online at `http://www.ctan.org/tex-archive/info/beginlatex/src/` includes XSLT which does exactly this.

A Configuring TEX search paths

TEX systems run on a huge variety of platforms, and are typically made up of a very large number of very small files in several separate 'trees' of directories (folders). This allows users to update parts of the system without having to update all of it, and to maintain their own tree of preferred files without having to have administrative rights on the computer.

To make sure TEX finds the right file, it uses a technique borrowed from the Unix world, based on a simple 'hash index' for each directory tree they need to look in. This is known as the ls-R database, from the Unix command (`ls -R`) which creates it. The program which does this for TEX is actually called after this command: *mktexlsr*, although it may be aliased as *texhash* or something else on your system. This is the program referred to in step 4 in the procedure on p. 82.

However, to know where to make these indexes, and thus where to search, TEX needs to be told about them. You don't normally need to change the configuration, but sometimes you might want to move directories between disks to free up space or use faster equipment, which would mean changing the configuration.

In a standard TEX installation this information is in the main (*not* the local) installation directory, in `texmf/web2c/texmf.cnf`. The file is similar to a Unix shell 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/share/texmf
TEXMFLOCAL = /usr/local/share/texmf
HOMETEXMF = $HOME/texmf
TEXMF = {$HOMETEXMF,!!$TEXMFLOCAL,!!$TEXMFMAIN}
SYSTEXMF = $TEXMF
VARTEXFONTS = /var/lib/texmf
TEXMFDBS = $TEXMF;$VARTEXFONTS
```

This defines where the main TEX/METAFONT (`texmf`) 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 may be set up in `/usr/share/texmf-local` or `/usr/share/texmf.local` instead of `/usr/local/`

Finding out where LATEX looks for stuff

There is a program distributed in all LATEX installations called kpsewhich, a TEX-specific variant of the standard Unix which command. If you type the command followed by a filename, it will tell you whereabouts in your TEX installation it is.

```
$ kpsewhich article.cls
/usr/share/texmf-texlive/tex/latex/base/article.cls
```

Better, there is an option to tell you where your main and local trees (directories) are installed, and even where LATEX puts its map files and format files (internal setups):

```
$ kpsewhich --expand-var '$TEXMFMAIN'
$ kpsewhich --expand-var '$TEXMFLOCAL'
$ kpsewhich --expand-var '$TEXMFSYSVAR'
```

So if you forget where to put something you are installing, **`$TEXMFLOCAL`** is where it should go.

share/texmf or similar variations, so you would see this name used instead. Under Microsoft Windows, the names will be full paths such as C:\ProgramFiles\TeXLive\texmf or C:\ProgramFiles\MikTeX2.8\tex. On an Apple Mac, it is ~/Library/texmf for each user.

B TEX Users Group membership

The TEX Users Group (TUG) 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 TEX typesetting system invented by Donald Knuth. TUG is run by and for its members and represents the interests of TEX users worldwide. There are many regional and sectoral user groups organised on a geographic or language basis: see the list at `http://www.tug.org/usergroups.html` for details.

B.1 TUG membership benefits

Members of TUG help to support and promote the use of TEX, METAFONT, and related systems worldwide. All members receive *TUGboat*, the journal of the TEX Users Group, the TEX Live software distribution (a runnable TEX 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.

B.2 Becoming a TUG member

Please see the forms and information at `http://www.tug.org/join.html`. You can join online, or by filling out a paper form. The NTG (Dutch) and UKTUG (United Kingdom) TEX user groups have joint membership agreements with TUG whereby you can receive a discount for joining both TUG and the local user group. 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 TEX memorabilia through the TUG store (`http://www.tug.org/store`).

The current (2011) TUG membership fee is $85 (US) per year for individuals and $55 for students, new graduates, seniors, and citizens of countries with modest economies. Add $10 to the membership fee after March 31 to cover additional shipping and processing costs. The current rate for non-voting subscription memberships (for libraries, for example) is $100. The current institutional rate is $500, which includes up to seven individual memberships.

B.3 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.

C The ASCII character set

The American Standard Code for Information Interchange (ASCII) was invented in 1963, and after some redevelopment settled down in 1984 as standard X3.4 of American National Standards Institute (ANSI). It represents the 95 basic codes for the unaccented printable characters and punctuation of the Latin alphabet, plus 33 internal 'control characters' originally intended for the control of computers, programs, and external devices like printers and screens.

Many other character sets (strictly speaking, 'character repertoires') have been standardised for accented Latin characters and for all other non-Latin writing systems, but these are intended for representing the symbols people use when writing text on computers. Most programs and computers use ASCII internally for all their coding, the exceptions being XML-based languages like XSLT, which are inherently designed to be usable with any writing system, and a few specialist systems like APL.

Although the TEX and LATEX file formats can easily be used with many other encoding systems (see the discussion of the inputenc in §2.6), they are based on ASCII. It is therefore important to know where to find *all* 95 of the printable characters, as some of them are not often used in other text-formatting systems. The following table shows all 128 characters, with their decimal, octal (base-8), and hexadecimal (base-16) code numbers.

Decimal number values are under each character. The index numbers in the first and last columns are for finding the octal (base-8) and hexadecimal (base-16) values respectively. Replace the arrow with the number or letter

from the top of the column (if the arrow points up) or from the bottom of the column(if the arrow points down).

Example: The Escape character (ESC) is octal '033 (03 for the row, 3 for the number at the top of the column because the arrow points up) or hexadecimal "1B (1 for the row, B for the letter at the bottom of the column because the arrow points down).

Table C.1: The ASCII characters

Oct	0	1	2	3	4	5	6	7	Hex
′00 ↑	NUL 0	SOH 1	STX 2	ETX 3	EOT 4	ENQ 5	ACK 6	BEL 7	″0 ↑
′01 ↑	BS 8	HT 9	LF 10	VT 11	FF 12	CR 13	SO 14	SI 15	″0 ↓
′02 ↑	DLE 16	DC1 17	DC2 18	DC3 19	DC4 20	NAK 21	SYN 22	ETB 23	″1 ↑
′03 ↑	CAN 24	EM 25	SUB 26	ESC 27	FS 28	GS 29	RS 30	US 31	″1 ↓
′04 ↑	Space 32	! 33	" 34	# 35	$ 36	% 37	& 38	’ 39	″2 ↑
′05 ↑	(40	) 41	* 42	+ 43	, 44	- 45	. 46	/ 47	″2 ↓
′06 ↑	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55	″3 ↑
′07 ↑	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63	″3 ↓
′10 ↑	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71	″4 ↑
′11 ↑	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79	″4 ↓
′12 ↑	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	″5 ↑
′13 ↑	X 88	Y 89	Z 90	[91	\ 92	] 93	^ 94	_ 95	″5 ↓
′14 ↑	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103	″6 ↑
′15 ↑	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111	″6 ↓
′16 ↑	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	″7 ↑
′17 ↑	x 120	y 121	z 122	{ 123	¦ 124	} 125	~ 126	DEL 127	″7 ↓
	8	9	A	B	C	D	E	F	

D GNU Free Documentation License

Version 1.3, 3 November 2008

D.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. Secondarily, 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

[1] `http://www.fsf.org/`

works whose purpose is instruction or reference.

D.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.

The 'publisher' means any person or entity that distributes copies of the Document to the public.

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.

D.2 VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, 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.

D.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.

D.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 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.

D.5 COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 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 mul-

tiple 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'.

D.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.

D.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.

D.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.

D.9 TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

[2] `http://www.gnu.org/copyleft/`

D.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 Copyleft[2].

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. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

D.11 RELICENSING

'Massive Multiauthor Collaboration Site' (or 'MMC Site') means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A 'Massive Multiauthor Collaboration' (or 'MMC') contained in the site means any set of copyrightable works thus published on the MMC site.

'CC-BY-SA' means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corpo-

ration, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

'Incorporate' means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is 'eligible for relicensing' if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

D.12 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 © YEAR YOUR NAME
>
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 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.

References

1. **Bull, RJ:** Accounting in Business. London: Butterworths, 1972, ISBN 0–406–70651–4

2. **Heller, Robert:** New To LaTeX. . . Unlearning Bad Habits. `comp.text.tex`, 11 March 2003, Nr. MPG.18d82140d65ddc5898968c@news.earthlink.net (all pages)

3. **Lamport, Leslie:** LaTeX: A Document Preparation System. 2nd edition. Reading, MA: Addison-Wesley, 1994, ISBN 0–201–52983–1

4. **Mittelbach, Frank** et al.: The LaTeX Companion. 2nd edition. Boston, MA: Addison-Wesley/Pearson Education, 2004, ISBN 0–201–36299–6

5. **Ryder, John:** Printing for Pleasure. London: Bodley Head, 1976, ISBN 0–370–10443–9

Index

The same fonts are used here as in the text of the book (see *About this book* on p. xxi) to distinguish between different meanings:

Notation	Meaning
CTAN	Acronyms (small caps in some typefaces)
`\command`	LaTeX control sequences (monospace font)
term	Defining instance of a specialist term (bold italics)
product	program or product name (italics)
`environment`	LaTeX environment (sans-serif bold)
package	LaTeX package (sans-serif; all available from CTAN)
options	Options to environments (sans-serif oblique)
`variables`	Variables (monospace oblique)

In the online version, the entries below are all hyperlinked to their source, with subsequent multiple occurrences giving the section number or name. Page or section numbers in **bold type** indicate a defining instance.