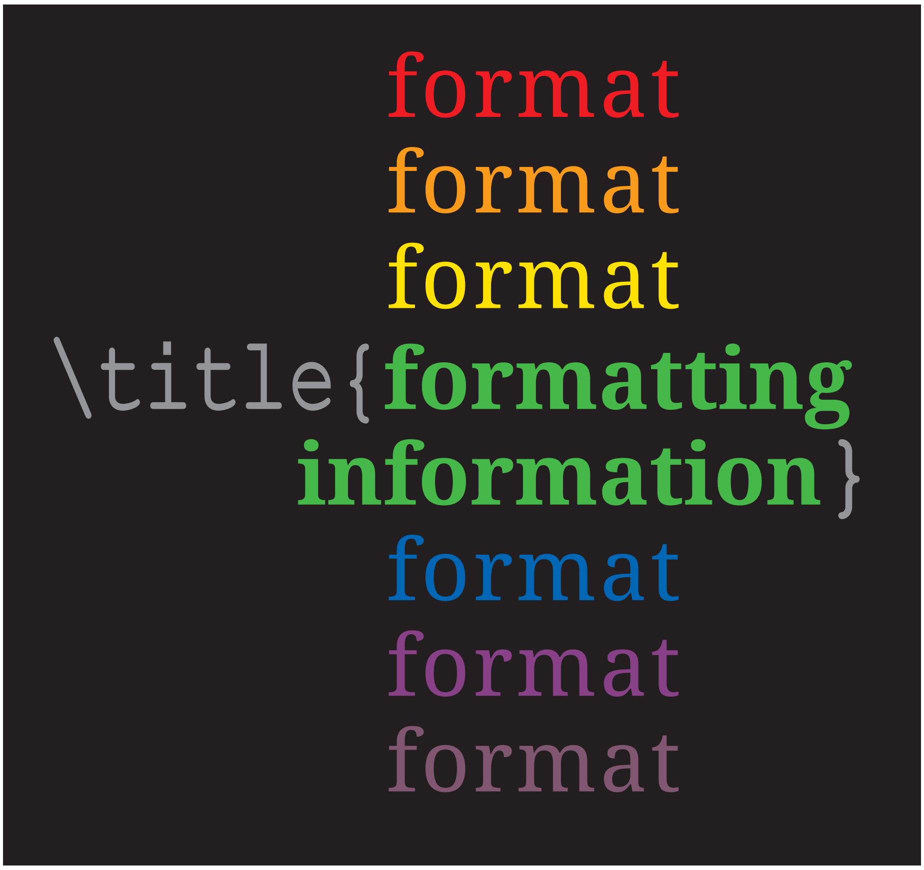


An introduction to typesetting with LaTeX

Peter Flynn

The PDF edition was typeset with XeLaTeX in Noto Serif, Noto Sans, and Nimbus Mono Narrow from the XML master document also used in the web and eBook editions, transformed with XSLT 3, *Calibre*, and *kindlegen*. Readers of the Kindle (`.mobi`) version may notice that the TeX and related logos are reproduced in plain text rather than with the special spacing and alignment normally seen; this is to compensate for the many deficiencies of the Kindle file format and the Kindle devices, which are not capable of reproducing microtypographic adjustments. Standard Electronic Publication v3 (EPUB v3) format eBook readers are not affected by this and show the logos in their correct form.

Formatting Information

An introduction to
typesetting with LaTeX

Peter Flynn

Silmaril Information Management Consultants

Eighth edition
26th August 2023

Contents

List of Exercises

List of Tables

List of Figures

Welcome to *Formatting Information*

This is the print version of *Formatting Information*, a book about how to use the LaTeX document preparation system. LaTeX takes over where wordprocessors and desktop publishing systems leave off, making it possible to automate your formatting consistently, accurately, and reusably, without the tedious and repetitive manual formatting required by other systems.

This book has helped thousands of users get started. It's now in its eighth edition (2023), but this is also a production release: everything has been tested but some details had to be rewritten and this was delayed by the COVID-19 pandemic until the release of of TeX Live (2023). The only things you need are a computer and a copy of LaTeX...and a document that you want to typeset. LaTeX works on almost any computer, and you can download it from the TUG web site, install it from the TUG DVD, or use one of the online in-browser versions like Overleaf.

In the web and eBook editions, this page doubles as the index, but in the print (PDF) edition, the index is at the end (p. 325). If you haven't done any typesetting before, I recommend that you start at the beginning. If you're itching to get started, and you feel you know enough about computers and text-editing already, you can try the Quick Start instead.

Either way, welcome to LaTeX. Take it gently for a while, and get used to being able to spend more time actually *writing* than formatting. If you find mistakes, please let me know so that I can correct them.

Some font conventions are used in the text and the index to distinguish between different meanings. These are listed in 'Symbols and conventions' on page xxxix. The entries in the index are all hyperlinked to their source. In the web and eBook editions, subsequent multiple occurrences give the section number or name. Page or section numbers in **bold type** indicate a canonical location where the entry is explained.

Credits

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 it (Flynn 2002), 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 have 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 someone felt to be indispensable.

The last few editions have been 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 `tex.stackexchange.com`, and the sometimes more obvious changes visible when one installs a new version of TEX. The previous edition came after a longer pause while I finished my research into editing interfaces for structured documents (Flynn 2014a), which took rather longer than I expected, so there was rather more to change; plus I switched the web version to a HTML5 mobile layout, which meant reprogramming the transformation. The new print editions now uses XƎLATEX, which has meant no more worrying about stray UTF-8 characters, and the ability to use different fonts. This edition now assumes the reader uses XƎLATEX and biblatex with *biber*, and all the examples and package references have been updated to match.

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 many online documents, some listed in 'Where's the math?' on page xxxiii, which deal with mathematical typesetting in TEX and LATEX in finer and better detail than I am capable of.

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. At that meeting, Peter Flom 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 subsequent Practical TEX Conferences, in 2006 and later, that more presenters, especially in the Humanities, have stepped up to Peter'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 with our 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, and lack the vocabulary and the experience to explain how to get 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.

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, perhaps most, users 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, mailing lists, or web forums, and do not download or read the excellent 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, Googling topics to try and find an answer, but finding it hard to express a query when they haven't learned the vocabulary yet.

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, as well as being expert in using LaTeX. Most beginners don't want extensive reasoning over all 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 at the annual TUG conference, held that year at the University of Delaware, on the marketing of LaTeX (Flynn 2001), and I showed an example of a draft brochure 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 the document you are now reading, 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 on the home page (at the front of the book, if you are reading this on paper or in an ebook). As I also noted earlier, it can be used as a one-day course if the users already have some experience of writing, editing, or reviewing formal documents for publication or assessment (eg reports, white papers, essays, books, 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 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. That was then; nowadays I would hope most people install the full works, so everything should work as-is.

However, as the careful reader may already have noticed, the master source of this 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 XSLT transformation to that it can be guaranteed to process with all of the current full LaTeX installations.

If you are just starting with LaTeX, at an early opportunity you should buy or borrow a copy of *LaTeXbook* which is Lamport's original manual. More advanced users should get the *Companion*, the *Graphics Companion* and the *Web Companion*. Mathematical users might want to start with the *Short Math Guide for LaTeX*. Details are in section E.13 on page 319.

There are hundreds if not thousands of web pages about how to use LaTeX, and the online version of this book is just one. There are dozens of books about LaTeX too, and these are a few of the printed ones that I strongly recommend:

- van Dongen's *LaTeX and Friends* covers LaTeX and mathematics with reference to the latest packages and techniques. I've known Marc for many years: he works at the same institution that I do, and this is a shameless plug for a really excellent book.
- Oetiker's *A (Not So) Short Introduction to LaTeX 2$_\varepsilon$* started as a translation and rationalisation of a ground-breaking German-language introduction to LaTeX. It has since taken on a momentum of its own, and has itself been translated into a many languages. It remains the best short guide, even if it has grown a bit.
- Grätzer's *Practical LaTeX* is a very approachable, non-technical introduction to everyday LaTeX usage. The first chapter is designed as a standalone guide to get you started.

There is a much bigger list of online resources on the TeX Users Group web site, and a much longer list of books there at `tug.org/books/`.

If you've just finished a LaTeX course, or read a book or web site, and you want a refresher or *aide-mémoire*, my leaflet *The Very Short Guide to LaTeX* is what it says it is: a 4–page guide to the bare essentials of document construction, designed as a reminder of what you've just learned.

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 40 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 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, not just to new computer users, and it's great fun, 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 may 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 software' that the major manufacturers foist on 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 was posted in response to this query, which I reproduce with permission below (the bold text is my own emphasis; full link in Heller (2003) below):

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

> vanilla text editor and treat it like an old-fashioned typewriter. Don't waste time playing with your mouse.
> Note: I am *not* saying that you should have no concerns about the appearance of your document, just that you should *write* the document (completely) first and tweak the appearance later...*not* [spend time on] lots of random editing in the bulk of the document itself. (Heller 2003)

More recently, an article reporting on a study of writing patterns between Microsoft Word users and LaTeX users reported that it was faster to use *Word* (Knauff and Nejasmic 2014). As a reviewer of that article, I asked the authors to make it clearer that the use of the proper templates (classes and packages) removed the need for LaTeX users to spend the time formatting that Word users do. The publication of the article upset a number of people in the TeX field, but I hope that it will spur the critical examination of how we write, and why it's better to do it in LaTeX than in other systems.

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* (Knuth 1980), and it remains pretty much the only typesetting program to include fully-automated mathematical formatting by default, done the way mathematicians do it. But he also brought out a booklet called *Mathematical Writing* (Knuth, Larrabee and Roberts 1989) 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.

But TeX is 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 TeX 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 for many years this also meant that there was little commercial publicity which would have got TEX noticed outside the technical field, because there was no great corporate marketing department to advertise its existence. Even now, some people who used it in college believe that it no longer exists!

Nowadays, however, there are several companies selling TEX software or services,[1] dozens of publishers accepting LATEX documents for publication, and hundreds of thousands of users using LATEX for millions of documents.[2]

There is occasionally some confusion among newcomers between the two programs, TEX and LATEX, and the other versions available, so the differences are explained in the list below.

TEX: The underlying typesetting program, originally written by Donald Knuth at Stanford in 1978–79. 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 while at Digital Equipment Corporation (DEC) in 1985 to automate 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. The documentation is

[1] See, for example, the list of consultants published by TUG.

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

Debunking the mythology

Naturally, over all the years, a few myths have grown up around LaTeX, often propagated by people who should know better. These *canards* make it harder to explain to potential users why they should look at LaTeX, 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 list section 6.2.2.1 on page 180), 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, people, get over it).

MYTH: 'LaTeX isn't WYSIWYG': Simply not true. TeX's Device-Independent (DVI) and PDF output are generally better quality WYSIWYG than any wordprocessor and most DTP systems. What people mean is that the typographic display (preview) is asynchronous with the editor window. This is only true for the default CLI implementations. See the list item 'Synchronous typographic displays' on page xxxvii for details of synchronous versions.

MYTH: 'LaTeX is obsolete': Quite the opposite: it's under constant development, with new features being added or updated almost daily. Check `comp.text.tex` for messages about recent uploads to CTAN, or follow @TeXUsersGroup on Twitter. 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.

More mythology

If you come across other myths from people who should know better, please let me know — I'm collecting them here!

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 and ancient mainframes right down to handhelds (even old PDAs like the Sharp *Zaurus* and the Nokia *N800*), and most Apple and Android smartphones). That includes Unix & GNU/Linux, including Apple Macintosh OS X, Windows, and all other desktop, mini, and mainframe 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 explain Byzantine politics; from librarians who can understand Library of Congress (LoC) and MAchine-Readable Cataloging (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"': Completely untrue. Although TeX grew up in the mathematical and computer science fields, because those were its author's 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.

less accessible than for LaTeX, but the author, Hans Hagen, provides excellent support at Pragma/ADE.

pdfTeX and pdfLaTeX: Extended versions of the *tex* and *latex* programs that create PDF instead of DVI files, written by Hàn Thế Thành. There are also enhancements for microtypographic extensions, native font embedding, and PDF support for hyperlinking. It is currently (2022) still the default TeX engine in most distributions.

XeTeX and XeLaTeX: A recent reimplementation of the *tex* and *latex* programs by Jonathan Kew which merges Unicode and modern font technologies. It is already in common use in editing environments such as *TeXshop* (Apple Macintosh OS X), *Kile* (Unix & GNU/Linux), and *WinEdt* (Windows). Details are at the Sourceforge web site.

XeLaTeX is used to produce the PDF edition of this book, and is the recommended system for new users as its Unicode features make the use of mnemonics for non-Latin characters largely unnecessary, and it works with any of the user's installed system fonts. Newcomers are unlikely to need the scripting facilities of LuaLaTeX but they are of course welcome to use it instead.

LuaTeX and LuaLaTeX: LuaLaTeX is a development of XeLaTeX which has a copy of the *Lua* scripting language built into it. This means you can write actual programs inside LaTeX, for example to calculate results and prepare statistics, avoiding the need for a separate set of external programs. The output is recalculated afresh each time, so if the data is rapidly changing, it can always represent the most recent values. It also implements a slightly different way of accessing TT and OT fonts.

TeXinfo: TeXinfo is the official documentation format of the GNU project.[3] It was invented by Richard Stallman and Bob

[3] 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.

> Chassell. It uses a single source file to produce output in a number of formats, both online and printed (DVI, HTML, [GNU] INFOrmation (INFO), PDF, XML, *DocBook*, EPUB v3, 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 underlying TeX engine so that users and developers now have a bug-free, rock-stable platform to work with.[4] Typographic programming development continues with the New Typesetting System (NTS), planned as a successor to TeX, and LaTeX3 as a long-term successor to current LaTeX. The LaTeX Project manages the ongoing development of LaTeX (see `www.latex-project.org`), and the current version is LaTeX 2$_\varepsilon$, which is what we are concentrating on here. Details of all developments can be had from the TUG web site at `www.tug.org`

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

Introduction

This book originally accompanied a two-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 been used as the basis for a 1-day intensive introductory course. It is aimed at users of Unix & GNU/Linux, including Apple Macintosh OS X, and Microsoft Windows systems, but it can be used with LaTeX on any platform, especially online systems such as *Overleaf*, but also including other Unix workstations, mainframes, Android and Apple smartphones, and even some older Personal Digital Assistants (PDAs).

Who needs this book?

The course was originally designed for computer-literate but non-IT 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;
- ☐ having fun playing around with fonts and formatting.

Table 1 – Using your computer — essential skills

Subject	**Detail**	**ECDL**
Using the mouse	know how to point and click with your mouse to run programs, pick from a menu, and highlight text (or how to use keyboard shortcuts to do the same)	1.1.4–1.1.6, 1.3.1, 1.4.1–1.4.2,
Handling files	know how to create, open, save, close, rename, copy, move, and delete files and folders (directories) using a directory browser (Windows: *My Computer* or just *Computer*; Mac: *Finder*; Linux: *Nautilus*, *Thunar*, or *Dolphin*, etc) or a typed command	2.3
Handling characters	know where to find all 95 of the printable (ASCII) characters on your keyboard, plus accents and symbols, if you need them	3.2.1.2
Using an editor	know how to use a good plaintext editor (*not* a wordprocessor like Microsoft *Word*, *Libre Office*, Lotus *Notes*, Apple *Pages*, or Corel *WordPerfect*; and *not* a context-insensitive editor like Apple *TextEdit* or Microsoft *Notepad*).	2.1.3

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 a little about writing, formatting, and readability. However, you do need to be completely familiar with using your computer, which means knowing the following topics thoroughly. Note that none of these is in any way specialist; they're all basic, fundamental, standard computer skills that everyone should know.

Table 2 – Using software (programs) — useful skills

Subject	Detail	ECDL
Downloading files	know how to use your Web browser and/or file transfer program to download and save files from the Internet	7.1.6 [7.3.1.6]
Unzip files	know how to uncompress and unwrap compressed 'archive' (zip) files	2.3.8
Install software	know how to install software, both manually and using automated installers	2.6
RTFM	know how to read and follow instructions and how (and where) to ask for help	1.7

If you don't know how to do these things yet, it's important to go and learn them first, at least the essential ones. 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 really 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 which everyone needs to be familiar with.

With the exception of software installation, they are all included in the 2000 European Computer Driving Licence (ECDL) (now the International Computer Driving Licence (ICDL)) course: the relevant module and section numbers of the original ECDL syllabus are noted in parentheses or in the margin above (Kelly and O'Connor 2005).

Objectives of this book

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

1. use your editor to create and maintain your documents;

2. use LaTeX markup to identify your document structure and formatting requirements;

3. typeset LaTeX documents, correct simple formatting errors, and display or print the results;

4. identify and use additional LaTeX packages (using the Internet for downloading where necessary and installing them);

5. recognise the limitations of procedural markup systems and choose appropriate generic markup methods where appropriate.

Synopsis

The original course covered the following topics as separate sessions. Earlier versions of this document kept to this structure in the book as chapters, but recent versions have moved Installation (originally chapter 1) to Appendix A and merged it with the details of configuration; and Typesetting, viewing, and printing (originally chapter 4) to a new Appendix; as the procedures in both cases have been so much simplified that the previous level of detail is no longer needed.

1. How to create LaTeX documents (with a Quick-Start Guide for the impatient);

2. Basic document structures (the Document Class Declaration and its layout options; the `document` environment with sections and paragraphs);

3. Using packages and CTAN to adapt formatting to your needs;

4. Other document structures (lists, tables, figures, images, and verbatim text);

5. Textual tools (footnotes, marginal notes, cross-references, indexes and glossaries, and bibliographic citations);

6. Typographic considerations (white-space and typefaces; in-line markup and font changes; extra font installation and automation);

7. Programmability and automation (macros and modifying LaTeX's behaviour);

8. Conversion and compatibility with other systems (XML, *Word*, etc).

A Where to get and how to install LaTeX (using the TeX Users Group's TeX Collection DVD);

B How to install new fonts;

C Typesetting, viewing, and printing (largely obsolete now that editors are better integrated with viewers and printers);

D User groups and the benefits of membership;

E The ASCII character set;

F The GNU Free Documentation License.

I have made a few other 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*, where all beginners should start;
- *A (Not So) Short Introduction to LaTeX 2$_\varepsilon$* is a good beginner's tutorial;
- *Gentle Intro* is a classic tutorial on Plain TeX (not LaTeX);

- *Imported graphics* 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;
- *TeX Symbol List* 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 at the end of the on page xvii.

Availability of LaTeX systems

The standard implementations of TeX and related systems are in the TeX Collection, published annually on DVD and online by the TeX Users Group (TUG). These are all derived from Knuth's master versions, and adapted for all major platforms (Unix & GNU/Linux, including 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 DVD image file in the ISO format from CTAN to burn your own copy.

Rather than install TeX Live on your laptop, desktop, or tablet, many users prefer to use one of the online systems such as *Overleaf* via a web browser. These provide an editor and typeset display in the same way as installed systems, but contained inside your web browser. LaTeX still works identically no matter which way you choose to use it.

There are no longer any commercial implementations of TeX that I am aware of. The most recently available ones are listed in 'Commercial implementations' below.

Systems included on the TeX Collection DVD

MiKTeX (Windows): This is the *MiKTeX* implementation plus the *TeXStudio* editor with its own built-in PDF viewer.

MacTeX (OS X): This is *TeX Live* for Mac plus the *TeXshop* editor (the Mac's own built-in *Preview* is used for the PDF display).

TEX Live (Unix & GNU/Linux, including Apple Macintosh OS X; and Windows): The full *TEX Live* system from TUG.

The TEX Collection DVD also includes a screenshot of the whole CTAN archive.

Choice of TEX Live for Linux users

Users of Linux (Red Hat/CentOS/Fedora, Debian/Ubuntu, and their derivative systems) can choose between the TUG TEX Live from the DVD are their system-supplied package (`texlive-full` for Debian (DEB) systems or `texlive-scheme-full` for RedHat Package Manager (RPM) systems). These system packages can be installed using the system's package manager (eg *dnf* or *yum* for RPMs; *apt-get* or *apt* for DEBs; or a graphical installer like *Synaptic* or *Software* etc), as these provide tighter integration with other software such as editors and viewers, but the version is usually tied to the version of the operating system. The *TEX Live* from TUG can be updated independently of the operating system, and SHOULD be used where the latest features are always required (in which case any system-installed packaged version MUST be removed first, to avoid conflicts and unresolved dependencies — see the notes in Appendix 1 starting on page 241).

Because the TEX program (the internal 'engine' which does the actual typesetting) is independent of any 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: see 'Graphical interfaces (editors)' below and section 1.3 on page 7 for details.

Graphical interfaces (editors)

Most users run LATEX with a graphical plaintext editor which has a toolbar and menus like other windowing applications. These usually include all the common formatting features of

LaTeX plus writing tools like spellchecking, thesaurus, indexing, and bibliographic citation. They generally all work in a very similar way. Text-only interfaces are available for use on servers and automated production systems (see 'Command-line interfaces').

The Windows and Mac systems described in 'Availability of LaTeX systems' on page xxxiv come with a recommended editor (*TeXStudio* and *TeXshop* respectively), but you can install any other suitable editor you prefer (see section 1.3 on page 7). The Unix & 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*, *TeXStudio*, or *Kile*.

Some fully synchronous typographic interfaces (editors) were available as commercial products, but so far as is known none of these is available any more: see the list item 'Synchronous typographic displays' on page xxxvii.

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 systems have to run in the background, unattended, with no human to click on buttons. In fact, the TeX typesetting engine is a Command-Line Interface (CLI) program, which can be used from any script or console or 'Command' window. You can type the command `xelatex` followed by the name of your document file (see Figure B.2 on page 288 for an example).

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

WYSIWYG displays

LaTeX usually displays your typeset results in a separate window such as a PDF viewer, updated automatically every time the document is retypeset, because the typesetting is kept separate from the editing. This is called an '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 HTML, where you reload the page in a browser to see it, or if you have used a spreadsheet, where the ReCalc button (F9) does something similar.

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, for example *Scientific Word* (see the list item 'Scientific Word' section 8.1.1.1 on page 221) or *VTeX* (see 'Commercial implementations' on page xxxviii).

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* with 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 with which the typographical display shows your document. Most modern PDF viewers are pretty good, given the fact that your screen is probably only a fraction of the accuracy of your printer — from 96 dots per inch (DPI) on an old desktop screen, 140 DPI on a small laptop, 240 DPI on a high-end 4k laptop or desktop, and up to around 300 DPI on some phones and tablets; as opposed to 600 DPI on your printer, or 1200 DPI or more in photo-quality, and 3,600 DPI up to well over 5,000 DPI on industrial digital printers and laser phototypesetters. More commonly, what people are really trying to express with WYSIWYG is What You See Is What You Mean (WYSIWYM); that is, your intent is accurately conveyed by the formatting.

Commercial implementations

Although the TEX Collection is available free of charge, there have some excellent commercial implementations of TEX and LATEX, listed below, which provided enhanced support and additional features, and many of these are still in use. These companies, founders, and staff were good friends of the TEX and LATEX communities for many years; but sadly, all the products listed are now at end-of-life, although they are in some cases still available for download (links in footnote).

Y&Y, Mackichan Software, PC-TEX, Blue Sky (Textures), True TEX, and BaKoMa TEX (RIP Basil K Malyshev), and MicroPress, Inc (VTEX), who produced TEX distributions for many years, all appear to have ceased trading.

Some of the Y&Y add-on fonts are still distributed by the TEX Users Group (see Appendix 3 starting on page 297), 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.

Symbols and conventions

There are several typographic conventions about how you represent computer-related material in print which are shown in below. Typed commands, keywords, examples of input, and related text are in a fixed-width (monospace) font, like a typewriter, because that's how program code is usually displayed and edited (this also helps avoid ambiguities, as explained in section 4.7.1.1 on page 130). Special values, like numeric quantities represented by a name or symbol, are in italics, as in mathematics. Terms or references to products, programs, packages, and other components of LaTeX have 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 5* DTD. Conversions are made to HTML and LaTeX using XSLT3 scripts and Michael Kay's *Saxon* processor.

The complete source, with all ancillary files, is available online at `www.ctan.org/tex-archive/info/beginlatex/src/`. If you want to try processing it yourself you will need *Java* and *Saxon* in addition to a full installation of LaTeX.

This document is published under the terms and conditions of the GNU Free Documentation License. Details are in Appendix 5 starting on page 305.

Table 3 – Typographic notations used in this document

Notation	Meaning
MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL	These keywords have a specific mandatory meaning when shown in THIS STYLE (small capitals of the prevailing font), according to RFC 2119 (Bradner 1997).
\command	LaTeX commands (control sequences) that you type which perform an action or identify your text eg \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 technical term
environment	A LaTeX formatting or identification environment, like *quotation*
class	A LaTeX document class (available from CTAN), like **memoir**
package	A LaTeX add-on package (available from CTAN), like footmisc
product	A program or product name
typewriter type	Literal examples
value	Mnemonic examples of things you must type, where you have to supply real-life values of your own, like \author{your name} means you must replace your name with your own real name.
x	A specific key on your keyboard
Ctrl + x	Two or more 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 to select
␣	A normal space, just so it's visible

1 Writing documents

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

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

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

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

The big advantage of plaintext is not just that it's readable; it's that the files can be copied, downloaded, or uploaded to any computer system running LaTeX and they will typeset exactly the same. Because they are plain text they cannot corrupt your system, and they cannot be used for hiding virus infections in the way that ***binary*** (coded non-plaintext) files can be. Everything you can see is in the file and everything in the file is there for you to see: there is nothing hidden or secret and there are no manufacturers' proprietary 'gotchas' like suddenly going out of date with a new version or imposing selective Digital Restrictions Management (DRM), leaving you unable to open your files.

Exercise 1 – Plaintext and wordprocessor files

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

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

Wordprocessors use markup too (*Libre Office* and Microsoft *Word* actually use XML internally nowadays) but it's extraordinarily complex, and encoded to prevent casual inspection. It's then packaged up (often into a zip file) with all the stylesheets, images, and other bits and pieces. LaTeX prefers to keep everything out in the open, making it very obvious how you are constructing your documents, so you can come back to them in the future and not have to worry about what you did.

1.1 Markup

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

Exercise 2 – Clarity check

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

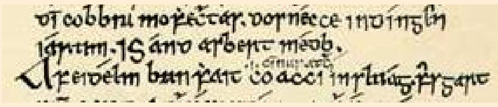
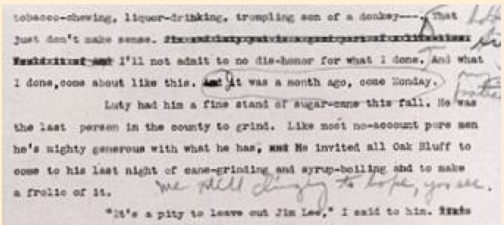
1. a footnote;
2. a section heading;
3. the Table of Contents;
4. an item in a list;
5. a URL (web link).

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

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

Markup

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

	Anon, 'Táin bó Cúailnge'	1100
 tobacco-chewing, liquor-drinking, trompling son of a donkey——. That just don't make sense. [illegible] [illegible] I'll not admit to no dis-honor for what I done. And what I done, come about like this. [illegible] It was a month ago, come Monday. Luty had him a fine stand of sugar-cane this fall. He was the last person in the county to grind. Like most no-account pore men he's mighty generous with what he has, and He invited all Oak Bluff to come to his last night of cane-grinding and syrup-boiling and to make a frolic of it. We still clinging to hope, you see. "It's a pity to leave out Jim Lee," I said to him. [illegible]	Rawlings, 'Varmints'	1932
`.h1 Interest Rates`	*Runoff*, *Script*	1960s
`:h1.Interest Rates`	*GML/DCF*	*c.*1975
`@Heading[Interest Rates]`	*Scribe*	*c.*1976
`\section{Interest Rates}`	LaTeX	1984
`<SEC><TTL>Interest Rates</TTL>...`	SGML (American Association of Publishers (AAP) DTD)	1985
`<div1><head>Interest Rates</head>...`	SGML (TEI)	1989
`<H1>Interest Rates</H1>`	SGML (HTML)	1989
`<sect1><title>Interest Rates</title>`	XML (DocBook)	1995
`<h1>Interest Rates</h1>`	XML (HTML5)	2005

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

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

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

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.

1.2 Choosing your LaTeX processor

Before you go any further there is one configuration you SHOULD check. As you may have seen in the list on page xxiii, there isn't just one flavour of LaTeX.

Plain LaTeX: For many years, there was only LaTeX, which (like TeX) produced a `.dvi` (Device-Independent) file, which had to be converted to Postscript in an additional step;

pdfLaTeX: In the 1990s, Hàn Thế Thành developed PDFTeX, which (along with *pdfLaTeX*) produced PDF directly, as well as adding the possibility of benefits like microtypographic adjustments;

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

XeLaTeX: More recently, Jonathan Kew developed XeTeX, which not only recognises Unicode Transformation Format — 8-bit (UTF-8) characters directly, but can also use your system's natively-installed TrueType Fonts (TTFs) and OpenType Fonts (OTFs) as well as those which come with LaTeX;

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

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

Exercise 3 – Set your LaTeX and BibTeX processors

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

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

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

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

There are still a few reasons some users stay with *pdfLaTeX* or even the original LaTeX. These include:

- a few packages (now a very small number) which positively require a processor which creates an old-style `.dvi` file;
- some specific packages still rely on raw Postscript features which need DVI-to-Postscript conversion first, before the PS output can be converted to PDF; and some DVI-to-Postscript converters cannot handle the (eXtended DVi (XDV)) format produced by XƎLaTeX;
- there are some other toolchains which depend on DVI files;
- some older editors do not yet make it possible to select XƎLaTeX or LuaLaTeX as the processor[1]

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

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

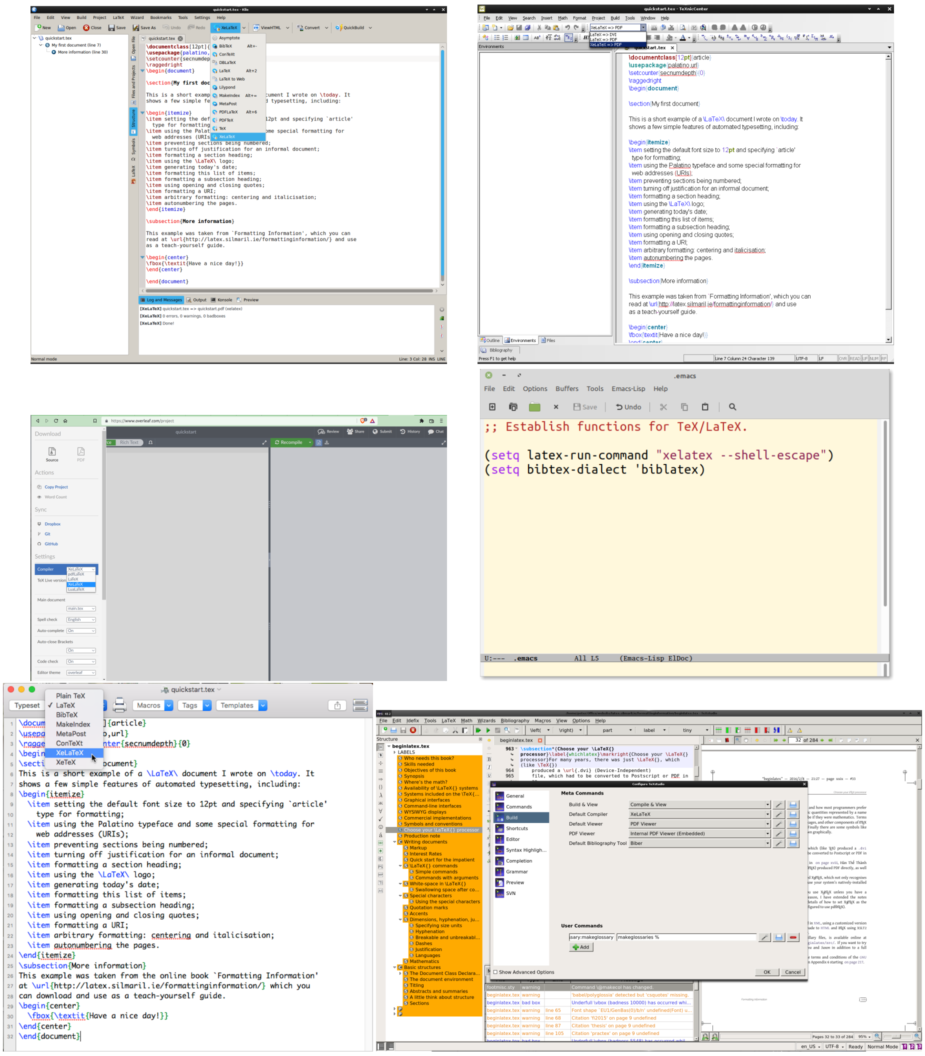
1.3 Picking an Editor

One of the best features of 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 come across this kind of flexibility in software before, and may be unfamiliar with the

[1] Some which are straightforward are shown in below.

Figure 1.1 – Some LaTeX editors being configured to use XeLaTeX



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

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

Table 1.1 – Default editors with different distributions of TEX

System	Package	Engine	Editor	Viewer
Microsoft *Windows*	*ProTEXt*	*MiKTEX*	*TEXStudio*	Adobe *Acrobat Reader*
Apple Mac OS X	*MacTEX*	*TEX Live*	*TEXshop*	*Preview*
Unix and GNU/Linux	*TEX Live*	*TEX Live*	*Kile*, *TEXStudio*, or *Emacs*	*Okular*, *Evince*, *qpdfview*, etc

So for beginners, in the case of *ProTEXt* (Windows) and *MacTEX* (Macs) 'the editor' is the one that comes with the distribution (*TEXStudio* and *TEXshop* respectively, see below). Unix and GNU/Linux users need to choose and install an editor separately.

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

Editor	Mac OS X	Linux	Windows
Emacs	✔	✔	✔
TEXMaker	✔	✔	✔
LyX	✔	✔	✔
texifier	✔	✗	✗
Kile	✗	✔	✔

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

Choose your editor carefully

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

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

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

   There are three classes of answer:

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

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

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

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

1.4 Quick start for the impatient

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

Exercise 4 – Quick start

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

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

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

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

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

1.5 LaTeX commands

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

1.5.1 Commands used in the example

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

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

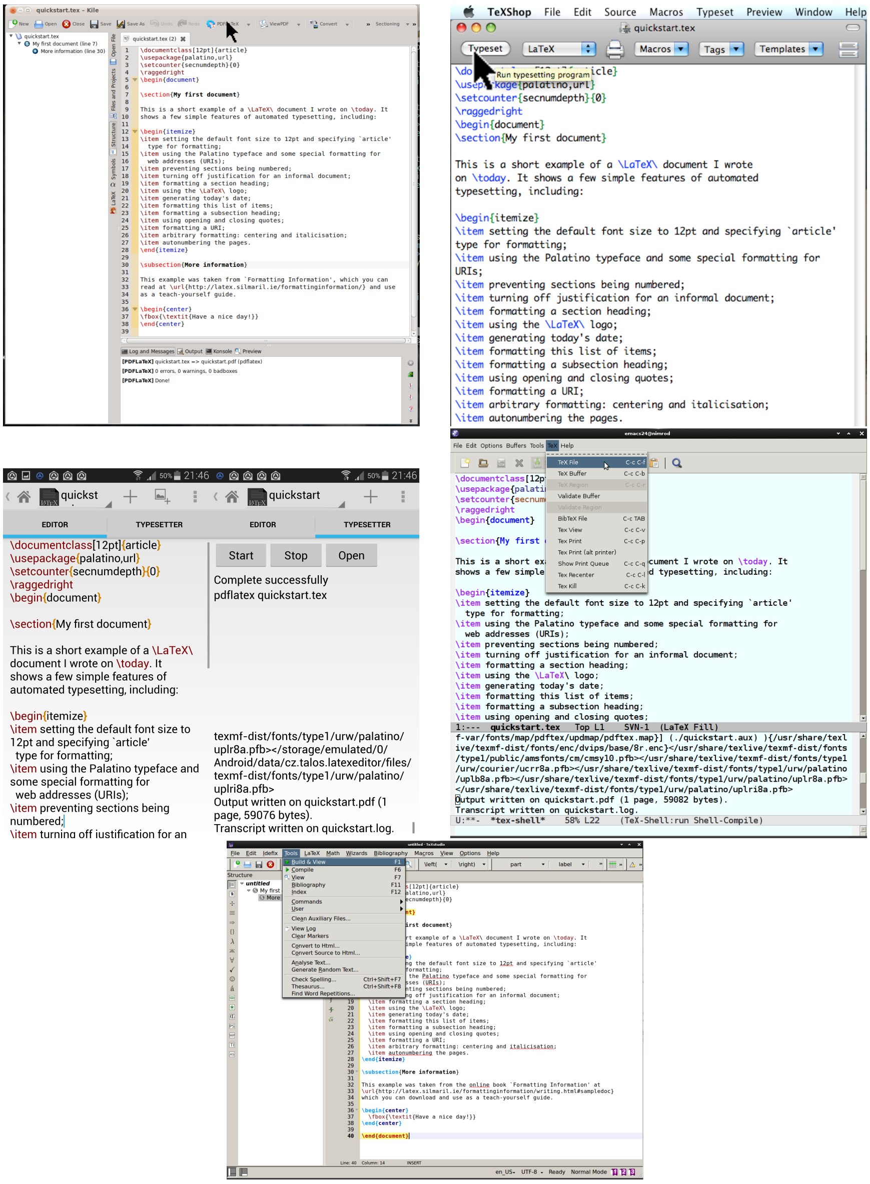
`\usepackage` tells LaTeX to use the named packages (plugins), here fontspec (to use all fonts) and url (provides a way to format URIs);

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

Figure 1.2 – Quick-start example document text

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

\section{My first document}

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

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

\subsection{More information}

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

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

\end{document}
```

Figure 1.3 – What to click on to typeset a document



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

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

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

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

The Preamble

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

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

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

`\section` identifies a section heading;

`\LaTeX` typesets the LaTeX logo;

`\today` typesets today's date;

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

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

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

`\subsection` identifies a subsection heading;

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

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

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

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

`\end` ends the *center* environment;

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

1.5.2 Simple commands

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

```
\today
```

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

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

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

Backslashes and forward slashes

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

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

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

1.5.3 Commands with arguments

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

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

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

Exercise 5 – Braces or not

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

Warning

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

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

1.6 White-space in LaTeX

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

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

There are three simple rules for spacing in LaTeX (see the Note on p. 19) — the first two say that, essentially, multiple spaces and blank lines are ignored in the typesetting. That means you are free to use them *in your editor* for optical ease and personal convenience when editing.

Three rules for spacing in LaTeX documents

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

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

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

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

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

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

\tableofcontents




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

The additional spacing or braces is not needed if

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

If you forget the white-space, like this:

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

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

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

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

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

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

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

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

```
\label{pform}

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

Exercise 6 – To space or not to space

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

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

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

Table 1.2 – Special characters in LaTeX

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

1.7 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 below.

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

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

% The ***comment character*** makes 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
```

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

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

~ The ***tilde*** in LaTeX prints as a normal space, but prevents a linebreak ever occurring at that point. It's often used between a person's initials and their surname, such as `P.~Flynn`, in case it might fall at the end of a line where a linebreak would make it harder to read.

& The ***ampersand*** is used in tabular setting (rows and columns) to separate the cell values within each row. We'll see how this works in section 4.2 on page 95.

The ***hash mark*** or ***octothorpe*** is the American 'pound' [weight] or 'number' sign.

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

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

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

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

Ordinal superscripts

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

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

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

1.8 Quotation marks

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

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

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

☞ He said, "I'm just going out."

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

DO NOT use the unidirectional typewriter single-quote ' key (apostrophe) or double-quote " key (quotes) for opening quotes: LaTeX treats these as closing quotes only.

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

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

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

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

1.9 Accents

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

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

Browser fonts and spacing

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

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

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

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

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

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

If you don't have accented letters on your keyboard

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

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

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

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

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

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

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

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

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

[3] Remember not everyone is lucky enough to be able to install new software: many users on business and academic networks still use old versions of 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 1.3 – Symbolic notation for Latin-alphabet accents

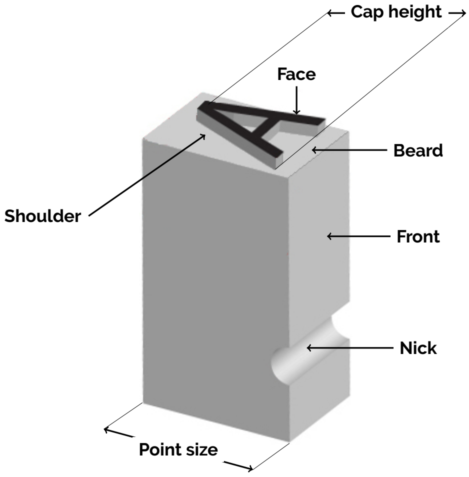
Accent	Example	Characters to type
Acute (fada)	é	`\'e`
Grave	è	`` \`e ``
Circumflex	ê	`\^e`
Umlaut or diæresis	ë	`\"e`
Tilde	ñ	`\~n`
Macron	ō	`\=o`
Bar-under	o̲	`\b o`
Dot-over (séımıú)	ṁ	`\.m`
Dot-under	ṣ	`\d s`
Breve	ŭ	`\u u`
Háček (caron)	ň	`\v n`
Long umlaut	ő	`\H o`
Tie-after	o͡o	`\t oo`
Cedilla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

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

1.10 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. While modern high-resolution displays use pixels smaller than you can easily see, many older screens cannot show dimensions of less than 1/96″ without resorting to magnification or scaling; and on printers, even at 600dpi, fine oblique lines or curves can still sometimes be seen to stagger the dots.

Figure 1.4 – Some parts of a piece of metal type



At the same time, many dimensions in 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. This is quite different from the 'grid' system used in many other typesetting and DTP systems.

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 most wordprocessors and DTP systems.

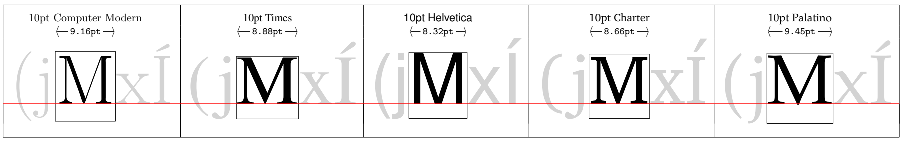
Occasionally, however, you will need to hand-correct an unusual word-break or line-break, and there are facilities for doing this on individual occasions as well as automating it for use throughout a document.

1.10.1 Specifying size units

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

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

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



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

The em can cause beginners some puzzlement because it's based on the 'point size' of the type, which is itself misleading. The point size refers to the depth of the metal body on which foundry type was cast in the days of metal typesetting, *not* the printed height of the letters themselves (see Figure 1.4 on the previous page). Thus the letter-size of 10pt type in one typeface can be radically different from 10pt type in another (look at above, where the widths are given for 10pt type). An em is the height of the type-body in a specific size, so 1em of 10pt type is 10pt and 1em of 24pt type is 24pt. A special name is given to the 12pt em, a 'pica' em, and a pica has become a fixed measure in its own right. An old name for a 1em space is a 'quad', and LaTeX has a command \quad for leaving exactly that much horizontal space.

Table 1.4 – Units in LaTeX

Unit	Size
	Printers' fixed measures
pt	Anglo-American standard points (72.27 to the inch)
pc	Pica ems (12pt)
bp	Adobe's 'big' points (exactly 72 to the inch)
sp	TeX's internal 'scaled' points (65,536 to the pt)
dd	Didot (European standard) points (67.54 to the inch)
cc	Ciceros (European pica ems), 12dd)
	Printers' relative measures
em	Ems of the current point size (historically the width of a letter 'M' but see Figure 1.5 on the facing page)
ex	x-height of the current font (height of a letter 'x')
	Other measures
cm	centimeters (2.54 to the inch)
mm	millimeters (25.4 to the inch)
in	inches (obsolete except in UK and parts of North America)

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

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

1.10.2 Hyphenation

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

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

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

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

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

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

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

1.10.3 Breakable and unbreakable text

Unbreakable text is the opposite of discretionary hyphenation. To force 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 size of the text: pneumonoultramicroscopicsilicovolcanoconiosis, although if you're reading this in a browser, you probably won't see the effect properly: look at the PDF.

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

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

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

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

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

1.10.4 Dashes

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

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

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

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

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

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

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

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

Em rules *vs* En rules

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

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

Leila Singleton wrote:

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

Christopher Maden wrote:

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

1.10.5 Justification

The default mode for typesetting in 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 ('tracking'), but this is best left to the expert.

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

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

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

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

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

Be careful when centering headings or other display-size material: it's one of the rare occasions when you may need to add a premature linebreak or forced newline (the double-backslash \\) to make the lines break at sensible pauses in the meaning (Flynn 2012). *Never* rely on the automated line-breaking of editors in these cases.

White-space and the double backslash

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

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

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

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

1.10.6 Languages

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

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

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

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

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

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

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

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

1.11 Mathematics

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

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

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

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

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

Delimiters

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

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

You SHOULD therefore use the LaTeX method of `\(…\)` and `\[…\]` — Barbara Beeton writes:

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

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

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

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

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

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

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

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

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

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

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

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

2 Basic structures

The Quick Start exercise in section 1.4 on page 11 was enough to show how a LaTeX document works. Now we're going to start looking at how a larger document is put together. If you skipped the whole of Chapter 1 starting on page 1, 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 based on *consistency*. This means that as long as you identify each component 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.

Consistency helps make documents easier to read and understand, as well as making them more visually attractive. Consistency is also what editors, reviewers, and publishers look for. Publishers 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 elements: parts, chapters, sections, subsections, headings, titles, subtitles, paragraphs, lists, tables, figures, sidebars, panels, exercises, and so on, even if they don't consciously think about them.

2.1 The Document Class Declaration

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 a computing science word for 'type'.

2.1.1 Document classes

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 a \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; and for theses[2] and dissertations;

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

book for books, booklets, or whole journals;

letter for letters.[3]

[1] Readers familiar with SGML, HTML, and XML will recognise 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. Many universities provide a special **thesis** class of their own.

[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 fairly basic in terms of layout and design, in order to make them easier to customise by adding *packages*, which are the style and layout plug-ins that LaTeX uses to let you automate formatting and change the design of your documents. Packages and classes are explained in more detail in Chapter 3 starting on page 65.

The **article** class in particular can be used for almost any short piece of typesetting by simply omitting the titling, changing the layout, and adding the relevant packages — like we saw in the Quick Start document in section 1.4 on page 11.

The **letter** class is not much used: it provides a very old-fashioned layout. There are other more up-to-date classes for letters available for download.

2.1.2 Extending the default classes

The built-in classes are intended as starting-points, especially for drafts, and for compatibility when exchanging documents with other LaTeX users. They come built into every installation of LaTeX and if left unmodified, are guaranteed to format identically everywhere. They are *not* intended as final-format publication-quality layouts, and should not be used as such. For most other purposes, especially for publication, you use LaTeX packages to extend these classes to do what you need. Some common ways to do this are:

- 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 (some come with LaTeX, others are on the publishers' web sites for download);
- Conference organisers may also provide class files for authors to write papers for submission, presentation, preprints, and proceedings;

- Many universities provide their own thesis document classes in order to ensure exact fulfilment of their formatting requirements (many of these are on CTAN);
- Businesses and other organisations can provide their users with private corporate classes on a central server and configure LaTeX installations to look there first for packages, fonts, etc (not usually available to the public, of course);
- There are nearly 300 document classes on CTAN (see www.ctan.org/topic/class).

The four default built-in document classes are therefore adequate for drafts or for sending to a colleague to edit, but they are not really usable for final-format publishing. For this you need to use packages to design it yourself, or (better) use a class file designed by your publisher or institution (or yourself!) to fit the type of publication. Quite often these are based on the default classes for compatibility, but typeset quite different output.

2.1.3 Document class options

The default layouts were originally designed to fit as drafts on US 'Letter' size paper.[4] To create documents with similar margins for A4 paper, you need to specify the paper size in an optional argument in square brackets before the document class name, eg

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

Many TeX systems now install the *a4paper* option as the default, so this may not be needed; on the contrary, North American users may now need to specify the *letterpaper* option instead. The

[4] 'Letter' size is 8½″×11″, which is the trimmed size of the long-obsolete Demy Quarto, still in use in North America. Other common US office sizes are 'Legal', which is 8½″×14″, a 'bastard' (variant) cutting close to the old Foolscap (8¼″×13¼″); Ledger or Tabloid (11″×17″, which is exactly twice 'Letter', in the same way that A3 is twice A4); and 'Executive' (7″×10″). International Organization for Standardization (ISO) standard 'A', 'B', and 'C' paper sizes, used everywhere else, are still largely unknown in many parts of North America.

geometry package, which we will see later, lets you specify other bigger and smaller paper sizes.[5]

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 and the font size is too small, because the finished job will normally be trimmed to a completely different size entirely — try printing a few pages of the PDF version of this chapter and then trimming the margins to make it 188 mm × 235 mm (the same as the *Companion* series) and you'll be amazed at how it changes the appearance.

The other default settings in the built-in classes are for:

1. 10pt type (all document classes);
2. two-sided printing (books and reports) or one-sided (articles and letters);
3. 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 (the *10pt* option is the default):

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

12pt to specify 12pt type (again, headings etc get scaled to match);

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

twoside to format articles or letters for two-sided printing;

titlepage to force articles to have a separate title page (books and reports get that automatically);

[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 the PostScript or PDF output. For this you need the geometry package 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.

draft makes LaTeX highlight any hyphenation or justification problems with a small square in the right-hand margin so they can be located quickly by you or a proofreader. This option also sets graphics to print as an empty rectangle containing just the filename of the image, so that image-heavy drafts will print more quickly and use less ink or toner.

So, 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 99.9% of all common text-document typesetting. There are extra options for other body type sizes in the extsizes bundle of document classes (extarticle, extbook, extreport, etc), and various national and international organisations supporting the visually-impaired have special large-type document class options.

Exercise 7 – Create a new document

1. Use your editor to create a new, empty document
   If your editor insists on filling your new document with template material, delete it all so that the file is empty;
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`.

2.2 The document environment

After the Document Class Declaration, the text of your document is enclosed between the two commands we saw in section 1.5

Global options

In addition to any options specific to the document class, it is also possible to put package options in the \documentclass options argument *instead of* in the \usepackage command (see section 3.1.2 on page 67), provided they are not implemented by more than one package. Packages which do not implement the named option at all are supposed to silently ignore it.

Picking suitable filenames

Never, *never*, NEVER create directories (folders) or file names which contain spaces or non-printing, non-ASCII characters. Although your operating system may support them, some don't, and they will only cause grief and tears, especially in automation software like document builders, web scripts, and app-based remote compilers.

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 (-), the underscore (_), and the dot (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, make your files more portable, and make it easier to use standard system utilities and applications, as well as those distributed with TEX

on page 12: \begin{document} and \end{document}. These identify the beginning and end of the text of your document (so in the example below, you would put your text where the dots are):

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

\begin{document}
...
```

```
\end{document}
```

The reason for marking the beginning of your document text is that LaTeX allows you to insert your setup and design specifications before it (where the blank line is in the example above: we'll be using this soon).

The reason for marking the end of your document text is that LaTeX stops processing at that point. You can therefore store comments or temporary text *underneath* the `\end{document}` in the knowledge that LaTeX will never see them and will never try to typeset them (they don't even need to be preceded by the `%` comment character), but they will remain in your file for you to see in your editor, maybe to copy and paste for re-use in a later edit.

```
...
\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 each time).

If you're familiar with HTML, SGML, or XML you'll recognise this technique: it's just like start-tags and end-tags.

Exercise 8 – Adding the document environment

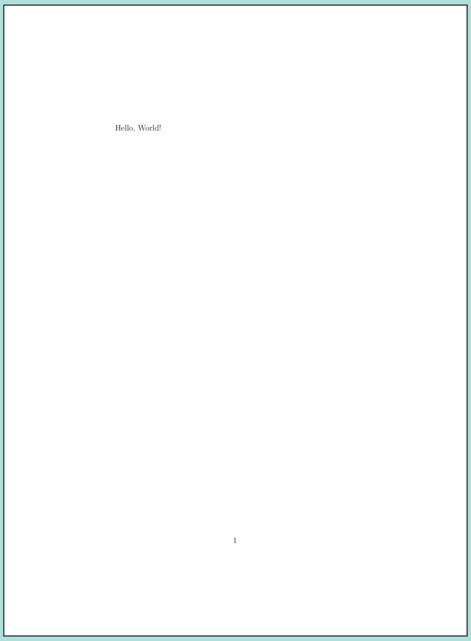
1. Add the *document* environment to your new file;
2. In between the Document Class Declaration and the `\begin{document}`, add the command which allows the use of UTF-8, TrueType, OpenType, and more:

```
\usepackage{fontspec}
```

3. In the *document* environment, type the phrase `Hello, World!`:

```
\begin{document}
Hello, world!
\end{document}
```

4. Save the file and typeset it with XƎLATEX and you should get some PDF output like this:



2.3 Titling

The first thing you actually 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* (information *about* information).

```
\documentclass[11pt,a4paper,oneside]{report}
\begin{document}
\title{Practical Typesetting}
\author{Peter Flynn}
\date{January 2022}
\maketitle
\end{document}
```

The `\title`, `\author`, and `\date` commands are 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. If you don't want a date at all, use `\date{}` (ie an empty date).

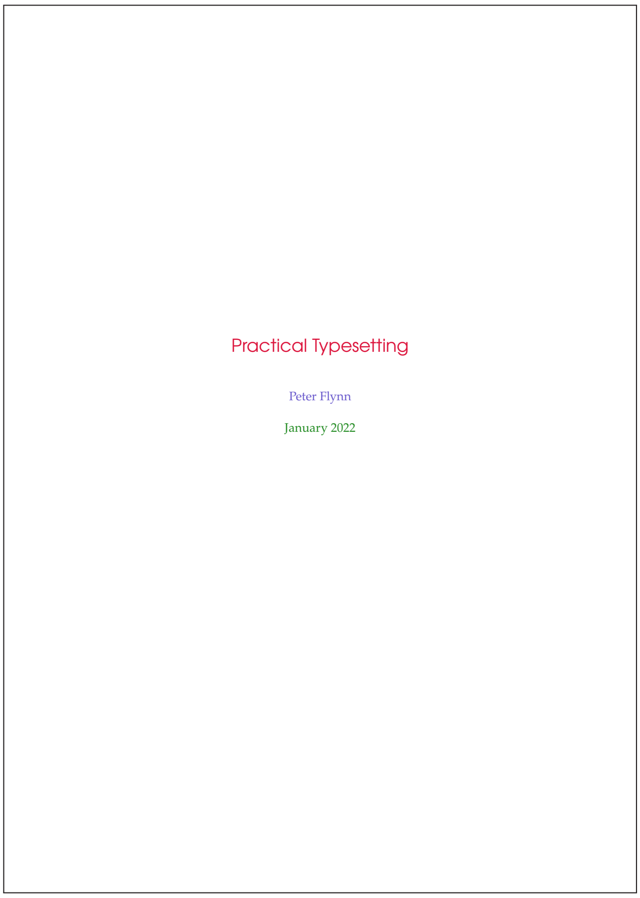
You MUST finish the metadata with the `\maketitle` command, which tells LaTeX that it's complete and it can typeset the titling information at this point. If you omit `\maketitle`, the titling will never be typeset.

Different title layouts

The `\maketitle` command, like all LaTeX commands, is reprogrammable, so you can alter the appearance of titles (like I did for the printed version of this document). It also means publishers can create new commands like `\datesubmitted` and `\editversion` in their own document classes. Details about reprogrammability are in Chapter 7 starting on page 203.

When this file is typeset, you get something like below (I've cheated and done it in colour for fun — yours will be in black

Figure 2.1 – Titling information typeset on the title page



Practical Typesetting

Peter Flynn

January 2022

and white for the moment). This is a report, so the title appears all by itself on a single page.

If you have mistyped a command, you may get an error message: see section B.3 on page 290 to resolve this.

2.4 Abstracts and summaries

In reports and articles it is usual for the author to provide an Summary or Abstract, which describes the content and explains its importance. Abstracts in articles are usually only a couple

Exercise 9 – Adding your 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.
3. Typeset the document and check that it's right.

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

of paragraphs long. Summaries in reports or theses can run to several pages, depending on the length and complexity of the document or the readership it's aimed at.

```
\documentclass[11pt,a4paper,oneside]{report}
\usepackage{fontspec}
\begin{document}
\title{Practical Typesetting}
\author{Peter Flynn}
\date{January 2022}
\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}
```

In both cases (reports and articles) 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, and it's used by web indexing engines to let people find your work. 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.

You add the *abstract* environment *after* the \maketitle command,[6] and type your Abstract or Summary in it, leaving a blank line between paragraphs if there's more than one (see section 2.7 on page 62 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 other suitable text. The heading associated with the *abstract* environment is called the \abstractname, and you can use the \renewcommand command in your Preamble to give it a new value:

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

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

Exercise 10 – Using an Abstract or Summary

1. Add the \renewcommand as shown above to your Preamble (call it something other than Summary if you prefer).

   (The Preamble is at the start of your document, in between the \documentclass line and the \begin{document}: see the panel 'The Preamble' on p. 15preamble).);
2. Add an *abstract* environment after the \maketitle and type in a paragraph or two of text.
3. Typeset the document.

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

[6] A few publishers' journal style packages ask for the Abstract to be typed *before* the \maketitle because they do special formatting with it along with the title block.

second set of curly braces (this is an example of a command with *two* arguments).

2.5 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 short-lived or relatively unimportant, then you just want to type it in and make it 'look nice' 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 not important for trivial, ephemeral, or short-term documents. Sometimes, though, you may need to write longer, more permanent, or more complex documents, or documents arranged to a regular pattern like reports or articles. Making them consistent by manual methods then becomes a nightmare, and an enormous waste of time, because everything has to be formatted and reformatted by hand.

LaTeX's automation is based on you providing the 'why' information, identifying the elements of your document by name, and letting the template or stylesheet take care of the formatting.

If you've got this 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.

2.6 Sections

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

Exercise 11 – Reasoning

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

- ☐ My document naturally divides into sections (parts, chapters, etc).
- ☐ My document is long.
- ☐ There is lots of repetitive formatting in my document.
- ☐ My document is complex (intellectually or visually).
- ☐ There are lots of figures or tables (or examples, exercises, panels, sidebars, etc) in my document.
- ☐ Accuracy is important in formatting my document.
- ☐ A master copy of my document is needed for future reference or reprinting.
- ☐ This is a formal or official document needing special care and attention.
- ☐ My document (or part of it) may need ongoing or occasional re-editing and republishing.
- ☐ It's *my* thesis, book, white paper, leaflet, pamphlet, paper, article, etc. *That*'s why I care.

of unstructured text. But even novels are normally divided into chapters, although short stories are often made up just of paragraphs.

Chapters are only available in the book and report document classes, because chapters don't have any meaning in articles or letters. Parts are also undefined in letters.[7]

[7] 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 section 7.6 on page 213).

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 are in below.

```
\section{New recruitment policies}
...
\subsection{Effect on staff turnover}
...
\chapter{Business plan 2020--2030}
```

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 section 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 section 6.1.2 on page 165). This alternative heading goes in [square brackets] before the curly braces:

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

Table 2.1 – LaTeX's sectioning commands

Depth	Division	Command	Notes
−1	Part	\part	Not in letters
0	Chapter	\chapter	Books, 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

```
on staff turnover at divisional headquarters}
```

Exercise 12 – 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. Typeset the document.

2.6.1 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 the depth is set to 3, using the depth column in Table 2.1 on the facing page. If you only want parts, chapters, and sections numbered, not subsections, subsubsections, or lower levels, you can change the value of the `secnumdepth` counter using the the `\setcounter` command, giving the depth value from Table 2.1 on the preceding page:

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

Notice that the `\setcounter` command, like `\renewcommand` which we saw earlier, has two arguments: the name of the counter you want to set, and the number you want to set it to.

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

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

To get a 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.

2.6.2 Table of contents

All auto-numbered headings (parts, chapters, sections, subsections, etc) get entered in the Table of Contents (ToC) automatically. You don't have to print a ToC, but if you want to, 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 typeset your document, and only reproduced the *next* time you typeset it, so you need to run LaTeX an 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 section 2.6 on page 56 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.

Table of Contents automation

Your editor should automatically run LaTeX twice when needed, if you are using the Build, Compile, Typeset, or Make button or menu entry (see Figure 1.3 on page 14). It is also done automatically by processing tools like *latexmk*, but if you are processing LaTeX manually by typing the commands in a terminal window, you will need to do this yourself.

Exercise 13 – Using a Table of Contents

1. Add the `\tableofcontents` command to your document, before or after the Abstract, as you prefer.
2. Typeset the document.
3. Check that the Table of Contents is now showing. If not, typeset the document again.

If what you expect doesn't appear, you should always check the log file or error display: you might have made a typing mistake in a command.

A `\tableofcontents` command normally shows only numbered section headings, and only down to the level defined by the `tocdepth` counter (see section 2.6.1 on page 59), 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 at some special place, 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.

There are several packages to help you restyle these lists of contents automatically; perhaps the best-known is tocloft.

2.7 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 'end the current paragraph here': it is *not* (repeat: ***not***) for creating a blank line in the typeset output.

Multiple blank lines

> Leaving multiple blank lines between paragraphs in your source document does *not* create extra white-space. As we saw in the Note on p. 19swallow, all extra blank lines are ignored by LaTeX: the space between paragraphs is controlled *only* by the value of `\parskip`.

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 and articles are normally typeset, but see below), but you can easily set it to any size you want with a `\setlength` command in your Preamble — like the `\setcounter` command we saw in section 2.6.1 on page 59 it takes two arguments: the name of the length, and the value to set it to:

```
\setlength{\parskip}{5mm}
```

This will set the space between paragraphs to 5mm. See section 1.10.1 on page 30 for details of the various size units LaTeX can use.

Most books and articles are set with no space between paragraphs (and indentation at the start of them). If you want to use the popular office-document style of having space between paragraphs (and no indentation), use the parskip package, which does it for you. It also makes adjustments to the spacing of lists and other structures which use paragraph spacing, so they don't get too far apart.

White-space in LaTeX can also be made flexible (what 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 can specify this in a `\setlength` command:

```
\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 chapter or section heading follows the standard Anglo-American publishers' practice of *no* indentation. Subsequent paragraphs are indented by the value of `\parindent` (default 18pt).[8] You can change the value of `\parindent` in the same way as any other length.

[8] Paragraph spacing and indentation are cultural settings. If you are typesetting in a language other than English, you should use the babel or polyglossia packages, which alter many things, including the spacing and the naming of sections, to conform with the standards of different cultures, countries, and languages.

In the printed version of this document, the paragraph indentation is set to 12.0pt and the space between paragraphs is set to 0.0pt plus 1.0pt. These values do not apply in the Web (HTML) version because not all browsers are capable of that fine a level of control, and because users can apply their own stylesheets regardless of what this document proposes.

Exercise 14 – Start typing!

By now you know enough about LaTeX's basic commands to write a whole document.

1. Give a title and your name as author (and a date if you want) and don't forget the `\maketitle` (see Figure 2.1 on page 53);
2. Add an Abstract if you need to (see section 2.4 on page 53).
3. Start with a `\chapter` (for books or reports) or a `\section` (for articles) with the title of the chapter or section.
4. Type your text in paragraphs. Leave a blank line between each. Don't bother about line-wrapping or formatting — LaTeX will take care of all that.
5. Use the geometry package to change the margins and size of the text body.

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 easily be able to tell where each paragraph begins!

3 Plugins and support

In earlier chapters we have already seen the use of 'classes' (document templates), and 'packages' (styling add-ons or plug-ins) which add features to your document. Most of the packages can be used with any document class. Both classes and packages are stored in subdirectories of your TEX installation directory.

Packages: A LATEX ***package*** is a file or collection of files containing extra commands and programming which add new formatting features, or modify those already existing. There over 6,000 packages, and over 5,000 of them are pre-installed with every full distribution of LATEX, ready to be used in your documents immediately.

Package files mostly end with the `.sty` filetype (some computers hide this, but it's there) — they used to be called 'style files'. Packages may also include other files as well, like fonts or configuration files.

Classes: A LATEX ***class*** is a special kind of package which provides formatting template features for a whole document. There are over 600 of these, and over 500 of them are pre-installed with LATEX.

Class files all end with the `.cls` filetype (some computers hide this, but it's there). Like ordinary packages, classes may also be distributed with ancillary files, like fonts or configuration files.

To find out if a particular package or class is already installed, type the kpsewhich command in a Terminal window (see Appendix 2 starting on page 285); for example to see if the noto package (or the Noto font family) or the **scrartcl** class (for the Koma-Script[1] 'article' class) is installed, type:

```
$ kpsewhich noto.sty
/usr/share/texlive/texmf-dist/tex/latex/noto/noto.sty
$ kpsewhich scrartcl.cls
/usr/share/texlive/texmf-dist/tex/latex/koma-script/scrartcl.cls
```

If the class or package is installed, this will tell you where the file is. If you get no response, it means it is not installed.

To find out what other packages are available and what they do, use the CTAN search page or Graham Williams' comprehensive package catalogue.

3.1 Using classes and packages

You should be familiar with this by now from earlier chapters.

3.1.1 Using a document class

We've already seen how to do this in section 2.1.1 on page 44: it's the document class name that you put in curly braces in the \documentclass line at the start of a LaTeX document.

```
\documentclass{book}
```

Most classes have *options* (we saw some in use in the Quick Start document in section 1.4 on page 11). The class documentation will explain what they are for and how to use them. Read it!

[1] Koma-Script is a bundle of alternative classes by Markus Kohm.

3.1.2 Using a package

To use a specific package, put its name in the argument of a \usepackage command in your Preamble, eg to use the Noto typeface:

```
\documentclass{book}
\usepackage{noto}
\begin{document}
...
\end{document}
```

Note

See section 3.2 on page 70 for how to install extra packages that weren't automatically installed with your distribution of TeX.

For another example, to use the xcolor package, which lets you typeset in colours (I warned you this was coming!), you would for example type:

```
\usepackage{xcolor}
```

This makes available the \color command and many others, and several sets of predefined palettes of colours which you can specify using options.

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.

If the package has options that you want to use, you must give the package its own separate \usepackage command, like geometry and xcolor shown below. How do you know if a package has options and what they are? Read the documentation!

```
\documentclass[11pt,a4paper,oneside]{report}
\usepackage{fontspec,graphicx}
\setmainfont{TeX Gyre Pagella}
\setsansfont{TeX Gyre Adventor}
```

```
\usepackage[svgnames]{xcolor}
\usepackage[margin=2cm]{geometry}
\begin{document}

\title{\sffamily\color{Crimson}Practical Typesetting}
\author{\color{SlateBlue}Peter Flynn}
\date{\color{ForestGreen}January 2022}
\maketitle

\end{document}
```

(Incidentally, don't actually do it this way: it's a very crude and cumbersome way to do colours in titling, and I only did it here for brevity. It's fine for a one-time short 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 7 starting on page 203 how it can be automated as part of the `\maketitle` command and kept out of the author's way.)

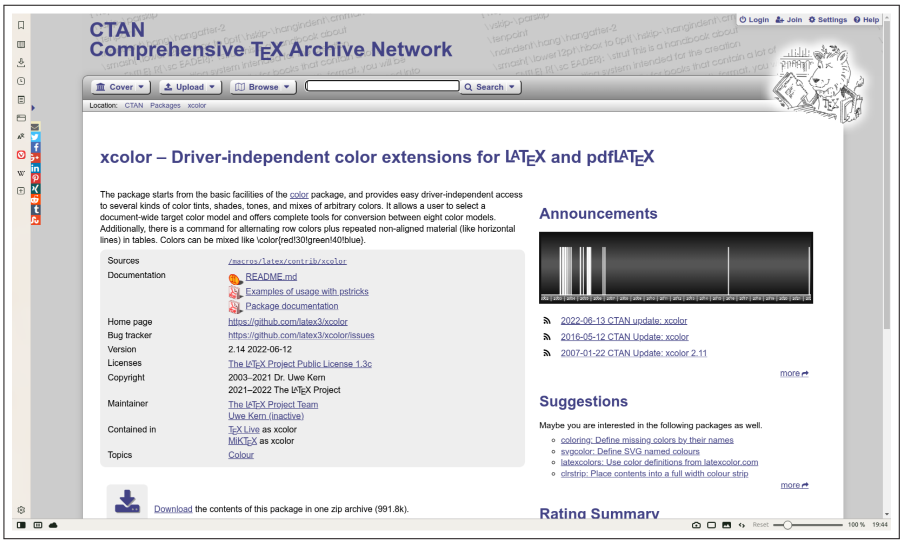
Exercise 15 – Add colour and change page shape

1. Use the xcolor package to add some colour to your document. Stick with primary colours for the moment, or read the xcolor documentation to see all the named colours available.
2. Use the geometry package to change the margins.
3. Reprocess and print your document if you have a colour printer (monochrome printers should print it in shades of grey).

The geometry package has options to let you specify margins, page and paper sizes, header and footer depths, and a lot of other page-geometry dimensions. The xcolor package has options to let you specify which of several standard palettes of colours you want to use.

It's really important to read the documentation for the package concerned to find out what can be done and how to do it: see section 3.1.3 on the facing page.

Figure 3.1 – The CTAN page for a package (xcolor)



3.1.3 Package documentation

To find out what features a package provides (and thus how to use it), you need to read the documentation. The simplest way is to search the web for CTAN followed by the package name. The top link should be the package folder on CTAN where you can click on the PDF where it says **Package documentation** (see below).

Alternatively, use your terminal (command) window and type `texdoc` followed by the package name. This will open your local copy of the documentation in your PDF viewer — assuming you have the package installed. You could also use your system's file finder to look for the package name — it should turn up the package directory itself as well as the documentation directory — what you're looking for is a PDF document.

If that doesn't find it, in the `texmf/doc/latex` directory of your installation there should be subdirectories of `.pdf` files, one for every package installed, which you can view or print. If your installation did not include the documentation, it's all on CTAN in `www.ctan.org/pkg/` followed by the package name as shown in above.

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.

See the next section for details of how to generate the documentation for additional packages you install yourself.

Exercise 16 – Read all about it

1. Find and view (or print) the documentation on the xcolor package you used in Exercise 15 on p. 68geoex.
2. Find and view (or print) the documentation on the geometry package you used in Exercise 15 on p. 68geoex.
3. Browse some of the other package documentation installed on your system.

3.2 Installing extra classes and packages

You will not need this section if...

- ☐ ...you are using MiKTEX, which has a package auto-installer (also applies to ProTEXt);
- ☐ ...you are using a system which has the TEX Live Package Manager *tlmgr* and you have been using it to keep your system up to date.

This section is for people who have neither MiKTEX nor *tlmgr*, and for those occasions when you need to install an extra, private, experimental, or non-standard package that cannot be installed automatically, or one that is on CTAN but not included in your distribution.

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 section B.3.3.7 on page 295).

You then need to download the package and install it using the instructions in section 3.2 on the facing page.

Some TEX distributions can now catch this error, and will download and install the missing class or package for you right there and then, and carry on typesetting as if it had always been there. Currently the best known implementation of this feature is in the MiKTEX distribution for Microsoft Windows (also part of the ProTEXt distribution which is based on it).

In other systems there is the TEX Live Package Manager (*tlmgr*), which can download and install packages for you, but not in the middle of a LATEX run: you have to stop and run it separately. The *tlmgr* program is not yet available in all distributions of LATEX, so check your documentation to see if it is working in your version.

(Package Managers like these 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. Updates occur when a class or package author finds and fixes a bug, or adds a new feature. All package updates on CTAN are automatically announced on the Usenet newsgroup `comp.text.tex`.)

There is no limit to the number of packages you can have installed on your computer (apart from disk space). There is probably a limit to the number that can be used inside any one LATEX document at the same time, although it may depend on how big each package is. In practice there is no problem in having even a few dozen packages active (this document uses over 250 of them).

The Comprehensive TEX Archive Network (CTAN) is a repository of TEX and related software from HyperText Transfer Protocol (HTTP) and File Transfer Protocol (FTP) servers worldwide. It contains copies of almost every piece of free software related to TEX and LATEX, and especially, all the public packages and classes.

Private, experimental, or non-standard packages are normally stored on their author's server, or on a shared resource like GitHub. To use these, you must get the URI of the download page.

(By unfortunate historical accident, both classes and packages are usually managed and stored on CTAN in the same file format (DocTEX: a `.dtx` filetype) and are referred to generically as 'packages' even though some of them contain classes. We have to live with this.)

Always try CTAN first

CTAN should *always* be your first port of call when looking for a software update or a class or package you want to use and don't have installed. To help prevent overload on the volunteers, please don't ask in the network help resources until you have checked CTAN and the FAQ (section 3.3.3 on page 84) first.

CTAN is based on three main servers but if you go to `www.ctan.org/` to start with, you will be redirected to your nearest mirror server for downloads.

3.2.1 Downloading packages

CTAN packages are available as `.zip` files from any CTAN server.

- some are in TDS format, which is the same directory layout as your TEX installation, so it is faster and easier to use because it unzips directly into your Personal TEX Directory;
- others are just zip files with no directory structure in them: after unzipping, you have to move the files to 'The Right Places' yourself;
- if you prefer, you can download all the needed files one by one and move the files to 'The Right Places' yourself.

3.2.1.1 Downloading a TDS package zip file

If you go to a package's CTAN page (http://ctan.org/pkg/name) this will show the package details, and if the package is available in

The TEX Directory Structure

> The TEX Directory Structure (TDS) is the set of folders and subfolders used by all modern distributions of TEX and LATEX so that there are known, fixed places where files are stored. The TDS is documented at `www.tug.org/tds/`.

TDS format, there will be a link labelled 'TDS archive' with a file ending in `.tds.zip`.

Download this and unzip it straight into your Personal TEX Directory, where LATEX will find it first, overriding any other version that is installed. The location of your Personal TEX Directory is described in section A.3 on page 271.

MiKTEX (and ProTEXt) users must also run the Refresh FNDB function (see **3.** on page 274).

3.2.1.2 Downloading a non-TDS package zip file

If there is no TDS zip file, there will be a prominent link at bottom right labelled 'Download the contents of this package in one zip archive' (see the bottom of Figure 3.1 on page 69).

1. Download the zip file to a ***temporary directory***. Mac and Linux systems already have a `/tmp` directory for this purpose. On Windows, create a folder for this purpose like `Computer\System\Users\your name\temp` (or just use `C:\tmp` or `C:\temp`).[2]

2. Unzip the file into the `tex/latex` folder of your Personal TEX Directory. It will create a new subfolder in there called after the package name, and put all the package files in it. You can delete the zip file from the temporary directory afterwards as it is no longer needed.

[2] *MiKTEX* users note that you cannot process TEX Installer (`.ins` and `.dtx`) files inside *MiKTEX*'s own installation folders: you have to process them elsewhere first, hence the need for a temporary directory.

Font packages need to be unzipped into `tex/fonts` instead; otherwise the procedure is the same;

3. For classes and style packages, you MUST complete installation by extracting the `.cls` or `.sty` file[s] as shown in section 3.2.2 on the next page.

4. Ancillary files such as documentation, and (for font packages) the font files, need to be moved to 'The Right Place' (see Table 3.1 on page 79).

3.2.1.3 Manual download

If there is no `.zip` file at all, as will usually be the case off-CTAN and for private 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 program (much smaller). You MUST always download *both* these files, if they exist (and maybe others in the download folder). There are two exceptions noted in below.

3.2.1.4 Other package downloads

If neither the two files nor the package `.zip` are there, it means one of two things:

- *Either* the package is part of a much larger *bundle* which you SHOULD NOT update yourself unless you are updating your entire LATEX system;[3]

- *or* it's one of a growing number of packages supplied as a single `.dtx` file alone (no `.ins`);

- *or* it's one of a few rare or unusual packages still supplied as a single hand-made `.sty` or `.cls` file possibly written for

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

the now obsolete LaTeX 2.09,[4] or perhaps by an author who has a doctrinal or philosophical objection to using DOCTeX.

Warning

On Unix-based systems (including Mac OS X and GNU/Linux), all you need to do is unzip the TDS zip file into your Personal TeX Directory. On Windows systems running MiKTeX, you MUST reindex your File Name Database (FNDB) (see step **4.** on page 77) before LaTeX will be able to find the new files.

3.2.2 Installing a class or package manually

There are four steps to installing a non-TDS LaTeX class or package:

1. **Extract the class or package files from the `.dtx`/`.ins` files**

   Use your directory browser or file manager (eg *My Computer*, *Finder*, *Thunar*, *Dolphin*, etc) to find the subfolder in your Personal TeX Directory below `tex/latex` into which you unzipped or downloaded the package file[s].

   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 the directory where the file is.

   This will extract all the files needed from the `.dtx` file (which is why you must have both of them present in the directory).

2. **Create the documentation if not already done**

   Some packages come with the PDF documentation already there. If so, ignore this step.

   To [re]create the PDF documentation, run LaTeX on the `.dtx` file *twice*. This will create a `.pdf` file of documentation explaining

[4] Almost all of these have been updated to work with LaTeX 2_ε, so they should be installed as in step **3.** on the following page, but there are a few remaining.

Warning

If this is a non-TDS zip file, or individually-downloaded files, 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).

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. If there is a BibTeX file of references, or if you need the Index, you will need to process *bibtex*, *biber*, *makeindex*, or other ancillary programs. I *very strongly recommend* doing this with the Build menu of your editor, or with the *latexmk* tool.

3. **Install the files if needed**

   This step is only needed if you unzipped and processed the file in some other location (eg your temporary directory), or if the processing extracted more than just the `.cls` or `.sty` file. Other types of files belong in different places in the TDS.

   Move the files created in step **1.** on the previous page to the right subdirectories in your Personal TeX Directory as shown in Table 3.1 on page 79. *Always* put the files in subdirectories of your Personal TeX Directory, *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*, NEVER put files into your TeX distribution's main directory tree. If you are a system administrator updating a shared system, however, you SHOULD put the files into the *local* (shared) TeX directory tree.

   'The Right Place' sometimes causes confusion, especially if your TeX installation is old or does not conform to the TDS. 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 & GNU/Linux systems: `~/texmf/`.
- Apple Macintosh OS X: `~/Library/texmf/`.
- Windows systems: `Computer\System\Users\your_name\texmf` (on obsolete Windows systems you can use `C:\texmf`).

Create this directory if it does not already exist. You will need to create subdirectories within this directory: see Table 3.1 on page 79.

Often there is just a `.sty` file to move but in the case of complex packages there may be more, and they belong in the different locations shown in Table 3.1 on page 79. It is important to create a subdirectory for the package within your Personal TEX Directory, rather than dump the files into `misc` along with other unrelated stuff.

4. **Shared systems and MIKTEX: update your index**

   On Unix & GNU/Linux systems (including Apple Macintosh OS X) you MUST NOT run the TEX indexer program or create an `ls-R` database in your Personal TEX Directory. These systems search your Personal TEX Directory automatically.

   Otherwise:

   - ***Windows MIKTEX users*** (only) MUST use the MIKTEX Administration program to add your new Personal TEX Directory to MIKTEX's search tree when you first create it.

     After that, each time you update files in there, you MUST run the File Name Database (FNDB) updater in the MIKTEX Administration program, otherwise TEX will never see your newly-installed files.
   - If you are updating a shared system, putting the files into the local TEX directory tree, you MUST run your TEX indexer program afterwards to update the package database.

     This program comes with every modern version of TEX and is variously called *texhash*, *mktexlsr*, or even *configure*, or it might just be a mouse click on a 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 MIKTEX and shared systems

In these installations, you MUST un your TEX indexer program after making changes. See step **4.** on the previous page for details.

This step is *essential*, otherwise none of your changes will work.

Files in your working folder

It is possible just to unzip package files into your current working folder, where your document is, because LATEX will look there first, before looking in your Personal TEX Directory or anywhere else, but it means that if you use LATEX in a different folder for another document, you'll have to copy all the packages you put in there. Far better to put them all in The Right Place to start with: your Personal TEX Directory.

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

Table 3.1 – Where in your Personal TEX Directory to put files you install manually from packages

Type	Subdirectory of `texmf`	Description
`.afm`	`fonts/afm/`*foundry*`/`*typeface*`/`	Adobe Font Metrics
`.bst`	`bibtex/bst/`*packagename*`/`	BIBTEX style file
`.clo`	`tex/latex/`*classname*`/`	Document class options
`.cls`	`tex/latex/`*classname*`/`	Document class file
`.dtx`	`tex/latex/`*packagename*`/`	Package DOCTEX file
`.dvi`	`doc/`*packagename*`/`	package documentation
`.fd`	`tex/latex/mfnfss/`	METAFONT Font Definition files
`.fd`	`tex/latex/psnfss/`	PostScript Type 1 Font Definition files
`.fd`	`tex/latex/`*typeface*`/`	Other Font Definition files
`.ins`	`tex/latex/`*packagename*`/`	Package DOCTEX installer
`.jpg`	`tex/generic/`	JPG images
`.log`	`doc/`*packagename*`/`	package documentation log
`.mf`	`fonts/source/public/`*typeface*`/`	METAFONT font outlines
`.otf`	`fonts/opentype/`*foundry*`/`*typeface*`/`	OpenType fonts
`.pdf`	`doc/`*packagename*`/`	package documentation
`.pfb`	`fonts/type1/`*foundry*`/`*typeface*`/`	*PostScript* Type 1 outlines
`.png`	`tex/generic/`	PNG images
`.sty`	`tex/latex/`*packagename*`/`	Package (style) file
`.svg`	`tex/generic/`	SVG images
`.tfm`	`fonts/tfm/`*foundry*`/`*typeface*`/`	TEX Font Metrics
`.ttf`	`fonts/truetype/`*foundry*`/`*typeface*`/`	TrueType fonts
`.vf`	`fonts/vf/`*foundry*`/`*typeface*`/`	TEX virtual fonts
others	`tex/latex/`*packagename*`/`	unless instructed otherwise

- Every user SHOULD have a Personal TEX Directory to put extra stuff into. Create yours now (see section A.3 on page 271).
- If there are configuration or other files (`.cnf`, `.cfg`, etc), read the documentation to find out if there is a special or preferred location to move them to.

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 policy or security reasons (such as a shared system where she cannot write to a locked disk or directory).

The local `texmf` directory (usually called `texmf-local` or `texmf.local`) is a place for local modifications *on a shared or managed multi-user system* (such as a server) which will override but otherwise not interfere with the main TEX installation directory. 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` or the local `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 section A.6 on page 281.

Exercise 17 – Install a package

1. Download and install the latest version of the enumitem package.

This implements inline lists, among many other extra features for list formatting.

3.2.3 Creating the TDS structure

If you have a TEX distribution which has an auto-updater like TEX Live (*tlmgr*) or MiKTEX, you'll probably never have to update a package manually, so you won't need this section unless you want to install something from outside CTAN such as a private, corporate, or commercial package or a typeface.

The TEX Directory Structure (TDS) means you can make the subdirectory structure of your Personal TEX Directory the same as that of your main TEX installation. This way you can have all the branches of the tree you need ready for future use.

If you install packages or fonts using a TDS zip file (see section 3.2.1.1 on page 72), this is the directory structure that will be used: look at the subdirectories of `texmf/tex/latex/` and `texmf/fonts/` in your main TEX installation for examples. LATEX will always use a package or font in your Personal TEX Directory before looking elsewhere.

On Unix & GNU/Linux systems (including Apple Macintosh OS X) it is straightforward to recreate the entire subdirectory structure ready for use. using the commands in Exercise 18 on p. 81.

Exercise 18 – Replicating the TDS

1. Find the installation directory of your main TEX distribution:
   ```
   $ find / -type d -name texmf-dist 2>/dev/null
   ```
2. Change to that directory:
   ```
   $ cd/usr/share/texlive/texmf-dist
   ```
   (using whatever directory the first command identified);
3. Replicate its subdirectories:
   ```
   $ find . -type d -exec mkdir -p ~/texmf/{} \;
   ```
   On a Mac, use `~/Library/texmf` instead of `~/texmf`.
   The operation may take a minute to complete.

Windows appears to provide no way of doing this, but it may be possible using a *Powershell* script; or you could install *Cygwin*, which provides you with the standard Unix tools in a Command window.

3.3 Where to go for help

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

3.3.1 Beginners start here

If you haven't used online help before, please read Eric Raymond's advice in *How To Ask Questions The Smart Way* (Raymond 2014) which will save you and the volunteers who answer you a lot of time.

Will Robertson posted on Twitter: 'Some recent LATEX experiences that made me happy' (see sidebars).

A very valuable list of Dos and Donts is maintained on StackExchange listing the most common mistakes that newcomers make. Once you've got started with LATEX, especially if you have learned it informally from colleagues, it's worth having a look at this just to make sure you avoid the easiest pitfalls.

3.3.2 The Minimal [Non-]Working Example or MWE

If you want to send an example of what you're trying to do to one of the forums, mailing lists, or newsgroups listed here, you MUST send an Minimal [Non-]Working Example (MWE). This is your LATEX document pared right down to the bare metal: remove *all* non-relevant packages, *all* non-relevant commands and formatting, and send ONLY the absolute bare minimum necessary to show what doesn't work. Unless you do this, you are wasting everyone's time, including your own.

There is an excellent article by Nicola Talbot at `tug.ctan.org/info/dickimaw/dickimaw-minexample.pdf` which explains the procedure in fine detail (Talbot 2014).

And guess what? While doing this, you often find you discover for yourself what the problem was, saving you and thousands of others the trouble of working it out afresh!

Happy discoveries (1)

Fonts: Matching the font I was supposed to use in the template but having it also be sized sensibly when I write maths:

```
\usepackage[matchlowercase]{tgheros}
\renewcommand\familydefault{\sfdefault}
```

(See Exercise 30 on p. 201tryfonts for more on the scaling of fonts.);

Captions: A single line to have sane caption formatting (in particular to visually identify the caption *vs* the surrounding text):

```
\usepackage[format=hang,font=small,
            tableposition=top]{caption}
```

(See section 4.2.2 on page 98 for more on tables, figures, and captions.);

Loading packages: Alphabetical loading of packages FTW:

```
\usepackage{amsmath,biblatex,booktabs,censor,
            enumitem,geometry,graphicx,siunitx}
```

Just makes it easier to see whether or not you've loaded a particular package.

biblatex The biblatex package continuing to make it oh so easy to customise reference lists:

```
\renewcommand\bibfont{\small}
\renewcommand\bibitemsep{0pt}
```

Squeezing it (vertically): 'Surely I can squeeze one more line onto this page?' Yes I can:

```
\enlargethispage\baselineskip
```

(Try that in MS Word, hah!).

Happy discoveries (2)

Squeezing it (horizontally): 'Surely this figure could nudge into the margins and no-one would notice?' Yes:

```
\vspace*{-1.5cm}
\centerline{\includegraphics
            [width=1.1\linewidth]{flowchart}}
```

Math: [No commentary required]

$$T_n^{(k)} = T_n^{(k-1)} + \frac{S_n^*}{0.4132} \operatorname*{mean}_j \left(\psi \left(\frac{m_j - T_n^{(k-1)}}{S_n^*} \right) \right)$$

Redacted: 'Oops, this paper is double-blind reviewed, how do I blank out the things I need to make it anonymous?' — thank you censor package and `\censor` command for making it so easy.

Tables: Finally, evergreen appreciation for booktabs and siunitx, which make it trivial to create just such a beautifully typeset table like this:

	Assessor 1	Assessor 2	Assessor 3
Student 1	$\frac{60}{57} = 1.05$		$\frac{54}{57} = 0.95$
Student 2	0.95	1.05	
Student 3	0.95	1.05	
Student 4		1.1	0.9
Student 5		1.1	0.9
Student 6	1.05		0.95
$w_j = \mathrm{loc}(\hat{A}_j)$	1	1.075	0.925

3.3.3 The FAQ

For general queries you should read the Frequently-Asked Questions (FAQ) document so that you avoid wasting your time and others' by asking about things for which there is already an easily-accessible answer.

3.3.4 StackExchange

The web site tex.stackexchange.com is a carefully-managed and well-structured question-and-answer site for TeX and LaTeX. You can vote answers up or down according to their quality or usefulness, but there are strict rules about how you ask questions, the same as for comp.text.tex below.

3.3.5 Discord

The Discord web site and the associated app (most devices) is a chat and discussion system originally aimed at the gaming community, but it now has a very active and useful server for TeX and LaTeX.

3.3.6 The TeXhax mailing list

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

3.3.7 TUG and other web sites

TeX Users Group (TUG), as well as most local user groups, maintains a web site (www.tug.org) with lots of information about various aspects of the TeX system and details of support groups, conferences, and journals in many languages and cultures. See Appendix 3 starting on page 297 for information on joining TUG.

3.3.8 Usenet News

The Usenet newsgroup comp.text.tex is the principal forum for other questions and answers about TeX and LaTeX, as well as the principal place where new CTAN packages are announced.

There is a very detailed guide to how to get the best out of asking questions on Usenet at www.catb.org/esr/faqs/smart-questions.html#intro.

To access Usenet news, type news:comp.text.tex into your browser's 'Location' or 'Address' window. If your browser doesn't

Note

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. It is also important that for specific queries you include a Minimal [Non-]Working Example — a very short whole LaTeX file that others can download and typeset, to see exactly what your problem is.

support Usenet news, install one of the many free newsreaders[5] from the list at en.wikipedia.org/wiki/List_of_Usenet_newsreaders). Google Groups also provides access to Usenet newsgroups (groups.google.com/g/comp.text.tex), but it is a web interface, not a newsreader in the normal sense of the word, and lacks most of the normal features of a newsreader.

3.3.9 Google LaTeX list

There is a Google Groups mailing list for LaTeX users at groups.google.com/group/latexusersgroup.

3.3.10 Commercial support

If you need commercial levels of support, such as a 24-hour phone contact, or macro-writing services, you can contact a consultancy which deals with TeX (details are on the TUG Web site and in issues of TUGboat).

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

4 Lists, tables, figures

It is perfectly possible to write whole documents using nothing but section headings and paragraphs. As mentioned in section 2.6 on page 56, 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[1] in nature or complex in structure.

In Chapter 2 starting on page 43 we saw how to create a hierarchical document structure with chapters and sections and paragraphs; this chapter covers the other building-blocks which you need within your structure: lists, tables, figures (including images), boxes like sidebars and panels, block quotations, and verbatim text (raw text like computer program listings). In Chapter 5 starting on page 135 we will cover the textual tools that you need *inside* text: footnotes, marginal notes, cross-references, citations, indexes, and glossaries.

---

[1] It's worth pointing out that 'technical' doesn't necessarily mean 'computer technical' or 'engineering technical', least of all 'mathematical technical': it just means it contains a lot of $\tau\acute{\epsilon}\chi\nu\eta$, Greek for specialist material or artistry. A literary analysis such as *La Textualisation de* Madame Bovary (on the marginal notes in the manuscripts of 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.

4.1 Lists

Lists are useful tools for arranging thoughts in a digestible format, usually a small piece of information at a time. There are four basic types of list, shown in below.

Table 4.1 – Types of list

Random or itemized lists (sometimes called 'arbitrary' 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 ordered lists** (sometimes called 'sequential' or '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 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; *b*) usually labelled with letters like this example; and *c*) often mutually inclusive or exclusive, with the final item prefixed by 'and' or 'or' respectively.

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.

4.1.1 Itemized lists

To create an itemized list, use 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 the normal round, solid one (•), which is also available with the command `\textbullet` if you load the textcomp package. See section 7.6.2 on page 214 for details of how to change the settings for list item bullets.

Nested itemized lists (see section 4.1.6 on page 93) used differing symbols for their bullets as well as more indentation and less spacing.

4.1.2 Enumerated lists

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

```
\begin{enumerate}
\item Enumerated lists use numbering on each item
  (can also be letters or roman numerals);
\item Long items use 'hanging 'indentation in just
  the same way that itemized lists do;
\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', in just the same way that itemized lists do;
3. The numbering system can be changed for any level.

See section 4.1.6 on page 93 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 enumitem package, which provides an environment option *noitemsep* for the three main list environments (there is also a *nosep* option for even more compact spacing). Both these options come *after* the environment name, not before; eg `\begin{itemize}[nosep]`

4.1.3 Description lists

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

```
\begin{description}
\item[Identification:] description lists require
  a topic for each item given in square brackets;
\item[Hanging indentation:] Long items use this
  in the same way as all other lists;
\item[Reformatting:] Long topic labels 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 topic labels can be reprogrammed to fold onto multiple lines.

I *very strongly* recommend using the enumitem package with its *unboxed* environment option for *description* lists, which avoids the spacing problems with LaTeX's default handling of long labels. This package has so many good features I tend to load it for virtually every document I create.

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 head item.

4.1.4 Inline lists

Inline lists are a special case, as they require the use of the enumitem or paralist packages.

The enumitem package with the *inline* option provides 'starred' versions of the three standard list types to do this: *enumerate**, *itemize**, and *description**. It uses a specification in the optional argument for formatting the labels (for example, italic letters and an upright parenthesis), and it also provides extensive support for the punctuation and conjunction between items, making it unnecessary to type it separately for each item (and differently for the last-but-one).

```
\usepackage[inline]{enumitem}
...
\textbf{\itshape Inline lists}, which are sequential
in nature, just like enumerated lists, but are
\begin{enumerate*}[label=\textit{\alph*}),
       itemjoin={{; }},itemjoin*={{; and }}]
\item formatted within their paragraph
\item usually labelled with letters
\item usually have the final item prefixed with
  `'and or `'or
\end{enumerate*}, like this.
```

☞ ***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.

See Chapter 6 starting on page 159 for details of the font-changing commands used in the optional arguments to the `enumerate*` shown in this example.

Exercise 19 – List practice

1. Add a list or two to your document.
2. Copy and paste any two of the ones described here to practice with.
3. Read the documentation for the enumitem package or the paralist package and use it to change the layout of the lists you have added.

4.1.5 Reference lists and segmented lists

Reference lists are visually indistinguishable from numbered or lettered lists, but the numbering or lettering does *not* imply a sequence. The numbers or letters are just used as labels so that the items can be referred to from elsewhere in the text (as in 'see item 501(c)3'). In this sense they are really a kind of sub-sectional division, and 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 section 5.3 on page 138).

Segmented lists are a highly specialised structure and outside the scope of this document. For details of their usage, see the 'TEI Guidelines' (Burnard and Sperberg-McQueen 1995).

4.1.6 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.

1. by default an outer enumerated list uses 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 slightly tighter for more deeply nested levels.

- by default the outer itemized list item has a bullet;
  – an embedded itemized list uses a dash;
    ∗ a third level uses an asterisk;
      · the fourth level uses a small bullet.

Table 4.2 – Default numbering for nested numbered lists

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

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]

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.

[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.

Exercise 20 – Nesting

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

4.2 Tables

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

> Printers charge extra when you ask them to typeset tables, and they do so for good reason: Each table tends to have its own peculiarities, so it's necessary to give some thought to each one, and to fiddle with the alternative approaches until finding something that looks good and communicates well. (Knuth 1986)

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. There are two things you need to know before you start: one is the terminology (see the panel

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 for tables and figures' on p. 96) and the other is what 'floats' are (see below

Terminology for tables and figures

LaTeX, in common with standard typesetting practice, uses the words 'Table' and 'Figure' with a very precise meaning:

Table: a formal textual feature, numbered, with a *caption*, and containing an aligned *grid* of numbers or text, referred to from the surrounding document (as in 'See Table 5'). A Table is the whole thing, not just the grid;

Figure: a formal graphical feature, numbered, with a caption, and an *image*, *diagram*, or piece of text, referred to from the surrounding document (as in 'See Figure 5'). A Figure is the whole thing, not just the image, diagram, or text.

Most importantly, Tables and Figures *float* (see section 4.2.1).

You can also have *informal tables*, which are simply grid alignments occurring between two paragraphs, with no caption or number or reference: there's one in section 1.5.2 on page 16; and you can have *informal figures* which are just images or diagrams done as *illustrations*. Informal figures and tables do *not* float.

The grid alignment of information in rows and columns in a Table is called a 'tabulation' or 'tabular matter', done with a `tabular` environment, and we'll deal with that in below.

4.2.1 Floats

Tables and Figures (and several other features of documents like sidebars) are what printers and publishers refer to as 'floats'. This means they are not part of the normal stream of your text, but separate freestanding entities, positioned in a part of the page to themselves (top, middle, bottom, left, right, or wherever the layout designer has specified). 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 by default moved to the top of the next page.

The positioning can be changed by moving the *table* or *figure* environment to an earlier or later point in the file, or by using the optional argument to the *table* or *figure* environment. This can be any mix of the letters `h` ('here'), `t` ('top'), `b` ('bottom'), `p` ('page by itself') to recommend where the Table or Figure should go (order is not significant: LaTeX will pick the best fit). To make your recommendation stronger, precede the first letter with an exclamation mark (`!`).

In this example you can see a Table requested to go here(!) or if not, at the top of the page; and a Figure requested to go at the bottom of the page or if necessary, on the next full page by itself.

```
\begin{table}[!ht]
...
\end{table}

\begin{figure}[bp]
 ...
\end{figure}
```

Authors sometimes want many figures or tables occurring in rapid succession, which is poor writing, as it's not just unfair to the reader, but raises the problem of how they are going to fit on the page and still leave room for text. In extreme cases, LaTeX will give up trying, and stack them all up until the end of the chapter or section for you to decide manually where to put them.

The skill is to space your tables and figures out within your text so that they intrude neither on the thread of your argument or discussion, nor on the visual balance of the typeset pages. But this is a skill few authors have, and it's one point at which professional typographic advice or manual intervention may be needed.

If you are unable to arrange things easily, as a last resort you can use the float package and the option letter capital H ('Here, dammit!'). Be aware that figures or tables using this package option *are no longer floats* so the onus is on you to ensure that the numbering sequence is not disrupted.

> Remember that if there really is not enough space 'here' on the page, then it *really* won't fit, and you will HAVE TO move things manually.

The float package also lets you create new classes of floating object (sidebars, examples, exercises, etc).

Please now read this section a second time. Getting the hang of floats can take a while if you've never come across the idea before. Most writers strongly recommend writing the document in its entirety first, and not worrying about where the floats end up until the text is complete and not likely to change any more. *Then* start moving any floats that are misplaced.

4.2.2 Normal tables

To create a LaTeX Table, use the *table* environment containing a \caption command followed by a \label command (the label is what you will use to refer to the table: we haven't done this yet, but see section 5.3.1 on page 138 if you're curious).

```
\begin{table}
  \caption{Project expenditure to year-end 2022}
  \label{ye2022exp}
  ...
\end{table}
```

Numbering is automatic, but the \label command MUST *follow* the \caption command, not the other way round. The numbering automatically includes the chapter or section number in document classes where this is appropriate. 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:

```
\caption[Project 2022]{Project expenditure to
  year-end 2022 showing utility costs as a
  separate item}
```

The caption is centered if it's shorter than one line; otherwise it is set full out. The caption and ccaption packages offer extensive customisation of the caption, including location and font.

below is an example that we can use to show how tables work (if you're reading this in on the web, ignore the shading: that's a part of the webpage style, not the LaTeX table).

Table 4.3 – Project expenditure to year-end 2022

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

The Institute also contributes to items at (b).

4.2.3 Simple tabular matter

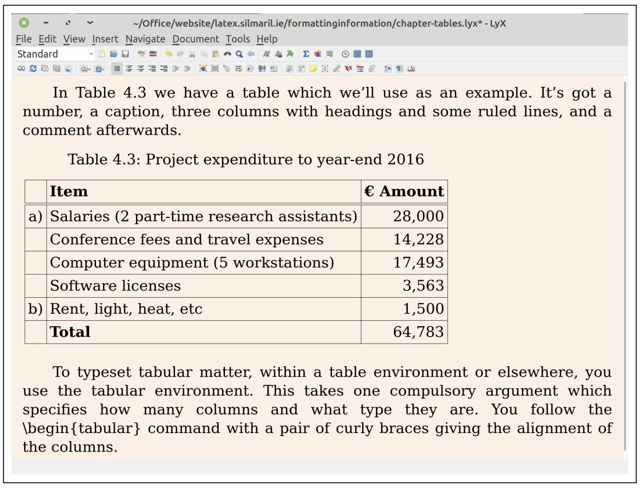
To typeset the grid within a *table* (or elsewhere),[3] you use the *tabular* environment. There are four ways to enter the data:

By hand: you can enter the tabular matter (cell data) by typing it in, which is perhaps the most common method, especially for small quantities of data;

In a grid tool: many LaTeX editors come with a pop-up grid tool like a miniature spreadsheet, which makes creating tabular matter easier, at the cost of some loss of fine control (see below).

[3] You can use the *tabular* environment anywhere you need stuff aligned in rows and columns, not just in a figure.

Figure 4.1 – Table being edited in LyX's tabular editor



~/Office/website/latex.silmaril.ie/formattinginformation/chapter-tables.lyx* - LyX

File Edit View Insert Navigate Document Tools Help

Standard

In Table 4.3 we have a table which we'll use as an example. It's got a number, a caption, three columns with headings and some ruled lines, and a comment afterwards.

Table 4.3: Project expenditure to year-end 2016

	Item	**€ Amount**
a)	Salaries (2 part-time research assistants)	28,000
	Conference fees and travel expenses	14,228
	Computer equipment (5 workstations)	17,493
	Software licenses	3,563
b)	Rent, light, heat, etc	1,500
	Total	64,783

To typeset tabular matter, within a table environment or elsewhere, you use the tabular environment. This takes one compulsory argument which specifies how many columns and what type they are. You follow the \begin{tabular} command with a pair of curly braces giving the alignment of the columns.

With a package: if the quantity of data is very large and is already in a spreadsheet or database, or if it is data which will change frequently before you are finished your document, you can use the datatool package (formerly known as csvtools) to read the data from a Comma-Separated Values (CSV) spreadsheet import/export file (see section 4.2.4 on page 103). If the data changes, you just re-export it and re-run LaTeX.

For large numbers of tables in big documents (eg theses) this is by far the most accurate and time-saving method.

As an image: it is also possible to include a 'table' which has actually been captured as an image from elsewhere, such as a screenshot from a spreadsheet (so it's not really a table just a picture of one). We will see how to include images in section 4.3 on page 113 on Figures, where they are more common.

In above and Table 4.3 on the preceding page there is the table which we'll use as an example. It's got a number, a caption, three columns with headings and some ruled lines, and a comment afterwards.

The tabular environment: This takes one compulsory argument which specifies how many columns there will be, and what type they are. You give one letter for each column using one of l, c, and r for a left-aligned, centered, or right-aligned column. The number of letters MUST be the same as the number of columns you are putting in the table.

```
\begin{tabular}{clr}
...
\end{tabular}
```

In the example in Table 4.3 on page 99, the tabular setting has three columns, the first one centered, the second left-aligned, and the third one right-aligned, so it is specified as {clr}. The dcolumn package provides a d column type for decimal alignment, and there are others we shall come across later.

Note

Each cell of these types (c, l, r, d) can hold only *one line of data* in its cell. If you need multi-line cells (like miniature paragraphs), see section 4.2.4 on page 103

Cell and row division: You can then type in each row, making sure each cell's data in the row is separated with an & character, and each row ends with a double backslash (\\).[4]

```
a)&Salaries (2 part-time research assistants)&28,000\\
```

You don't need to add any extra spaces or do any manual formatting, although you can if you want: LaTeX just uses the column specifications to know how to format it.

If a cell has nothing to go in it, you just don't type anything, but the ampersand must still be there:

[4] Note that this use of the double backslash to signal the end of a row is subtly different from the use we saw in section 1.10.5 on page 36 to terminate a normal text line prematurely. Here it marks the end of a table row.

```
&Total&64,783\\
```

Column headings: These are often set in **bold type**, as in the example (see 'Cell formatting' below).

```
&\textbf{Item}&\textbf{\EUR\ Amount}\\[6pt]\hline
```

In this case there is also some extra space (6pt, see 'Row spacing' below) to make it look nicer, and a horizontal line across the table (see the list item 'Table rules' below).

The data for a row may be longer than the width of the screen window in your editor, but it can take up as many lines on the screen as needed; the end of the row is always signalled by the double backslash, so LaTeX knows when it's time for the next row.

Cell formatting: Font changes can be done within a cell (bold, italic, etc; we'll come on to these later, see section 6.2.4 on page 189) and these changes are limited to the cell in which they occur: they do not 'bleed' across cells (in the example, the column headings have each been made bold separately).

Row spacing: Additional vertical white-space *below* a row (but above a rule) can be specified by giving a dimension in [square brackets] immediately after the double backslash which ends the row (3pt in the case of the last row before the totals in the example). A negative value will decrease the spacing below that row.

If the line below a horizontal rule looks too close, it can be optically spaced by adding a *strut* at the start of the next line (that is, *after* the `\hline`). A 'strut' is hidden vertical rule a little bit higher than the row-height; hidden because its width is zero, making it invisible, as in the example code. Use the `\rule` command for this, with a width of 0pt and height of 1.2em, just a fraction higher than the text, which will force the rows apart by 0.2em.

```
\begin{table}
\caption{Project expenditure to year-end 20}
\label{ye2022exp}
\centering\smallskip
\begin{tabular}{clr}
  &\textbf{Item}&\textbf{\EUR\ Amount}\\\hline\rule{0pt}{1.2em}
a)&Salaries (2 part-time 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\\[3pt]\cline{2-3}
\rule{0pt}{1.2em}&Total&64,783\\
\end{tabular}
\par\medskip\footnotesize
The Institute also contributes to (a) and (b).
\end{table}
```

Table rules: A line across the whole table is done with the `\hline` command after the double-backslash which ends a row.

For a line which only covers some of the columns, use the `\cline` command (in the same place), with the column range to be ruled in curly braces. If only one column needs a rule, it must still be given as a range (eg in the example, `{3-3}`).

Vertical rules (between columns) can be specified in the column specifications with the vertical bar character (`|`) before, after, or between the `l`, `c`, `r` letters. This character creates rules which extend the whole height of a table: it is not necessary to repeat them every row.

I have indented the code example given just to make the elements of the table clearer to read: this is for editorial convenience, and has no effect on the formatted result (see Table 4.3 on page 99). If you copy and paste this into your example document, you will need to add the marvosym package to your Preamble, which will let you use the official CEC-conformant Euro symbol command `\EUR` (€ as distinct from €).

4.2.4 More complex tabular formatting

TeX's original *tabular* environment was designed for classical numerical grids, where each cell contains a single value. If you need a cell to contain multiline text, like a miniature paragraph, you can use the column specification letter p (paragraph) followed by a width in curly braces instead of an l, c, or r. So p{3.5cm} would mean a column 3.5cm wide, where each cell can contain paragraph-style text, for example:

```
\begin{tabular}{cp{3.5cm}r}
```

These p column specifications are *not* multi-row (row-spanned) entries: they are single cells which can contain multiple lines of typesetting: the distinction is extremely important. These paragraphic cells are typeset justified (two parallel margins) and the baseline of the top line of text is aligned with the baseline of neighbouring cells in the row.

The array package provides some important enhancements which overcome the limitations of the p cells:

Vertical alignment: In addition to the p, whose vertical alignment baseline is the the top line of text, the array package provides the m and b letters. These work the same way as p (followed by a width in curly braces), but their vertical alignment baseline is the middle or bottom of the cell respectively.

Prefixes and suffixes: With the array package, any column specification letter can be preceded by >{} with some LaTeX commands in the curly braces. These commands are applied to every cell in that column, so to make a p column typeset ragged-right you would say, for example, >{\raggedright}p{3.5cm} (or \raggedleft, or \centering).

Note that if you do this, the last column specification MUST include a prefix or suffix containing the \arraybackslash command, to revert the meaning of the double-backslash, which gets redefined by horizontal formatting commands like \raggedright, otherwise you will get errors when the end-of-row double-backslash is not recognised.

There is a suffix format as well: you can follow a column letter with <{} with code in the curly braces (often used to turn off math mode started in a prefix).

The colortbl package lets you colour rows, columns, and cells; and the dcolumn package mentioned above provides decimal-aligned column specifications for scientific or financial tabulations. Multi-column (column-spanning) is built into LaTeX tables with the \multicolumn command; but for multi-row (row-spanning) cells you need to add the multirow package. Multi-page and rotated (landscape format) tables can be done with the longtable, rotating, and landscape packages.

The LaTeX table model is very different from the HTML auto-adjusting model used in web pages; it's closer to the Continuous Acquisition and Life-cycle Support (CALS) table model used in technical documentation formats like DocBook. However, auto-adjusting column widths are possible with the tabularx and tabulary packages, offering different approaches to dynamic table formatting.

You do not need to format the tabular data in your editor: LaTeX formatting will typeset the table using the column specifications you provided. You can therefore arrange the layout of the data in your file for your own convenience: you can give the cell values all on one line, or split over many lines: it makes no difference so long as the cells are separated with the & and the rows are ended with the double-backslash.

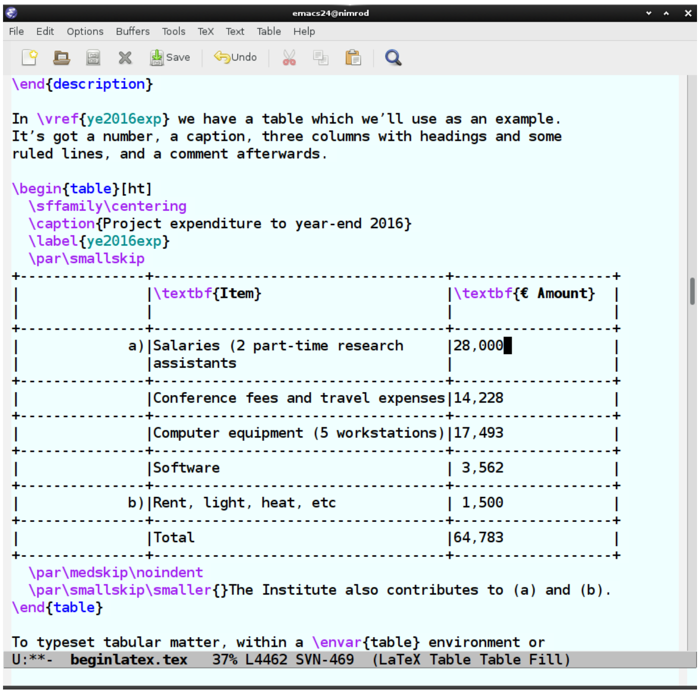
As mentioned earlier, some editors have a grid-like array editor for entering tabular data. Takaaki Ota provides an excellent *tables-mode* for *Emacs* which uses a spreadsheet-like interface and can generate LaTeX table source code (see below).

4.2.5 More on tabular spacing

Extra space, called a 'shoulder', is automatically added on both sides of all columns by default. The initial value is 6pt, so you get that amount left and right of the tabulation; because it is added left and right of every cell, the space between columns is therefore 12pt by default. This can be adjusted by changing the value of the \tabcolsep dimension *before* you begin the tabular environment.

```
\setlength{\tabcolsep}{3pt}
```

Figure 4.2 – Tables mode for *Emacs*



```
\end{description}

In \vref{ye2016exp} we have a table which we'll use as an example.
It's got a number, a caption, three columns with headings and some
ruled lines, and a comment afterwards.

\begin{table}[ht]
  \sffamily\centering
  \caption{Project expenditure to year-end 2016}
  \label{ye2016exp}
  \par\smallskip
+----------------+------------------------------------+--------------------+
|                |\textbf{Item}                       |\textbf{€ Amount}   |
|                |                                    |                    |
+----------------+------------------------------------+--------------------+
|              a)|Salaries (2 part-time research      |28,000              |
|                |assistants                          |                    |
+----------------+------------------------------------+--------------------+
|                |Conference fees and travel expenses|14,228              |
+----------------+------------------------------------+--------------------+
|                |Computer equipment (5 workstations)|17,493              |
+----------------+------------------------------------+--------------------+
|                |Software                            | 3,562              |
+----------------+------------------------------------+--------------------+
|              b)|Rent, light, heat, etc              | 1,500              |
+----------------+------------------------------------+--------------------+
|                |Total                               |64,783              |
+----------------+------------------------------------+--------------------+
  \par\medskip\noindent
  \par\smallskip\smaller{}The Institute also contributes to (a) and (b).
\end{table}

To typeset tabular matter, within a \envar{table} environment or
U:**-  beginlatex.tex   37% L4462 SVN-469  (LaTeX Table Table Fill)
```

The shoulder can be omitted in specific locations by adding the code `@{}` in the appropriate place[s] in the column specification argument. For example to omit it at the left-hand and right-hand sides of a tabular setting, put it at the start and end of the column specifications (putting it between two column specifications will remove all space between those columns).

```
\begin{tabular}{@{}clr@{}}
```

You can also use `@{}` to insert different spacing between columns (or at the right-hand and left-hand sides) by enclosing a spacing value; for example, `@{\hspace{2cm}}` could be used to force a 2cm space between two columns.

To change the row-spacing in a tabular setting, you can redefine the `\arraystretch` command (using `\renewcommand` because it's defined as a command, not a length). The value of `\arraystretch` is actually a ***multiplier***, preset to 1, so

`\renewcommand{\arraystretch}{1.5}` would set the baselines of your tabular setting one and a half times further apart than normal.

Exercise 21 – Calculate vertical spacing in a tabular environment

Assume that you are making a table in the default size of 10pt type on a 12pt baseline. You want a 14pt baseline, so what value would you set `\arraystretch` to?

It is conventional to centre the tabular setting within a Table, using the *center* environment (note US spelling) or the `\centering` command (as in the example) — the default is flush left — but this is an æsthetic decision. Your journal or publisher may insist instead that all tabular material is set flush left or flush right (not the individual columns; the whole tabular setting inside the table).

If there is no data for a cell, just don't type anything — but you still need the `&` separating it from the next column's data. The astute reader will already have deduced that for a table of n columns, there must always be $n-1$ ampersands in each row. The exception to this is when the `\multicolumn` command is used to create cells which span multiple columns, when the ampersands of the spanned columns are omitted. The `\multicolumn` command takes three arguments: the number of columns to span; the format for the resulting wide column; and the contents. So to span a centred heading across three columns you would write `\multicolumn{3}{c}{The new heading}`.

The `\multicolumn` command can also be used to replace a *single* column if you need to vary some prefixing or suffixing or alignment specified in the column specification. For example if you have a right-aligned column (eg numbers) but you want one of the cells to be some text centered, you could write `\multicolumn{1}{c}{no data}`. In this case, of course, you keep all the ampersands, because you are not actually spanning columns.

4.2.6 Techniques for alignment

As mentioned earlier, it's perfectly possible to use the *tabular* environment to typeset any grid of material — it doesn't have to be inside a formal table. There are also other ways to align material without using a tabular format.

4.2.6.1 Using tabular alignment outside a table

By default, LaTeX typesets *tabular* environments *inline* to the surrounding text. That is, the *tabular* environment acts like a single character within the paragraph. This also means if you want an alignment displayed by itself, not as part of a formal table, you can put it between paragraphs (with a blank line or \par before and after) so it gets typeset separately; or put it inside a positioning environment like *center*, *flushright*, or *flushleft*.

One side-effect of this is that small and intricately constructed micro-tabulations can be used to good effect when creating special effects like logos, as they they get treated like a character and can be typeset anywhere.

Tabular setting can 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 side to line up with each other. One common way to implement 'spring margins'[5] like this is to create two columns of whatever fraction of the page width you need (but adding to 1, of course), and removing for the extra space that would otherwise be added automatically between columns and at the edges:

```
\begin{tabular}{
          @{}
          >{\raggedright}p{.75\textwidth}
          @{}
          >{\raggedleft\arraybackslash}p{.25\textwidth}
          @{}}
```

[5] The term 'spring margins' comes from the DOS wordprocessor *PC-Write* and seems to be due to its author, the late Bob Wallace. I am not aware of any other mainstream system at the time that implemented them.

```
left-hand material
&
right-hand material\\
\end{tabular}
```

As mentioned earlier, the `@{}` suppresses the inter-column gap (or the shoulder left or right) so that the total width available will be the full text width of the page.

Exercise 22 – Create a tabulation

Create one of the following in your document:

1. a formal Table with a caption showing the number of people in your class broken down by age and sex;
2. an informal tabulation showing the price for three products;
3. the logo YEAR 2000 (hint: section 4.6.2 on page 127).

4.2.6.2 Alignment in general

Within the two-dimensional plane of conventional typesetting, there are two sets of axes to which the elements of the document should align: horizontal and vertical.

- The vertical axes are the left and right edges of the paper, the left and right margins of the text area, indentation, any internal temporary left and right margins (as for lists, block quotation, displayed mathematics, the left and right edges of illustrations, etc), and any internal column boundaries of a *tabular* environment.
- The horizontal axes are the top and bottom edges of the paper, the top and bottom margins of the text area, the space for running headers and footers, the top and bottom edges of all 'pool' items (see the start of this chapter), the baseline

of the text, and any internal row boundaries of a `tabular` environment.

Warning

If someone says they want something 'aligned', you need to ask 'aligned to *what*, exactly'? It's not always obvious, and in unusual cases it's not always easy to find out how to calculate or access an axis without careful study of the internal programming of a class or package.

By default, LaTeX starts each line up against the left-hand margin: if indentation is used, then the first lines of paragraphs will be indented, *except* for the first paragraph after a heading.[6]

- Depending on the language you select in the babel or polyglossia packages, the first lines of first paragraphs after a heading may *not* be indented (for example in French typesetting).
- In right-to-left languages, the alignment is reversed, and lines start up against the right-hand margin, and (see below) end against the left-hand margin.

The typeset line extends to the right-hand margin, and the process of justification ensures that all line-ends, apart from the last in a paragraph, align with this margin. The exception is when a *raggedright* or *raggedleft* or *centering* alignment has been specified.

Alignment to the four paper edges is extremely rare except in magazines and specialist formats like corporate reports or white papers, where images may be positioned to the edge[s] of the paper, and are said to 'bleed' off the sheet. It is of course possible in LaTeX but it is well outside the scope of this introductory text for beginners.

[6] This is known as Anglo-American usage (and applies to those countries and their [legacy] colonies, when using the English language).

4.2.6.3 Alignment within pool items

While typesetting a paragraph, LaTeX has no way to become aware of whereabouts a particular word or letter is being placed, for two reasons:

1. The justification of the paragraph does not start until after the whole paragraph has been typeset; only then does TeX start testing for line-end breakpoints, assigning them penalties, and inserting the variable spacing between words. This process is synchronous with the typesetting of the paragraph, and the next paragraph will not be started until justification of the one just ended is complete.

2. The positioning of the paragraph vertically on the page does not start until well after at least a whole page's worth of material has been typeset and justified, and the 'galley' of accumulated material comes close to filling up. At this point, TeX pauses typesetting of the next page (which it has already started), finds the optimum place to break the page, sends the completed page to the output, resets the accumulator to the remaining material, and then resumes typesetting. This process of page-building is therefore *a*synchronous with the process of typesetting, and the point at which access to already-typeset material ceases to be possible is not predictable in meaningful terms.

This means that doing things with stuff that has already been dealt with really isn't possible, and requests for it have to be respectfully declined. Anything you need to do with an item, whether it's a letter or a word or a paragraph, like applying a font change or putting it in a box, for example, needs to be done *in situ*, *before* it disappears down TeX's throat.

While there are packages for dealing with *completed* paragraphs, such as reledpar for typesetting synchronised parallel-page (eg dual-language) editions, access to the inside of the paragraphs is not possible at this stage. It *is*, however, possible to typeset material into a box, and then do things with it, including emptying it all back out again, in a limited manner. This makes it possible to see how much space a particular item is going to occupy, and then decide whether or not to treat it in a certain way. Standard LaTeX does this when deciding if a table or figure caption is narrow enough to fit centered on one line, or if it needs to set it full-out.

Packages which provide their own alignment options, such as enumitem for finer control of lists, usually specify in the documentation how to manipulate the shape and appearance of their environments. A substantial amount of this is about how to align one atomic value, such as a heading or title, with another one, such as the word which comes after it. In the case of the lettrine for dropped initial capitals, it's about how to adjust the capital (up, down, right, left) with respect to the indented rectangle into which it is to fit. In the case of the colortbl package for coloured rows and columns and cells in tables, it includes details of how to get the coloured block microadjusted.

4.2.6.4 Alignment to margins

The geometry package has extensive features for specifying the paper size, page size, margin sizes (left and right, if you are typesetting for double-sided work), marginal gaps, the head and foot settings for headers, footers, footnotes, and the gaps between them.

The description of line-alignment in the preceding section holds true for all text typeset inside further environments, for example in an *abstract* or a *quotation*, and within all lists, as well as the *p*/*m*/*b* column formats within a *tabular* setting. So long as you remain aware of the possible effect of unscoped formatting commands on lower-level nested environments, you can nest one environment inside another to an unspecified depth, and the rules of alignment will continue to be applied as much as possible. However, as with HTML and CSS, it is possible to overuse or abuse nesting, as it makes the code obscure.

Because the nesting of environments implies encapsulation, access to the alignment points (eg margins) of an outer environment is often not possible inside a deeper-nested environment. The TeX language model allows for the inheritance of settings defined at a higher level, but where these values are implemented as part of the code creating both the current environment *and* a higher one (eg lists inside lists), they will occupy the same space, and only the local value will be accessible. In such cases, any values needed would have to be saved in a variable accessible to the lower-level environment. In 30+ years of using LaTeX I have only ever needed to do this once.

4.2.6.5 Grids

Outside the *tabular* environment, LaTeX does *not* use a grid system. Its origins in mathematics mean that because displayed equations can occupy non-integer numbers of 'lines' (compared with text, which always occupies a whole number of lines), it was judged better for quality to allow flexible space *between* headings and paragraphs. Over the depth of a whole page, this minute amount of flexibility usually absorbs the fractional part of a line-height due to overspill in formulas (part of which 'rubberisation' led Leslie Lamport to choose LaTeX as the name for his set of macros).

There is a grid package available which enables grid setting in double-column documents, but overall there is no easy way to 'snap' pool elements to arbitrarily distanced gridlines. The flexibility of `\parskip` and the dozen or more other 'skips' (flexible lengths) in the LaTeX source (`latex.ltx`) could be removed, and display mathematics set in boxes of an integer number of `\baselineskip`s, and special environments could be written to anchor themselves to a specific corner, but in general, the model of flexibility has proved itself over nearly 40 years, and requirements for grid models should be transferred to the NTS in the care of TUG and the LaTeX development specialists.

4.3 Figures

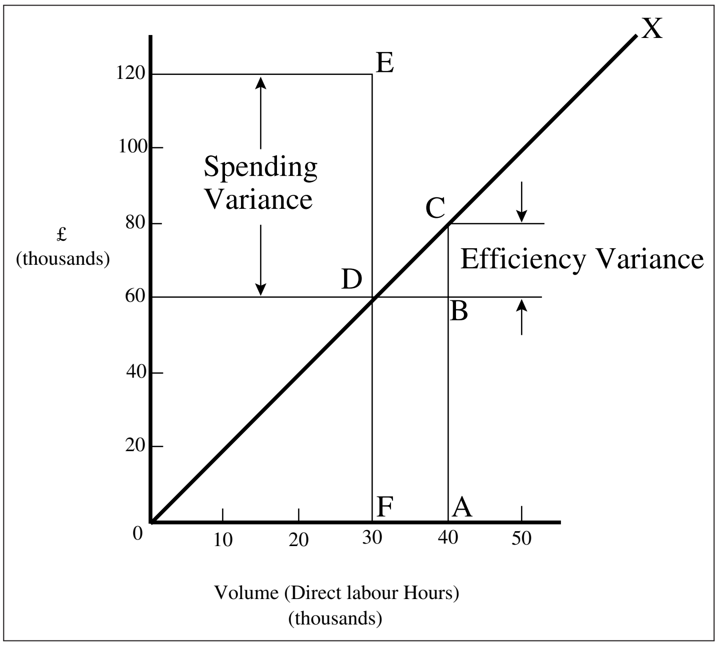
As explained in section 4.2 on page 95, Tables and Figures *float* to a vacant part of the page, as they are not part of your normal text, but illustrative objects that you refer to.

If you haven't already read the panel 'Terminology for tables and figures' on p. 96floatterms and section 4.2.1 on page 96, please read them now before going any further.

To create a figure, use the *figure* environment. Like Tables, they automatically get numbered, and they MUST include a \caption (with a \label after it, if needed for cross-referencing). Like Tables, it is conventional to centre the material, but that is a personal choice.

```
\begin{figure}
\caption{Total variable overhead variance (after
         \textcite[p.191]{bull}}
\label{workeff}
\centering
\fbox{\includegraphics[width=.75\columnwidth]{diagram}}
\end{figure}
```

Figure 4.3 – Total variable overhead variance (after Bull (1972, p. 191))



You can see that the structure is very similar to the *table* environment, but in this case we have a graphic included with the \includegraphics command. Here, it's also enclosed in an \fbox, which places a frame box around it (see section 4.6.2 on page 127). Details of including graphics are in the next section because they can occur in many places, not just Figures: you need the graphicx package. Details of the bibliographic citation mechanism used in the caption are will be covered in section 5.3.2 on page 140.

Figures can contain text, diagrams, pictures, or any other kind of illustration, even a *tabular* environment — LaTeX is agnostic on this point, so Tables can contain an image (of a table, presumably) and Figures can contain a tabulation. What matters is that you describe them properly.

The content of the Figure could of course also be textual, in the form of lists, paragraphs, or other blocks of text. For drawings, LaTeX has a very simple drawing environment called *picture*, which lets you create a very limited set of lines and curves, but for a diagram of any complexity, you can use any normal vector drawing program (see section 4.4.3 on page 120), save the image as a PDF vector image, and include it in your Figure with `\includegraphics` as illustrated.

Much more common is to create your diagrams within the LaTeX document using *TikZ*, which is a TeX interface to the Portable Graphics Format (PGF) graphics language. *TikZ* is much more powerful than the *picture* environment, but it's a whole language of its own, so it needs learning. (I used it to create the labelling on the image in Figure 1.4 on page 29.) The manual is in the tikz and there is a self-help blog at `latexdraw.com` and extensive help via the TeX Stackexchange.

4.4 Images

Images (graphics) can be included anywhere in a LaTeX document, although in most cases of formal documents they will occur in Figures (see section 4.3 on page 113). To use graphics, you need to use the graphicx package in your Preamble: `\usepackage{graphicx}`.[7] This package provides the command

[7] You may find a lot of old files which use a package called epsf. Don't use it: it's obsolete. The graphicx package has an 'x' because it's an extension on the older package called graphics.

\includegraphics which is used to insert an image in a document. The command is followed by the name of your graphics file *usually without the filetype* (we'll see in itemsection 4.4.1 on the facing page why you don't normally need to include the filetype).

```
\includegraphics{myhouse}
```

In most cases you should just make sure the image file is in the same folder (directory) as the document you use it in. This avoids a lot of messing around remembering where you put the files; but you could instead put them all in a single subfolder (for example, called `images`) and include that as part of the filename you use in the command.

```
\includegraphics{images/myhouse}
```

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 below).

4.4.1 Supported image file formats

The type of image file you use depends on LaTeX processor you are using (see section 1.2 on page 5 for how to choose). The common file types are:

- Joint Photographic Experts Group (JPG), used for photographs and scanned images;
- Portable Network Graphic (PNG), used for photographs and scanned images;
- Portable Document Format (PDF), used for vector graphics (drawings, diagrams) and typographic output from other programs;
- Encapsulated Postscript (EPS), an old publishing industry standard for many years, and the forerunner of PDF, still used by some older programs that generate diagrammatic or typographic output.

See below and below for other file formats. For more details, see the answers to the question on StackExchange, Which graphics formats can be included in documents processed by latex or pdflatex?. Basically, for modern systems running XƎLATEX, LuaLATEX or *pdflatex* (creating PDF output) graphics files MAY be in PNG, PDF, or JPG (JPEG) format ONLY. (You may come across very old systems running plain (original) LATEX (creating DVI output) in which graphics files MUST be in EPS format ONLY: no other format will work: see below)

- PNG actually gets converted to the PDF internal format automatically (at a small penalty in terms of speed) so for lots of images, or very large images, use JPG format or preconvert them to PDF;
- It is also of course possible to convert (repackage) your JPG pictures to PDF, using any of the standard graphics conversion/manipulation programs (see below for details). Preconverting all your images to PDF makes them load into your document slightly faster.
- LATEX will search for the graphic file by file type, in this order (check for the newest definition in your `pdftex.def`): `.png`, `.pdf`, `.jpg`, `.mps`, `.jpeg`, `.jbig2`, `.jb2`, `.PNG`, `.PDF`, `.JPG`, `.JPEG`, `.JBIG2`, and `.JB2`.[8]

See section 4.4.3 on page 120 for more about how to create and manage your image files.

4.4.1.1 Other file formats

Convert them to one of the supported formats using a graphics editing or conversion tool such as the GNU Image Manipulation Program (GIMP), the *NetPBM* utilities, *ImageMagick*, or a utility like Péter Szabó's *sam2p* (not available with TEX Live but downloadable for Windows and Linux).

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;

[8] Thanks to Enrico Gregorio and Philipp Stephani on `comp.text.tex` for locating this for me.

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.

It is in fact possible to tell LaTeX to generate the right file format by itself during processing, but this uses an external graphics converter like one of the above, and as it gets done afresh each time, it may slow things down.

4.4.1.2 Postscript

Since TeX 2010, EPS files will be automatically converted to PDF if you include the epstopdf package. This avoids need to keep your graphics in two formats, at the expense of a longer compile time while it converts every EPS image (not recommended).

All good graphics packages (eg *GIMP*, *PhotoShop*, Corel *Draw*, etc) 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 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 first by LaTeX instead of the PDF file, because of the order in which it searches.

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 archive 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 addressing a higher level of directory (one beginning with ../). The *dvips* driver will not accept these because they pose a security risk to *PostScript* documents. Unlike PDF, *PostScript* is a real programming language, capable of opening and deleting files, and the last thing you want is to create a document able to mess with your filesystem (or someone else's).

4.4.2 Resizing images

The \includegraphics command can take optional arguments within square brackets before the filename to specify the height or width, as in the example below. This will resize the image that prints; whichever dimension you specify (height or width) the other dimension will automatically be scaled in proportion to preserve the aspect ratio.

The file on disk does not get changed in any way, and nor does the copy included inside the PDF: what gets changed is just the size that it displays at in the finished document. So if you include a huge JPG but tell LaTeX to print it at a small size, your PDF will still include the whole image file at full size — all that changes is the way it shows it. This is very inefficient: normally you should create images at the right size for the document.

```
\begin{center}
  \includegraphics[width=5cm]{twithcat}
\end{center}
```

☞ 

If you specify both height *and* width, the image will be distorted to fit (not really useful except for special effects). You can scale an image by a factor (using the *scaled* option) instead of specifying height or width; clip it to specified coordinates; or 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 *Companion*. The package also includes commands to rotate, mirror, and scale text as well as images.

4.4.3 Making images

There are two types of image: bitmaps and vectors.

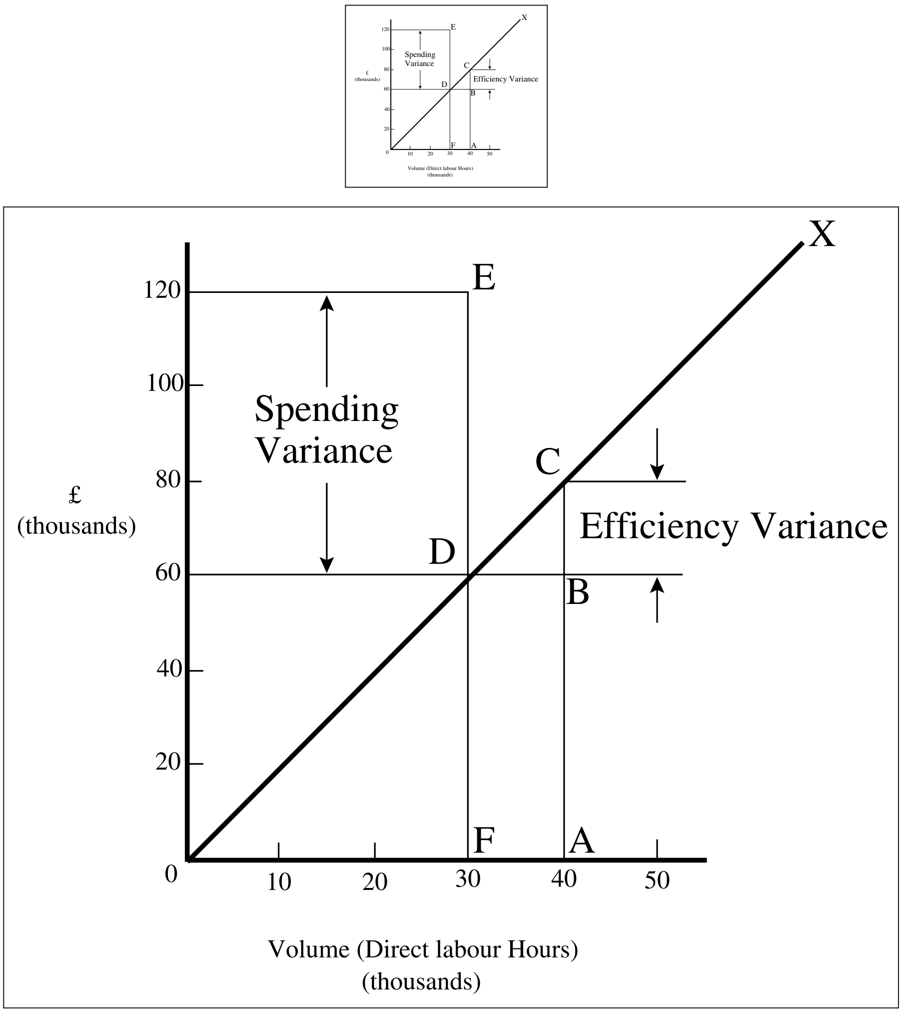
Bitmaps: Bitmap images are made of coloured dots, so if you enlarge them, they go jagged at the edges, and if you shrink them, they may go blurry. Bitmaps are fine for photographs, where every tiny dot is a different colour, and the eye won't notice so long as you don't shrink or enlarge too much. Bitmaps for diagrams and drawings, however, are almost always the wrong choice, and often disastrously bad.

Vectors: Vector drawings are made from instructions, just like LaTeX is, but using a different language (eg 'draw this from here to here, using a line this thick'). They can be enlarged or reduced 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 vectors are 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 their own vector language internally). All the major graphics-generating packages in all disciplines output vector formats: *AutoCAD*, *ChemDraw*, *MathCAD*, *Maple*, *Mathematica*, *ArcInfo*, and so on.

EPS was for decades the universally-accepted format for creating vector graphics for publication, with PDF a close second. PDF is now the most common format, but most of the major graphics (drawing) packages can still save as EPS, such as *PhotoShop*, *PaintShop Pro*, Adobe *Illustrator*, Corel *Draw*, and *GIMP*. There

Figure 4.4 – The vector diagram shrunk and enlarged



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 if possible, partly because it is even vaster than BMP, and partly 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 23 – 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.

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 section 4.3 on page 113.

4.4.4 Graphics storage

I mentioned earlier that there was a way to tell LaTeX where to look if you want to store some images centrally for use in many different documents.

If you want to be able to use some images from any of your LaTeX documents, regardless of the folder[s], then you should store the images in the subdirectory called `tex/generic` within your Personal TeX Directory.

Otherwise, you can use the command \graphicspath with a list of one or more names of additional directories you want searched when a file uses the \includegraphics command.

Put each path in its own pair of curly braces within the curly braces of the \graphicspath command. I've used the 'safe' (MS-DOS) form of the Windows My Pictures folder in the example here because you should never use directory or file names containing spaces (see the panel 'Picking suitable filenames' on p. 49filenames).

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.

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

If you use original LaTeX and *dvips* to print or create PostScript files, be aware that some versions will not by default handle EPS files which are outside the current directory, and will issue the error message saying that it is 'unable to find' the image. As we mentioned above, 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, eg `dvips -R0 ... dvifile`

4.5 Quotations

Direct speech and short quotes within a sentence 'like this' are done with simple quotation marks as described in section 1.8 on page 24. 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 section 6.2.5 on page 191) as shown in the example.

```
\begin{quotation}\small\noindent
At the turn of the century William Davy, a Devonshire
parson, finding errors in the first edition of his
\citetitle{davy}, asked for a new edition to be printed.
His publisher refused and Davy purchased a press, type,
and paper. He harnessed his gardener to the press and
apprenticed his housemaid to the typesetting. After
twelve years' work, a new edition of fourteen sets of
twenty-six volumes was issued---which surely indicates
that, when typomania is coupled with religious fervour,
anything up to a miracle may be
achieved.\hfill\textcite[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 1976, p. 76)

The inclusion of a bibliographic citation at the end is optional but commonplace, especially in academic or research documents where it may be compulsory because of the need to cite everything you quote. It's also possible in LaTeX for this to be tucked into the space at the end of the last line of the quotation, if there is room (if it's too long, or not predictable [like a web page], it should go on a line by itself, as it does in the web version of this document).

The *quotation* environment sets the whole block of text indented, and each paragraph of it also has its own indentation on the first line, even the first paragraph. This is rather unconventional as a default, so it is common to add a `\noindent` command at the start of the quotation so that the first paragraph does *not* get indented (others still will).

4.6 Boxes, sidebars, and panels

LaTeX, like most typesetting systems, works by setting text into boxes. Each character is also a box, with a height and a width, just like it was in metal type; characters are assembled into lines, which are also boxes; and lines are assembled into pages, which are also boxes. The page-making mechanism also works like an old compositor's galley (tray) from the days of metal type: the box accumulates lines of typeset text until it's a bit longer than the height of the page. TeX then works out how much of it really will fit on the page, cuts it off and ships it out to the PDF file, and puts the remainder back into the galley (box) at the top, ready to start accumulating more material for the following page.

4.6.1 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.

```
\parbox{3in}{Please make sure you send in your
completed forms by January 1st next year, or the
penalty clause in 2(a) will apply.}
```

☞

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

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, so if you include a \parbox in mid-sentence, the centre of the box will be lined up with the line of type currently being set. You can specify that the top or bottom should be the alignment point by adding an optional *t* (top) or *b* (bottom) in square brackets before the width. For example, `\parbox[t]{3in}{...}` will produce a box with the baseline aligned with the top line of the text in the box.

Where the contents is more complex, use the *minipage* environment.

```
\begin{minipage}{4in}
  Please make sure you send in your completed forms
  by January 1st next year, or the penalty clause
  2(a) will apply:
  \begin{itemize}[noitemsep]
    \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.

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. Fortunately, it is rare for LaTeX to need this level of attention.

Within a *minipage* environment you can use virtually everything that occurs in normal text (eg lists, paragraphs, tabulations, etc) with the exception of floats like Tables and Figures. The *minipage* environment takes a compulsory 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 \parboxes, the paragraph indentation (\parindent) is reset to zero. If you need to change it, do so *inside* the *minipage* or \parbox using the \setlength command (see section 2.7 on page 62).

Because a minipage is typeset independently from the rest of your text, any footnotes inside a minipage will be typeset at the end of the minipage, *not* at the foot of the containing page, and they will be done using lowercase letters by default, to keep them separate from the normal footnotes. We haven't done footnotes yet, but they're in section 5.1 on page 135.

There are 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 technique of setting the material into a special box that remembers it, and then emitting it where you want it (this is implemented by the standard *lrbox* environment or by the *Sbox* environment from the fancybox package, but these are advanced techniques).

4.6.2 Framed boxes

To put a frame round $\boxed{\text{some text}}$, use the \fbox command:

```
\fbox{some text}
```

We already saw this used in the Quick Start document and also to frame an image in Figure 4.3 on page 114. For text, this works for a few words in mid-line, but the framed box and its contents won't break over the end of a line. To typeset multiline text in a box, put it in a \parbox, or use a *minipage* or *tabular* environment as described above, and enclose the whole thing in a \fbox.

```
\fbox{\parbox{3in}{Please make sure you send in your
completed forms by January 1st next year, or the
penalty clause 2(a) will apply.}}
```

The spacing between text and box is controlled by the value of \fboxsep, and the thickness of the line by \fboxrule, both of which can be reset with the \setlength command.

If you are using colour, the xcolor package extends boxing to the \colorbox command, which takes two arguments: a colour name or code for the background colour, and the text (which will need a foreground colour if black would not be suitable):

```
\colorbox{green}{\textcolor{white}{some text}}
```

The package also provides \fcolorbox which puts a frame around a coloured box; in this case the first argument is the frame colour, the second the background colour, and the third the contents.

4.6.3 Panels and sidebars

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

You can create panels of any size with these borders by typesetting your text in a *minipage* environment inside the *Sbox* environment which fancybox provides. The *minipage* formats the text but the *Sbox* 'captures' it, allowing you to delay putting the frame around until it is complete (so it knows the size).

The printed version of this document uses this extensively and there is a worked example shown in section 7.5 on page 210.

Sidebars are blocks of text, usually explaining something too detailed to put in the main text. They sometimes go at the side of the page as their name suggests, but can also be floats like Tables and Figures. The float already mentioned lets you create your own new types of float as environments, so you could define one called *sidebar* and use that to handle your sidebars, putting a boxed block of text in each one.

4.7 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, filename, an email address, or raw text containing characters which you don't want interpreted by LaTeX and which needs to be typeset specially, using a fixed-width font.

LaTeX includes two features for handling fixed-format text: inline verbatim and display verbatim. There are many more variations available in other packages.

4.7.1 Inline verbatim

To specify a word or phrase as verbatim text in typewriter type within a sentence, use the special command \verb, followed by the word or phrase surrounded by any suitable character which does *not* occur in the word or phrase itself. This is a very rare exception to the rule that arguments go in curly braces.

For example, you could use the plus sign to show a LaTeX command in a manual like this one:

```
You can typeset a phrase verbatim, even if it includes
\LaTeX\ command characters, for example the command to
insert an image: \verb+\includegraphics[width=3in]{myhouse}+
```

You can typeset a phrase verbatim, even if it includes LaTeX command characters, for example the command to insert an image: `\includegraphics[width=3in]{myhouse}`

The plus sign is 'safe' to use here because it doesn't appear in the code you want to typeset but you could use the grave accent or backtick key ` or the vertical bar | if the phrase already had a plus sign in it:

```
for example \verb|\(a=b+c)| when illustrating the
\LaTeX\ equation \(a=b+c\).
```

for example `\(a=b+c\)` when illustrating the LaTeX equation $a = b + c$.

The \verb command has the advantage that it turns off all special characters (see section 1.7 on page 22) except the one you use as the delimiter, so you can easily quote sequences of characters in any computer syntax — including TEX. However, LATEX will never break the argument of \verb at a line-end when formatting a paragraph, even if it contains spaces, so if it happens to be long, and falls towards the end of a line, it *will* stick out into the margin. See section 1.10.2 on page 32 for more information on line-ends and hyphenation. The argument to \verb MUST NOT contain a linebreak in your editor: this will cause it to fail. See also the warning about using \verb inside \footnote.

4.7.1.1 Typesetting URIs

Typesetting URIs causes two problems which LATEX overcomes with the url package (and the hyperref package which also handles URIs).[9]

Visible accuracy: It is important for them to be visibly accurate, so they can be copied and retyped from print.

It is therefore essential 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, but many sans-serif fonts do not. It is a common error by designers not to distinguish URIs in this way.

Intrusive hyphens: URIs (especially long ones) must never have a hyphen added if they have to be broken over a line-end, because it might be mistaken (and retyped) by the user for a real hyphen needed as a part of the address. Conversely, any existing hyphen must never be used at a line-end.

Handling a URI therefore means being able to perform a hyphenless line-break, but only at some existing punctuation characters — principally the slash and the dot — but not usually the hyphen.

[9] The original term Uniform Resource Locator (URL) is now deprecated in favour of the more accurate Uniform Resource Identifier (URI). For details see www.w3.org/Addressing/. The older term still persists, especially in this LATEX package and its command, and in some XML markup vocabularies.

The url package provides the command \url which works in the same way as \verb, but uses the standard curly braces to enclose the address, eg \url{http://latex.silmaril.ie} — the command understands the syntax of a URI (Berners-Lee, Fielding and Masinter 2005) and will never break mid-way through an unpunctuated word, only at slashes and full points (and never at embedded hyphens unless in exceptionally rare circumstances when the *hyphen* package option can be used).

Bear in mind, however, that spaces and non-ASCII characters are (currently) invalid in URIs, so using spaces in a \url argument will cause it to fail, as will using other non-URI-valid characters.

The hyperref package make links for cross-references, citations, and URIs clickable in LaTeX PDFs (it's used in the PDF version of this document). It can also make footnote links clickable, although as they are usually on the same page, this is probably not needed in most documents.

You can load it with footnote links turned off, and the different classes of link (cross-references, citations, and URIs) in different colours:

```
\usepackage[hyperfootnotes=false]{hyperref}
\hypersetup{linkcolor=DarkGreen,citecolor=DarkRed,
            urlcolor=Blue,pdfborder=0 0 0,
            allbordercolors=0 0 0,colorlinks=true}
\AtBeginDocument{\renewcommand{\UrlFont}{\ttfamily}}
```

In some systems, links get borders drawn around them, but in the example above they are turned off by making their colour invisible (0 0 0). The hyperref package likes to use the roman font for URIs (normally A Bad Idea as we saw above in the list item 'Visible accuracy' above) but this can be turned off by changing the vaue of \UrlFont back to ttfamily.

4.7.1.2 Enhanced inline verbatim

The listings package, which we look at more below for display verbatim, also has an inline form. This can use colour to highlight your examples based on the language you are documenting — I am using it extensively in the PDF of this book.

The command \lstinline uses the same syntax as \verb (two matching but otherwise unused characters) to enclose the argument, but it provides for very extensive options to specify the language, font, size, style, and formatting. The most useful is the language, of which about 100 are predefined, from *ADA* to *Verilog*, and you can add new keywords and even whole new languages.

This is probably the most effective way to show computer-language examples inline, because it handles the syntax-based enhancement for you. It is, however, still subject to the same limitations as \verb, in that the code must fit on the space available in the line, or it will stick out into the margin.

```
For example, you could use the plus sign to show a \LaTeX\ command:
\lstlisting[language={[LaTeX]TeX}]`\verb+\includegraphics[width=3in]{myhouse}+`
in order to display
\lstlisting{[LaTeX]TeX}]`\includegraphics[width=3in]{myhouse}`, because the
plus sign does not occur in the command, and is therefore free to be used.
```

For example, you could use the plus sign to show a LaTeX command: \verb+\includegraphics[width=3in]{myhouse}+ in order to display \includegraphics[width=3in]{myhouse}, because the plus sign does not occur in the command, and is therefore free to be used.

4.7.2 Display verbatim

For longer (multiline) chunks of fixed-format text like examples of programming, use the *verbatim* environment. 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}, which MUST occur on a line by itself, starting at the beginning of the line (*not* indented)[10].

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

[10] A number of environments which do complicated things with the category codes of characters have this requirement to end the environment with the \end {...} command on a line by itself, at the start of the line. These include the *Verbatim* environment from the fancyvrb package, and any of the \end {...} commands taken over by the endfloat package.

```
\begin{document}

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

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

For more control over formatting there are two useful packages: the verbatim package, which overcomes a few of the limitations of the built-in *verbatim* environment; and the fancyvrb package, which provides much greater flexibility with a *Verbatim* environment (note the capital letter).

Exercise 24 – Try some fixed-format text

1. Add your email address and home page URI using the `\verb` and `\url` commands. You'll need to add `\usepackage{url}` to your Preamble for the latter.
2. Load the listings package and try the `\lstinline` command to do the same.

However, as I mentioned above, for a much more powerful verbatim environment, I use the listings package for its ability to colour the keywords of a program according to the language used. It can also add rules, interpret internal formatting, and include external files, and let you add your own language definitions for new languages. The penalty is a slightly more complex configuration, but if you are documenting any kind of computer code in significant quantities, the quality and usability of the result is well worth it.

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

\title{Practical Typesetting}
```

```
\author{Peter Flynn\\Silmaril Consultants}
\date{December 2004}
\maketitle

\end{document}
```

5 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:

- footnotes and end-notes;
- marginal notes;
- cross-references, both normal ones and bibliographic citations;
- indexes and glossaries;
- typesetting in multiple columns.

5.1 Footnotes and end-notes

The command `\footnote{Like this}`,[1] 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,Like this. and the numbered text automatically printed at the foot of the page. The number is reset to 1 at the start of each

[1] Like this.

chapter (but there are packages to 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.

Footnotes in titling and sectioning commands (`\title`, `\caption`, and `\author`; `\chapter`, `\section`, etc) are regarded as Bad Style, and you SHOULD try to avoid them. If you can't, in `\title` and `\author` you MUST prefix the `\footnote` command with `\protect` to prevent it being taken as part of the text. In `\caption`, `\chapter`, `\section`, etc, you MUST use the optional argument to provide the un-footnoted version for the ToC (see section 2.6 on page 56).

Depending on the book or journal style, footnotes in titles may need to produce the symbols *, †, ‡, §, ¶, ||, **, ††, and ‡‡ instead of the values 1–9 (and an error message for the tenth such footnote). In accordance with standard publishing practice, footnotes inside a *minipage* environment (see section 4.6.1 on page 126) 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 (endnote) to hold over your footnotes and make them print at the end of the chapter instead or at the end of the whole document, and there is a package (fnpara) to print many short footnotes in a single footnoted paragraph so they take up less space. It is also possible to have several separate series of footnotes active simultaneously, which is useful in critical editions or commentaries: for example, a numbered series for the original author's original footnotes; a lettered series for footnotes by subsequent commentators or authorities in later editions; and a roman-numeral series for your own footnotes. Note that some disciplines put their bibliographic references in footnotes (notably Historians) and they even call bibliographic references 'footnotes' (see section 5.3.2.3 on page 146).

It is also possible to format footnotes within footnotes, although this is considered Really Bad Style.

Verbatim inside footnotes

Because LaTeX reads the whole footnote before doing anything with it, you can't use the \verb (inline verbatim) command inside footnotes on its own: either use the \VerbatimFootnotes command from the fancyvrb package, or prefix \footnote with \protect, or use (abuse?) the \url command instead (which you should be using for Web and email addresses in any case).

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 (with the footnote counter as its argument) in the Preamble of your document:

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

There are also ways to refer to a 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, but these techniques are out of scope here.

5.2 Marginal notes

You can add marginal notes to your text instead of footnotes. or as well as You need to make sure that you have a wide-enough margin, of course: use the geometry package (see section 3.1.2 on page 67) to allocate enough space, otherwise the notes will be too cramped.

There are several packages to help with formatting marginal notes, but you can also define one yourself. Add this new command to your Preamble:

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

Then you can use `\marginal{Some text}` where you need it. Be careful, however, because marginal notes are aligned with the line where the command starts, so a very long one followed too closely by another will cause LaTeX to try and adjust the position so they don't overlap.

We're jumping ahead a bit here, as we haven't covered how to define your own commands yet. I won't even try to explain it here, although the astute reader can probably deduce it by inspection. See Chapter 7 starting on page 203 for more information about making up your own commands.

5.3 References and citations

This is one of the most powerful features of LaTeX. As we mentioned when discussing Figures and Tables, you can label any point in a document with a name you make up, so that you can refer to it by that name from anywhere else in the document (or even from another document) and LaTeX will always work out the right cross-reference number for you, no matter how much you edit the text or move it around.

As we well see later, a similar method is also used to cite documents for a bibliography or list of references, and there are packages to sort and format these in the correct style for different journals or publishers.[2]

5.3.1 Cross-references

You label the place in your document you want to refer *to* by adding the command `\label` followed by a short name you make up, in curly braces:[3] exactly as we did for labelling Figures and Tables in section 4.2.2 on page 98.

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

[2] Be aware that in some disciplines where cross-references are not much used, the word 'references' may be used to mean 'bibliographic references'.

[3] This section is labelled `normalxref`, for example.

You can then refer to this place from anywhere in the same document with the command \ref followed by the name you used, eg

- 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 when being typeset.

- The \S command can be used if you want the section sign § instead of the word 'section' (there is also a \P command that produces the paragraph sign or pilcrow ¶).

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

☞ In section 13.5 there is a list of recent projects.

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. If you are familiar with HTML, this is the same concept as the internal linking mechanism using # labels (or IDs in XHTML or HTML5).

Labels in normal text: If the label is in normal text, as above, the reference will give the current chapter/section/subsection/appendix number (depending on the current document class).[4]

Labels in Tables or Figures: If the label is inside a Table or Figure, the reference provides the Table number or Figure number prefixed by the chapter number (remember that in Tables and Figures the \label command MUST come *after* the \caption command).

The \ref command does not produce the word 'Figure' or 'Table' for you: you have to type it yourself, or use the varioref package which automates it.

[4] So I can refer here to the label of this section as \ref {normalxref} and get the value 'Section 5.3.1 on the facing page'.

Labels in lists: A label in an item in an enumerated list will provide the item number. In other lists its value is null or undefined.

Labels elsewhere: If there is no apparent countable structure at the point in the document where you put the label (in a bulleted list, for example), the reference will be null or undefined.

The command `\pageref` followed by any of your label values will provide the page number where the label occurred, instead of the reference number, regardless of the document structure. This makes it possible to refer to something by page number as well as by its `\ref` number, which is useful in very long documents like this one (varioref automates this too).

Process twice!

> LaTeX records the label values each time the document is processed, so the updated values will get used the *next* time the document is processed. You therefore need to process the document one extra time before final printing or viewing, if you have changed or added references, to make sure the values are correctly resolved. Most LaTeX editors handle this automatically by typesetting the document twice when needed.

Unresolved references are printed as two question marks, and they 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, as is defining two labels with the same value.

5.3.2 Bibliographic references

The mechanism used for references to reading lists and bibliographies is very similar to that used for normal cross-references. Instead of using `\ref` you use `\cite` or one of the variants explained in below; and instead of `\label`, you attach a label value to each of the reference entries for the books, articles, reports, etc that you want to cite. You keep these reference entries in a ***bibliographic reference database*** that uses the BIBTEX data format (see below).

Bibliographic reference databases

Although it is possible to type the details of each reference manually, it's much easier to use a program designed for the purpose. There are several available (see Wikipedia's list), including Zotero and Mendeley. Both are open-source, but *Mendeley* was recently bought out by Elsevier; although it was still free of charge at the time of writing there have been concerns about Elsevier's use of your data. Their features vary but *Zotero*'s primary benefit is that it can grab bibliographic metadata from web pages, so that you don't have to type it in. Both can extract the metadata from the PDFs of articles you download from journal sites. And both can export the data in BIBTEX format, which is essential.

Once your data is in BIBTEX format, you can manage your collection of references with any of the many free BIBTEX-based database programs such as JabRef (see Figure 5.1 on page 149).

Endnote and Reference Manager are commercial products which do not use the BIBTEX data format, but which can export the Research Information Systems (RIS) format which *Zotero*, *JabRef*, and *Mendeley* can all import.

You add your entries to whichever system you choose, usually by downloading references from an online database like *Web of Science*, *JSTOR*, *PubMed* etc, or by using *Zotero* or similar to gather the entry from a web page (you can also type references in by hand). *JabRef* lets you click an icon or menu entry and the LATEX citation command will be inserted into your document editor at the cursor location.

This does away with the time needed to maintain and format references each time you cite them, and dramatically improves accuracy. It means you only ever have to enter the bibliographic details of your references once, and you can then cite them in any document you write, and the ones you cite will get formatted automatically to the style you specify for the document (eg Harvard, Oxford, Institute of Electrical and Electronics Engineers (IEEE), Vancouver, Modern Language Association (MLA), American Psychological Association (APA), etc).

5.3.2.1 Choosing between BIBTEX and *biblatex*

LATEX has two systems for doing citations and references, BIBTEX (old) and biblatex (new): both of them use the same file format, also called BIBTEX. Both support the four common ways of indicating a citation: author-year, numeric, abbreviated alphabetic, and footnoted; plus a wide range of others.

BIBTEX: the older BIBTEX has been in use for many decades and is still specified in some publishers' document classes, especially for journal articles and books, and particularly those which have not been updated for a long time. While it will continue to work, it has several drawbacks:

1. it doesn't handle non-ASCII characters easily, so accents and non-Latin words are a problem;
2. the same applies to the sort-and-extract program it uses (also called *bibtex*);
3. the style format files (`.bst` files) are written in BIBTEX's own rather strange, unique, and largely undocumented language, making it extremely difficult to modify them or write new ones;
4. many of the style format files are now very old and out of date;
5. the range of data fields in references is limited and also out of date.

biblatex the newer biblatex system is now a well-established LATEX package to replace almost all of old BIBTEX. The main advantages are:

1. it uses the same `.bib` files as for old BIBTEX, but lets you use many new document types and data field names.

2. it works with UTF-8, so non-ASCII, non-Latin, and other writing systems are handled natively when using XƎLATEX or LuaLATEX.
3. there is a new sort-and-extract program (*biber*) to replace *bibtex*, which also handles UTF-8 natively.
4. the style format files are written entirely in LATEX syntax, and are under active development, so updating or writing layout formats it is much easier than with BIBTEX.
5. it supports the four popular citation formats listed above natively, without the need for additional packages.

> The only current drawback is that there are still a few uncommon and less-used reference formats that are still only supported in BIBTEX and not yet available in biblatex. If you are required to use one of these, you are going to be stuck with BIBTEX (unless you'd like to write a new biblatex add-on to handle it).

The biblatex package with the *biber* program is therefore recommended, especially with XƎLATEX or LuaLATEX. From here on, I shall be concentrating on biblatex.

5.3.2.2 The BIBTEX file format

The same file format for BIBTEX files is used for both BIBTEX and biblatex, regardless of whether you use *biber* or *bibtex*, so if you have existing BIBTEX files, they will continue to work, but it's a good idea to update old files with some of the more accurate field names provided by biblatex.

The file format is specified in the original BIBTEX documentation (look on your system for the file `btxdoc.pdf`). The biblatex package and its updated style formats provide many more fields and document types than we can describe here.

Each BIBTEX entry starts with an `@` sign and the type of document (eg `article`, `book`, etc), followed by the whole entry in a single set of curly braces. The first value MUST be a unique BIBTEX key (label) that you make up, which you will use to cite the reference with; followed by a comma:

```
@book{fg, ... }
```

Then comes each field (in any order), using the format:

```
fieldname = {value},
```

There MUST be a comma after each line of an entry *except the last line*:

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

Some TeX-sensitive editors have a BibTeX mode which understands these entries and provides menus, templates, and syntax colouring for writing them. The rules are:

- There MUST be a comma after each line of an entry *except the last line*;
- There MUST NOT be a comma after the last field in the entry (*only* — eg after {London} in the example);
- Some styles recapitalise the title when they format: to prevent this, enclose the title in double curly braces as in the example;
- You MUST use extra curly braces to enclose multi-word surnames, otherwise only the last will be used in the sort, and the others will be assumed to be forenames, for example the British explorer can be sorted under T as

  ```
  author = {Ranulph {Twisleton Wykeham Fiennes}},
  ```

- Multiple authors MUST go in a single author field, separated by the literal word and (see example below);

- Values which are purely numeric (eg years) may omit the curly braces;
- Months and editions MUST be numbers (and may therefore omit the curly braces); DO NOT include ordinal indicators like th or st;
- Fields MAY occur in any order but the format MUST otherwise be strictly observed;
- 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 MAY safely delete them as unused lines).

There is a required minimum 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, conference paper (in a Proceedings), etc, exactly as with all other reference management systems. These are all (entry types and entry fields) listed in detail in the biblatex documentation (Lehman et al. 2015, sections. 2.1 & 2.2, 8).

Here's another example, this time for a book on how to write mathematics — note the multiple authors separated by and. Long entries can spread over several lines: the extra spaces and line-breaks are ignored, so long as the value ends with the matching curly brace (and comma, if needed).[5]

```
@book{mathwrite,
  author    = {Donald E Knuth and Tracey
               Larrabee and Paul M Roberts},
  title     = {{Mathematical Writing}},
  publisher = {Mathematical Association of America},
  address   = {Washington, DC},
  series    =  {MAA Notes 14},
  isbn      = {0-88385-063-X},
  year      = 1989
}
```

[5] The major differences between BIBTEX's use of these files and biblatex's use of them is that biblatex allows many more different types of fields, and is generally more up-to-date; and *biber* sorts UTF-8 correctly, and is more configurable.

Every reference in your reference database MUST have a unique key value (label or ID): you can make this up, just like you do with normal cross-references, but some bibliographic software automatically assigns a value, usually based on an abbreviation of the author and year. These keys are for *your* convenience in referencing: in normal circumstances your readers will not see them. You can see these labels in the right-hand-most column and at the bottom of the screenshot in Figure 5.1 on page 149, and in the examples above. You use these labels in your documents when you cite your references (see section 5.3.2.3).

There are many built-in options to the biblatex package for adjusting the citation and reference formats, only a few of which are covered here. Read the package documentation for details: it is possible to construct your own style simply by adjusting the settings, with no programming required (unlike the older BibTeX styles, which are written in a programming language used nowhere else).

Many users keep their BibTeX files in the same directory as their document[s], but it is also possible to tell LaTeX and BibTeX that they are in a different directory. This is a directory specified by the `$BIBINPUTS` shell or environment variable set up at installation time. On Unix & GNU/Linux systems (including Apple Macintosh OS X), and in TeX Live for Windows, this is your TeX installation's `texmf/bibtex/bib` directory — the same one that old-style BibTeX `.bst` style files are kept in — but to avoid permission conflicts you should use your Personal TeX Directory and create a subdirectory of the same name in there for your own `.bib` files. MiKTeX also uses the same `$BIBINPUTS` variable, but it is not set on installation: you need to set it using the Windows Systems Settings (see for example `www.computerhope.com/issues/ch000549.htm`).

5.3.2.3 Citation commands

The basic command is `\cite`, followed by the label of the entry in curly braces. You can cite several entries in one command: separate the labels with commas.

```
\cite{fg}
```

```
\cite{bull,davy,heller}
```

For documents with many citations, use the Cite button or menu item in your bibliographic reference manager, which will insert the relevant command for you (you can see it activated for the *TEXStudio* editor in below).

How the citation appears is governed by two things:

1. the reference format (style) you specify in the options to the biblatex package (see section 5.3.2.4 on page 150);

2. the type of citation command you use: `\cite`, `\textcite`, `\parencite`, `\autocite`, `\footcite`, etc, as shown below.

There are four built-in formats in biblatex:

authoryear: There are two basic types of author–year citation:

author as text, year in parentheses: used in phrases or sentences where the name of the author is part of the sentence, and the year is only there to identify what is being cited; this command is `\textcite{fg}`

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

This is sometimes called 'author-as-noun' citation.

whole citation in parentheses: used where the phrase or sentence is already complete, and the citation is being added in support: this command is `\parencite{fg}`

...as others have already clearly shown (Fothergill 1929).

The references at the end of the document are sorted into surname order of the first author, and by year after that.

numeric: This format is popular in some scientific disciplines where `\cite` produces just a number in square brackets, eg [42]. The references at the end of the document are usually numbered *either* in order of citation *or* in order of first surname;

alphabetic: This format is also popular in some scientific disciplines and `\cite` produces a three- or four-letter abbreviation of the author's name and two digits of the year, all in square brackets, eg [Fot29]. The references at the end of the document are sorted using abbreviated key value as their label. This format is also called 'abbreviated'.

footnoted: Footnoted citations are common in History and some other Humanities disciplines, to the extent that scholars in these fields actually call their references 'footnotes'.[6] The command `\footcite` produces a superscript number like an ordinary footnote, and a short reference at the foot of the page. It is only relevant when using author-year styles (in numeric style it would just produce the reference number at the foot of the page, which would be misleading). The references at the end of the document are given in full, and are usually sorted alphabetically by first surname.

To direct your reader to a specific page or chapter, you can add a prefix and/or a suffix as optional arguments in square brackets before the label.

```
...as shown by \textcite[p 12]{mathwrite}.
```

A prefix gets printed at the start of the citation and the suffix gets printed at the end, but all still within the parentheses, if any. As they are both optional arguments, and as suffixes are far more common than prefixes, when only one optional argument is given, it is assumed to be the suffix. The example above therefore produces:

. . . as shown by Knuth, Larrabee and Roberts (1989, p. 12).

There are many variant forms of the citation commands, either for specific styles like Chicago, Vancouver, Harvard, IEEE, APA, MLA, etc; or for grammatical modifications like capitalising name prefixes, omitting the comma between name and year, or adding multiple notes; or for extracting specific fields from an entry (eg `\titlecite`). If you have requirements not met by the formats described here, you can find them in the documentation for the biblatex package.

[6] This can be confusing to outsiders: it's not clear how they refer to conventional footnotes, or if they even use them.

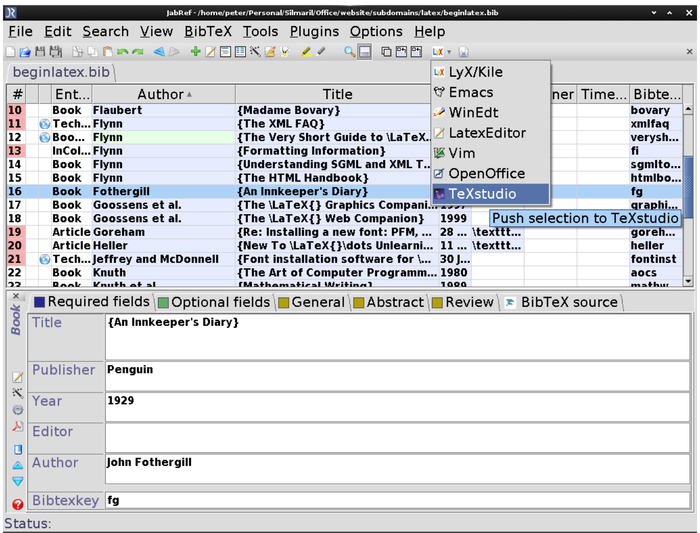
Table 5.1 – Built-in biblatex style commands and formats

Style	Command	Result
authoryear	`\parencite{fg}`	(Fothergill 1929)
authoryear	`\textcite{fg}`	Fothergill (1929)
authoryear	`\footcite{fg}`	[1]
numeric	`\cite{fg}`	[42]
alphabetic	`\cite{fg}`	[Fot29]
authoryear	`\cite{fg}`	Fothergill 1929

[1] Fothergill 1929.

Modern Language Association (MLA) citation is a special case, as it omits the year and instead REQUIRES the location of the citation within the document (eg the chapter, section, page, or line). It may include the title, if there would otherwise be ambiguity. The biblatex format for MLA citation handles the context-dependent formatting with the command `\autocite`.

Figure 5.1 – JabRef displaying a file of references, ready to insert a citation of Fothergill's book into a LaTeX document being edited with TeXStudio



Your reference management software will have a display something like above (details vary between systems, but they all do roughly the same job in roughly the same way), showing all your references with the data in the usual fields (title, author, date, etc).

Your database, which contains all your bibliographic data, MUST be saved or exported as a BIBTEX format (`.bib`) file from your reference management software (*JabRef* uses this format automatically), It looks like the examples in section 5.3.2.2 on page 143. Your `.bib` file works with both biblatex *and* BIBTEX, but biblatex provides more field types and document types so that your references can be formatted more accurately.

If your bibliographic management software doesn't save BIBTEX format direct, save your data in RIS format, then import the `.ris` file into *JabRef* and save it as a `.bib` file from there.

Cheatsheet

Clea F Rees has written an excellent cheatsheet with virtually everything on it that you need for quick reference to using biblatex. This is downloadable as the package biblatex-cheatsheet from CTAN.

5.3.2.4 Setting up biblatex with *biber*

For more complex citation requirements, you may need to set up your document with the following packages:

1. the babel or polyglossia package with appropriate languages, *even if you are only using one language*. The default language is American English, so there are commands to map this to other language variants (the example below shows this for British English);

2. the csquotes package, which automates the use of quotation marks around titles or not, depending on the type of reference;

Exercise 25 – Using biblatex

1. Use your bibliographic database program (eg *JabRef* or similar) to create a file with your references in it (see section 5.3.2.2 on page 143). Make sure each entry has a unique short keyname (**bibtexkey**) to make citations with;
2. Save the file with a name ending in `.bib` (eg `myrefs.bib`) in the same folder as your LaTeX document;
3. Add these two lines to the Preamble of your LaTeX document:

```
\usepackage[backend=biber,style=authoryear]{biblatex}
\addbibresource{myrefs.bib}
```

4. In the body of your document, where you want to cite a work, use `\parencite{keyname}` or `\textcite{keyname}` as appropriate;
5. Just before the end of your document, add the command `\printbibliography` at the point where you want the bibliography printed ;
6. Typeset the document: run *xelatex*, then *biber*, and then *xelatex* again (most editors automate this);
7. When you've got the citations and references working, read the BibTeX documentation.

3. the biblatex package itself, specifying the *biber* program and the style of references you want, either `numeric`, `alphabetic`, or `authoryear`; or a publisher's style; and any options for handling links like DOIs, URIs, and ISBNs;

4. the language mapping command, if needed (see the documentation for the style you have chosen to find out if you need this);

5. finally, the name of your BIBTEX file[s] (see the panel 'Bibliographic reference databases' on p. 141bibdb) with one or more \addbibresource commands.

```
\usepackage[frenchb,german,british]{babel}
\usepackage{csquotes}
\usepackage[backend=biber,doi=true,isbn=true,
            url=true,style=apa]{biblatex}
\DeclareLanguageMapping{british}{british-apa}
\addbibresource{myrefs.bib}
```

At the end of your document you can then add the command \printbibliography (or elsewhere that you want the full list of references you have cited to be output). See section 5.3.2.5 for details of how LATEX produces the references.

Versions of *biber* and biblatex

One critically important point to note is that biblatex and *biber* are step-versioned; that is, each version of the biblatex package only works with a specific version of the *biber* program. There is a table of these dependencies in the biblatex documentation PDF. If you manually update biblatex for some reason (perhaps to make use of a new feature), you MUST also update your copy of *biber* to the correct version, and *vice versa*, otherwise you will not be able to produce a bibliography.

5.3.2.5 Producing the references

Because of the record→extract→format process (the same as used for cross-references), you will get a warning message about 'unresolved references' the first time you process your document after adding a new citation for a previously uncited work. LATEX inserts the label of the reference in bold as a marker or placeholder until you run *biber* and re-typeset the document. This is why most editors have a Build function to do the job for you.

This function should therefore handle the business of running *biber* and re-running XƎLaTeXfor you. If not, here's how to do it manually in a Command window: you run XƎLaTeXthen run *biber* to extract and sort the details from the BibTeX file, and then run XƎLaTeXagain:

```
xelatex myreport
biber myreport
xelatex myreport
```

In practice, authors tend to retypeset their documents from time to time during writing anyway, so they can keep an eye on the typographic progress of the document. So long as you remember to click the Build or equivalent button after adding a new \cite command, all subsequent runs ofXƎLaTeXwill incrementally incorporate all references without you having to worry about it.

If you work from the command line, the *latexmk* script automates this, running *bibtex* or *biber* and re-running LaTeX again when needed.

5.4 Indexes and glossaries

Indexes and glossaries are tools for directing or helping the reader. Any book or report sized document should have an index, although they are uncommon in theses. Glossaries are usually only needed where there is a substantial number of technical terms needing formal definition and cross-referencing.

5.4.1 Indexes

LaTeX has an automated indexing facility which uses the standard *makeindex* program for sorting and collation. 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, use 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}`. You can create another level as well, so you can have subsubentries like `\index{beer!lite!American}`;

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 typographic 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* as a third-level entry and italicises it at the same time. Any of the standard `\text...` font-change commands work here: see Table 6.3 on page 190 for details.

You can also change the font of the page number on its own, for example for first-usage references, by using the vertical bar in a similar way to the 'see' entries above, but instead of 'see', substituting a font-change command name alone (*without* backslash or curly braces) such as `textbf` for a page number in bold (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 clicking the Index button or menu entry in your editor, or by typing the makeindex command followed by your document name without the .tex filetype. Most editors will do this automatically as they process your document if they spot that you have generated a .idx file.

The *makeindex* program creates a file with the same name as your document, but with the .ind filetype, containing the sorted index. This is what 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[7] for the *makeindex* program has details of how to add letter headings to each alphabet group.

5.4.2 Glossaries

Glossaries can be done in a similar manner to indexes, using the command \makeglossary in the Preamble and the command \glossary in the same way as \index. There are some subtle differences in the way glossaries are handled: both the books by Lamport (1994) and by Mittelbach et al. (2004) duck the issue, but there is some documentation on glotex on CTAN. There is also a gloss package based on BibTeX which uses \gloss in the same way as \cite.

However, by far the best way is to use the glossaries package (not glossary, which is obsolete; and not gloss either). This is a relatively complex package, as glossaries are a relatively complex tool, but there is extensive help in the documentation. It requires the *makeglossaries* script from CTAN (there is also a *makeglossariesgui* Java GUI).

Basically, you need to create a set of definitions, one per item to be glossed, using the \newglossaryentry command. Think

[7] On Unix & GNU/Linux systems (including Apple Macintosh OS X, just type the command man makeindex ; the page is also available in many reference sites on the web.

of this as being the equivalent to a reference entry in your bibliography.

```
\newglossaryentry{esis}{name={ESIS},description={The
  \textbf{Element Structure Information Set} of a
  marked-up document, originally defined for SGML
  (replaced for XML using W3C Schemas by the
  Post-Schema-Validation InfoSet, PSVI). See
  \url{http://xml.coverpages.org/WG8-n931a.html}}}
```

This specifies *a*) the label you will use (esis); *b*) the name of the item as it will be printed (ESIS); and *c*) the textual description to go in the glossary. Probably the bext place to put these is in a separate file like mygloss.tex, which you can get LaTeX to read with an \input{mygloss.tex} command in your Preamble.

You can then use the \gls command in your text to produce the printable name of any entry, using the label to refer to it, so \gls{esis} will produce 'ESIS'. It is possible to use or define variant commands to handle references at the start of a sentence (where you need a capital letter if the name is not an acronym), and grammatical alteration like unusual plurals or forms ending in '—ing'.

At the end of your document, where you want the glossary printed, you use the command \printglossaries. The glossary needs to be processed separately from the main document, using the *makeglossaries* script, exactly the same way as you do for *biber*, *bibtex*, and *makeindex*. The Build function of your editor should do this for you, or you can use a Makefile or a utility like *latexmk*. As with all these tools, there are many more facilities built into them: read the documentation.

The acronym package can also be used to create a kind of glossary containing a list of acronyms and their expansions, as well as driving the use of expansion on first mention. However, the glossaries package now contains built-in support for acronyms.

For Linnaean taxa, use the biocon package, which automates the way in which plant and animal species names and ranks are typeset and referred to in formal writing. There are other more complex packages for managing taxonomies.

5.5 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. There is a *multicols** environment which does not try to balance the columns.

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.

6 Layouts and fonts

This is the chapter that most users think they 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 you will have seen by now, this is normally unnecessary in LaTeX, which does most of the 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.

6.1 Changing layout

The design of the page can be a very subjective matter, and also a rather 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 1978 or 1991 may look rather dated in 2023.

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.[1] However, there are a

[1] Some authors — and perhaps some designers — believe that consistency is undesirable, and that double-page layouts in printed books should each be

number of different places where the layout usually *does* change, related to the frequency with which the format appears.

1. In books, the title page, the half-title, copyright and legal pages, dedication, acknowledgements, 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.

2. The Table of Contents and related lists like the List of Figures, List of Tables, List of Acronyms, Bibliography, References, Glossary, etc SHOULD all share one design with the preliminary sections like Preface, Introduction, and Foreword, which SHOULD be at section level, not chapter level (or in an article, at subsection level, not section level).

3. Chapter and Appendix start pages SHOULD always share a layout.

4. Other (normal) pages have a single layout, but within the page there MAY be individual variations to handle tables, lists, figures, sidebars, exercises, footnotes, etc.

The things that normally never change are the page size, margins, and body font size. There are very rare and exceptional circumstances when you can do this, but normally, once set, they stay fixed.

The exceptions to this are newspapers and magazines, where page layout is done individually, page by page (or pairs of facing pages together), but even here, most publications have strict rules about what blocks of material can be placed where, and use a carefully-designed set of templates to achieve this. While it would be technically possible to implement the huge range of page layouts needed by newspapers and magazines in LaTeX, it would be impracticable to use them under the publication

designed independently. Kirschenbaum's magnificent *Goodbye Gutenberg* expresses this both eloquently and attractively, but the cost of such design labour and the cost of four-colour printing on all pages places it beyond the reach of most publishers' budgets until the economics of on-demand four-colour 'printing' makes it possible.

Layout conventions

Contrary to popular assumption (and contrary to LaTeX's defaults), navigation lists and any prelims and postlims (item above) SHOULD NOT be chapter-level headings but section-level. Only chapters and appendices should be at chapter-level in terms of layout and their rank in the Table of Contents.

deadline pressures in this field, as there is a constant need for modifications which would require a large number of LaTeX-skilled programmers to implement.

If you are going to design a whole document yourself, 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 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.

6.1.1 Margins and spacing

We mentioned in section 5.2 on page 137 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 section 3.1.3 on page 69 for how to read package documentation). Here is an example:

```
\usepackage[a4paper,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.

Once it knows the paper size, if you give it the text width and the left-hand margin, for example, it can work out the right-hand margin. The package also provides the \newgeometry command, which lets you reset the margin settings in mid-document (at a page break, of course). This probably isn't something you want to do very often, though.

The spacing around the individual textual components (paragraphs, lists, footnotes, tables, figures, etc) can also be changed on a document-wide basis, as we saw with paragraph spacing and indentation in section 2.7 on page 63. There are a lot of packages available to do various aspects of this, far too many to go into detail here: search CTAN 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 enumitem 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 *Companion* or ask for expert help, as there are many extra settings which can also be changed to fine-tune your design, but which need some understanding 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:

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, so 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, eg \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, eg \vspace*{19pt}.

Double line-spacing: LaTeX's \baselinestretch value governs the amount of *extra* line-spacing based on the current font size (see section 6.2.5 on page 191). By default it is null, meaning no extra space. It is possible to set it to a *multiplier*, like \renewcommand{\baselinestretch}{1.75} to make it 1.75 times normal. However...

Double-spacing normal lines of text is usually A Bad Idea, as it looks very ugly, but increased line-spacing does become important if you are typesetting very wide lines, otherwise the reader's eye will not be able to pick up the start of a new line easily.

Double-spacing is still a requirement in many universities for thesis submission, partly because of the tendency of writers to use very wide lines on office-type paper sizes, and partly because the reviewers needed space to write in corrections. With the growth of electronic submission and editorial corrections in PDF files, it should become less necessary. Nowadays, 1⅓ or 1½ line spacing is considered acceptable, according to your font size.

Use the setspace package to do this. It has commands for double line-spacing (\doublespacing) and for one-and-a-half line spacing (\onehalfspacing): the \singlespacing command resets them). There is also a *spacing* environment to let you specify a different multiple as the argument:

```
\begin{spacing}{1.333}
...
\end{spacing}
```

Be aware that you may not want footnotes to be spaced by the same multiple as your normal text, and you may want other elements like lists, tables, figures, or quotations spaced differently.

As with theses, there are some perfectly genuine and normal reasons for wanting bigger 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, 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 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` (1/6em), which we saw between single and double quotes in section 1.8 on page 25. It's also sometimes used between the full point after abbreviations and a following number, as in page references like p. 42, where a word space would look too big, and setting it solid would look too tight.
- `\enspace` (1/2em). 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 numbers in listings are easier to line up.
- `\quad` (1em) was originally the width of a capital M in metal type.
- `\qquad` (2em) is double a `\quad`.

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 adjustable 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]

6.1.2 Headers and footers

LaTeX has built-in settings to control the page style of its default page layouts, and space at the top and bottom of the page is provided automatically for them (it can also be adjusted or turned off in the geometry package). These settings are implemented with the `\pagestyle` command, which can take one of the following arguments in curly braces:

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 has no meaning;

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 extracted or printed or copied separately;

myheadings 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

[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.

left-hand, centre, and right-hand headers and footers for both odd-numbered (left-hand) and even-numbered (right-hand) pages (twelve objects in all).

Table 6.1 – Header and footer locations in the fancyhdr package

top left, even	top centre, even	top right, even	top left, odd	top centre, odd	top right, odd
	LH page, even-numbered			RH page, odd-numbered	
bottom left, even	bottom centre, even	bottom right, even	bottom left, odd	bottom centre, odd	bottom right, odd

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.

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

```
                \footnotesize\sffamily\thepage}}
\fancyfoot[L]{\footnotesize\sffamily\@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 commands 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.

6.1.3 List spacing

The different types of list are explained in section 4.1 on page 88.

To change the format of lists, use the enumitem package as recommended in section 4.1 on page 88. LaTeX's default list layouts are generously spaced, with wide indentation, and a blank line above and beneath and between items. They do, however, cope excellently with continuation paragraphs (additional paragraphs within an item), which many other systems confuse with unnumbered items.

1. A very common requirement is the unspaced or compact list: enumitem provides two: *noitemsep*, which removes the vertical white-space between items; and *nosep*, which additionally removes the vertical white-space above and below the list.

2. Bullets and numbering can be changed using the *label* option. Margin spacing can be changed to accommodate very wide or very narrow bullets or numbers.

3. Description lists can be restyled in a variety of ways. However, you should probably pick one way of formatting for the whole document, and not go changing it for every list.

All of the factors controlling the list shape can be reset, but you need to be careful that you don't make the list unreadable by closing up the spacing too much when the items are large (multi-line).

6.2 Using fonts

> 'Why do we need more fonts?' asked Bill Gates. 'We've got a serif, a sans, and a monospace font. Why do we need more?'
>
> (Berry 2017)

The default typeface in LaTeX is Computer Modern (CM). This typeface was created by Knuth for use with TeX. It is based on a Victorian book typeface, Monotype Series 8, because he designed TeX originally for typesetting books. Because it is one of the very few book typefaces with a comprehensive set of mathematical fonts, it has remained the default, rather than the variations on Times that you find in wordprocessors and other DTP systems (until recently the full set of mathematical symbols for Times were an expensive commercial add-on).

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.

The standard distribution of TeX Live comes with about 130 OpenType and 75 TrueType typefaces (see section 6.2.2.1 on page 174). There are also some 300 Postscript Type 1 typefaces (many of these are PS versions of the OT and TT faces), and about

165 legacy METAFONT (Postscript Type 3) typefaces, to preserve compatibility with older documents which use them.

LaTeX can use more different types of font than any other system

- ☐ The original LaTeX could use any METAFONT font;
- ☐ pdfLaTeX could use any METAFONT or Postscript Type 1 font;
- ☐ XeLaTeX and LuaLaTeX can use any METAFONT or Postscript Type 1 or TrueType or OpenType font.

6.2.1 First time only: setting up fonts

XeLaTeX and LuaLaTeX let you use all your *system fonts* (those that came preinstalled with your computer and your non-TeX software) as well as the ones that came with your TeX distribution. There is one small piece of preparation to do, the very first time you use LaTeX: index them (properly speaking, cache them) for fast access. After that, you only need to re-index them if you buy or download a new font.

Without this indexing, you can still use your system fonts in LaTeX documents but you would have to type in where to find each font file every time, which is tedious.

There are four places where fonts are usually installed on Linux TeX systems:

1. the system fonts directory (installed by your operating system, wordprocessor, and other 'office' software);

2. the TeX distribution font directory (where the fonts go that came with TeX);

3. the shared TeX additional fonts directory (only relevant for multi-user shared systems; traditionally this is where system managers would put extra fonts for everyone to use);

4. the 'local' shared TEX fonts directory (only relevant for multi-user shared systems: traditionally this is where authorised users would put extra fonts for everyone to use);
5. your Personal TEX Directory fonts subdirectory (where you put fonts you buy or download for your own use).

The last one (your Personal TEX Directory) is automatically searched by LATEX and never needs indexing, so it is not in the examples below.

Each `fonts` directory will normally have subdirectories for the different types of font, eg `truetype`, `opentype`, `type1`, etc.

6.2.1.1 Indexing your fonts under Linux

This section covers three types of Linux installation:

1. TEX Live installed from the TUG DVD or download on any type of Linux (see Exercise 26 on p. 171);
2. Debian and its derivatives like Ubuntu that share the `.deb` repositories (see Exercise 27 on p. 172);
3. Red Hat and its derivatives like CentOS that share the `.rpm` repositories (see Exercise 28 on p. 173).

The principles are the same but the names of the directories differ slightly. If you have information on how this works on other types of distribution (eg Arch, SuSE, etc) please contact the author.

6.2.1.2 Indexing your fonts under Windows

TEX Live installed from the TUG DVD or download on Windows systems needs no separate configuration, as the *fc-cache* program (included with TEX Live) is run automatically after installation.

If you intend adding new fonts that you have bought or downloaded, you should create a Personal TEX Directory with the appropriate subdirectories (eg `fonts/truetype` etc) and put the fonts in there. No indexing is then needed.

(Be aware that MiKTEX *does* require that you update MiKTEX's File Name Database (FNDB) when you add new fonts or personal (non-CTAN) packages: see **3.** on page 274 for details.)

Exercise 26 – Font indexing in TeX Live installed from TUG DVD or download on Linux systems

tug.org/texlive/doc/texlive-en/texlive-en.html#xetexfontconfig contains TUG's online details which you can check for updates.

1. Open a Command or Terminal window;
2. Become root by typing `sudo su –` and giving your password when asked;
3. Determine the location of your TeX Live installation by typing

```
$ kpsewhich -var-value TEXMFSYSVAR
```

4. Using the value of the location of `$TEXMFSYSVAR` found in step above, copy the file at `TEXMFSYSVAR/fonts/conf/texlive-fontconfig.conf` to `/etc/fonts/conf.d/09-texlive.conf`;
5. Update the font cache:

```
$ fc-cache -fsv
```

   (wait a few minutes while it caches your fonts).
6. Type `exit` to leave superuser mode;
7. Close the window if you want.

6.2.1.3 Indexing your fonts under Apple Mac OS X

The Apple Mac distribution of TeX Live, MacTeX, can already use the Mac systems fonts, but you need to add the TeX Live fonts via the *FontBook* app.

Exercise 27 – Font indexing in TEX Live installed from `.deb` repositories on Debian-based Linux systems

1. Open a Command or Terminal window;
2. Become root by typing `sudo su -` and giving your password when asked;
3. Open your favourite text editor (eg *emacs*, *vi*, *kate*, *gedit*, etc);
4. Create a new, empty file `/etc/fonts/09-texlive.conf`;
5. Copy and paste this configuration into the file:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
  <!-- Fonts installed by Debian's tex* packages -->
  <dir>/usr/share/texlive/texmf-dist/fonts/opentype</dir>
  <dir>/usr/share/texlive/texmf-dist/fonts/truetype</dir>
  <dir>/usr/share/texlive/texmf-dist/fonts/type1</dir>
  <!-- Fonts installed on shared multiuser TeX systems -->
  <dir>/usr/share/texmf/fonts/opentype</dir>
  <dir>/usr/share/texmf/fonts/truetype</dir>
  <dir>/usr/share/texmf/fonts/type1</dir>
  <!-- Fonts pre-installed by the operating system -->
  <dir>/usr/share/fonts/opentype</dir>
  <dir>/usr/share/fonts/truetype</dir>
  <dir>/usr/share/fonts/type1</dir>
  <!-- Fonts added by users on some systems -->
  <dir>/usr/local/share/fonts/opentype</dir>
  <dir>/usr/local/share/fonts/truetype</dir>
  <dir>/usr/local/share/fonts/type1</dir>
</fontconfig>
```

6. Save the file;
7. Update the font cache:

```
fc-cache -fsv
```

   (wait a few minutes while it caches your fonts).
8. Type `exit` to leave superuser mode;
9. Close the window if you want.

Exercise 28 – Font indexing in TeX Live installed from `.rpm` repositories on RedHat-based Linux systems

1. Open a Command or Terminal window;
2. Become root by typing `sudo su -` and giving your password when asked;
3. Open your favourite text editor (eg *emacs*, *vi*, *kate*, *gedit*, etc);
4. Create a new, empty file in `/etc/fonts/` called `09-texlive.conf`;
5. Copy and paste this configuration into the file:

```
<?xml version="1.0"?>
<!DOCTYPE fontconfig SYSTEM "fonts.dtd">
<fontconfig>
  <dir>/usr/local/texlive/YYYY/texmf-dist/fonts/opentype</dir>
  <dir>/usr/local/texlive/YYYY/texmf-dist/fonts/truetype</dir>
  <dir>/usr/local/texlive/YYYY/texmf-dist/fonts/type1</dir>
  <dir>/usr/local/texlive/texmf-local/fonts/opentype</dir>
  <dir>/usr/local/texlive/texmf-local/fonts/truetype</dir>
  <dir>/usr/local/texlive/texmf-local/fonts/type1</dir>
  <dir>/usr/share/fonts</dir>
</fontconfig>
```

6. Replace YYYY with the year of your TeX Live distribution;
7. Save the file;
8. Update the font cache:

```
fc-cache -fv
```

   (wait a few minutes while it caches your fonts).
9. Type `exit` and close the window.

6.2.2 Set the default font family for a document

As explained in section 6.2 on page 168, Computer Modern is the built-in default typeface, so that's what you get if you don't specify anything else. There are three ways to specify other typefaces and individual fonts: *a*) by using a package; *b*) by font name; or *c*) by filename.

Using a package is more convenient, especially for whole typefaces, because the configuration of all the component fonts (eg roman, italic, bold, bold-italic, math, etc) has already been done by the package author, but font names or filenames let you specify your system (non-TeX) fonts, which packages cannot do.

6.2.2.1 OpenType and TrueType typeface packages for XeLaTeX and LuaLaTeX

The list below gives packages for about 40 OpenType (OT) and TrueType (TT) typefaces installed with a full distribution of TeX, and below that another 20 or so which can be downloaded from CTAN. Both sets are listed in `pkks.de/fontpackages.html`.

(A few packages are not included here because they are not actually fonts in themselves, but 'enabling' packages which make specific combinations available for special purposes, such as the hep-font package for math combinations for the High Energy Physics community.)

Most of these packages support a *default* option, which sets them as the default font for the document, eg

```
\usepackage[default]{cabin}
```

will set the Cabin typeface as the default for the document. The fontspec package, which is required for using OT and TT fonts, is built into these packages and does not need to be specified separately.

OpenType and TrueType faces available at installation

(Samples are links to the package pages)

`Alegreya` Alegreya	Sphinx of black quartz, judge my vow

`almendra` Almendra	*Sphinx of black quartz, judge my vow*
`bitter` Bitter	Sphinx of black quartz, judge my vow
`cabin` Cabin	Sphinx of black quartz, judge my vow
`cantarell` Cantarell	Sphinx of black quartz, judge my vow
`Chivo` Chivo	Sphinx of black quartz, judge my vow
`cinzel` Cinzel	SPHINX OF BLACK QUARTZ, JUDGE MY VOW
`cochineal` Cochineal	Sphinx of black quartz, judge my vow
`coelacanth` Coelacanth	Sphinx of black quartz, judge my vow
`comfortaa` Comfortaa	Sphinx of black quartz, judge my vow
`crimson` Crimson Test	Sphinx of black quartz, judge my vow
`CrimsonPro` Crimson Pro	Sphinx of black quartz, judge my vow
`dejavu-otf` DejaVu	Sphinx of black quartz, judge my vow
`droidsans` Droid Sans	Sphinx of black quartz, judge my vow
`droidserif` Droid Serif	Sphinx of black quartz, judge my vow

`ebgaramond` EB Garamond	Sphinx of black quartz, judge my vow
`forum` Forum	Sphinx of black quartz, judge my vow
`lato` Lato	Sphinx of black quartz, judge my vow
`libertine` Linux Libertine	Sphinx of black quartz, judge my vow
`libertinus-otf` Libertinus	Sphinx of black quartz, judge my vow
`LibreBodoni` Libre Bodoni	Sphinx of black quartz, judge my vow
`librecaslon` Libre Caslon Text	Sphinx of black quartz, judge my vow
`linguisticspro` Linguistics Pro	Sphinx of black quartz, judge my vow
`marcellus` Marcellus	Sphinx of black quartz, judge my vow
`merriweather` Merriweather	Sphinx of black quartz, judge my vow
`newpxtext` New PX Text	Sphinx of black quartz, judge my vow
`newtxtext` New TX Text	Sphinx of black quartz, judge my vow
`noto-serif` Noto	Sphinx of black quartz, judge my vow
`OldStandard` Old Standard	Sphinx of black quartz, judge my vow

`opensans` Open Sans	Sphinx of black quartz, judge my vow
`quattrocento` Quattrocento	Sphinx of black quartz, judge my vow
`Rosario` Rosario	Sphinx of black quartz, judge my vow
`sourceserifpro` Source Serif Pro	Sphinx of black quartz, judge my vow
`TheanoDidot` Theano Didot	Sphinx of black quartz, judge my vow
`TheanoModern` Theano Modern	Sphinx of black quartz, judge my vow
`TheanoOldStyle` Theano Old Style	Sphinx of black quartz, judge my vow
`tinos` Tinos	Sphinx of black quartz, judge my vow

These packages represent a selection of typefaces from suppliers like Google; donated TEX sources like CTAN; foundries like Impallari, Summer Institute of Linguistics (SIL), Ascender, and many others; and learned societies and individuals. Those in the list above are installed with TEX Live; those in the list below can be downloaded and installed from CTAN.

More OpenType and TrueType faces available from CTAN

(Samples are links to the package pages)

`accanthis` Accanthis ADF Std	Sphinx of black quartz, judge my vow
`andika` Andika	Sphinx of black quartz, judge my vow
`caladea` Caladea	Sphinx of black quartz, judge my vow

`CharisSIL` Charis SIL	Sphinx of black quartz, judge my vow
`CormorantGaramond` Cormorant Garamond	Sphinx of black quartz, judge my vow
`fourier-otf` Fourier OTF	Sphinx of black quartz, judge my vow
`garamondlibre` Garamond Libre	Sphinx of black quartz, judge my vow
`gfsneohellenicot` GFS Neo-Hellenic OT	Sphinx of black quartz, judge my vow
`heros-otf` TEX Gyre Heros (Helvetica)	Sphinx of black quartz, judge my vow
`ibarra` Ibarra Real Nova	Sphinx of black quartz, judge my vow
`imfellEnglish` IM Fell English	Sphinx of black quartz, judge my vow
`kpfonts-otf` Kepler Project OT	Sphinx of black quartz, judge my vow
`lexend` Lexend	Sphinx of black quartz, judge my vow
`librebaskerville` Libre Baskerville	Sphinx of black quartz, judge my vow
`noto` Noto	Sphinx of black quartz, judge my vow
`pagella-otf` TEX Gyre Pagella (Palatino)	Sphinx of black quartz, judge my vow

`PlayfairDisplay` Playfair Display	Sphinx of black quartz, judge my vow
`plex-serif` Plex Serif	Sphinx of black quartz, judge my vow
`roboto` Roboto	Sphinx of black quartz, judge my vow
`schola-otf` TEX Gyre Schola (Century)	Sphinx of black quartz, judge my vow
`spectral` Spectral	Sphinx of black quartz, judge my vow
`xcharter-otf` Xcharter	Sphinx of black quartz, judge my vow

LATEX includes versions of the popular (some would say overused) Adobe '35' fonts which have been built into PDF readers, laser printers, printer drivers, and most DTP systems since the dawn of desktop publishing shortly after TEX was written. These comprised eight text (Latin-alphabet) typefaces and two fonts of symbols or dingbats (35 fonts in total). They are now provided by non-Adobe versions known as the 'TEX Gyre' collection, derived from the Unternehmensberatung Rubow Weber (URW) equivalents shown in the list below.

The replacements for the old Adobe '35'

(Samples are links to the package pages)

`tgadventor` URW Gothic L (ITC Avant Garde)	Sphinx of black quartz, judge my vow
`tgbonum` URW Bookman L (Bookman Old Style)	Sphinx of black quartz, judge my vow

`tgchorus` URW Chancery L Medium Italic (ITC Zapf Chancery)	*Sphinx of black quartz, judge my vow*
`tgcursor` URW Nimbus Mono L (IBM Courier)	Sphinx of black quartz, judge my vow
`tgheros` URW Nimbus Sans L (Helvetica)	Sphinx of black quartz, judge my vow
`tgschola` URW Century Schoolbook L (Century Schoolbook)	Sphinx of black quartz, judge my vow
`tgpagella` URW Palladio L (Palatino)	Sphinx of black quartz, judge my vow
`tgtermes` URW Nimbus Roman No9 L (Times New Roman)	Sphinx of black quartz, judge my vow

(If you need the old Microsoft Symbol font, it can be downloaded, but Scott Pakin's Comprehensive LaTeX Symbol List is probably a better place to find symbols. The same applies to the Zapf Dingbats font, for which the bbding and marvosym packages provide alternatives. Jonathan Kew has posted details of how to access the actual Zapf Dingbats individually if required.)

The original Computer Modern typeface family was a METAFONT design by Donald Knuth, and was accompanied by a selection of other fonts, also made using METAFONT, some of which are in the list below

Some of the original fonts still in use

`cmr` Computer Modern Roman	Sphinx of black quartz, judge my vow

cmss Computer Modern Sans-Serif	Sphinx of black quartz, judge my vow
cmtt Computer Modern Typewriter	`Sphinx of black quartz, judge my vow`
panr Pandora Roman	Sphinx of black quartz, judge my vow
pss Pandora Sans	Sphinx of black quartz, judge my vow
pntt Pandora Typewriter	**Sphinx of black quartz, judge my vow**
uni Universal	Sphinx of black quartz, judge my vow
ccr Concrete	Sphinx of black quartz, judge my vow
eiad Eiad	Nil aon tinteán mar do ṫinteán féin
rust Rustic	**SPHINX OF BLACK QUARTZ, JUDGE MY VOW**
uncl Uncial	**sphinx of black quartz, judge my vow**
cdr Dürer	SPHINX OF BLACK QUARTZ JUDGE MY VOW

The X Consortium donated a number of Latin-alphabet fonts in Postscript Type 1 format: have a look at Charter, Utopia,[3] URW Antiqua Condensed, and URW Grotesk. There are hundreds of other fonts downloadable from CTAN: see Palle Jørgensen's comprehensive LaTeX Font Catalogue published by the Danish TeX Users Group, categorised by type (serif, sans, monospace, decorative, etc) with samples and links to the packages.

[3] The licence for Utopia does not allow it to be distributed automatically to users, but you can download it personally.

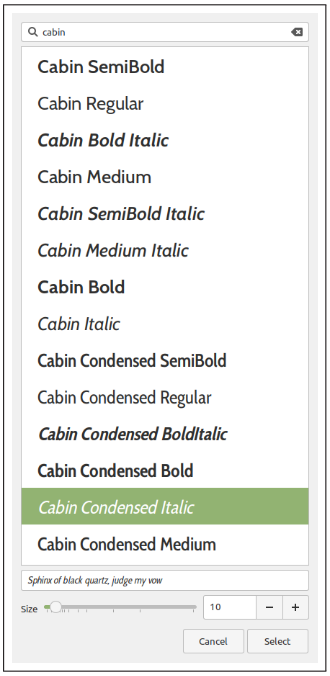
At any given time, TEX expects there to be three typefaces available: `rm` for Roman (Serif), `sf` for Sans-Serif, and `tt` for Typewriter (Monospace).

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`

It is common to want to change all three defaults at the same time in order for new fonts to match each other. If a typeface provides Roman, Sans, and Monospace all in its own style, the package will change all three defaults automatically. Packages loading a single font family just load that one, so the others would remain Computer Modern. You can load another package for other fonts to replace them, or specify them individually as shown in the rest of this section.

6.2.2.2 OpenType and TrueType fonts and faces by fontname

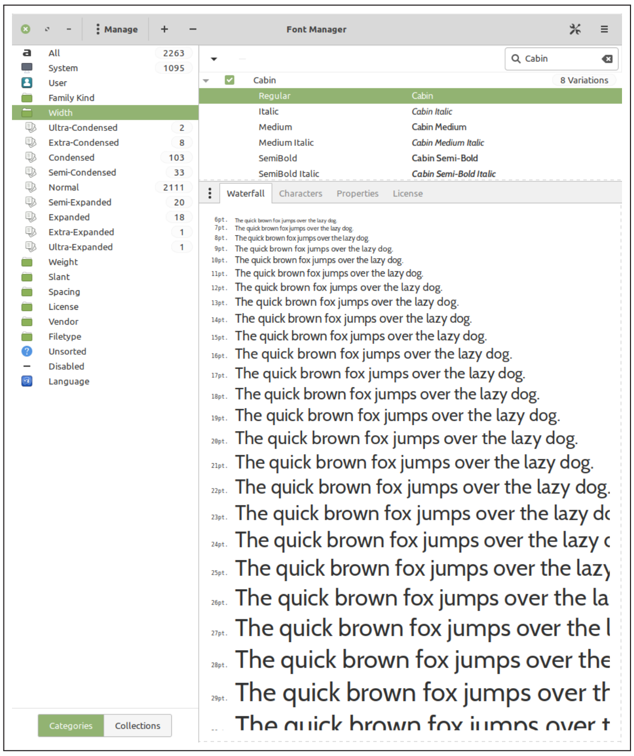
The *fontname* of a font is the name that the designer declares is the name of the font family (like Cabin), or sometimes the name of the individual font (like Almendra-Bold). It is *not* the filename, although sometimes they happen to be the same. The fontname is the name you see in lists of fonts like the font dialog drop-down menu in editors.



You need the fontspec package to use OT and TT faces and fonts by fontname. This package provides three commands to select font families: \setmainfont (for the roman or main face); \setsansfont (for the sans-serif face); and \setmonofont (for the typewriter or monospace face). These all take one compulsory argument: the fontname of the font family or typeface (we will see in section 6.2.2.3 on page 185 how to do this with filenames).

Most of the time that's all you need.

You can find the fontnames of any of your installed fonts by using your font browser or indexing command provided by your operating system.



On Unix & GNU/Linux systems, and on Windows systems with TeX Live installed, the command `fc-list` can be used to list your installed fonts, giving the filename (location), fontname (and synonyms), and the style, eg

```
$ fc-list Cabin
/usr/share/fonts/opentype/
  cabin/Cabin-Bold.otf: Cabin:style=Bold
  cabin/Cabin-BoldItalic.otf: Cabin:style=Bold Italic
  cabin/Cabin-Italic.otf: Cabin:style=Italic
  cabin/Cabin-Medium.otf: Cabin:style=Medium
  cabin/Cabin-MediumItalic.otf: Cabin:style=Medium Italic
  cabin/Cabin-Regular.otf: Cabin:style=Regular
  cabin/Cabin-SemiBold.otf: Cabin:style=SemiBold
  cabin/Cabin-SemiBoldItalic.otf: Cabin:style=SemiBold Italic
/usr/share/texlive/texmf-dist/fonts/opentype/
  impallari/cabin/Cabin-Bold.otf: Cabin:style=Bold
  impallari/cabin/Cabin-BoldItalic.otf: Cabin:style=Bold Italic
  impallari/cabin/Cabin-Italic.otf: Cabin:style=Italic
  impallari/cabin/Cabin-Medium.otf: Cabin:style=Medium
  impallari/cabin/Cabin-MediumItalic.otf: Cabin:style=Medium Italic
  impallari/cabin/Cabin-Regular.otf: Cabin:style=Regular
  impallari/cabin/Cabin-SemiBold.otf: Cabin:style=SemiBold
  impallari/cabin/Cabin-SemiBoldItalic.otf: Cabin:style=SemiBold Italic
```

From which you can see that, on my own system, I have Cabin installed twice: once by the operating system and once by TEX Live. The important bit is between the first and second colons: the fontname Cabin, in this case a font family because there are no other values. Each entry gives the filename, the fontname, and any variants, separated by colons. If I do the same for Almendra, I get:

```
$ fc-list Almendra
/usr/share/texlive/texmf-dist/fonts/truetype/public/
  almendra/AlmendraSmallCaps.ttf: AlmendraSmallCaps:style=Regular
  almendra/Almendra-Bold.ttf: Almendra:style=Bold
  almendra/Almendra-Regular.ttf: Almendra:style=Regular
  almendra/Almendra-BoldItalic.ttf: Almendra:style=Bold Italic
  almendra/Almendra-Italic.ttf: Almendra:style=Italic
```

Some font entries have multiple values for the fontname, separated by commas, like Alegreya,Alegreya Black. In these cases the font family's fontname is the first one (before the first comma).

6.2.2.3 OpenType and TrueType fonts and faces by filename

Both TT and OT fonts are usually named in a pattern where the font family name (eg LiberationSerif, all one word) is followed by the variant (regular, italic, bold, bold-italic, etc) separated by a hyphen or underscore character, or sometimes by 'camel-casing', where descriptors starting mid-word use a capital letter. Here is the directory listing of my installation of the Liberation typeface:

```
$ ls -l /usr/share/fonts/truetype/liberation
total 2088
-rw-r--r-- 1 root root 118296 Feb 22  2020 LiberationMono-BoldItalic.ttf
-rw-r--r-- 1 root root 105460 Feb 22  2020 LiberationMono-Bold.ttf
-rw-r--r-- 1 root root 124012 Feb 22  2020 LiberationMono-Italic.ttf
-rw-r--r-- 1 root root 108172 Feb 22  2020 LiberationMono-Regular.ttf
-rw-r--r-- 1 root root 135124 Feb 22  2020 LiberationSans-BoldItalic.ttf
-rw-r--r-- 1 root root 137052 Feb 22  2020 LiberationSans-Bold.ttf
-rw-r--r-- 1 root root 162036 Feb 22  2020 LiberationSans-Italic.ttf
-rw-r--r-- 1 root root 128468 Feb 22  2020 LiberationSansNarrow-BoldItalic.ttf
-rw-r--r-- 1 root root 110252 Feb 22  2020 LiberationSansNarrow-Bold.ttf
-rw-r--r-- 1 root root 132452 Feb 22  2020 LiberationSansNarrow-Italic.ttf
-rw-r--r-- 1 root root 113028 Feb 22  2020 LiberationSansNarrow-Regular.ttf
-rw-r--r-- 1 root root 139512 Feb 22  2020 LiberationSans-Regular.ttf
-rw-r--r-- 1 root root 151452 Feb 22  2020 LiberationSerif-BoldItalic.ttf
-rw-r--r-- 1 root root 147132 Feb 22  2020 LiberationSerif-Bold.ttf
-rw-r--r-- 1 root root 145028 Feb 22  2020 LiberationSerif-Italic.ttf
-rw-r--r-- 1 root root 152408 Feb 22  2020 LiberationSerif-Regular.ttf
```

As you can see, there are four families of the Liberation typeface: Mono, Sans, Sans Narrow, and Serif. In each case there is a Regular (roman), Italic, Bold, and Bold Italic variant. The fontspec lets you load a font by specifying the filetype (extension) of the font file, where it is installed (the path), and how the font filenames fit the pattern:

```
\setmainfont{LiberationSerif}[Extension=.ttf,
  Path=/usr/share/fonts/truetype/liberation/,
  UprightFont=*-Regular,
  BoldFont=*-Bold,
  ItalicFont=*-Italic,
  BoldItalicFont=*-BoldItalic]
```

The pattern-matching is done by an asterisk which gets replaced by the font family name you give in the first argument, and the filetype (extension) is added to the end, so that fontspec can construct the whole filename. Repeating this for the sans and mono variants gives us the full set (the Narrow face is an exception and less commonly used):

```
\setsansfont{LiberationSans}[Extension=.ttf,
  Path=/usr/share/fonts/truetype/liberation/,
  UprightFont=*-Regular,
  BoldFont=*-Bold,
  ItalicFont=*-Italic,
  BoldItalicFont=*-BoldItalic]
\setmonofont{LiberationMono}[Extension=.ttf,
  Path=/usr/share/fonts/truetype/liberation/,
  UprightFont=*-Regular,
  BoldFont=*-Bold,
  ItalicFont=*-Italic,
  BoldItalicFont=*-BoldItalic]
```

These three now automatically fill the LaTeX roles of the `rm`, `sf`, and `tt` families. The advantage of using filenames is that you do not have to rely on XeLaTeX and LuaLaTeX accessing the fonts via the font index/cache system, as it goes straight to the filename in the directory you specify.

If you are mixing OT or TT fonts from different families, you can use the *Scale=MatchLowercase* option for the sans and mono setups, making them scale to match the roman face, exactly as in Exercise 30 on p. 201tryfonts.

6.2.3 Changing the font-family temporarily

To use a different font [family] for a specific purpose, use the command \newfontfamily. This works exactly the same as the commands above for setting the main, sans, and mono font families but takes an extra parameter first, to specify the command you want it known by, so it does *not* replace the Roman, Sand, or Monospace defaults but is available *in addition* to them. To make the command \tablesfont invoke Liberation Sans Narrow, for example, you would use:

```
\newfontfamily{\tablesfont}{Liberation Sans Narrow}
```

If you want to use the filename, the format is:

```
\newfontfamily{\tablesfont}{LiberationSansNarrow}[
  Extension=.ttf,
  Path=/home/peter/texmf/fonts/truetype/liberation/,
  UprightFont=*-Regular,
  BoldFont=*-Bold,
  ItalicFont=*-Italic,
  BoldItalicFont=*-BoldItalic]
```

Then the new command (here, \tablesfont) can be used to switch to that typeface.

To load a solitary font (that is, not a whole family), there is a command \newfontface, which also works in the same way, by creating a new command to switch to it.

```
\newfontfamily{\headlinefont}{LobsterTwo-Bold}
  [Extension=.otf,
   Path=/usr/share/fonts/opentype/lobstertwo/]
...
{\centering\headlinefont\fontsize{20}{24}\selectfont
  Lobster Rolls\dotfill \$3.95\par}
```

Lobster Rolls **$3.95**

These commands created by \newfontfamily, like the ones in Table 6.2 on the next page, are called 'unscoped' because they have global effect from that point on. In order to restrict the effect to a smaller scope (a few words, for example), you MUST put the command *and* the text inside a *group* (enclosed in curly braces as in the example, or within an environment), otherwise they will apply to the end of the document. See the panel 'Grouping' on p. 202 for more detail.

In a normal document, of course, arbitrary typeface changes like this are rare: people don't (or at least, probably shouldn't) randomly flip from one font to another. You select your default typefaces once, using packages or commands, at the start of the document, and stick with them — bold and italics are handled by the document class or stylesheet packages you use.

However, in advertising or magazines, a wide range of typefaces changes is common, but they are usually part of predefined styles for handling that type of formatting, built into the document class, so it is rare to have to do them manually.

Most cases where people want unusual typeface changes involve things like special symbols or effects on a repetitive basis, and LaTeX provides much easier (programmable) ways to make these changes into shorthand commands (called macros: see Chapter 7 starting on page 203).

This is jumping ahead a bit, but you could, for example, define a new macro called \product which would let you typeset product names in a distinct typeface (usually italics):

```
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 the font-family name in the macro, 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.

Vastly more common are changes to type *style*, while staying within the same font-family.

6.2.4 Changing type style

Within each typeface or font family there are usually several different 'looks' to the type design. LaTeX distinguishes mainly between ***font shape*** and ***font series***. *Italics* is a shape (look carefully: the actual shape of the letters changes, as well as their slope); whereas **bold** is a series (same shapes, same slope, just thicker strokes).

Table 6.2 – Typeface styles, families, shapes, and series (unscoped)

Type style	Command	Example
Upright	`\upshape*`	The quick brown fox jumps over the lazy dog
Italic	`\itshape`	*Quick brown fox jumps over the lazy dog*
Slanted	`\slshape*`	*Quick brown fox jumps over the lazy dog*
Small Caps	`\scshape*`	QUICK BROWN FOX JUMPS OVER LAZY
Bold	`\bfseries*`	**Quick brown fox jumps over the**
Extended	`\bfseries†`	**Quick brown fox jumps over the lazy**
Sans-serif	`\sffamily`	The quick brown fox jumps over the lazy dog
Monospace	`\ttfamily`	`quick brown fox jumps over the lazy`

These commands are *unscoped*, so they must be used within a group.
* 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.

The 'shape', 'series', and 'family' commands in above are ***commutative*** (each adds to the preceding form), so you can combine a shape with a series and/or a family, without the need to use `\selectfont`:

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

☞ This is **bold *italic* *sans-serif*** type.

Beware of pushing your fonts beyond their limits unless you have typographic skills. 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 also sometimes impossible to combine one family with another (such as seriffed sans-serif type!). Slanted plus italics, for example, doesn't make any sense, as italics are already slanted; and while some typefaces may well possess sans-serif italic small caps, they are not in common use.

If you really feel you need such combinations, try the fontaxes,which splits the 'shape' axis into a primary axis (upright, italic, slanted, upright italic, etc) and a secondary axis (small caps on or off). It redefines the `\itshape` and `\scshape` commands to combine instead of override each other. The fontspec package loads fontaxes. So do many legacy font packages. (Thanks to @Davislor on tex.stackexchange for this information.)

Sans-serif and monospace (typewriter) are not just different fonts, they are often different typeface families entirely.[4]

To avoid the problem of forgetting to put curly braces around the commands *and* text you want formatted, there is an alternative set of ***scoped*** commands for the most common type shape and series commands. These use curly braces in the 'argument' manner, so their effect applies only to the text in curly braces. These are the normal commands for changing the style of a word or phrase.

Table 6.3 – Typeface styles, families, shapes, and series (scoped)

Type style	Command	Example (using Computer Modern)
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.

[4] Although if you're a typographer wanting to experiment with typewriter typefaces with and without serifs, you can use METAFONT or *FontForge*to do exactly this kind of thing. But that's way outside the scope of this document.

These are commutative too, so you can nest them inside one another:

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

What we know as ***underlining*** isn't a font: it was used in the `days of typewriters` where italics were not available, and it is extremely rare in typography except for specialist purposes. If you think you need it, use the ulem package with the *normalem* option, which provides a `\uline` command.

6.2.5 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 experience in 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 in the METAFONT program. It often comes as a surprise to new users that many typefaces are not designed as a single font and just scaled up or down, but specially drawn at different sizes to make them more legible.

As an example, here's 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.

Modern type formats have ***hinting*** parameters that allow scaling to implement the effects of design-sizes, but in general, you probably don't want to go scaling fonts too far beyond their design size because the spacing 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 default size for book work.

- ☐ Using the larger document class options (11pt and 12pt) will use 11pt and 12pt designs (explicit or hinted), with the other sizes (such as for headings) rescaled to match.

Table 6.4 – LaTeX font step sizes

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 lazy dog	7	7
\footnotesize	The quick brown fox jumps over the lazy dog	8	8
\small	The quick brown fox jumps over the lazy dog	9	9
\normalsize	The quick brown fox jumps over the lazy dog	10	10
\large	The quick brown fox jumps over the lazy dog	12	12
\Large	The quick brown fox jumps over the lazy dog	14	14.40
\LARGE	The quick brown fox jumps over the lazy dog	18	17.28
\huge	The quick brown fox jumps over the lazy dog	20	20.74
\Huge	The quick brown fox jumps over the lazy dog	24	24.88

- Note that these are unscoped commands, so they should be used inside a group, either an environment or a set of curly braces terminated with a \par inside the closing brace. There are no scoped equivalents.
- Mathematics users should not confuse the text-mode \scriptsize command here with mathematics-mode \scriptstyle. [Thanks to Doug McKenna and David Carlisle on the TeXhax mailing list.]

- The exact sizes used are listed in the macros in the Class Option files size10.clo, size11.clo and size12.clo.
- LaTeX's default fonts above 10pt are in fact scaled by a factor of 1.2, as shown in the fourth column of above.

While these shorthand commands relieve the beginner of having to worry about the appropriate point-size for a given task, if you need very specific sizes you can use the \fontsize command to specify exact sizes. This takes two arguments: the point size and the baseline distance. The example below gives you 22pt type on a 28pt baseline (ie with 6pt extra space or 'leading' between the lines).

```
{\fontsize{22}{28}\selectfont The example below gives you
  22pt type on a 28pt baseline (ie with 6pt extra space or
  `leading' between the lines).\par}
```

The example below gives you 22pt type on a 28pt baseline (ie with 6pt extra space or 'leading' between the lines).

The term 'leading' comes from the old metal-type practice of adding a strip of typemetal between the lines, or casting the type on a deeper body, to increase the line spacing, so it's pronounced 'ledding' after the metal.

If you are using *pdflatex* or the original *latex* processor, you will need to use the fix-cm package to override the step sizes. This needs special placement: it MUST come at the start of the document, *even before* the \documentclass command, and MUST be invoked with the \RequirePackage command normally used only by document class designers:

```
\RequirePackage{fix-cm}
```

```
\documentclass{article}
...
```

6.2.6 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	*Accounting in Business*
Product names	Corel *WordPerfect*
Variables in maths	$E = mc^2$
Subtitles or headings	*42. 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, and we've been exposed to many of these meanings since we started to read. Computers cannot (yet) do this reliably, 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 in typesetting.

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

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

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

It also makes it possible to find and act on groups of meanings — such as making an index of scientific names, or retrieving all product names — if they are identified as such. Otherwise you'd spend weeks hunting manually through every `\textit` command to find the ones you wanted. This is the bottom line of automation: it can save you time and money.

In Chapter 7 starting on page 203 we will see how to make your own simple commands to do things like this.

6.2.7 Colour

You can typeset anything in LaTeX in any colour you want using the xcolor package. Adding the command `\usepackage{xcolor}` to your Preamble (note the US spelling of color) 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}` to get red like this. There is an unscoped `\color` command as well, for use within groups:

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

There are several package options for additional colours: two popular ones are *dvipsnames*, which provides a 64-colour palette of predefined colour names exactly matching the big box of

64 *Crayola*™ colouring pencils much favoured by artists and designers; and *svgnames*, which provides the 256 colours defined in the specification for the Scalable Vector Graphics (SVG) drawing and diagramming language (which includes the 64 colours of *dvipsnames*). There are others too: see the documentation for the xcolor package.

If you want the *Crayola* colour Crimson, and you have loaded xcolor with the *svgnames* or *dvipsnames* option, you can use it as a colour name (colour names are case-sensitive):

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

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 named option you loaded the package with.

Using the `\definecolor` command, you can also 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 of 255, separated by commas*. For example, an RGB colour given as (37,125,224) in decimal integer form can be given as:

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

To get the fractional value, divide the integer 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. Alternatively, use the HTML hexadecimal colour model, the same as used in web pages and CSS stylesheets:

```
\definecolor{midblue}{HTML}{250FE0}
```

The xcolor package also provides two colour versions of \fbox (see section 4.6.2 on page 127) called \colorbox and \fcolorbox which create a box with a coloured background:

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

The material in the second argument can have its own text colour, as in the example. The \fcolorbox has an extra first argument to specify the colour of the frame or border placed around the box. The border width is controlled by the \fboxrule setting and the separation between rule and content is controlled by the \fboxsep setting as we already saw in section 4.6.2 on page 127.

However, combining colours is an art and a skill: using the command above to get the effect magenta on midblue illustrates why it is important to learn about colour models and palettes before trying to use them!

6.3 The LATEX font catalogue

The LATEX Font Catalog is a web site created and maintained by Palle Jørgensen at 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 spend many fascinating hours downloading and installing them and trying them out in your documents.

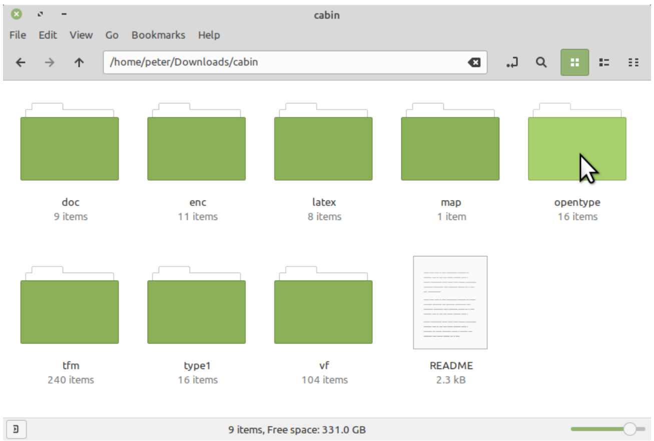
Installing a font from the Font Catalogue automatically

Over 100 of the typefaces in the LATEX Font Catalog are prebuilt for LATEX using the TEX Directory Structure (TDS)

1. Download the .tds.zip file for the typeface you want from mirrors.ctan.org/tex/install/fonts/;

2. Unzip it directly into your Personal TEX Directory.

Figure 6.1 – Layout of a font zip file downloaded from CTAN



If no TDS zip file is provided (check on the typeface's web page on CTAN at `www.ctan.org/tex-archive/fonts`), you will need to use the plain zip file and move the subdirectories into The Right Places yourself.

Installing a font from the Font Catalogue manually

1. Download the zip file from the link **download the contents of this package in one zip archive** on the typeface's CTAN web page, which is either at the bottom or the right-hand side, depending on the width of your screen.

2. Open the zip file in your directory browser (see row 2 in Table 1 on p. xxxhandlingfiles). Inside the zip file there will be several subdirectories, shown in below

   You will want three folders: `doc`, `latex`, and *either* the `truetype` *or* the `opentype`, whichever one is there.

3. Open the `doc` zip directory and check the `README` or `AUTHORS` file to identify the company or individual responsible for the typeface. This is commonly called the 'foundry', following the habits of the old hot-metal type era. It should be one word, or an acronym.

4. In your Personal TEX Directory, create places to put these, replacing foundry and typeface names with meaningful one-word values:

Zip file directory	TDS directory
doc	fonts/doc/foundry/typeface
latex	tex/latex/typeface
opentype[1]	fonts/opentype/foundry/typeface
truetype[1]	fonts/truetype/foundry/typeface

[1] Font zip files normally provide *either* a `truetype` folder *or* an `opentype` folder. If both, pick one.

5. Extract the contents of each of the three zip folders in turn into the folders you have created in your Personal TEX Directory according to the table in step above.

   The foundry name is not essential, but it helps to use a separate folder because it identifies where the fonts originated, which can help with choosing fonts.

If there was a package (`.sty`) file in the `latex` zip directory, you can use it in your documents, otherwise load the font[s] you want as shown in section 6.2.2.2 on page 182.

Exercise 29 – Font indexing in MacTEX on Apple Mac OS X systems

1. Open the Finder;
2. Click on the Macintosh `HD` entry;
3. Press ⇧ + ⌘ + . (shift splat dot together) to reveal system folders;
4. Navigate to `/usr/local/texlive/YYYY/texmf-dist` (replacing YYYY with the year of your MacTEX distribution);
5. In the menus, click File › Add to Sidebar;
6. Open FontBook;
7. In FontBook, click + to add fonts;
8. In the Finder panel that opens, click the texmf-dist folder in the sidebar and navigate to the `fonts/truetype` subfolder;
9. Click Open;
10. If the system prompts for your password, type it in;
11. Wait. If a window opens saying some fonts fail validation, DO NOT install them[a]
12. Repeat from step above for `fonts/opentype`;
13. When done, browse FontBook for fonts available to XƎLATEX and LuaLATEX.

[a] In some cases, FontBook may believe them to have missing tables, and in others, they may duplicate existing Mac system fonts, and FontBook's behaviour seems to be to use the TEX Live fonts in preference to its own.

Exercise 30 – Try setting up fonts by fontname

1. Open a new LaTeX file in your editor (pick a blank or epty one if your editor offers templates);
2. Copy and paste this text into the file:

```
\documentclass[12pt]{article}
\usepackage{fontspec}
\setmainfont{Crimson Pro}
\setsansfont{Cabin Regular}
\setmonofont{TeX Gyre Cursor}
\AtBeginDocument{\LARGE}
\begin{document}
This is the main (default) font

\sffamily This is the sans-serif font

\ttfamily This is the monospace font
\end{document}
```

3. Process the document and examine the PDF. You may notice that the sans-serif font (Cabin) and the monospace font (Cursor)looks larger than the main font (Crimson), even though they are all set to the `\LARGE` size (about 18pt, see Table 6.4 on page 192). This is because fonts are designed with different heights to the lowercase and uppercase characters.
4. Compensate for this by adding the option *Scale=MatchLowercase* to the sans and mono commands:

```
\setsansfont{Cabin Regular}[Scale=MatchLowercase]
\setmonofont{TeX Gyre Cursor}[Scale=MatchLowercase]
```

5. Reprocess and see that the sans and mono fonts have now been loaded at a size which matches the main font.

There are packages for the TeX Gyre fonts, like tgcursor (they all start with tg), have a *matchlowercase* package option which does this scaling.

Grouping

The use of curly braces to restrict the scope of a typographic change is called a TeX ***group***. This is different to putting the text into the argument of a command. Inside a group, the effect of any changes is local, so they will not interfere with the text following the closing curly-brace.

This is a different way of using curly braces to how we have used them before.

If you use a paragraph-formatting command like `\centering`, `\flushleft`, or `\flushright` inside a group, you MUST end the text with a `\par` command *inside the group* to cause the paragraph to be typeset with the desired format, otherwise the formatting simply will not take effect.

Environments like *`center`*, *`quotation`*, *`table`*, or *`figure`* are themselves groups, so the same rules apply, except that you do *not* need the `\par` at the end because most such environments are inherently paragraph-based and will do it for you.

A 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 and productive to have control of the cause than the effect.

7 Programmability

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 programming systems of their own. It's also the one that needs most foreknowledge, which is why this chapter is in this position.

TeX is basically a programming language for typesetting. As such, it allows you to define ***macros***, which are little (or large) program-like sequences of commands with a name which can be used as a command itself. This in effect makes a macro a shorthand for a sequence of operation you wish to perform more than once. LaTeX is in fact just a large collection of such macros.

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 one-liners can very simply automate otherwise tedious chores and allow the author to concentrate on the most important thing; *writing*.

7.1 Simple replacement macros

In its simplest form, a LaTeX macro can just be a straightforward text replacement of a phrase to avoid lengthy retyping with the possibility of misspelling something each time you need it, eg

```
\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 the Note on p. 19swallow). If you want to force a space to be printed after the expansion, use a backslash followed by a space, eg

```
The \EF\ is a member institution of the Commission
 of the European Union.
```

As you can see from this example, the \newcommand command takes two arguments: the name you want to give the new command, and the expansion to be performed when you use it, so there are always two sets of curly braces after a \newcommand. The names of new commands created like this MUST be made of the letters A–Z and a–z *only*, and must not be the names of existing commands.

7.2 Macros using information gathered previously

A more complex example is the macro \maketitle which is used in almost every LaTeX document to format the title block. In the default 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 section 2.3 on page 52.

If you inspect one of the default document class files, such as 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 *before* using the \maketitle command.

This use of one command (eg \title) to store the information in another (eg \@title) 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 because they designed for use in packages and classes, and are disallowed in document text. To use them in your Preamble (eg to redefine \maketitle), you have to allow the @ sign to temporarily become a 'letter' using the \makeatletter command so the @ can be recognised in a command name (and remember to turn it off again afterwards with the \makeatother command — see item below below).

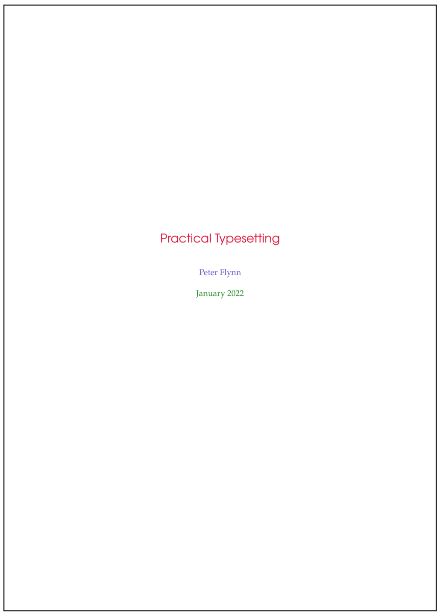
```
\makeatletter
\renewcommand{\maketitle}{%
   \begin{flushleft}%
      \sffamily
      {\Large\bfseries\color{red}\@title\par}%
      \medskip
      {\large\color{blue}\@author\par}%
      \medskip
      {\itshape\color{green}\@date\par}%
      \bigskip\hrule\vspace*{2pc}%
   \end{flushleft}%
}
\makeatother
```

Insert this immediately before the \begin{document} in the sample file in the first listing, and make sure you have used the xcolor package. Typeset the file and you should get something like below.

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 the first pair of curly braces after it;
3. Opened a second pair of curly braces to hold the new definition. The closing curly brace of this pair 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;

Figure 7.1 – Example of reprogrammed title layout



[1] If you move all this Preamble into a package (.sty) file of your own, you don't need these commands: the use of @ signs in command names is allowed in package and class files.

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` of each one makes sure that multiline title and authors and dates would get typeset with the relevant line-spacing;

7. Added some flexible space between the lines, and around the `\hrule` (horizontal rule) at the end;

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.

Exercise 31 – Rewriting the title

Add the code above to your test document (or create a new one), then add title, author, and date, and make your new title.

7.3 Macros with arguments

But macros are not limited to text expansion or the reproduction of previously-stored values. 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 mentioned earlier (in section 6.2.6 on page 195) the idea of making new commands to put specific classes of words into certain fonts, such as `\foreign` or `\product`. Here's an example for a new 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}, 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 below 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 item 'Font changes' section 5.4.1 on page 154 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 item '%' section 1.7 on page 22), 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 section 1.10.2 on page 32 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.

7.4 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 minimise 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, eg \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.

7.5 Macros and environments

As mentioned in section 4.6.3 on page 128, 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 box commands to display your

[2] Don't try this at home alone, folks! This one is safe enough, but you should strictly avoid `\def` for a couple of years. Stick to `\newcommand` for now.

text, but they require a pre-formed box as their argument, so the package provides a special environment *Sbox* which 'captures' your text for use in these boxes.

Here we put the text in a *minipage* environment because we want to change the width; this occurs *inside* the *Sbox* environment so that it gets typeset into memory and stored in a box. It can then be 'released' afterwards with the command \TheSbox as the argument of the \shadowbox command (and in this example it has also been centred).

```
\begin{Sbox}
\begin{minipage}{3in}
This text is formatted to the specifications
of the minipage environment in which it
occurs.

Having been typeset, it is held in the Sbox
until it is needed, which is after the end
of the minipage, where you can (for example)
align it and put it in a special framed box.
\end{minipage}
\end{Sbox}
\begin{center}
\shadowbox{\TheSbox}
\end{center}
```

☞

This text is formatted to the specifications of the minipage environment in which it occurs.

Having been typeset, it is held in the Sbox until it is needed, which is after the end of the minipage, where you can (for example) centre it and put it in a special framed box.

The point about this kind of construct is that it can be turned into an environment of your own, so you can reuse it wherever you need (the colouring is left as an exercise to the reader):

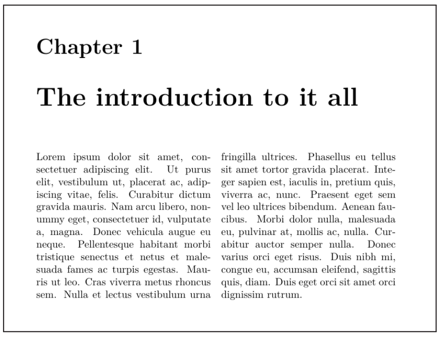
```
\usepackage{fancybox,fontawesome}
\newenvironment{warning}[1]{%
  \begin{Sbox}\begin{minipage}{8cm}%
    \subsubsection*{\fa-fire-extinguisher\ #1}}
  {\end{minipage}\end{Sbox}\begin{center}
   \shadowbox{\TheSbox}\end{center}}
...
\begin{warning}{Open and Close}
Make sure all your open-curly-braces are matched
by close-curly-braces and that every \verb+\begin+
is matched by an \verb+\end+.
\end{warning}
```

Open and Close

Make sure all your open-curly-braces are matched by close-curly-braces and that every `\begin` is matched by an `\end`.

Here's another one, this time to create a two-column chapter with the title spread across both columns (thanks to users on Discord for this one):

```
\usepackage{multicol,lipsum}
\newenvironment{spanchapter}[1]
  {\begin{multicols}{2}[\chapter{#1}]}
  {\end{multicols}}
...
\begin{spanchapter}{The introduction to it all}
\lipsum[1]
\end{spanchapter}
```


Chapter 1

The introduction to it all

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7.6 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. These examples can also be done with the appropriate package, but they are done by hand here as examples of how it works.

7.6.1 Changing numbering levels

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 (see footnote7 on page 57). 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 the (missing) chapter number, because no chapter has been started. But this numbering is controlled by macros, and you can redefine them. In footnote2 on page 94 we said that every counter automatically gets a related command beginning with `\the...`. In this case what we need to change is that command `\thesection` because the way the counters are set up makes it reproduce the value of the counter `section` (see section 4.1.6 on page 94) *plus* any higher-level value (eg `chapter`). It's redefined afresh in each document class file, using the command `\renewcommand` (in this case in `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:

```
\renewcommand{\thesection}{\arabic{section}}
```

I've used the more formal modern 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' version of the \arabic command 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

7.6.2 Changing list item bullets

As mentioned earlier, here's how to redefine a bullet for an itemized list, with a slight tweak:

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

Here we use the bbding package which has a large selection of 'dingbats' or little icons, and we change the label for top-level itemized lists (\labelitemi, find these in any document class file) to make it print a right-pointing pencil (the names for the icons are in the bbding package documentation: see section 3.1.3 on page 69 for how to get it).

Never change the installation files

What's important in making this type of modification is that you DO NOT (ever) alter the original installed document class file `report.cls`. Instead you copy the command you need to change into your own document Preamble, or a private package file, and modify that instead. It will then override the default because it will get loaded *after* the document class and *replace* the original definition.

(XML users may recognise that this is the *reverse* of the way that Local Subset modifications work with an XML Document Type Description (DTD), where the first declaration always holds, and subsequent redeclarations are ignored.)

In this case, we are also 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 separate `\lowerbox` command); and the second is the text you want to affect. Here, we are shifting the symbol down by ¼ex (see section 1.10.1 on page 30 for a list of dimensional units LaTeX can use). We also make the icon slightly smaller to go better with the font we are using.

There are label item commands for each level of lists (1–4) which have command names ending in Roman numerals (i–iv) because of the rule that command names can only use letters. Thus to change the icon for the fourth-level list, modify `\labelitemiv`. There is a vast number of symbols available, not just bbding: see the *TeX Symbol List* for a comprehensive list.

8 Conversion

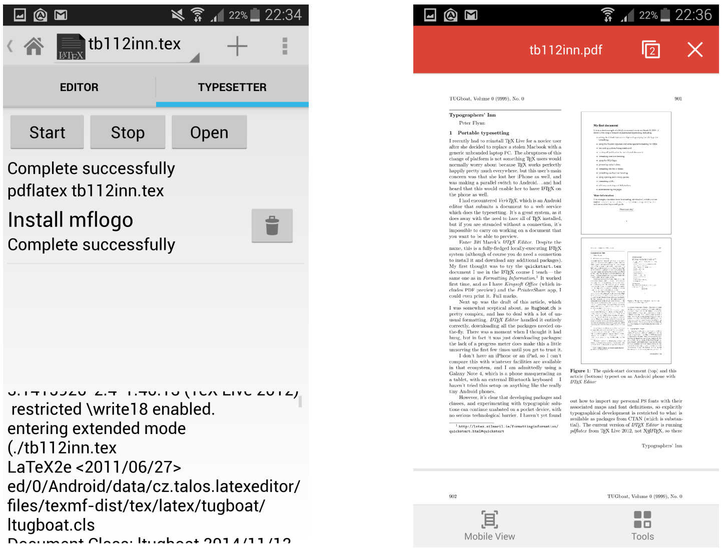
As we saw right at the start, LaTeX uses plaintext files, so they can be read and written by any standard application that can open text files. This helps preserve your information over time, as the plaintext format cannot be obsoleted or hijacked by any manufacturer or sectoral interest, and it will always be readable on any computer, from your smartphone (LaTeX is available for many handhelds, from old PDAs, see below, to Android devices, see below) through all desktops and servers right up to the biggest supercomputers.

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 Compact Disc (CD) or Digital Versatile Disc (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 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.

Figure 8.1 – LaTeX editing and processing on the Sharp Zaurus 5500 PDA

Figure 8.2 – LaTeX editing and processing on the Samsung Galaxy Note 4



Like other typesetters and formatters (Quark *XPress*, Adobe *InDesign* and *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 or compensate for it. Similarly, converting other formats into LaTeX often means editing back the stuff the other formats omit because they only store appearances, not structure.

Most converters are one-way: that is, they convert into LaTeX or out of LaTeX, and there are several excellent systems for doing the conversion from LaTeX directly to HyperText Markup Language (HTML) so you can at least publish it on the web, as we shall see in below.

Most of the utilities listed below are Open Source or free-to-use software. There are many commercial solutions as well, either the software itself or a service where they do it for you, but they are

Pandoc

There is one system that does conversion in both directions, and includes a huge range of formats: *Pandoc*. This is a large library of *Haskell* routines for handling conversions, with a command-line front end. Supported formats include *Word*, *Libre Office*, LaTeX, *DocBook*, EPUB v3, *InDesign*, *Markdown*, and dozens of others, even JavaScript Object Notation (JSON).

Before trying the systems described in section 8.1 and section 8.2 on page 230, see if *Pandoc* will handle your files.It doesn't have the same levels of speciality as some of the other converters, but it does provide those additional input formats. The exception is probably converting from XML to LaTeX for which a robust Extensible Stylesheet Language 3 (XSLT 3) script is really the only reliable solution.

pandoc.org/

mostly aimed at large-scale business conversion, and are ususally too expensive for domestic or academic single documents.

8.1 Converting into LaTeX

Turning other document formats into LaTeX is generally several orders of magnitude easier than the other way round, because almost all other document-handling systems know and understand their own features.

The methods vary from wordprocessors or plugins with a menu entry for File > Save As... > LaTeX, to a custom XSLT script for a bespoke solution.

8.1.1 Conversion from wordprocessors

Several wordprocessor systems can save their text in LaTeX format using the File > Save As... > LaTeX or Export As... menus. A very few actually create LaTeX natively, but there are also some stand-alone converters.

Microsoft Word

Microsoft Word in particular does not have any LaTeX export facility at all. Instead, you can either open the document in one of the other systems listed below, and use that to export into a LaTeX document, or use a converter. If you prefer a commercial solution which runs as a plugin within Word, see the the list item 'GrindEQ' section 8.1.1.2 on the next page or the the list item 'TeX2Word' section 8.2.1 on page 232.

8.1.1.1 Native LaTeX

LyX: LyX is probably the best-known wordprocessor-like interface to LaTeX (not quite WYSIWYG, more What You See Is What You Meant). It is already basically LaTeX internally, and its LaTeX export is very good, offering several flavours (LaTeX, XeTeX, LuaTeX, etc) as well as Word and Libre Office formats.

`www.lyx.org/`;

Scientific Word: A writing and editing front-end interface to LaTeX specifically designed for math and science papers. With LaTeX as the backend, the companion product *Scientific Workplace* provides computational and plotting facilities.

`sciword.co.uk/`.

8.1.1.2 Export via plugin

AbiWord: *AbiWord* can load a Word document and provides an extensive list of export formats, so it provides a good pathway for single-file conversion. Export formats include Word, HTML, XHTML, RTF, EPUB v3, *DocBook*, *Libre Office*, and others including LaTeX.

`www.abisource.com/`;

Libre Office: *Libre Office*[1] has a LaTeX plugin called *Writer2LaTeX*, so it can be used to open Microsoft *Word* documents (as well Office Document Text (ODT) and other formats), and export them to LaTeX.

www.libreoffice.org/;

GrindEQ: *GrindEQ* is a commercial plugin for Microsoft Word to allow the loading and saving of LaTeX documents. It is oriented primarily towards mathematics.

www.grindeq.com/.

Several maths packages like the *EuroMath* editor, and the *Mathematica* and *Maple* analysis packages, can also save material in LaTeX format.

Pandoc: See the Note on p. 220pandoc.

pandoc.org/;

docx2tex: This system converts Microsoft Word's `.docx` to LaTeX (only). It runs in *Java* (standalone or as an XProc pipeline with XML *Calabash*), so it works on all platforms. It is extremely configurable, with customisable directories, and a config file that lets you map your Word styles to LaTeX `\begin` and `\end` commands, just like the old *PCWriTeX* driver.

github.com/transpect/docx2tex/releases.

8.1.1.3 Failing that...

If you can't get the kind of conversion you want from the existing utilities, or you need to make your own (perhaps to embed in another system), these are some alternatives.

Using HTML: Use the File › Save As... › HTML (or export) menu from your wordproessor to save the file as HTML, then rationalise the HTML into the XML version of HTML (XHTML) using Dave Raggett's *HTML Tidy*, and convert the resulting XHTML

[1] The former *OpenOffice* was taken over by Apache, and is no longer regarded as a contender.

file to LaTeX with the technique shown below in section 8.1.3 on the next page.

tidy.sourceforge.net/;

Using PDF: Apache provides a Java utility called *pdfbox* which can extract the text from a PDF document into HTML format, preserving the bold and italics, which *pdftotext* does not do. This can save a lot of time in post-editing before using the HTML conversion mentioned above.

pdfbox.apache.org/;

Using RTF: 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 description says it has support for figures and tables; equations are read as figures; and it can handle the latest RTF versions from Microsoft *Word* 97/98/2000, *StarOffice* (so presumably *OpenOffice* and *Libre Office*), and other wordprocessors. It runs on Windows and Unix & GNU/Linux systems, including Apple Macintosh OS X

rtf2latex2e.sourceforge.net/ (also available as package rtf2latex2e from CTAN);

Using Word or ODF: If you can get the files into *Word* or *Libre Office* format (which are both basically Zip files containing XML), write a transformation in XSLT to convert the internal XML directly into LaTeX. This is by far the most robust way to do it but it requires that the author or editor has rigorously used Named Styles. Unfortunately, without them, the quality of most wordprocessing files is generally poor when it comes to identifying which bits do what (which is what LaTeX needs to know), so some guesswork or heuristics may be needed.

At the extreme end are very simplistic systems that are incapable of outputting any kind of structured document, because they only store what the text looks like (basically, font, size, and style), rather than *why* it's there or what role it fulfils. In those cases you may be able to save the file as a PDF, and use the *pdfbox* utility as in the list item 'Using PDF' above above.

8.1.2 Bulk conversion

Converting large numbers of related documents using most of the non-graphical (command-line) utilities is often straightforward using a shell script in (eg) *bash* or *Powershell*. At the simplest level it can just be a few lines like

```
for f in *.docx; do
  pandoc -f docx -t latex $f ${f/docx/tex};
done
```

However, if you have large numbers of obsolete Word `.doc` files (too many to open and save as `.docx`), you can try to use a specialist conversion tool like EBT's *DynaTag* (supposedly still 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 in Providence, RI). It's old and expensive and they don't advertise it, but for the Graphical User Interface (GUI)-driven bulk conversion of consistently-marked *Word* (`.doc`, *not* `.docx`) files into usable XML it beats everything else hands down. But whatever system you use, the *Word* files MUST be consistent, though, and MUST use Named Styles from a stylesheet (template), otherwise no system on earth is going to be able to guess what they mean.

There is of course an external way, suitable for large volumes only: send it off to the Pacific Rim to be scanned, retyped, or hand-edited 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/if document is already in electronic format, but it's a good example of the problem of low quality of wordprocessor markup that this solution exists at all.

8.1.3 Getting LaTeX out of XML

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 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, since the early 2000s the internal format for both *OpenOffice* (now *Libre Office*) 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.

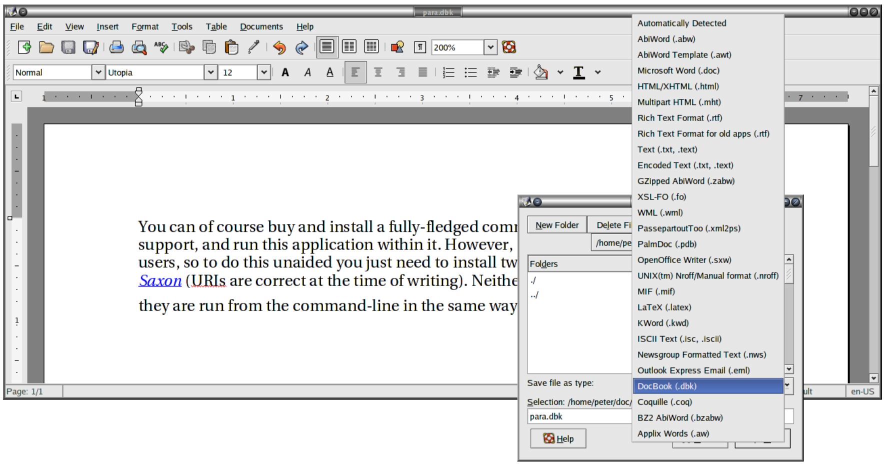
Assuming you can get your document out of its wordprocessor format into XML by some method, here is a very brief example of how to turn it into LATEX.

You can of course buy any fully-fledged commercial XML editor with XSLT support, and run transformations within it. However, this is beyond the reach of many individual users, although *oXygen* is available at a discounted price to academic sites.

To do the job unaided you need to install three pieces of software: *Java*, *Saxon* or another XSLT processor, and the *DocBook* 5.0 DTD (links are correct at the time of writing). None of these has a graphical interface: they are run from the command-line.

As an example, let's take the a sample paragraph, as typed or imported into *AbiWord* (see below). 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.

Figure 8.3 – Sample paragraph in *AbiWord* being converted to XML



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 to 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. In this example I've changed the linebreaks to keep it within the bounds of the page size of the PDF 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 the templates accordingly.

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 flavours of XML).

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="2.0">

  <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>&#x0a;</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>
```

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 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).[2]

2. The `book` template outputs the `\begin{document}` command, invokes `apply-templates` to make it carry on processing the contents of the `book` element, and then at the end, outputs the `\end{document}` command.

3. The `para` template just outputs its content, but follows it with a linebreak, using the hexadecimal character code `x0A`.

[2] 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.

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.

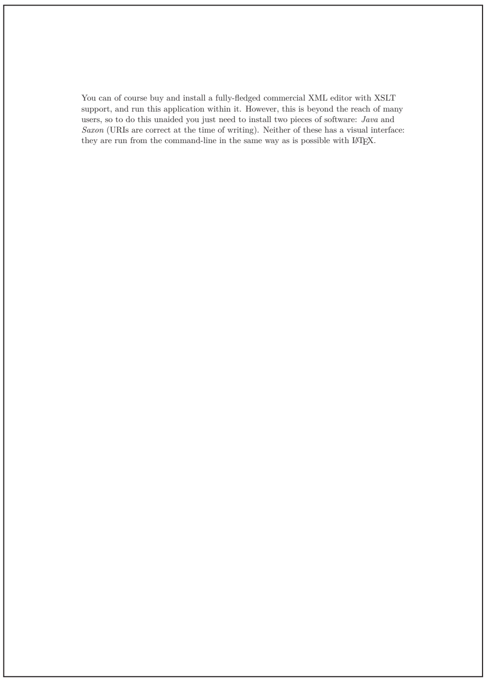
5. The emphasis template surrounds its content with \emph{ and }.

If you run this through *Saxon*, which is an XSLT processor, you can output a LaTeX file which you can typeset (see below).

```
$ java -jar saxon-he-10.3.jar -o para.ltx para.dbk para.xsl
$ xelatex para.ltx
This is XeTeX, Version 3.14159265-2.6-0.999991 (TeX Live
  2019/Debian) (preloaded format=xelatex) \write18 enabled.
entering extended mode
LaTeX2e <2020-02-02> patch level 2
L3 programming layer <2020-02-14>
(./para.tex
(/usr/share/texlive/texmf-dist/tex/latex/base/article.cls
  Document Class: article 2019/12/20 v1.4l
  Standard LaTeX document class
(/usr/share/texlive/texmf-dist/tex/latex/base/size11.clo))
(/home/peter/texmf/tex/latex/geometry/geometry.sty
(/usr/share/texlive/texmf-dist/tex/latex/graphics/keyval.sty)
(/usr/share/texlive/texmf-dist/tex/generic/iftex/ifpdf.sty
(/usr/share/texlive/texmf-dist/tex/generic/iftex/iftex.sty))
(/usr/share/texlive/texmf-dist/tex/generic/iftex/ifvtex.sty)
(/usr/share/texlive/texmf-dist/tex/generic/iftex/ifxetex.sty)
(/home/peter/texmf/tex/latex/geometry/geometry.cfg))
(/usr/share/texlive/texmf-dist/tex/latex/base/textcomp.sty)
(/usr/share/texlive/texmf-dist/tex/latex/l3backend/l3backend-xdvipdfmx.def)
(./para.aux)
(/usr/share/texlive/texmf-dist/tex/latex/base/ts1cmr.fd)
*geometry* driver: auto-detecting
*geometry* detected driver: xetex
[1] (./para.aux) )
Output written on para.pdf (1 page).
Transcript written on para.log.
$
```

This is a relatively trivial example, but it serves to show that it's not hard to output LaTeX from XML — this document is produced in exactly this way.

Figure 8.4 – The typeset paragraph and its generated source code


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 two pieces of software: *Java* and *Saxon* (URIs are correct at the time of writing). Neither of these has a visual interface: they are run from the command-line in the same way as is possible with LaTeX.

```
\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}} (links
are correct at the time of writing). None of these has a graphical
interface: they are run from the command-line in the same way as is
possible with \LaTeX.
\end{document}
```

8.2 Converting out of LaTeX

Converting LaTeX to other formats is much harder to do comprehensively than converting *into* LaTeX. As noted before, the LaTeX file format really requires a LaTeX processor in order to handle all the packages and macros, because LaTeX is entirely reprogrammable, and there is therefore no telling what complexities authors have added themselves by redefining things (what a lot of this book is about!).

However, if you have stuck to the standard commands given in the *LaTeXbook*, and not defined anything extra or redefined anything else, and have used only a small set of the most common packages, most of the converters shown here will do a good job.

Many authors and editors rely on custom-designed or homebrew converters, often written in the standard shell scripting languages (Unix shell commands, *Perl*, *Python*, *Tcl*, *Lua*, etc). Some of the public packages presented here are also written in the same languages, but 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 one-off or *ad hoc* transformations, but it is easier to obtain consistency and a more sophisticated final product using a converter written to handle a wider range of LaTeX features.

- Embedding homebrew non-standard control sequences in LaTeX source code (a common habit of authors) may be tempting, to make it easier for the author to edit and maintain, but will always make it harder to convert to another system.

- Most of the converters mentioned here provide a fast and reasonably reliable way to get LaTeX documents into *Word*, HTML, and other forms of XML in an acceptable — if not optimal — format, even if your primary target is eventually to convert to TEI, Journal Article Tag Suite (JATS), *DocBook*, or some other vocabulary, because once it's in well-formed XML of one kind, translation to another vocabulary is much easier.

- Above all, it is essential to understand that no conversion will produce an error-free result except for the most trivial of documents. All other output *will* require post-editing to correct things the converter was unable to handle, add things it missed, change formatting it was unable to apply, and delete things that should not have appeared.

There is a useful discussion of some of the alternatives mentioned below in § 3 of the lwarp package documentation (Dunne 2020, p. 71–73). If you actually want to *author* in a hybrid format that enables XML output (rather than converting existing native LaTeX documents), see below.

8.2.1 Conversion to *Word*

This is the most frequently-requested conversion, and also one of the hardest to do, because there are things LaTeX can do that simply cannot be represented in *Word* in any meaningful manner. However, *Word* uses XML internally, and it is also possible to convert your LaTeX to XHTML and import it (see below).

There are several programs on CTAN to do LaTeX-to-*Word* and similar conversions, but they do not all handle everything LaTeX can produce, and some only handle a subset of the built-in commands of default LaTeX. in particular, however, have a good reputation:

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). This Open Source program preserves layout and formatting for most LaTeX documents using standard built-in commands and obsolete codepages (not Unicode), but it has little support for redefined commands and common packages, including fontspec; and because it doesn't recognise the array its support for advanced column specifications in tables is limited, although it does a good job on simple tables. If you need a conversion for wordprocessors that can't read `.docx` files, this is a good place to start.

`latex2rtf.sourceforge.net/`;

TeX2Word by Kirill A Chikrii for Microsoft Windows is a commercial converter plug-in for *Word* to let it import 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.

www.chikrii.com/products/tex2word/;

Pandoc: See the Note on p. 220pandoc.

pandoc.org/;

TeX4HT: See the list item 'TeX4HT' section 8.2.2 on page 235.

tug.org/tex4ht/.

One easy route into wordprocessing, however, is the reverse of the procedures suggested in the preceding section: convert LaTeX to HTML, which many wordprocessors can read easily, using any of the packages in section 8.2.2. Once it's in HTML, run it through *Tidy* (see the list item 'Using HTML' section 8.1.1.3 on page 222) to make it well-formed XHTML, add some embedded Cascading Style Sheets (CSS) styling to the header manually, and rename the file to end in the obsolete .doc filetype, which can fool *Word* into opening it natively as if it were a *Word* file.

8.2.2 Conversion to HTML

This probably runs *Word* a close second in frequency of demand. Conversion to HTML — or more probably XHTML — lets you put your LaTeX document on the web in a format everyone can read, but as with *Word*, not everything you can do in LaTeX can be represented in HTML, although with CSS3 you can get close.

lwarp: This is a LaTeX *package* for producing HTML v.5 (HTML5) output, using external utility programs for the final conversion of text and images. Strictly speaking this is an authoring package, not a conversion: you have to write your document using the commands defined in the package, rather than normal LaTeX. This makes it hard to use for handling existing documents, as they will need extensive editing before they can be processed. However, the package

Circular conversion

To the best of my knowledge, there is no off-the-shelf system that can convert circularly from LaTeX to *Word* or XML/XHTML *and back* and back again, and back again, without serious loss of formatting.[a] At each conversion, some document features will unavoidably be regularised to conventions of the target format which can no longer be represented in the source format. It may be possible for a very trivial document, but not for any real-life application.

This means that corporate, technical, or academic applications which depend on features of the *Word* interface such as Change Recording or the Style Margin cannot use LaTeX as a distributed editing format. However, after all edits have been made, the *Word* document can of course be converted to LaTeX for final typesetting.

[a] There *was* one once, in the mid-1990s, actually made by Microsoft: *SGML Author for Word*. It wasn't an editor as its name suggested, but a converter that used Named Styles to convert losslessly from SGML to *Word* and back, repeatedly, so that non-tech management could edit tech documents. Just as XML was taking off, they dropped it on the floor. Go figure. See `cora.ucc.ie/bitstream/handle/10468/1690/Human-Interfaces-to-Structured-Documents.pdf#page=393` and `xml.silmaril.ie/downloads/sgml-author-review.pdf` for more details

is under active development and supports a wide range of formatting packages.

The lwarp package is included in all TeX Live-based distributions.

LaTeX2HTML: *LaTeX2HTML*'s main task is to eproduce the document structure as a set of interconnected HTML files, so it is popular for creating multi-page web sites from a single large LaTeX document. It outputs a directory named after the input filename, and all the output files are put in that directory, so the result is self-contained and can be uploaded to a web server as it stands. It supports mathematics via images, and can deal with the built-in commands and a small range of packages.

github.com/latex2html/latex2html/;

TeX2page: This converts Plain TeX LaTeX. or Texinfo documents to HTML. Complex requirements can be configured in the TeX2page extension language (Common Lisp or Scheme). The authors have tried to make running TeX2page as similar as running LaTeX as possible.

github.com/ds26gte/tex2page;

TeX4HT: (TeX-for-HyperText) is an Open Source program which converts TeX and LaTeX documents to various kinds of XML and to *Libre Office* format (among others), which *Word* can open.

It operates differently from most other converters: It uses the TeX/LaTeX program itself to process the file, and handles conversion in a set of postprocessors for the common LaTeX packages. It can also output to XML, including TEI, *DocBook*, and the *Libre Office* and *Word* XML formats, and it can create Texinfo-format manuals.

By default, documents retain the single-file structure of the original, but there is a set of configuration directives to make use of the features of hypertext and navigation, and to split files for ease of use on the web.

tug.org/tex4ht/;

This is an Open Source translator from LaTeX to HTML by Luc Maranget at the *Institut national de recherche en sciences et technologies du numérique* [originally Institut de recherche en informatique et en automatique] (INRIA) in Paris. runs on Unix & GNU/Linux systems, and supports most of LaTeX 2$_\varepsilon$, including macro definitions. and outputs HTML5. It can be customised via style files using LaTeX code.

pauillac.inria.fr/~maranget/hevea/;

This is an Open Source translator running on most platforms, predominantly for converting mathematical LaTeX documents into HTML. works with both Plain TeX and LaTeX. Instead of using images of equations, it claims to translate them to actual HTML.

The author's link at hutchinson.belmont.ma.us/tth/ seems to be dead, but the package is available as tth from CTAN.

GELLMU: Generalized Extensible LaTeX-Like MarkUp (GELLMU) is a LaTeX-like markup for authoring structured document types which can be converted to HTML, *DocBook*, TEI, or GELLMU's own LaTeX-like document type 'article'. The source language markup most closely resembles actual LaTeX source markup: much of the markup vocabulary is the same as that of actual LaTeX.

Documents are processed via the *SGMLS* Perl module and *elisp* routines in *Emacs*, and can output LaTeX, classic HTML, or XHTML+MathML.

Available as package gellmu from CTAN;

plasTeX: plasTeX is a LaTeX document processing framework. It comes bundled with an XHTML renderer (including multiple themes), as well as a way to simply dump the LaTeX document to a generic form of XML. Other renderers can be added, including Unix & GNU/Linux *man* pages, *Docbook*, and EPUB v3.

It works by processing LaTeX documents into XML Document Object Model (DOM)-like objects which can be used to generate various types of output. Many options can be set, including controlling splitting into multiple files and adding CSS files.

plastex.github.io/plastex/.

8.2.3 Conversion to XML

LaTeXML: LaTeXML provides a conversion to an intermediate XML vocabulary which can be used to create industry publishing

XML formats such as DocBook, TEI, and JATS, and even XHTML for EPUB v3.

LaTeXML is at math.nist.gov/~BMiller/LaTeXML/.

Tralics: *Tralics* comes from the Apics and Marelle teams at INRIA. It creates an XML document of its own design, representing everything it finds in the LaTeX file, using an error element type for anything it cannot handle. In a way it is similar to LaTeXML but using a different vocabulary, and it too has an extensive configuration mechanism to tune it for specific types or classes of document.

www-sop.inria.fr/marelle/tralics/;

TeX4HT: See the list item 'TeX4HT' section 8.2.2 on page 235.

tug.org/tex4ht/;

Pandoc: See the Note on p. 220pandoc.

pandoc.org/.

8.2.4 Conversion to plain text

When all else fails, you can always convert your document to plain, unmarked text.

Text extraction: If you have the full version of Adobe *Acrobat Reader* (or one of several other commercial PDF products), you can open a PDF file created by LaTeX, select and copy all the text, and paste it into your wordprocessor, and it will retain some common formatting of headings, paragraphs, and lists. This still requires the text to be edited into shape, but enough of the formatting should be preserved to make it worthwhile for short documents.

Otherwise, use the Open Source *pdftotext* utility to extract everything from the PDF file as plain (paragraph-formatted) text, and open that in a plain-text editor.

If you need HTML, there is a Java utility from Apache, the web server project, called *pdfbox* (see the list item 'Using PDF' section 8.1.1.3 on page 223), which can extract the text from a PDF document in HTML format, preserving the bold and italics, which can save a lot of time.

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.

8.2.5 Authoring with LaTeX and XML

A number of systems have been mentioned which allow you to write your documents in a slightly different way to standard LaTeX but still using the same syntax (the list item 'GELLMU' section 8.2.2 on page 236 and the list item 'lwarp' section 8.2.2 on page 233, for example). There is also a document class called **internet** but reports indicate that it is no longer being developed.

8.3 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 section 8.2 on page 230.

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 1995–7, 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 one route to PDF via Formatting Objects with Extensible Stylesheet Language: Formatting Objects (XSLFO) but at the expense of reinventing many of the wheels which LaTeX already possesses, so the sibling XSLT (Transformations) language can be used instead to translate to LaTeX source code, as shown in the example in section 8.1.3 on page 224. This is usually much faster than writing your own formatting from scratch, and it means that you can take full advantage of the packages and typographic 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 `www.ctan.org/tex-archive/info/beginlatex/src/` includes XSLT which does exactly this.

A Installation

TEX Live installs on all modern desktop, laptop, and server platforms. You can get a copy of the TUG DVD from your local TEX user group or another TUG member; or you can download the platform distribution you need from the TUG web site at `tug.org`; or you can download the ISO image at `tug.org/texlive/acquire-iso.html` and burn your own DVD. All the individual platform distributions are available from CTAN in `www.ctan.org/tex-archive/systems/`.

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

TEX Live: for all Unix & GNU/Linux systems including Apple Macintosh OS X, and for Microsoft Windows;

TEX Live is also available specially packaged for installation on Linux systems through the Linux repositories for each system.

MiKTEXt: for Microsoft Windows by Christian Schenk, which includes the *TEXStudio* editor, *Ghostscript*, and *GSview*;

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

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 www.tug.org/lugs.html for addresses), and edited by Karl Berry (TEX Live), Richard Koch (MacTEX), Manfred Lotz (CTAN), and Christian Schenk (MiKTEX). 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. A special debt is owed to the dedication of the late Sebastian Rahtz, one of the prime movers in TUG, the TEX Collection, and the support of TEX and LATEX for many decades.

There was also a selection of commercial distributions you could buy, as described in 'Commercial implementations' on page xxxviii, and while they are no longer available for sale, many are still in widespread use: they all process LATEX identically, but there are some differences in size, speed, packaging, installation, support, and extra software provided.

A.1 Size and space

A full installation of TEX Live (eg the TUG CD image or the texlive-full Linux package) is just over 7.5GB, but there are two much smaller but more limited variants, 'scheme-basic' and 'scheme-small'. These contain the programs, fonts, and a small subset of classes and packages, but you can download anything else you need from CTAN.

Scheme-full: At 7.5GB, this includes all the programs, classes, packages, support and ancillary files, 500+ typeface families, and support for hundreds of languages including Chinese, Japanese, and Korean (CJK), Hebrew, Arabic, and many others — enough to support typesetting any document in (more or less) any script anywhere in the world.

Scheme-small: If all you want to do is typeset a math papers, you can use *scheme-small* (BasicTEX) which is about 550MB. Admittedly still not tiny, but plausible nowadays even on fairly small machines.

Scheme-basic: The most minimal scheme including LaTeX is *scheme-basic*, some 265MB. Perfectly capable of typesetting straightforward documents, but of course lacks almost all add-on packages, say, revtex, which is included in *scheme-small*.

You can select the scheme on the command line invocation of *install-tl* by typing one of the scheme (`-s`) options:

```
install-tl -s small
install-tl -s basic
```

This can be more convenient than doing it via the menus.

If you cannot install TeX at all (for example, your computer is corporate issue and locked down to prevent software being installed), there are several interactive online systems available such as Overleaf (formerly known as *ShareLaTeX*), Papeeria, and Authorea. These systems have an edit window and a PDF window, just like an installed system, and they all run LaTeX exactly as if it was installed on your desktop.

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 item 'Use the latest versions' section A.5 on page 280 for more details.

A.2 Installing the software

A.2.1 Unix and GNU/Linux

Users of modern GNU/Linux systems may not even need the TeX Collection DVD, as TeX installation packages are available online, built into the package manager for your system (see the Note on p. xxxvdvdordeb).>

If your system has a graphical package manager (eg *Synaptic* or *Software*, see above), run it and install `texlive-full` (Ubuntu etc) or `texlive-scheme-full` (Red Hat etc), `ghostscript`, `gv`, `okular`, `biber`, `kile` (or `texstudio` or `emacs`), and `jabref`. Some of them may already be installed on your system. Go

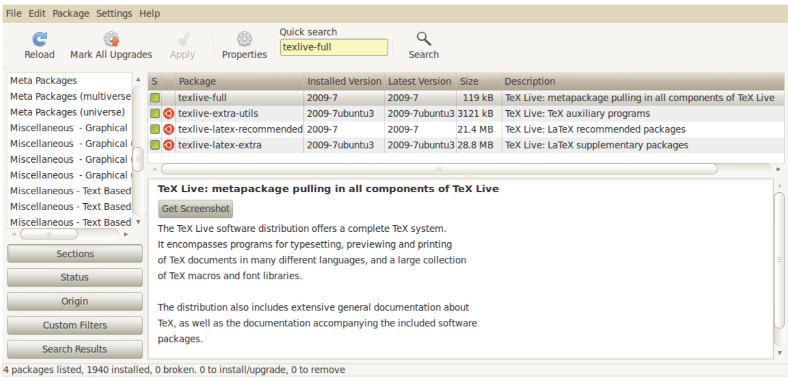
Package managers

If you pick the 'full' option when you installed TEX, you will have pretty much everything there is already installed.

Some distributions of LATEX like MiKTEX (Windows) can recognise when you try to use a new or updated class or package in your document that isn't installed, and automatically download and install it for you right there and then, and carry on compiling your document.

Others like *TEX Live* and MacTEX have a separate package manager (*tlmgr*) which can be used to add classes and packages if you find you need them.

Figure A.1 – Installing TEX Live from *Synaptic*, an *Ubuntu* package manager



and have a cup of coffee while they automatically install all the necessary components.

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

```
sudo apt install texlive-full biber ghostscript gv \
  okular biber kile texstudio emacs jabref

sudo dnf install texlive-scheme-full biber ghostscript gv \
  okular biber kile texstudio emacs jabref
```

Unfortunately, the *Okular* and *Evince* PDF/DVI viewers have cumbersome interfaces: *Okular* prematurely replaced two older and much better-designed viewers, *kdvi* and *kpdf*; but you could install *qpdfview* instead, which is faster and lighter.

Unix and GNU/Linux installers

I strongly recommend this method for most GNU/Linux users. You should only need to install from the TeX Collection DVD if you are using an older, hand-built, or commercial Unix system which has no package manager, or where the version provided by the package manager is seriously out of date.

The date of the version can be a drawback: the Linux repository versions of TeX Live can be up to a year out of date, because of the enormous volunteer effort required to put them together. For new users this should not be a concern, as most updates do not seriously affect core facilities. If there are very recently-updated packages you badly need, you can install them separately, using the instructions in section 3.2 on page 70.

If you do decide to switch to the *TeX Live* DVD or download, make sure you *completely* uninstall and purge the `texlive-full` or `texlive-scheme-full` package first, otherwise your system will get hopelessly confused. This will leave your system in a conflicted state, because to preserve dependencies it will want to uninstall all the packages above that depend on TeX. The solution is a 'shim' Debian package that pretends that the Debian distribution of *TeX Live* is still installed, while actually using the DVD-installed version.

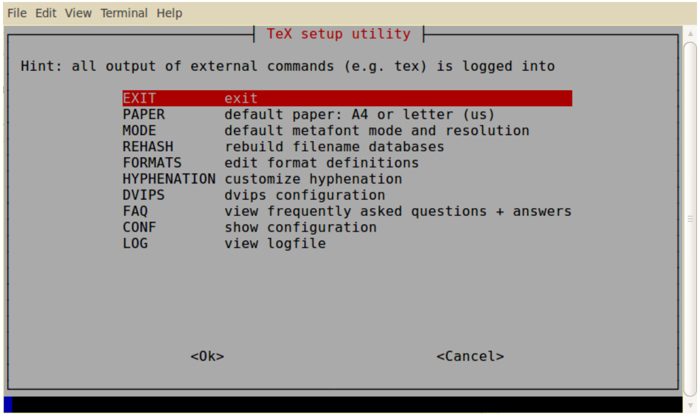
After installation, run *texconfig* (see below) in a terminal window to adjust your local settings. This is a console utility, so type `texconfig` just to adjust your own personal settings, or sudo texconfig to adjust them system-wide (for all users). In the utility, use the arrow keys to go up and down 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 ↵ 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.

You may also need the *REHASH* option later on. It is used to update TEX's fast-find database (see step **4.** on page 77) on multiuser shared systems 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 usually needed with PDF output.

Figure A.2 – Running the post-installation program *texconfig*



A.2.2 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.

If you don't have the DVD, download the `MacTeX.pkg` package from `tug.org/mactex/mactex-download.html` and click it in your Downloads to start the installation.

Depending on how your Mac is set up, you may be asked to download and install the developer tools during the TeX installation. Do so.

When it's all 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 supplied with *MacTeX* which you use for writing your documents.

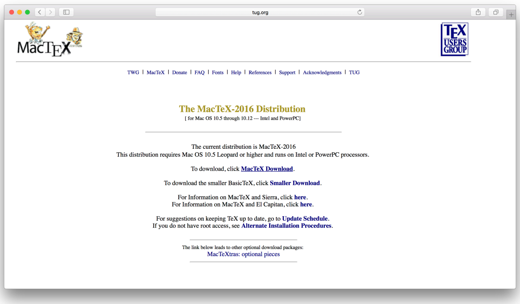
If you are going to use Adobe *Acrobat Reader* instead of *Preview*, make sure you clear the font cache and set the resolution to the system default (112dpi) otherwise you may get very weird displays.

If you are using the *El Capitan* version of the operating system, it is possible that *TeXshop* will not recognise the folder where MacTeX installed the system. If you get an error claiming your preferred LaTeX engine does not exist in `/usr/texbin`, then use the *TeXshop* Preferences ❯ Engine menu to change the first **Path Setting** to `/Library/TeX/texbin`.

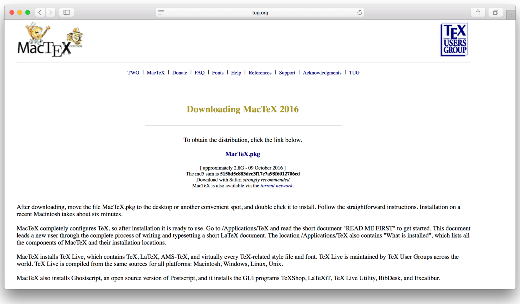
If you are unused to installing Mac software, here's a click-by-click guide:

A.2.2.1 Download the MacTeX installer

1. In the TeX Users Group web page for MacTeX at `tug.org/mactex`, click on MacTeX Download



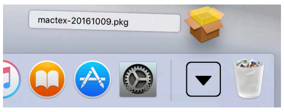
2. Click on the MacTeX.pkg link to download the installer



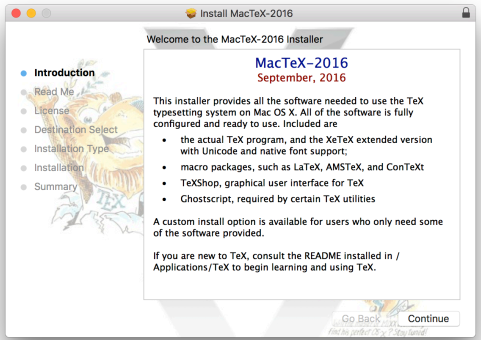
3. The installer will be downloaded into your Downloads area

A.2.2.2 Running the installer

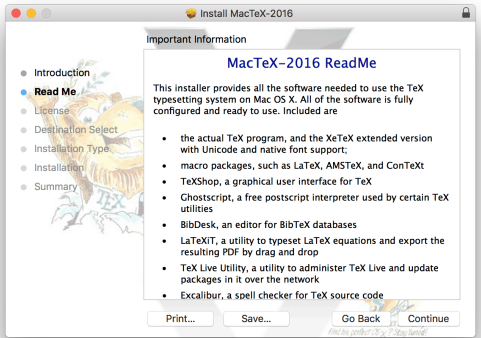
1. Click on the Downloads icon to see the package and click on it to start installation



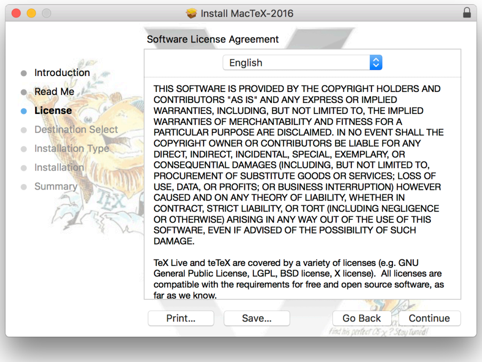
2. The Introduction screen explains what to do



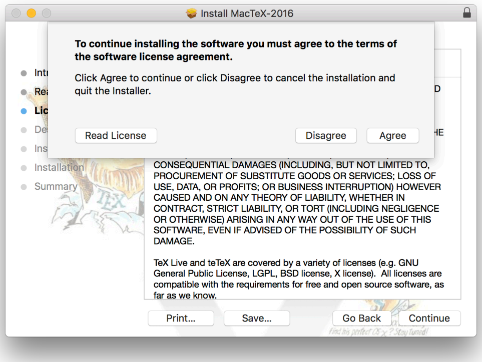
3. The ReadMe screen lists what will be installed



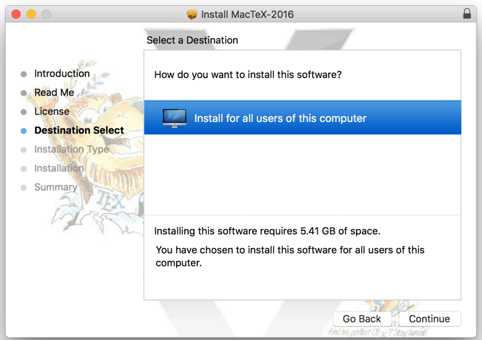
4. The License screen explains your rights: read it



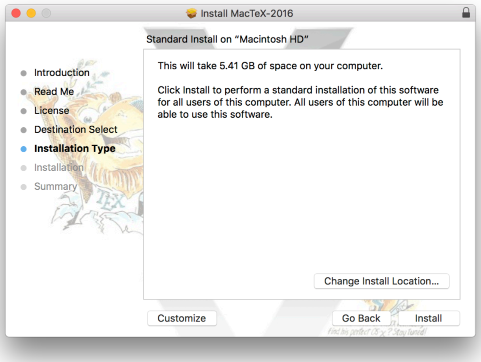
5. Click Agree to continue



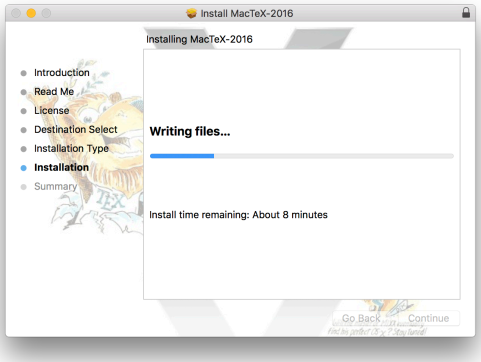
6. The Destination screen lets you choose where to install. Normally you want to install for all users, even if you are the only user on the machine



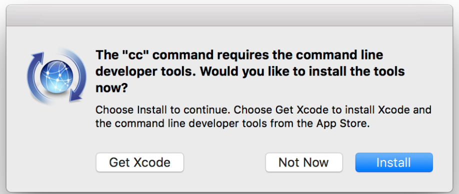
7. Pick the Standard Installation



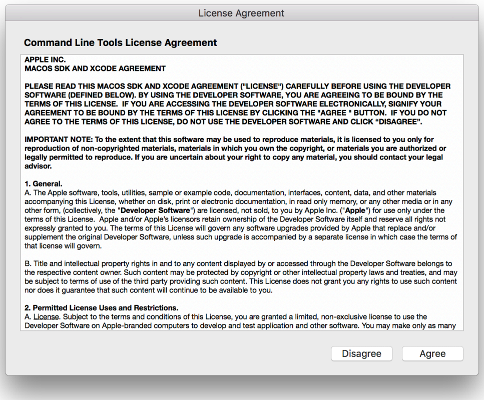
8. Wait while the files are installed



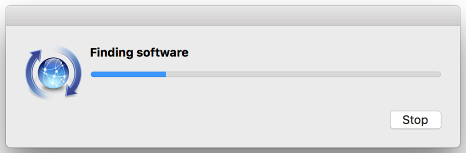
9. Install the developer tools (this step will not occur if you have them already installed)



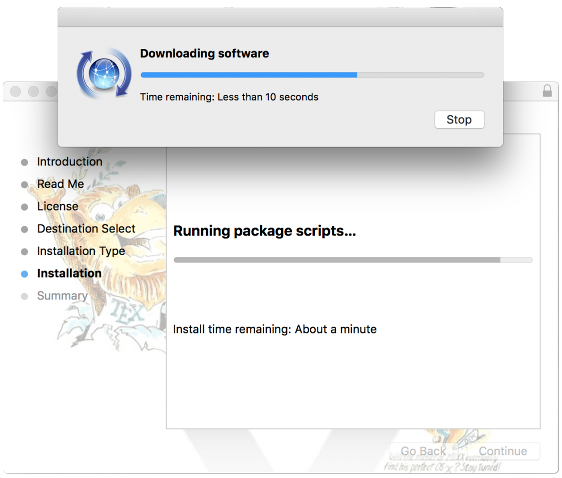
10. Agree to the developer tools license



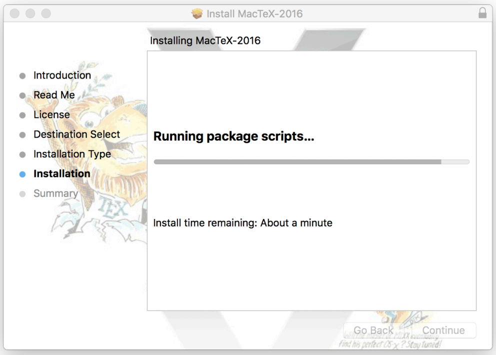
11. The installer will check the location of the developer tools



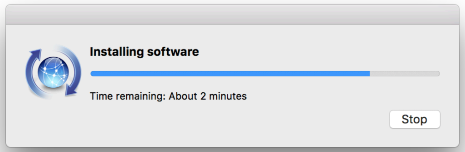
12. The developer software will be installed



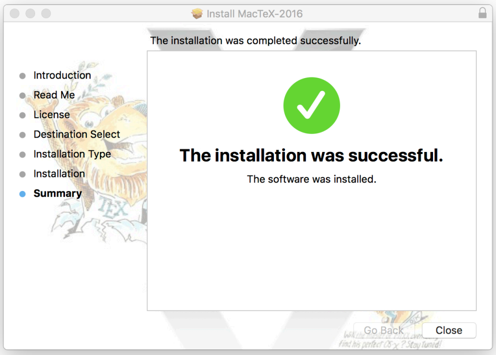
13. The MacTEX installation scripts will be run



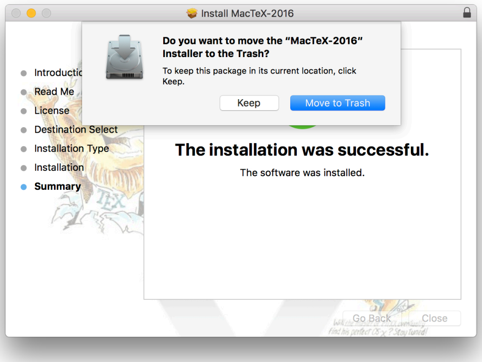
14. The developer software installation will continue



15. The MacTEX installer will complete

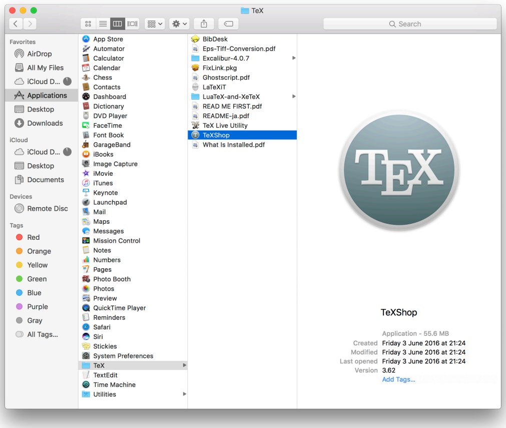


16. You can now dispose of the installer package

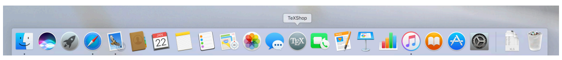


A.2.2.3 Move the TEXShop editor into the Dock

1. Open the Finder and locate the TeX folder in Applications. In it is the *TEXShop* application



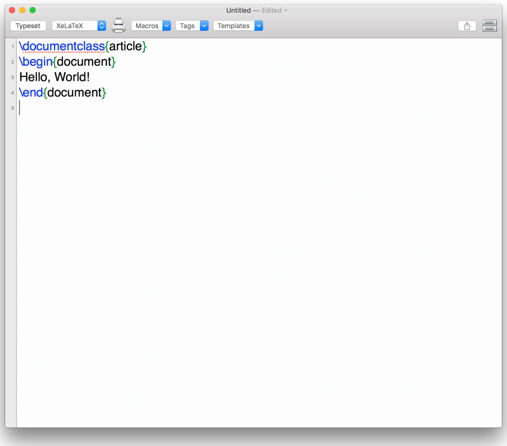
2. Drag and drop the *TEXShop* application into your Dock



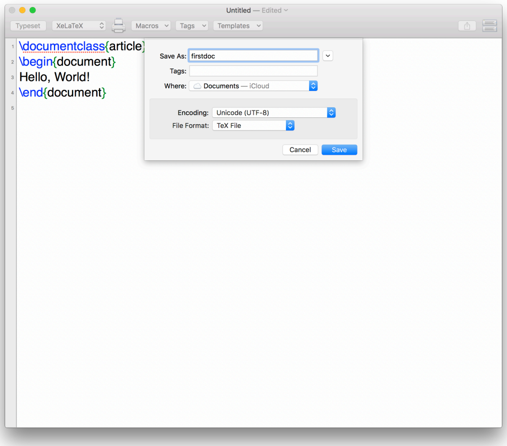
A.2.2.4 Testing MacTEX

```
\documentclass{article}
\begin{document}
Hello, World!
\end{document}
```

1. Click *TEXShop* in the Dock to open it. Type the 4–line document as shown above



2. Set the LATEX processor to XƎLATEX in the drop-down menu and click Typeset. You will be asked to name and save the document: type a name, check the Encoding is set to UTF-8, and click Save



3. The processing log will record what LATEX does (this is also where any error messages get shown)

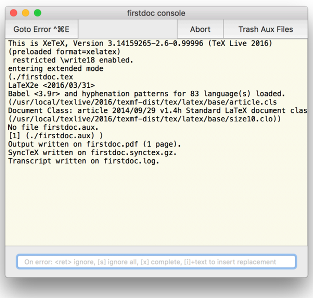
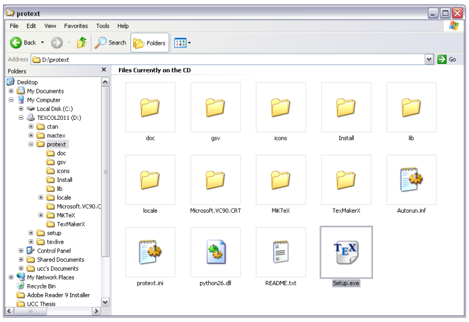
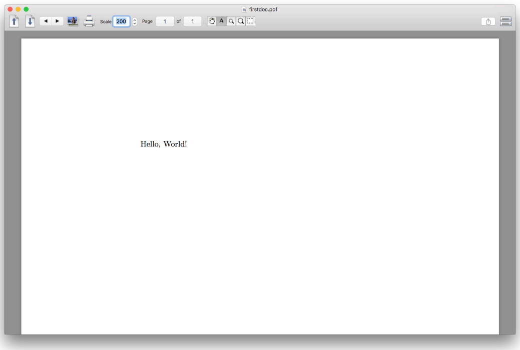


Figure A.3 – The ProTEXt setup program on the TEX Collection DVD



4. The typeset document is shown in a Preview window



A.2.3 Microsoft Windows

You can install MiKTEX from the TEX Collection DVD or by download from the MiKTEX web site.

A.2.3.1 Installing from the DVD with Autorun turned on

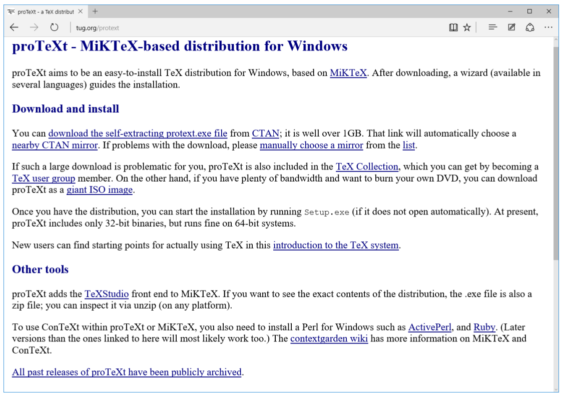
If your system has auto-run enabled, inserting the TEX Collection DVD should start the setup program automatically. Click on the Open proTeXt button to start. Continue from below below.

If your system has auto-run turned off, insert the DVD, go to the `protext` folder, and double-click on the `Setup.exe` program as shown in below. Continue from below below.

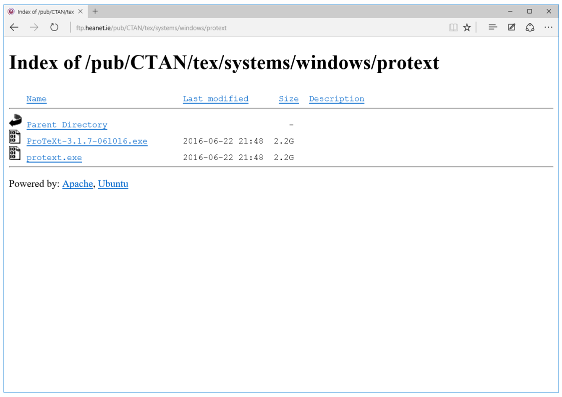
A.2.3.2 Downloading the ProTEXt installation program

Skip this if you have the TEX Collection DVD and go straight to section A.2.3.5 on page 260

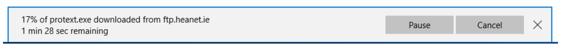
1. In the TEX Users Group web page for proTEXt at tug.org/protext/, click on download the self-extracting protext.exe file



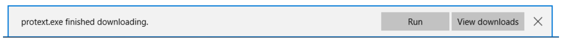
2. Click on the protext.exe link to download the installer



3. The installer will be downloaded into your Downloads area

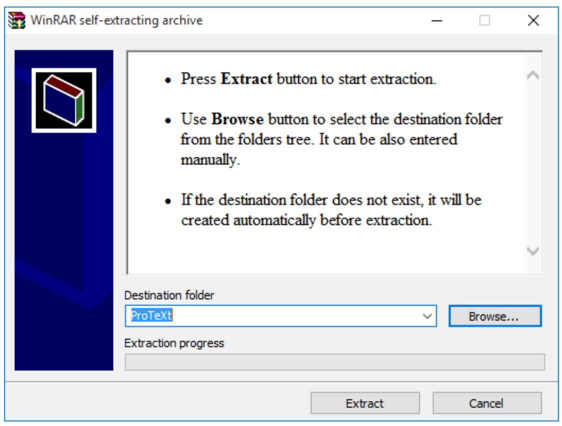


4. When it has finished downloading, wait while it is checked for malware. When the Run button appears, click it to start installation. You may be challenged by Windows Security asking if it is OK to run this program. Answer Yes

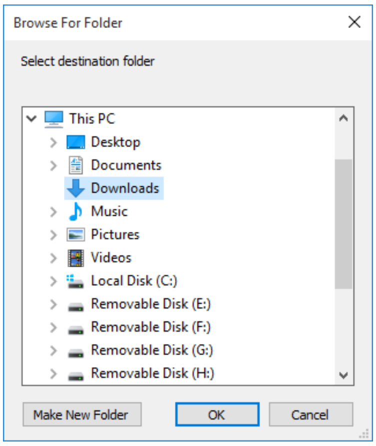


A.2.3.3 Extracting the installation files

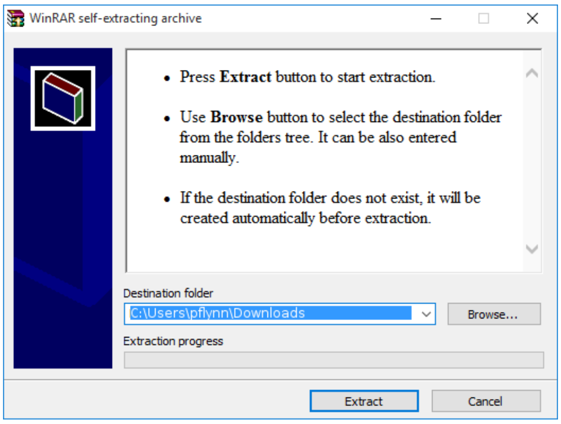
1. The first thing is to extract the installation files. Click on Browse to pick a folder



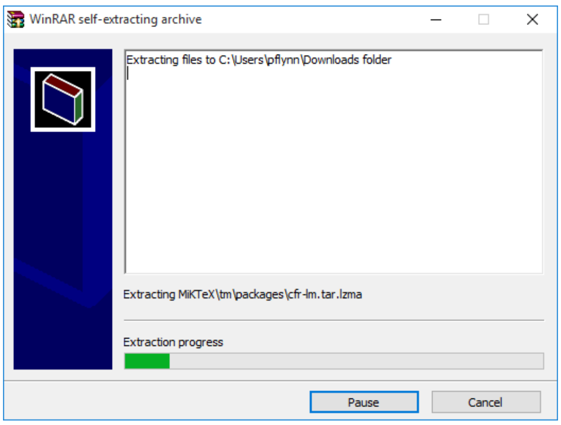
2. Choose your Downloads folder and click OK



3. Click Extract to start extracting the installation files



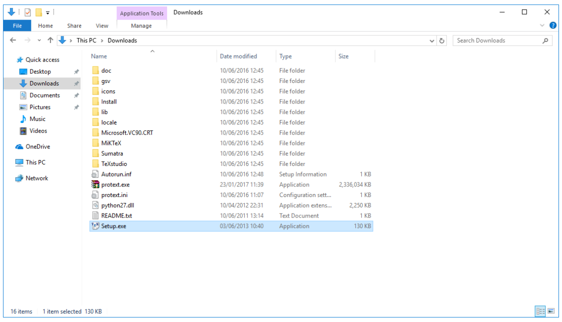
4. The installation files will be unpacked to the folder you chose



A.2.3.4 Locating your unpacked installation files

1. When unpacking has finished, run your File Explorer. In older versions of Windows, this was called *My Computer* or just *Computer*

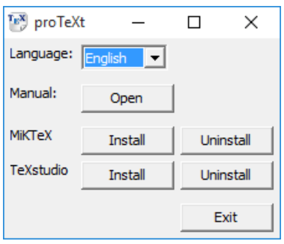
2. Go to the folder where you unpacked the installation files and double-click on `Setup.exe`



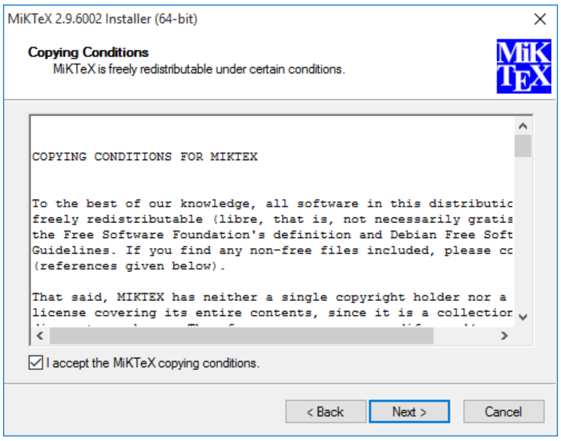
A.2.3.5 Installing proTEXt

By now you should be able to double-click the `Setup.exe` program in the `proTeXt` folder.

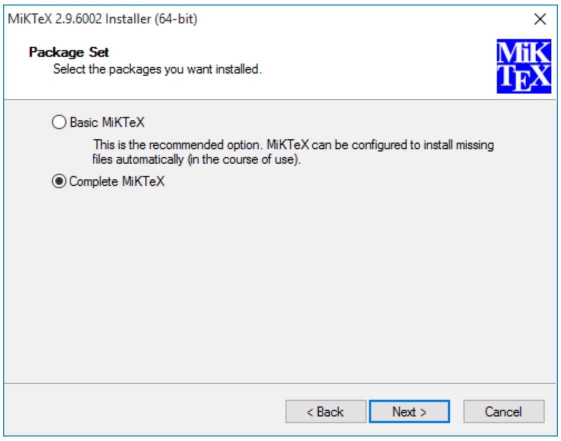
1. In the ProTEXt setup window, select your language and click on the *MiKTEX* Install button



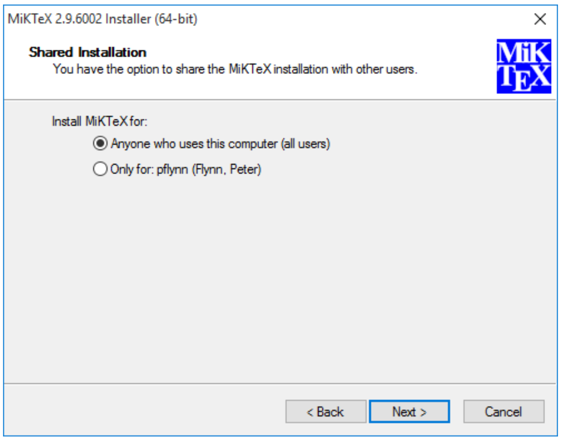
2. Click in the box to accept the MiKTEX licence and click Next to continue



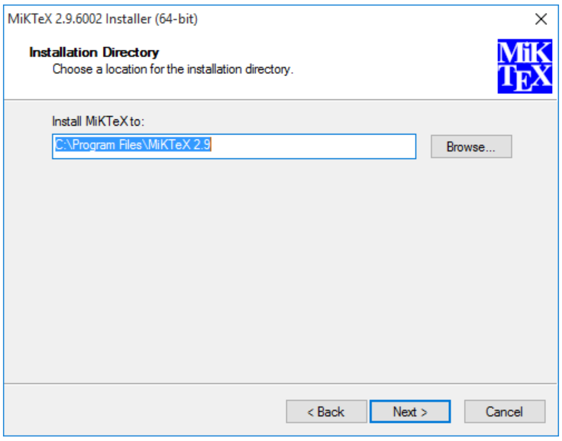
3. Make sure the **Complete MiKTEX** option is selected and click Next to continue



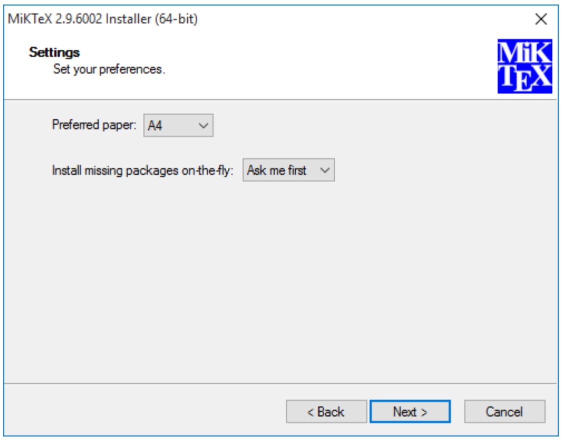
4. Choose a private installation or one that everyone who uses your computer can use, and click Next to continue



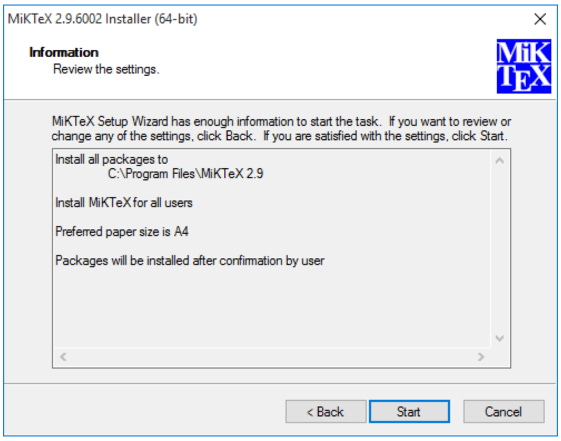
5. Accept the installation folder that MikTEX suggests (unless you are an expert or have a special disk setup) and click Next to continue



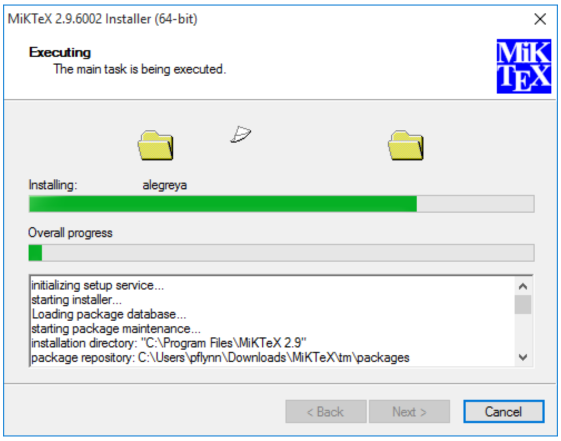
6. In the Options screen, select your paper size (A4 or US Letter), and whether or not you want extra packages to be downloaded and installed automatically (Yes or No) — on a laptop where a network connection is not always present, choose 'Ask first' instead, then click Next to continue



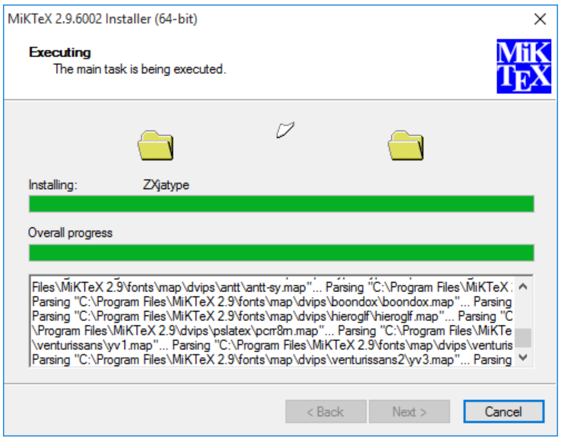
7. Finally, accept the settings as shown (or change them by clicking Back), and then click Start to start the installation process



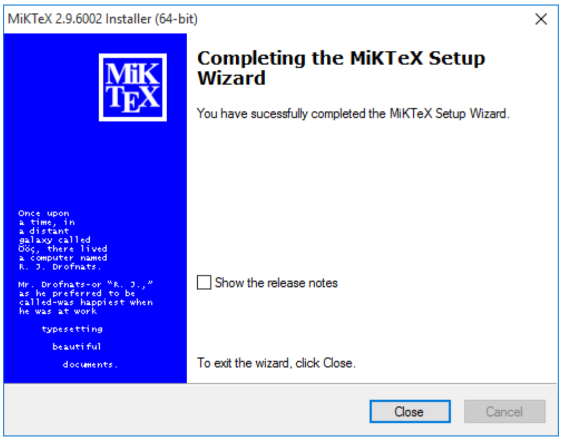
8. During installation, MiKTEX will list the files it is installing and show a progress bar



9. When it is all done, wait while it indexes your fonts, then click Finish

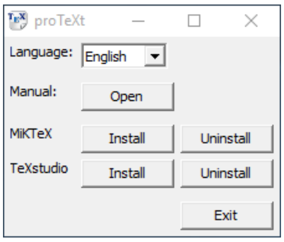


10. Click Close to finish the installation of MiKTeX

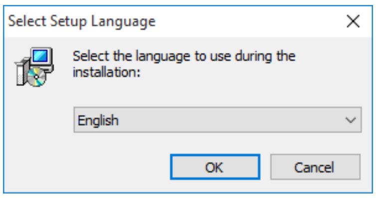


A.2.3.6 Installing TeXStudio

1. Go back to the *ProTeXt* setup window and click on the *TeXStudio* Install button

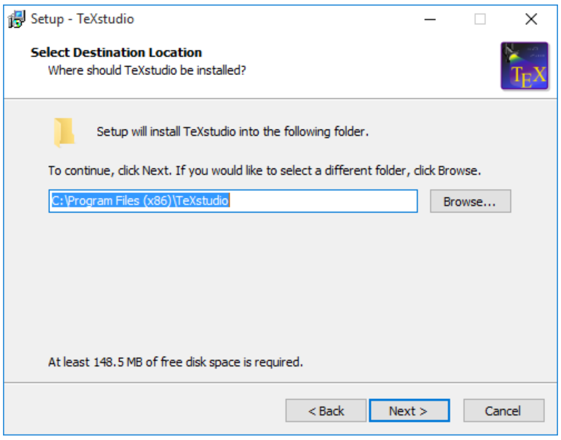


2. The *TeXStudio* installation program will start and ask you to select the language to use during installation and click OK to continue

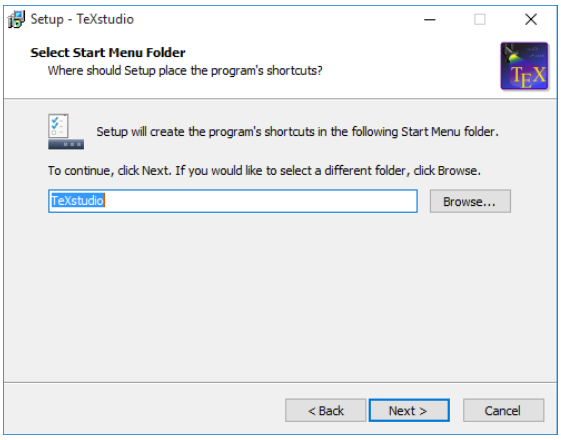


3. Click Next in the following screen to continue

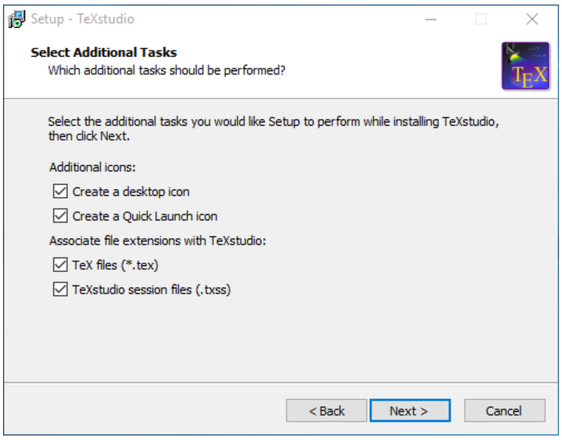
4. Accept the installation folder that *TeXStudio* suggests (unless you are an expert or have a special disk setup) and click Next to continue



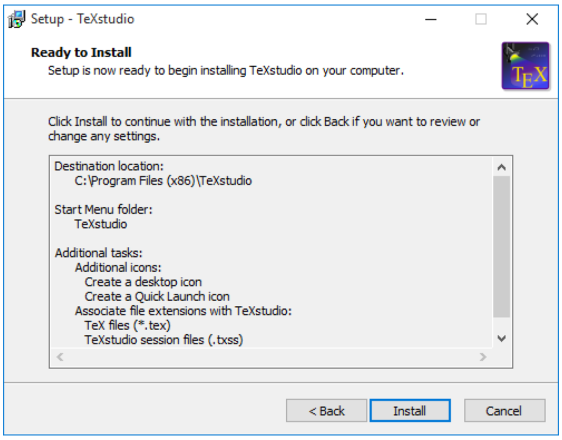
5. Also accept the suggested location for the *TeXStudio* shortcut and click Next to continue



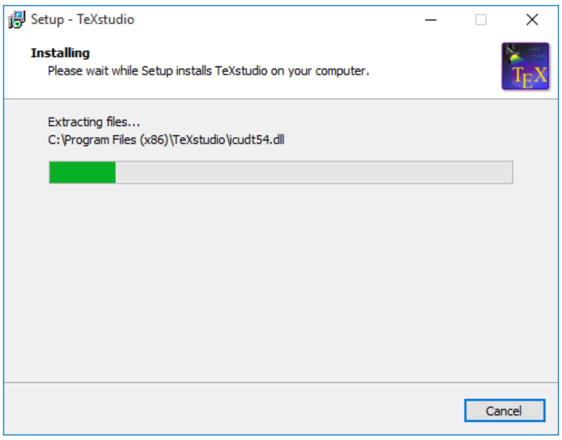
6. Add icons to your desktop and Quick Launch bar by selecting the boxes, then click Next to continue



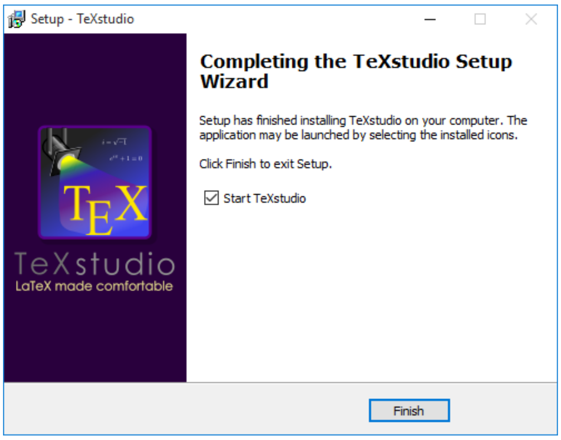
7. Finally, accept the settings as shown (or change them by clicking Back), and then click Install to start the installation process



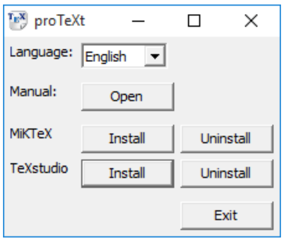
8. Wait a few moments while *TeXStudio* installs



9. When it is all done, click Finish and *TeXStudio* will start

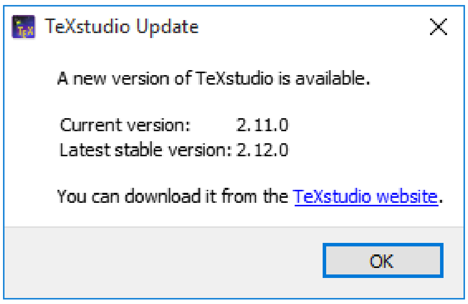


10. Close down the installer

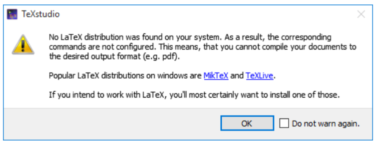


A.2.3.7 Verifying the version of TeXStudio

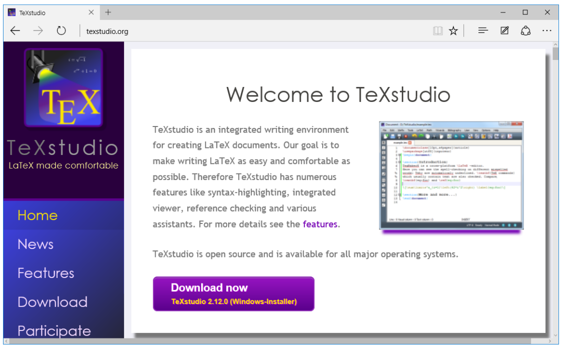
1. If there is a new version available, you will be notified. You download it from the *TeXStudio* web site at `texstudio.org/` as described below



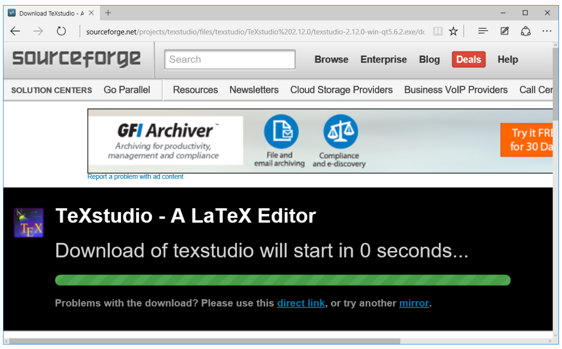
2. If *TEXStudio* claims it cannot find the installation of LATEX, it is out of date and should be replaced by an updated version from their web site as described here



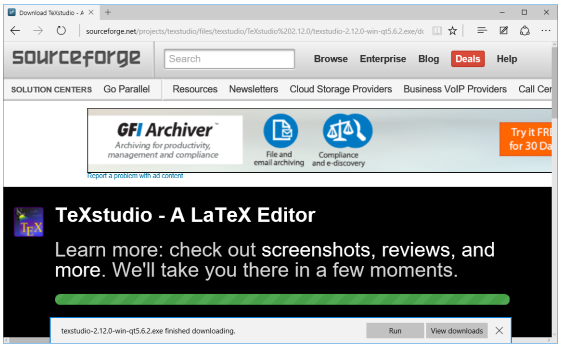
3. Download a new version from the *TEXStudio* web site at `texstudio.org/`. Click on Download now to get the latest version



4. Downloading a new version of *TEXStudio*



5. When the download has finished, click Run

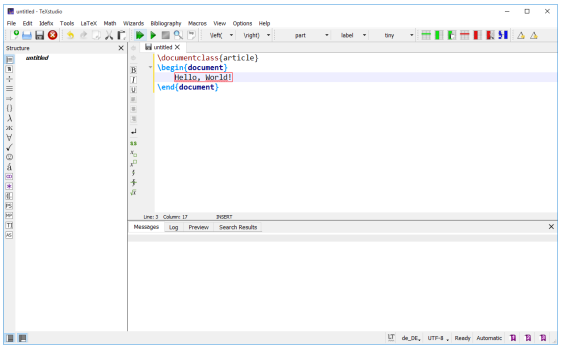


Continue the procedure from step 2. on page 264.

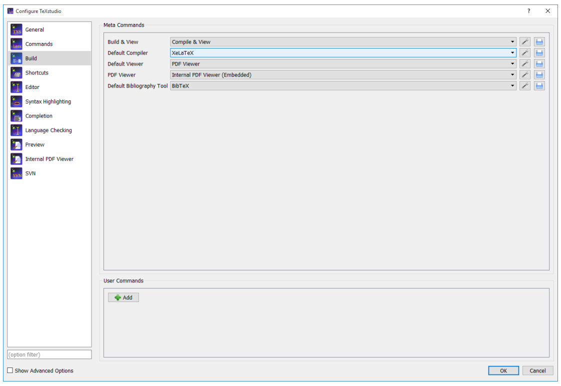
A.2.3.8 Testing TEXStudio

```
\documentclass{article}
\begin{document}
Hello, World!
\end{document}
```

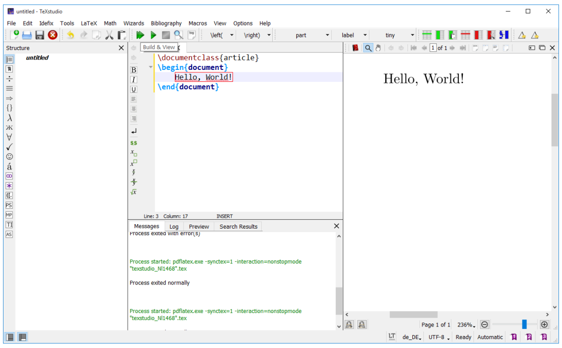
1. When *TEXStudio* runs, click File ❯ New and type the 4–line document as shown above



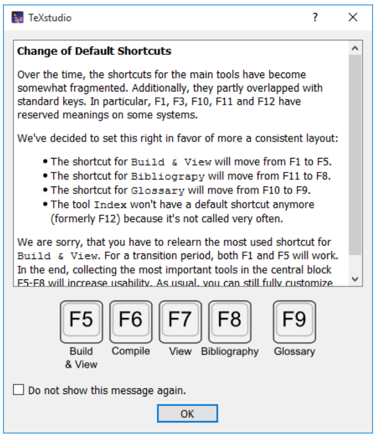
2. Click the Options ❯ Configure TeXStudio menu and set the processor (Default Compiler) to XƎLATEX



3. Typeset the document by clicking on the green double-arrow icon (Build & View)



4. Shortcuts make life simpler



One last thing to do: add a Personal TeX Directory for extra downloads such as additional fonts.

MiKTeX and TeXnicCenter

If you plan on using the *TeXnicCenter* editor instead of *TeXStudio*, you must make sure when you install *MiKTeX*, that you *make a careful note of the folder you install* MiKTeX *into* 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 this has moved on since the time or writing). *You need to know this*, because *TeXnicCenter* won't guess it for you.

A.3 Your Personal TEX Directory

There are always new packages coming out, and others being updated. There are also times when you may want to add a new or uncommon class or package by hand — perhaps a private one from a company or organisation (so LATEX and CTAN won't know about it), or even one you are writing yourself.

To do this, you need a place to put the files where they won't get mixed up with your documents or with TEX's own files. This is the 'The Right Place' to put files mentioned in step **3.** on page 76, and it's known as your Personal TEX Directory (PTD) or Personal TEX Folder (PTF).

LATEX will automatically check this place first for classes and packages, so anything you put in your Personal TEX Directory will be found *before* any file of the same name in your main TEX installation. This is why it's important for manual updates, and for special or private classes, packages, styles, and fonts.

The folder is called `texmf` (short for TEX and METAFONT), and it goes in your home or login directory. On Unix & GNU/Linux systems, including Apple Macintosh OS X, and on *TEX Live* systems on Microsoft Windows, you just need to create the folder. In MiKTEX and ProTEXt you need to tell MiKTEX where it is (see section A.3 on page 274).

Creating a Personal TEX Directory

Unix and GNU/Linux: You can either use the terminal or the file manager:

Either: open a terminal (console) window and type

```
mkdir ~/texmf
```

Or: use a file-manager:

1. Open a file-manager window (eg *Thunar*, *Nautilus*, *Dolphin*, etc) on your Home directory;
2. Right-click in an empty area of your Home directory so the menu dialog appears;
3. Click Create New Folder;

4. Type the new folder name `texmf`;
5. Press the ⏎ key;
6. Close the file-manager.

Now create the TDS subdirectory structure: At the barest minimum, you MUST create the directory and subdirectory `tex/latex` *inside* your new Personal TEX Directory (PTD).

If you want to recreate the entire subdirectory tree, there are instructions in section 3.2.3 on page 80.

Apple Mac OS X: You can either use the terminal or the file manager:

Either: open a Terminal window (find *Terminal* in Applications › Utilities or type `Terminal` into *Spotlight*) and type

```
mkdir ~/Library/texmf
```

Or: use the Finder:

1. Open the Finder on your Home folder;
2. Click on View › As Columns;
3. Click on View › Show View Options and in the Options dialog which appears, make sure that Show Library Folder is checked, then close the dialog window and select Library in the list of folders;
4. Click File › New Folder;
5. Name the new folder `texmf`;
6. Close the Finder.

Now create the TDS subdirectory structure: At the barest minimum, you MUST create the directory and subdirectory `tex/latex` *inside* your new Personal TEX Directory (PTD).

If you want to recreate the entire subdirectory tree, there are instructions in section 3.2.3 on page 80.

Microsoft Windows: You can either use the terminal or the file manager:

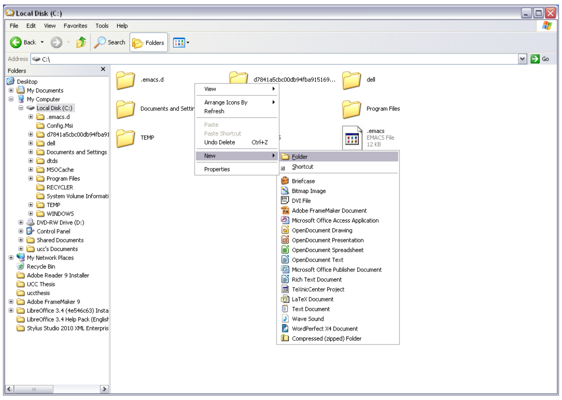
Either: open a Command window (find *Command* in All Programs) and type

```
cd %USERPROFILE%
md texmf
```

On older Windows systems (7/8/95/XP/ME) use `%HOME%` instead of `%USERPROFILE%`.

Or: use the directory manager:

1. Open *My Computer* (just called *Computer* in older Windows systems);
2. Create a new subfolder called `texmf`
   - ☐ in the `C:` drive (older Windows).
   - ☐ in `Computer\System\Users\your name\texmf` (Win 7/8).
   - ☐ in `C:\Users\your name\texmf` (Win 10 and up).



3. Type in the new folder name `texmf` (make sure it is called `texmf` in all lowercase) and press the ⏎ key. If Windows makes the `T` capital, change it to lowercase `t`.

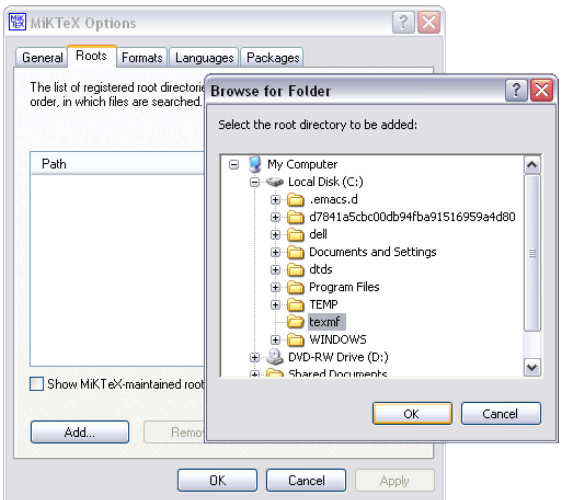
Now create the TDS subfolder structure: At the barest minimum, you MUST create the folder and subfolder `tex/latex` *inside* your new Personal TEX Directory (PTD).

If you want to recreate the entire subdirectory tree, there are instructions in section 3.2.3 on page 80 which can be adapted for Windows systems.

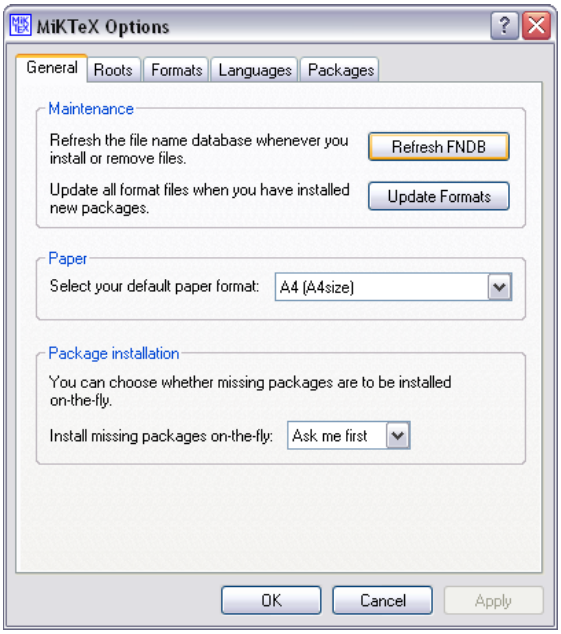
Finally...if you use MiKTEX or ProTEXt

If you use MiKTEX or ProTEXt you MUST now tell MiKTEX to add this folder to its File Name Database (FNDB). This step is compulsory: without it, nothing will work.

1. Click the Start or Windows button and run the MiKTEX Options (maintenance) program (it should be shown among your recent programs).
2. Click the **Roots** tab and the Add button, and navigate in the window to the place where you created the `texmf` folder above



3. Click on the **General** tab and tell MiKTEX to update its FNDB along with its other folders by clicking the Refresh FNDB button (below)



Warning

In MiKTeX 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.

A.4 Installing new fonts

Fonts come in a variety of formats. The earlier *PostScript* Type 3 (METAFONT) and Type 1 (PS) fonts are now being superseded by fonts in TrueType (`.ttf`) and OpenType (`.otf`) formats. How you install them and where they go depends on how and where you installed LaTeX: all I can deal with here are the standard locations within the TeX Directory Structure (TDS).

TrueType and OpenType typefaces: These are a single `.ttf` or `.otf` file per font (so there may be many for a whole typeface or font family).

METAFONT typefaces: These are normally used only with older distributions of LaTeX but there are still some fonts which are not yet available in OpenType or TrueType format.

They have a number of `.mf` source (outline) files and possibly also some `.fd` (font definition) files. There may be `.tfm` (TeX Font Metric) files but these are not needed at installation, as they get generated from the outlines automatically the first time you use the font.

PostScript typefaces: They are normally used only with *pdflatex* and are no longer covered in this book.

They come as a pair of files per font (so there may be many for a whole typeface): a `.pfb` (PostScript font ***binary***) outline, and an `.afm` (Adobe Font Metric) file.

A `.sty` file (if present) SHOULD be used in a `\usepackage` command to tell LaTeX what font features are available, and there is often a PDF showing examples and describing how to use the font or family.

The use of Type 1 and METAFONT typefaces assumes the use of the New Font Selection Scheme (NFSS) used in LaTeX 2_ε. If you are running the obsolete LaTeX 2.09, you MUST upgrade it now, because none of this will work.

A.4.1 TrueType and OpenType fonts

These types of font are already recognised by your computer, so you just install them in the normal way for your computer system (usually double-click or right-click on the font file and select Install from the menu).

On Windows and Apple Macintosh OS Xsystems, the fonts are then available immediately.

On Unix & GNU/Linux systems you must run the *FontConfig* utility *fc-cache* to give LaTeX fast-loading access to all the fonts you have installed. Run the program like this:

```
fc-cache -fv
```

It can take several minutes, especially if you have a lot of fonts. Once it has finished, you can search the font database with the `fc-list` utility and then use the `grep` command to refine the search for the font name or part, eg:

```
$ fc-list|grep -i comic|sort
/home/peter/texmf/fonts/truetype/
  windows/comicbd.ttf: Comic Sans MS:style=Bold
  windows/comici.ttf: Comic Sans MS:style=Italic
  windows/comic.ttf: Comic Sans MS:style=Regular
  windows/comicz.ttf: Comic Sans MS:style=BoldItalic
```

You can then see the exact font name to use in your documents, eg

```
\usepackage{fontspec}
\setsansfont{Comic Sans MS}
```

A.4.1.1 Font management

If you want to keep your additional fonts in several different locations, you will need to tell *FontConfig* where to look. You may install fonts *a*) in your Personal TEX Directory; *b*) in the system's shared folder; or *c*) in your TEX distribution's shared tree. You should *never* install fonts in your TEX distribution's master tree, as this will get overwritten when you upgrade, whereas the shared tree is not touched.

On Unix & GNU/Linux systems there are already `.conf` font configuration files in `/etc/fonts/conf.avail/` for the fonts that come with the installation. You can create one or more configuration files for the installation locations you have used. Some examples are shown in section 6.2.1.1 on page 170.

A.4.2 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).

Installation of METAFONT fonts

1. In your Personal TeX directory, create a new subdirectory called `fonts/source/public/name` named after the typeface you're installing[1]
2. Copy all the `.mf` files to this directory;
3. Copy any `.fd` file[s] to your `tex/latex/mfnfss` subdirectory (create it if it doesn't already exist);
4. Copy any `.sty` (style) files to a subdirectory (create it too), named after the typeface, eg `tex/latex/name`;
5. If you are using MiKTeX, run your TeX indexer program (see step **4.** on page 77).

   Unix & GNU/Linux systems, including Apple Macintosh OS X, don't need this last step.

That's it. Unlike *PostScript* fonts, METAFONT fonts generate their own font metric files (`.tfm` files) automatically on-the-fly the first time the font is used, so there is normally 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 item1 on page 73 and step **2.** on page 75).

If the font came *without* `.fd` or `.sty` files, you'll need to find someone who can make them for you.

A.5 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 been able to install it without problems. I've

[1] On Unix & GNU/Linux systems, including Apple Macintosh OS X, the easiest way to do this is in a Terminal window, in your Personal TeX Directory, using the command `mkdir -p fonts/source/public/whatever`, as this creates any intervening subdirectories for you. Under Windows, you have to create each subsubdirectory individually.

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 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 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 at `www.ctan.org/` for the latest version of ProTEXt (Windows), MacTEX (Macs), or TEX Live (all platforms) for the latest copy of the installation program. 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 the Note on p. 245tlc 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`.

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.

64-bit Windows: The *MiKTEX* distribution for Windows is a 64-bit system. Support for the earlier 32-bit distribution was ended in 2022.

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 Universal Serial Bus (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 Windows eXPerience (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 3 starting on page 297), who may be able to help.

A.6 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.** on page 77.

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 a slightly different directory to the one given in the example. Under Microsoft Windows, the names will be full paths such as `C:\Program Files\TeXLive\texmf` (for TEX Live) or `C:\Program Files\MikTeX 2.9\tex` (for MiKTEX). On an Apple Mac, it is `~/Library/texmf` for each user.

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* (1) 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, local, system, and personal trees (directories) are installed, and even where LaTeX puts its map files and format files (internal setups), and where shared BibTeX files can be stored:

```
$ kpsewhich -var-value TEXMFMAIN
$ kpsewhich -var-value TEXMFLOCAL
$ kpsewhich -var-value TEXMFSYSVAR
$ kpsewhich -var-value TEXMFHOME
$ kpsewhich -var-value BIBINPUTS
```

So if you forget where to put something you are installing, `$TEXMFHOME` is where it should go on a single-user system, and `$TEXMFLOCAL` on a multi-user system.

B Commands and errors

Most people these days do their LaTeXing in a graphical windowing editor with menus, running in a modern operating system that uses windows, icons, fonts, and a mouse which moves a pointer. This probably works fine 95% of the time, when you're dealing with one or two documents at a time, and everything you want to do is accessible through the menus, and you explicitly *don't* want to see LaTeX spilling its guts all over the place every time it reformats the document. Click here, move to there, cut, move somewhere else, paste, edit the text, write some more, click Typeset and you're done.

However, life isn't always that easy. Sometimes things go wrong, and you need to open up the lid and find out what it was. This appendix is a short description of how to run LaTeX manually, via the command-line, instead of through your editor, and it also covers error messages, and a few internal details about viewing and printing.

The editor wasn't always the primary interface to TeX, except for actually writing and editing the document. Before editors with built-in LaTeX controls became available, you had to leave your editor — or at least go to another window — and type a command to process your document, then another to view it or print it. For a small but significant number of people, running LaTeX this way is still the order of the day.

- Maybe they're working on a remote mainframe or supercomputer console with no graphics, just a 3270 or VT-100 terminal like those in Figure B.1 on the next page;
- They might be using a smartphone where the editing facilities are limited and the scope for full menus entirely absent;
- Perhaps they are simply uninterested in all the bells and whistles of the modern interface, with too many menus doing things they can actually do faster typing instructions by hand;
- Possibly they're using automation facilities that most LaTeX editors don't have, like the ability to apply the same edit to thousands of documents while you go and have a coffee or get on with something else;
- Or perhaps they are writing a system where LaTeX is the embedded typesetter, so they're actually working in a completely different scripting or programming language which does a lot of other things before calling on LaTeX behind the scenes to do some typesetting.

Before I go any further I'm going to assume at this stage that you have typed a document (for example Figure 1.2 on page 13), and that you have saved it as a plaintext file with a filetype of `.tex` and a name of your own choosing, following the rules in the panel 'Picking suitable filenames' on p. 49filenames.

B.1 LaTeX from the Terminal

Originally a terminal was a screen and a keyboard, looking very much like a standard desktop computer in the days before flat screens and windowing systems. There are still a surprising number of these around. The important point is that it was (is) a text-only interface to the computer. You got 25 lines of 80 fixed-width white-on-black or green-on-black characters, no fonts, no colours, and no mouse; maybe reverse-video as a form of highlighting (see below).

Nowadays the word usually means a 'virtual terminal': a window that behaves like a terminal — 25 lines of 80 fixed-width characters in monochrome (see Figure B.2 on the following page). It's a window into the heart of your computer. Even though you still have all your other windows visible, it knows nothing about them and can't interact with them (except for copy and paste). But instead of being a padded cell, most terminals can do things many other windows can't, like handling files in bulk, or to a schedule, or unattended, even forcing things to happen even when the graphical world outside has got itself jammed solid.

B.1.1 So where is the terminal window?

Apple Macintosh OS X: Click on Finder » Applications » Utilities » Terminal;

Microsoft Windows: Click on the Windows or Start button, All Programs » Accessories » Terminal (in older versions it's called Command Prompt);

Figure B.1 – Text-only display terminals

Images courtesy of Wikipedia. *Left*: IBM 3279 display by Retro-Computing Society of Rhode Island (CC BY-SA 3.0); *Right*: DEC VT100 terminal by Jason Scott (Flickr `IMG_9976`, CC BY 2.0), at the Living Computer Museum (apparently connected to the museum's DEC PDP-11/70).

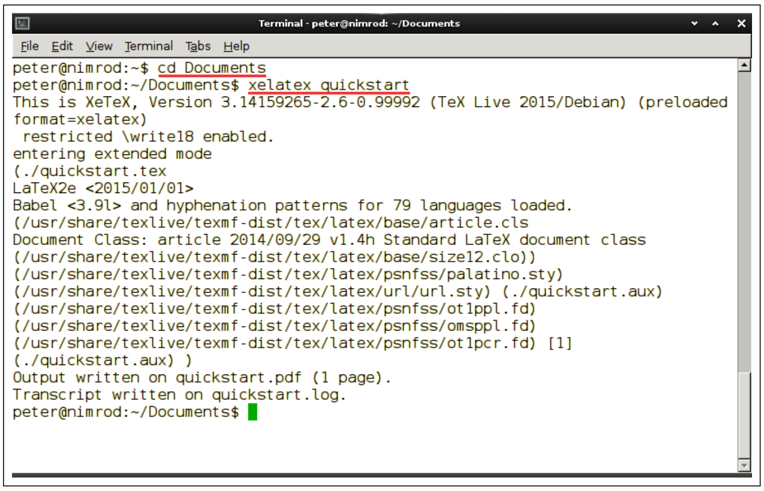
Unix and GNU/Linux: In most graphical interfaces, click on the menu Applications › Accessories › Terminal (in some systems it's called Console).

When you have finished using the terminal, it's good practice to type `exit` (and press ⏎).

B.1.2 Using the terminal window

On a physical terminal you usually have to log in first (very much like today: username and password). In a terminal window this isn't usually necessary (see below) because it's inside your graphical system, and you've already logged in to that.

Figure B.2 – Virtual terminal in a window



In this example I'm logged into a computer called `nimrod` with my username `peter`. The system prompt is the directory name plus a dollar sign (the tilde indicates that I'm in my home directory system). For visibility, I underlined in red here the commands I typed, one to change to my `Documents` folder, and one to run XƎLATEX on the `quickstart.tex` document.

The first thing you see is the prompt (usually a dollar sign or percent sign, or maybe a greater-than pointer `C:\>` like MicroSoft

Disc Operating System (MS-DOS) used to use). When the prompt appears, you can type an instruction (command) and press the ⏎ key at the end of the line to send if off to the computer for processing. Until you press ⏎, the computer has no idea you've finished typing: you MUST press ⏎ at the end of each line of command for it to take effect.

The results, if any, are displayed on the screen, and the prompt is displayed again ready for your next command. Some commands don't have any output: if you change directory or delete a file, for example, you just get another prompt. There's no message confirming the action, and no check to see if you really meant it. You said to do it, and it's done.

B.2 Typesetting

Which LaTeX command you type depends on what output you want and how you want it to be created — see the list on page xxiii. Whichever way you run LaTeX, it will process your file and display a log or record of what it's doing (see Figure B.2 on the preceding page: it looks much the same no matter what system you use).

To typeset your document:

1. Make sure you are in the right directory (folder) in your terminal, then type the command (`xelatex` followed by your filename; or `latexmk` or one of the other workflow-management commands);

2. If you are using citation and reference commands for a bibliography, you will then need to run `biber` or `bibtex` (followed by a space and the name of your document), whichever you have chosen to use (see section 5.3.2.1 on page 142);

3. Run XeLaTeX again as in item above so that the citations are picked up;

4. If you are creating an index, you will then need to run makeindex (followed by a space and the name of your document);

5. Run LaTeX again as in item above so that the citations and index references are resolved.

LaTeX and all the ancillary programs write a transcript of what goes, and this will be shown in the window as well as being written into a log file.

If LaTeX reports any errors — easily identifiable as lines in the log beginning with an exclamation mark (!) — *don't panic!* Turn to section B.3, identify what went wrong, and fix it in your input file. Then re-run LaTeX.

Exercise 32 – Running LaTeX in a terminal or console window

- ☐ Open a command window;
- ☐ Type cd followed by the name of the folder where you saved your sample document;
- ☐ Type your LaTeX command followed by the name of your LaTeX document.

B.3 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 as well as seasoned users 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 below 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.

There is an extensive guide to how to handle errors in LaTeX in (Beeton 2017) (her presentation from TUG 2017) which also has a lot of useful information about how to work with LaTeX in general.

B.3.1 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 e***x***it the program.
q	Carry on ***q***uietly 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 ***e***ditor at the point where you found the error (this only works if you're using an editor which LaTeX can communicate with).
h	Try to give me more ***h***elp.
i	(followed by a correction) means ***i***nput the correction in place of the error and carry on (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.

B.3.2 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.

B.3.3 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: 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.

B.3.3.1 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!

B.3.3.2 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!

B.3.3.3 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 section B.3.3.1 on the preceding page, 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.

B.3.3.4 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 unreasonably long, or macros which are too complex to fit in memory (or more likely just badly-written). I had it happen once (admittedly on an older system) 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...but in fact the current version of LaTeX is capable of handling the longest known footnote at 173 pages without any strain (Flynn 2023b).

B.3.3.5 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 according to the definitions of the fontname package.

B.3.3.6 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 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).

B.3.3.7 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 3 starting on page 65).

C User Groups

The TEX Users Group (TUG) was founded in 1980 for educational and scientific purposes: to provide an organisation 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, language, or topical basis: see the list at `www.tug.org/usergroups.html` for details.

C.1 Conferences and meetings

TUG and many other user groups hold annual meetings where members, newcomers, and visitors can hear about new features, discuss developments, and learn about TEX, typography, and related topics. Full details of TUG-sponsored meetings are on the TUG web site.

This list is maintained by the author, based on public announcements in mailing lists, web forums, and newsgroups. If you have details of a meeting which is not shown here, please contact me. Previous meetings which have taken place are also included here for the record.

Upcoming

ConTEXt 2023: The 17th International ConTEXt meeting will be held on September 10-16th, 2023, in Prague-Sibřina, Czech Republic.

Future of Text vol IV: The fourth annual Symposium on the Future of Text will be held in London (and online) on the 4th of October 2023. This is to announce 'The Future of Text' Vol IV which is now open for proposals (see below).

This year we are focusing on text to extend cognition in extended environments, from paper in notebooks to reams of paper across tables and onto walls, with screens, projections and head mounted devices, interacted with directly as well as through AI.

If you are able to attend on the day in London, please contact `mailto:frode@hegland.com` as soon as possible so that your name can be on the door for the venue.

- The Symposium
- Invitation to attend
- Invitation to contribute

You may choose to propose an article for the book, a presentation for the symposium or both.

Declarative Amsterdam 5: The fifth edition of Declarative Amsterdam will take place on 2 and 3 November 2023 at the Science Park, Amsterdam. It will be a hybrid conference with the opportunity to attend live or online, for both attendees and presenters.

Declarative techniques are a style of computing that expresses the purpose of computation without describing its control flow. It allows you to focus on the 'what', rather than the 'how'. While the conference is mainly focused on XML and related technologies, much of the underlying principle holds true for the LATEX user interface also.

Done and dusted

Balisage 2023: Balisage is the premier conference on the theory, practice, design, development, and application of markup.

We solicit papers on any aspect of markup and its uses, including XML, XSLT, xQuery, JSON, TeX, Markdown, and many others. The conference will run from 31 July to 4 August, and will be virtual again this year, so local watch-parties are encouraged.

Many aspects of LaTeX markup are closely related to the use of XML markup and the handling of structured documents.

TUG 2023: Following the very successful online meetings of past years, the TeX Users Group 2023 meeting will take place at the Hotel Collegium Leoninum, Noeggerathstrasse 34, 53111 Bonn, Germany from July 14–16, with a Tagged PDF developers' workshop on July 13; physically if possible but the conference will be streamed.

DANTE 2023: The 65th annual meeting of the Deutschsprachige Anwendervereinigung TeX e.V. (DANTE e.V.) will take place on July 13, 2023 in Bonn (`www.dante.de/veranstaltungen/dante2023/`), immediately prior to the list item 'TUG 2023' section C.1.

MarkupUK 2023: MarkupUK 2023: although primarily related to XML, all forms of markup are discussed. From 2022, Markup UK and XML Prague will be held in alternate years, starting with XML Prague. We are looking forward to meeting you in June 2022 in Prague and on 1–3 June 2023 at the Mile End campus of Queen Mary University of London.

GuIT 2023: The GuIT meeting 2023, the 19th Italian conference on TeX, LaTeX and digital typography, will be held in Rome on May 20, 2023. For further details: `www.guitex.org/home/en/meeting`.

BachoTeX 2023: Jerzy Ludwichowski writes: The war in Ukraine is raging still, but none the less we decided to organize BachoTeX 2023, and to make up for the lost time by keeping the same theme as the cancelled 2020 conference, "A model kit: modeling and implementing text typesetting in TeX and other systems". The dates are from 29th of April until 3rd of

May, 2023, at Bachotek near Brodnica, in the north-east of Poland. Registrations will be announced separately as soon as possible.

BachoTEX 2022: Once again the Polish Grupa Użytkowników Systemu TEX (GUST) Board and the BachoTEX conference Organizing Committee decided to call it off because of the returning dangerous waves of SARS-CoV-2 infections.

Those of you who have been to Bachotek know that one of the main strengths of our conferences is the proximity of dozens like minded people during several days of intensive work and discussions, with almost no ways for evasion. This creates an environment for generating new ideas, gaining knowledge effortlessly as if by inhaling and — last but not least — supporting friendships. Citing Bogusław Jackowski: "ideas are created between heads".

Unfortunately such physical conditions in the times of an epidemic are a serious weakness. Just try to imagine problems with even one case of an infection in such conditions…

For the future: BachoTEX in 2023 will be, on standard dates, with the same leading theme as for 2020, "A model kit: modeling and implementing text typesetting in TEX and other systems". Keep fingers crossed!

Balisage 2022: The Balisage markup conference is the principal technical meeting specifically about markup, including XML, TEX, Markdown, and many others. The 2022 meeting will be held August 1–5. We don't yet know if it will be all virtual or hybrid, with some in-person component(s).

TUG 2022: Following the very successful online meetings in 2020 and 2021, the TEX Users Group 2022 meeting will also be held online, on 20–24 July via *Zoom*, under the auspices of the University of Adelaide, Australia.

C.2 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 (on DVD), and the CTAN software distribution (containing most of the CTAN archive).

In addition, TUG members can 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 US. See the TUG Web site for details.

C.3 Becoming a TUG member

Please see the forms and information at www.tug.org/join.html. You can join online, or by filling out a paper form. The Nederlandstalige TEX Gebruikersgroep (NTG) (Dutch) and United Kingdom TEX User Group (UKTUG) (UK) 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 www.ntg.nl/newmember.html (the NTG membership page) or 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 occasionally the case). You can order older issues of TUGboat and TEX memorabilia through the TUG store (www.tug.org/store).

The current (2015) TUG membership fee before March 31st is $85 (US) per year for individuals and $55 for students, new graduates, seniors, and citizens of countries with modest economies. Add $20 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 $105. The current institutional rate is $500, which includes up to seven individual memberships.

C.4 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.

D ASCII characters

The American Standard Code for Information Interchange (ASCII) was invented in 1963, and after some development settled down in 1984 as standard X3.4 of American National Standards Institute (ANSI). It represents the 95 codes for the printable characters (A-Z, a-z, 0-9, and punctuation) of the unaccented Latin alphabet, plus 33 internal 'control characters' originally intended for the control of computers, programs, and external devices like printers, screens, disks, modems, etc.

Many other character sets (strictly speaking, 'character repertoires') have been used for accented Latin characters and for other (non-Latin) writing systems, for representing the symbols people use when writing text on computers, but the current standard is ISO 10646 (Unicode), which covers pretty much all the marks the human race makes when communication, and I strongly recommend you use only Unicode UTF-8 when writing for LaTeX.

Although the TeX and LaTeX file formats can easily be used with many other encoding systems (see the discussion of the inputenc package in section 1.9 on page 25), their markup is based on ASCII. It is therefore important that you know where to find all 95 of the printable characters, as some of them are not often used in other text-formatting systems. The printable characters are:

- Uppercase (capital) letters A–Z;
- Lowercase (small) letters a–z;
- Digits 0–9;
- Punctuation, white-space, and symbols:
  - TAB, space, newline (line-end), linefeed;
  - exclamation mark, unidirectional apostrophe, unidirectional quote, comma, hyphen, fullpoint (period), slash, colon, semicolon, question mark, at (@);
  - left and right parentheses, left and right square brackets;
  - hash, dollar, percent, ampersand, asterisk;
  - plus, less-than, equals, greater-than, backslash, caret, underscore, grave accent (backtick), and tilde.

No other ASCII characters can be used (invisible control characters), but using LuaLaTeX or XeLaTeX any other Unicode character is valid (but you may need extra fonts).

E GNU Free Documentation License

Version 1.3, 3 November 2008

E.1 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 works whose purpose is instruction or reference.

E.2 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, eXperimental Computing Facility (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.

E.3 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.

E.4 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.

E.5 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.

E.6 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 multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled 'History' in the various original documents, forming one section Entitled 'History'; likewise combine any sections Entitled 'Acknowledgements', and any sections Entitled 'Dedications'. You must delete all sections Entitled 'Endorsements'.

E.7 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.

E.8 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.

E.9 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.

E.10 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.

E.11 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.

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.

E.12 RELICENSING

The Massive Multiauthor Collaboration (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 (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 Corporation, 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.

E.13 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

American Mathematical Society (2002). *Short Math Guide for LaTeX*. Providence, RI: AMS. URL: ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf.

Anderson, Chris, ed. (1993). *WIRED*. San Francisco, CA: Condé Nast.

Anon (1100). 'Táin bó Cúailnge'. In: *Leabhar na h-Uidhri: The Book of the Dun Cow*. Dublin: Royal Irish Academy [1870], p. 55.

Beeton, Barbara, ed. (1980). *TUGboat*. Portland, OR: TeX Users Group.

Beeton, Barbara (Apr. 2005). 'Re: Guillemets in British English?' In: *Discussion of Type and Typographic Design*.

Beeton, Barbara (2017). 'Debugging LaTeX files: Illegitimi non carborundum'. In: *TUG 2017*. Vol. 38. 2. Portland, OR: TeX Users Group. URL: http://tug.org/TUGboat/Contents/contents38-2.html.

Berners-Lee, Tim, Roy Fielding and Larry Masinter (2005). *Uniform Resource Identifier (URI): Generic Syntax*. Reston, VA.

Berry, John (2017). 'TYPE Magazine'. In: History of Type 1. URL: https://www.typemag.org/post/the-faces-of-microsoft.

Bradner, Scott (1997). *Key words for use in RFCs to Indicate Requirement Levels*. Reston, VA.

Bull, RJ (1972). *Accounting in Business*. London: Butterworths. ISBN: 0-406-70651-4.

Burnard, Lou and Michael Sperberg-McQueen (1995). 'Segmentation and Alignment'. In: *Guidelines for the Text Encoding Initiative*. Ed. by Lou Burnard and Michael Sperberg-McQueen. Oxford: OUP. Chap. 16. URL: https://tei-c.org/release/doc/tei-p5-doc/en/html/SA.html.

Carnes, Lance and Karl Berry, eds. (2004). *The PracTEX Journal*. Portland, OR: TEX Users Group. URL: http://www.tug.org/pracjourn/.

Davy, William (1806). *A System of Divinity*. Lustleigh, Devon: Published by the author.

Doob, Michael (2002). *A Gentle Introduction to TEX: A Manual for Self-Study*. Tech. rep. Portland, OR. URL: http://www.ctan.org/tex-archive/info/gentle/.

Dunne, Brian (2020). *The lwarp Package: LATEX to HTML*. 0.79. Portland, OR: TEX Users Group. URL: http://mirrors.ctan.org/macros/latex/contrib/lwarp/lwarp.pdf.

Flaubert, Gustave (1857). *Madame Bovary*. Paris.

Flynn, Peter (2001). 'TEX — a mass market product?: Or just an image in need of a makeover?' In: *TUG 2001*. Vol. 22. 3. Portland, OR: TEX Users Group. URL: http://tug.org/TUGboat/tb22-3/tb72flynn.pdf.

Flynn, Peter (2002). 'Formatting Information'. In: *TUGboat*. Vol. 23. 2, pp. 115–250. URL: http://www.ctan.org/tex-archive/info/beginlatex/.

Flynn, Peter (2012). 'Typographers' Inn'. In: *TUGboat*. Vol. 33. 1, pp. 8–10. URL: http://tug.org/TUGboat/tb33-1/tb103inn.pdf.

Flynn, Peter (2014a). 'Human Interfaces to Structured Documents: The usability of software for authoring and editing'. PhD thesis. University College Cork, School of Applied Psychology. URL: https://cora.ucc.ie/handle/10468/1690.

Flynn, Peter (2014b). *The Very Short Guide to LATEX*. URL: http://latex.silmaril.ie/veryshortguide/veryshortguide.pdf.

Flynn, Peter (2023a). *Formatting Information*. Cork, Ireland: Silmaril Consultants. URL: http://latex.silmaril.ie/formattinginformation/.

Flynn, Peter (2023b). 'Typographers' Inn'. In: *TUGboat*. Vol. 44. 1, pp. 3–5. URL: http://tug.org/TUGboat/tb44-1/tb136inn.pdf.

Fothergill, John (1929). *An Innkeeper's Diary*. 3rd ed. London: Penguin.

Goossens, Michel, Sebastian Rahtz and Frank Mittelbach (1997). *The LaTeX Graphics Companion*. Reading, MA: Addison-Wesley. ISBN: 0-201-85469-4.

Goossens, Michel, Sebastian Rahtz, Ross Moore et al. (1999). *The LaTeX Web Companion*. Reading, MA: Addison-Wesley. ISBN: 0-201-43311-7.

Grätzer, George (2014). *Practical LaTeX*. Heidelberg: Springer International, p. 216. ISBN: 978-3-319-06425-3.

Hartke, Stephen G (2006). 'The PracTeX Journal'. In: 1. URL: http://www.tug.org/pracjourn/2006-1/hartke/hartke.pdf.

Heller, Robert (Mar. 2003). 'New To (La)TeX...Unlearning Bad Habits'. In: *comp.text.tex* MPG.18d82140d65ddc5898968c@news.earthlink.net, (all pages). URL: https://groups.google.com/g/comp.text.tex/c/RPZMdIYtX8c/m/xzRqJeVHm5sJ.

Kelly, Paula and Denise O'Connor (2005). *Training for ECDL*. Dublin: Blackrock Education Centre. ISBN: 0-9540287-9-1.

Kirschenbaum, Valerie (2005). *Goodbye Gutenberg*. New York: Global Renaissance Society. ISBN: 0974575038. URL: http://www.goodbyegutenberg.com.

Knauff, Markus and Jelica Nejasmic (Dec. 2014). 'An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development'. In: *PLoS One*. DOI: 10.1371/journal.pone.0115069. URL: http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0115069.

Knuth, Donald E, Tracey Larrabee and Paul M Roberts (1989). *Mathematical Writing*. MAA Notes 14. Washington, DC: Mathematical Association of America. ISBN: 0-88385-063-X.

Knuth, Donald Ervin (1980). *The Art of Computer Programming*. 2nd ed. Vol. 1. Reading, MA: Addison-Wesley. ISBN: 0-201-89685-0.

Knuth, Donald Ervin (1986). *The TEXbook*. 7th ed. Reading, MA: Addison-Wesley. ISBN: 0-201-13447-0.

Lamport, Leslie (1994). *LATEX: A Document Preparation System*. 2nd ed. Reading, MA: Addison-Wesley. ISBN: 0-201-52983-1.

Lehman, Philip et al. (2015). *The biblatex Package: Programmable Bibliographies and Citations*. 3rd ed. Portland, OR: TEX Users Group. URL: http://mirrors.ctan.org/macros/latex/contrib/biblatex/doc/biblatex.pdf.

Mac Namara, Matthew (2003). *La Textualisation de* Madame Bovary. Amsterdam: Rodopi. ISBN: 90-420-0984-5.

Mittelbach, Frank et al. (2004). *The LATEX Companion*. 2nd ed. Boston, MA: Addison-Wesley/Pearson Education. ISBN: 0-201-36299-6.

Oetiker, Tobias (2014). *A (Not So) Short Introduction to LATEX 2$_\varepsilon$*. Portland, OR: TEX Users Group. URL: http://www.ctan.org/pkg/lshort.

Pakin, Scott (2009). *A comprehensive list of symbols in TEX*. Tech. rep. URL: http://www.ctan.org/tex-archive/info/symbols/comprehensive/.

Rawlings, Marjorie Kinnan (Dec. 1932). 'Varmints'. In: *Scribner's Magazine*.

Raymond, Eric (May 2014). *How To Ask Questions The Smart Way*. URL: http://www.catb.org/esr/faqs/smart-questions.html (visited on 07/02/2023).

Reckdahl, Keith (2006). *Using imported graphics in LATEX 2$_\varepsilon$*. Tech. rep. URL: http://www.ctan.org/tex-archive/info/epslatex.pdf/.

Ryder, John (1976). *Printing for Pleasure*. London: Bodley Head. ISBN: 0-370-10443-9.

Sperberg-McQueen, Michael (2016). *Choosing an editor*. Pers. Comm.

Talbot, Nicola (2014). *Creating a LaTeX Minimal Example*. Tech. rep. Portland, OR. URL: http://tug.ctan.org/info/dickimaw/dickimaw-minexample.pdf.

TeX Users Group (2003). *Getting Started with TeX, LaTeX, and friends*. Tech. rep. Portland, OR. URL: http://www.tug.org/begin.html.

van Dongen, Marc (2012). *LaTeX and Friends*. Berlin: Springer. ISBN: 978-3-642-23815-4. URL: http://www.springer.com/us/book/9783642238154.

Index

See Table 3 on page xl for the meanings of the typographic formatting used here and in the text.

Revision history

v8.07 — 1 August 2023 Switched back to XeLaTeX on KB's recommendation because of speed concerns with LuaLaTeX. Started work on an appendix on implementing designs, with a section of metadata and titling.

v8.06 — 27 July 2023 Extensively rewrote the section on alignment outside a tabular environment.

v8.05 — 12 June 2023 Added Barbara Beeton's comment on math delimiters; extracts from TYPO-L about em rules and en rules.

v8.04 — 7 May 2023 Added GUIT23 and DANTE meetings, corrected expansion of plural/suffix on defining instances of acronyms.

v8.03 — 18 April 2023 Added ConTEXt meeting (Sept 23), corrected misleading markup on Personal TeX Directories.

v8.02 — 14 April 2023 Resolved personname references to xml:ids, replaced more character entity references.

v8.01 — 4 April 2023 Removed (obsolete) list of commercial systems, edited surrounding text, and replaced links to the table with links to the subsection. Corrected @startinglinenumber on included code; resolved the right incantation for Windows to place the PTD; replaced dozens of character entity references no longer needed with their actual Unicode character.

v8.0 — 1 April 2023 Finally switched to LuaLaTeX, edited references to fit. Tidied up acronym crossrefs, updated lots of package references.

v7.99 — 8 March 2023 Added a few more exercises. Clarified some explanations. Completely rewrote the section on installation. Tidied up cross-reference linking in the PDF.

v7.98 — 31 January 2023 Added many more exercises. Corrected many bits of poor grammar and wordy explanations.

Completely rewrote the section on fonts. pdflatex and plain LaTeX are now omitted (ie XeLaTeX only).

v7.97 — 1 November 2022 Started work on adding exercises. Added Discord to help section.

v7.96 — 11 October 2022 Added docx2tex converter and rationalised the section.

v7.95 — 23 September 2022 Rewrote the terminology and description of floats.

v7.93 — 13 August 2022 Rewrite of the TTF and OTF font installation section, updates to the accuracy of the other font installation sections.

v7.92 — 9 August 2022 Updated BachoTEX conference cancellation.

v7.91 — 9 June 2022 Updated BachoTEX conference.

v7.9 — 20 February 2022 Updated conferences. Changed the Quick Start to use XeLaTeX and updated the commentary on the commands used.

v7.85 — 10 July 2021 Updated list of commercial implementations to remove companies no longer trading.

v7.84 — 16 February 2021 Added new favicon.

v7.83 — 10 December 2020 Updated reference to csvtools in tables to datatool, and added image of tabled being edited in LYX.

v7.82 — 23 April 2020 Added footnote in verbatim about the alignment of the end command in other packages

v7.81 — 19 November 2018 Updated and rewrote the section on image file formats

v7.8 — 25 February 2018 Updated BachoTEX and MarkupUP

v7.7 — 14 February 2018 Updated conference dates; added details of lwarp; fixed bibref pointer to PDF pages where given.

v7.6 — 14 May 2017 Added salient comments about how to choose an editor from Michael Sperberg-McQueen, and slides from TUG 2017 from Barbara Beeton on how to handle LaTeX errors.

v7.5 — 21 January 2017 Full update of Mac and Windows installation with new screenshots and a new layout for procedures; introduction of an XSLT routine to test when an element is immediately preceded by another element with only white-space intervening, and to generate a space token if needed (this overcomes the design flaw alluded to in the revision comments to v3.7 below); many updates to phraseology, and removal of obsolete mentions of packages and practices.

v7.45 — 4 November 2016 Updated details of TDS package installation

v7.44 — 11 October 2016 Updated details of meetings

v7.43 — 15 April 2016 Updated comments on superscripted ordinals, spacing around em rules, and details of loading OTF/TTF fonts

v7.42 — 1 March 2016 Edited all sections, corrected spellings, updated package details and all examples to XeLaTeX/biblatex/biber, fixed stray typos, checked links,. This is preparatory to v8, due for later in 2016.

v7.41 — 30 January 2016 Minor typos, updated event dates

v7.4 — 16 November 2015 Started updating converters

v7.3 — 26 October 2015 Updated meetings for 2016

v7.2 — 12 July 2015 Several sections re-ordered to present the material is a more logical fashion. Numerous grammatical elisions corrected, and some late typos (thanks to Rob Borland).

v7.1 — 10 June 2015 Minor changes to accommodate revised PDF format.

v7.0 — 30 July 2014 Completely re-edited, large sections rewritten, obsolete material removed, including installation changes for the 2014 DVD, and a completely new responsive web site launched.

v6.0 — 30 December 2013 Updated links, replaced references to obsolescent packages, rewrote installation for the 2013 DVD

v5.7 — 21 December 2011 Moved and expanded the details of creating a Personal TEX Directory. Added new section on using the LaTeX Font Catalogue.

v5.6 — 1 November 2011 Revised installation details for TL2011.

v5.5 — 25 May 2011 Minor revision; added details of page references in citations, a warning about the broken harvard.sty and its solution with natbib and har2nat, and a reference to bibunits.

v5.4 — 27 April 2011 Minor revision; added details of packages for body type size options; fixed bug in HTML bibrefs which were failing to retrieve the date.

v5.3 — 22 March 2011 Minor revision; removed mention of VTeX as a synchronous typographic editor and replaced with BaKoMa TeX. Located and fixed XSLT bug which was preventing cross-reference IDs being used correctly. Finally tracked down the non-appearance of italics in some places (no Lite Italic in my copy of Antique Olive).

v5.2 — 13 March 2011 Minor revision; even finer details of the problems Windows users face at installation.

v5.1 — 5 March 2011 Minor revision; spellings and font selection errors repaired; missing rule in HTML table example; better details of the problems Windows users face at installation.

v5 — 28 January 2011 Major revision; Installation and Editors sections rewritten, remaining package references updated, and more new ones added.

v4 — 1 April 2009 Major revision; Installation and Editors sections reorganised, all package references updated, and new ones added.

v3.7 — 22 December 2006 There have again been some small but significant improvements, both in the LaTeX code, and in the default installations and implementations. The default colour package is now xcolor; the default output for many people is now PDF; and the advent of XeTeX means that TrueType fonts and Unicode are now much more easily supported. The DocBook DTD has been updated to 4.4, and the TypeBook DTD shim likewise, and the IGNOREd code from 3.5 and earlier versions has now finally been dropped. XSLT still has the notorious design flaw of ignoring whitespace nodes in mixed content when a DTD is used, but this seems to have gone unnoticed except by the publishing industry. The use of the `citetitle` element for bibliographic references has been replaced by `biblioref`.

v3.6 — 31 March 2005 Since the publication of the November 2003 edition in TUGboat, several new books on LaTeX have been released, and this edition reflects some of the new material and approaches contained in them. See the Bibliography for details of these publications. The only technical change has been to use empty elements for the TeX, LaTeX, and other logos instead of the more usual entities so that the HTML version can use CSS to produce better logos. Thanks to whoever wrote the CSS for TeX4ht, which is where I found the styles.

v3.5 — 29 July 2004 Modified DTD to add span element type to allow use of external entities for formatted TeX, LaTeX, and other logos in the HTML version. Changed entity declaration in the internal subset to enable this, and switched declarations and marked sections in the DTD. This now means it needs Saxon 7 or 8 to process, as Saxon 6 does not handle parameter entities values used as parameter entity declarations.

v3.4 — 9 November 2003 Applied all Barbara Beeton's corrections (see separate emails) and rewrote a few formatting macros to allow the document to fit more easily into the US Letter shape. It would be nice if it would also format for A5 so that it could become a paperback but that's another day's work. Started on writing the missing sections (Installing Type 1 CM Fonts and Going beyond LaTeX, but these are not finished yet) and rewrote entirely the existing (non-CM) Type 1 font installation procedure in line with the new (unreleased) Gutta-Percha script. Added hidden meanings for CD, DVD, IBM.

v3.3 — 20 August 2003 Fixed XSLT bug which wrongly lettered appendices. Fixed problem which called wrong font for examples of Times and Helvetica (thanks to William Adams). Updated numerous typos, added comments about pdftex option to color. Rewrote formatting for TUGboat.

v3.2 — 5 March 2003 Finished rewrite. Revised and expanded almost everything.

v3.1 — 28 August 2002 Recast in DocBook and reworded some sections. Started the big rewrite.