

# L<sup>A</sup>T<sub>E</sub>X for Complete Novices

Nicola L. C. Talbot

School of Computing Sciences  
University of East Anglia  
Norwich. NR4 7TJ. U.K.

<http://theoval.cmp.uea.ac.uk/~nlct/>

Tuesday 15<sup>th</sup> January, 2008 (version 1.3)

Copyright (c) 2007 Nicola L. C. Talbot

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “[GNU Free Documentation License](#)”.

The base URL for this document is: <http://theoval.cmp.uea.ac.uk/~nlct/latex/novices/>

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Overview . . . . .	7
1.2	Recommended Reading . . . . .	9
<b>2</b>	<b>Some Definitions</b>	<b>11</b>
2.1	Source Code . . . . .	15
2.2	DVI File (or Output File) . . . . .	15
2.3	Commands (also called macros or control sequences) . . . . .	16
2.4	Grouping . . . . .	19
2.5	Arguments (also called parameters) . . . . .	20
	2.5.1 Mandatory Arguments . . . . .	20
	2.5.2 Optional Arguments . . . . .	22
2.6	Moving Arguments and Fragile Commands . . . . .	26
2.7	Robust Commands . . . . .	27
2.8	Short and Long Commands . . . . .	27
2.9	Declarations . . . . .	28
2.10	Environments . . . . .	28
2.11	Preamble . . . . .	31

## CONTENTS

2.12	Class File	31
<b>3</b>	<b>From Source Code to Typeset Output</b>	<b>33</b>
3.1	Text Editor and Terminal Approach	36
3.2	TeXnicCenter	52
3.3	WinEdt	67
<b>4</b>	<b>Creating a Simple Document</b>	<b>75</b>
4.1	Using Simple Commands	81
4.2	Special Characters and Symbols	84
4.3	Lists	92
4.3.1	Unordered Lists	93
4.3.2	Ordered Lists	98
4.3.3	Description Environment	102
4.4	Simple font changing commands	107
4.4.1	Changing the Font Style	108
4.4.2	Changing the Font Size	116
4.5	Aligning Material in Rows and Columns	118
4.6	Boxes and Mini-Pages	128
<b>5</b>	<b>Structuring Your Document</b>	<b>136</b>
5.1	Author and title information	136

## CONTENTS

5.2	Abstract . . . . .	140
5.3	Chapters, Sections, Subsections . . . . .	142
5.4	Creating a Table of Contents . . . . .	148
5.5	Cross-Referencing . . . . .	152
5.6	Creating a Bibliography . . . . .	162
5.7	Page Styles and Page Numbering . . . . .	170
<b>6</b>	<b>Packages</b>	<b>175</b>
6.1	Using Packages . . . . .	175
6.1.1	The <code>graphicx</code> Package . . . . .	176
6.1.2	Multi-Lingual Support: using the <code>babel</code> package . . . . .	186
6.1.3	Changing the format of <code>\today</code> . . . . .	189
6.2	Downloading and Installing Packages . . . . .	191
6.2.1	Refreshing the <code>T<sub>E</sub>X</code> Database . . . . .	196
<b>7</b>	<b>Floats</b>	<b>200</b>
7.1	Figures . . . . .	201
7.1.1	Side-By-Side Figures . . . . .	206
7.1.2	Sub-figures . . . . .	210
7.2	Tables . . . . .	214

## CONTENTS

<b>8</b>	<b>Defining Commands</b>	<b>219</b>
8.1	Defining Commands with an Optional Argument . . . . .	232
8.2	Redefining Commands . . . . .	237
<b>9</b>	<b>Mathematics</b>	<b>243</b>
9.1	In-Line Mathematics . . . . .	244
9.2	Displayed Mathematics . . . . .	246
9.3	Mathematical Commands . . . . .	249
9.3.1	Maths Fonts . . . . .	250
9.3.2	Greek Letters . . . . .	253
9.3.3	Subscripts and Superscripts . . . . .	254
9.3.4	Functional Names . . . . .	260
9.3.5	Fractions . . . . .	267
9.3.6	Roots . . . . .	272
9.3.7	Mathematical Symbols . . . . .	275
9.3.8	Delimiters . . . . .	283
9.3.9	Arrays . . . . .	298
9.3.10	Vectors . . . . .	301
9.3.11	Mathematical Spacing . . . . .	304
<b>10</b>	<b>Defining Environments</b>	<b>308</b>

<b>11 Counters</b>	<b>316</b>
<b>12 Lengths</b>	<b>323</b>
<b>13 Common Errors</b>	<b>328</b>
13.1 * (No message, just an asterisk prompt) . . . . .	331
13.2 Argument of <code>\cline</code> has an extra <code>}</code> . . . . .	332
13.3 Argument of <code>\multicolumn</code> has an extra <code>}</code> . . . . .	332
13.4 <code>\begin{...}</code> ended by <code>\end{...}</code> . . . . .	332
13.5 Bad math environment delimiter . . . . .	333
13.6 Can only be used in preamble. . . . .	333
13.7 Command ... already defined . . . . .	334
13.8 Display math should end with <code>\$\$</code> . . . . .	335
13.9 Environment ... undefined. . . . .	335
13.10 Extra alignment tab has been changed to <code>\cr</code> . . . . .	336
13.11 Extra <code>\right</code> . . . . .	336
13.12 File ended while scanning use of ... . . . . .	337
13.13 File not found. . . . .	337
13.14 Illegal character in array arg . . . . .	339
13.15 Illegal parameter number in definition . . . . .	339
13.16 Illegal unit of measure (pt inserted). . . . .	340
13.17 Lonely <code>\item</code> . . . . .	340

## CONTENTS

13.18	Misplaced alignment tab character &	340
13.19	Missing } inserted	341
13.20	Missing \$ inserted	341
13.21	Missing \begin{document}	343
13.22	Missing delimiter	344
13.23	Missing \endcsname inserted	345
13.24	Missing \endgroup inserted	346
13.25	Missing number, treated as zero	346
13.26	Paragraph ended before \begin was complete	347
13.27	Runaway argument	348
13.28	Something's wrong—perhaps a missing \item.	350
13.29	There's no line here to end.	351
13.30	Undefined control sequence	351
13.31	You can't use 'macro parameter character #' in horizontal mode	354
<b>14</b>	<b>Need More Help?</b>	<b>355</b>
	<b>Bibliography</b>	<b>356</b>
	<b>History</b>	<b>358</b>



## CONTENTS

<b>GNU Free Documentation License</b>	<b>360</b>
1. APPLICABILITY AND DEFINITIONS . . . . .	361
2. VERBATIM COPYING . . . . .	364
3. COPYING IN QUANTITY . . . . .	365
4. MODIFICATIONS . . . . .	366
5. COMBINING DOCUMENTS . . . . .	370
6. COLLECTIONS OF DOCUMENTS . . . . .	371
7. AGGREGATION WITH INDEPENDENT WORKS . . . . .	372
8. TRANSLATION . . . . .	372
9. TERMINATION . . . . .	373
10. FUTURE REVISIONS OF THIS LICENSE . . . . .	374
ADDENDUM: How to use this License for your documents . . . . .	374
 <b>Index</b>	 <b>376</b>

# List of Figures

3.1	Starting vim from a terminal . . . . .	39
3.2	Starting a new file in vim . . . . .	40
3.3	Input mode in vim . . . . .	41
3.4	Creating a sample document in vim . . . . .	42
3.5	Saving your document in vim (the file name should be omitted if the file already has a name) . . . . .	43
3.6	Listing the contents of the current directory . . . . .	45
3.7	Running $\text{\LaTeX}$ . . . . .	46
3.8	Running $\text{\LaTeX}$ . . . . .	47
3.9	Load a DVI file into a DVI viewer . . . . .	50
3.10	Viewing a DVI file in kdvi . . . . .	51
3.11	Loading a PostScript file . . . . .	53
3.12	TeXnicCenter Tip of the Day Window . . . . .	55
3.13	TeXnicCenter Configuration Wizard . . . . .	56
3.14	TeXnicCenter Configuration Wizard . . . . .	57
3.15	TeXnicCenter Configuration Wizard . . . . .	58
3.16	TeXnicCenter . . . . .	59
3.17	New Project Dialog Box . . . . .	60
3.18	New Project Dialog Box . . . . .	61

## LIST OF FIGURES

3.19	TeXnicCenter — New Project Started . . . . .	62
3.20	TeXnicCenter — Typing in Source Code . . . . .	64
3.21	TeXnicCenter — Selecting Output Type . . . . .	65
3.22	TeXnicCenter (using L <sup>A</sup> T <sub>E</sub> X and dvips) . . . . .	66
3.23	TeXnicCenter — Showing Error . . . . .	68
3.24	WinEdt . . . . .	70
3.25	WinEdt . . . . .	71
3.26	WinEdt — Saving the File . . . . .	72
3.27	WinEdt — L <sup>A</sup> T <sub>E</sub> X Output . . . . .	73
4.1	TeX views each letter as a box . . . . .	128
6.1	The TeX Directory Structure showing the main LaTeX- related sub-directories . . . . .	193
6.2	MiKTeX: Updating the database . . . . .	198
7.1	Some shapes . . . . .	205
7.2	A Circle . . . . .	209
7.3	A Rectangle . . . . .	209
7.4	Two Shapes: (a) A Rectangle and (b) A Circle . . . . .	213

# List of Tables

4.1	Symbols . . . . .	86
4.2	Ligatures and Special Symbols . . . . .	89
4.3	Accent Commands . . . . .	91
4.4	Font changing commands . . . . .	111
4.5	Font changing declarations . . . . .	112
4.6	Font size changing declarations . . . . .	117
7.1	A Sample Table . . . . .	215
7.2	A Sample Table . . . . .	217
8.1	Predefined Names . . . . .	241
9.1	Maths Font Changing Commands . . . . .	250
9.2	The amfonts and amsmath Font Commands . . . . .	253
9.3	Lower Case Greek Letters . . . . .	254
9.4	Upper Case Greek Letters . . . . .	255
9.5	Function Names . . . . .	260
9.6	Relational Symbols . . . . .	275
9.7	Binary Operator Symbols . . . . .	276

## LIST OF TABLES

9.8	Arrow Symbols	277
9.9	Symbols with Limits	278
9.10	Ellipses	281
9.11	Delimiters	285
9.12	Mathematical Spacing Commands	305
12.1	Units of Measurement	324

# List of Exercises

1	Simple Document . . . . .	78
2	Using Simple Commands . . . . .	82
3	Using Special Characters . . . . .	90
4	Lists . . . . .	106
5	Fonts . . . . .	118
6	Aligning Material . . . . .	126
7	Creating Title Pages . . . . .	138
8	Creating an Abstract . . . . .	140
9	Creating Chapters, Sections etc . . . . .	145
10	Creating a Table of Contents . . . . .	149
11	Cross-Referencing . . . . .	159
12	Creating a Bibliography . . . . .	169
13	Page Styles and Page Numbering . . . . .	174
14	Using the <code>graphicx</code> Package . . . . .	185
15	Using the <code>datetime</code> package . . . . .	191
16	Creating Figures . . . . .	206
17	Creating Sub-Figures . . . . .	214
18	Creating Tables . . . . .	217
19	Defining a New Command . . . . .	230

## LIST OF TABLES

20	Defining Commands with an Optional Argument . . . . .	234
21	Renewing Commands . . . . .	242
22	Maths: Fractions and Symbols . . . . .	282
23	Maths: Vectors and Arrays . . . . .	304
24	More Mathematics . . . . .	304
25	Defining a New Environment . . . . .	314
26	Using Counters . . . . .	322

# Chapter 1

## Introduction

The aim of this document is to introduce  $\text{\LaTeX}$  to a non-technical person. To begin with it may be best to give a quick overview of  $\text{\TeX}$  and  $\text{\LaTeX}$ , and how they are related, as newcomers to  $\text{\LaTeX}$  are often confused by the two terms. (Don't worry if this paragraph sounds too technical, just skip to the next paragraph, and come back later when you're feeling a bit more confident.)  $\text{\TeX}$  is a typesetting application written by **Donald Knuth**, which typesets text via a set of instructions called primitives. In general, these primitives are too complicated to use, so there are several formats that allow you to access  $\text{\TeX}$  in a slightly more user friendly way. These formats basically define a set of commands based on  $\text{\TeX}$ 's primitives that you can use to create your document. The original format is called "Plain  $\text{\TeX}$ ", however many people find this format difficult to use, so some opt to use the format called " $\text{\LaTeX}$ " which was written by Leslie Lamport. There are also other formats available, such as ConTeXt, but this document only covers  $\text{\LaTeX}$ . You can think of  $\text{\LaTeX}$  as an intermediary between you and  $\text{\TeX}$ , translating your instructions into  $\text{\TeX}$ 's primitive commands.

[Should I use  
Plain TeX or  
LaTeX?]



## 1. INTRODUCTION

L<sup>A</sup>T<sub>E</sub>X is excellent for producing professional looking documents, however it is a *language* not a word processor, so it can take a bit of getting used to, particularly if you have never had any experience using programming languages.

[Why is TeX  
not a  
WYSIWYG  
system?]

L<sup>A</sup>T<sub>E</sub>X does take a while to learn which can be quite daunting, but if you are prepared to invest some time studying the basics, then you will find that the harder tasks which you find so frustrating using a word processor are much easier using L<sup>A</sup>T<sub>E</sub>X. Here are a few reasons why I prefer to use L<sup>A</sup>T<sub>E</sub>X, although it is not an exhaustive list:

L<sup>A</sup>T<sub>E</sub>X is far better at typesetting mathematical equations than word processors. I wrote my Ph.D. thesis back in the days of L<sup>A</sup>T<sub>E</sub>X2.09 (the old version of L<sup>A</sup>T<sub>E</sub>X) and given the high quantity of mathematics that I had to typeset, it would have taken me considerably longer to write it in a word processor, and the resulting document wouldn't have looked nearly as good.

For example, compare the following equations:

1. Using equation editor in Microsoft Word:<sup>1</sup>

---

<sup>1</sup>I was unable to find a calligraphic font for the  $\mathcal{L}$ . The font looks a little ragged because I had to convert it to bitmap to include it in this document.

## 1. INTRODUCTION

$$\frac{\partial^2 L}{\partial z_i^{\rho^2}} = -\frac{\partial \rho_i}{\partial z_i^{\rho}} \left( \frac{\partial v_i}{\partial \rho_i} \frac{e^{v_i}}{1 - e^{v_i}} + v_i \frac{e^{v_i} \frac{\partial v_i}{\partial \rho_i} (1 - e^{v_i}) + e^{2v_i} \frac{\partial v_i}{\partial \rho_i}}{(1 - e^{v_i})^2} \right)$$

2. Using L<sup>A</sup>T<sub>E</sub>X:

$$\frac{\partial^2 \mathcal{L}}{\partial z_i^{\rho^2}} = -\frac{\partial \rho_i}{\partial z_i^{\rho}} \left( \frac{\partial v_i}{\partial \rho_i} \frac{e^{v_i}}{1 - e^{v_i}} + v_i \frac{e^{v_i} \frac{\partial v_i}{\partial \rho_i} (1 - e^{v_i}) + e^{2v_i} \frac{\partial v_i}{\partial \rho_i}}{(1 - e^{v_i})^2} \right)$$

(Incidentally, this equation was taken from some kernel survival analysis, so it is a genuine piece of mathematics. You will find out how to create this equation on page 288 in [section 9.3.8](#).)

That's all very well and good if you want to typeset some equations, but if your work doesn't involve maths, does that mean that L<sup>A</sup>T<sub>E</sub>X is not for you? Although I am a mathematician, I have written plenty of non-mathematical documents, including fictional work and newsletters, but I still opt for L<sup>A</sup>T<sub>E</sub>X because using L<sup>A</sup>T<sub>E</sub>X ensures consistent formatting, and

## 1. INTRODUCTION

the style of the document can be completely changed by simply using a different class file, or loading additional packages. This means that I can concentrate on writing the document, rather than worrying about how it will look. It also means that if, after having written a 200 page document, I then find that I need to change all the figure captions so that they are labelled “Fig” instead of “Figure”, all I need to do is edit a single line, rather than going through 200 pages to individually edit every single figure caption! In fact, if you browse The T<sub>E</sub>X Catalogue Topic Index [1], you will see that T<sub>E</sub>X users span both the sciences and the arts.

L<sup>A</sup>T<sub>E</sub>X makes it easy to cross-reference units such chapters, sections, equations, figures and tables. It is also easy to generate a table of contents, list of figures, list of tables, index, glossary and bibliography. You don't need to worry about numbering anything, as this is done automatically, which means that you can insert new sections or swap sections around without having to worry about updating the rest of your document. Furthermore, if you use B<sub>I</sub>B<sub>T</sub>E<sub>X</sub> in combination with L<sup>A</sup>T<sub>E</sub>X, and you have, say, 100 or more citations, it doesn't matter if you are then told that the citations have to be re-ordered (say, in order of citation rather than alphabetically.) All that is required is a minor edit to change the appropriate style file rather than ploughing through the entire document changing all the citations by hand.

When you are editing a document using a word processor, the word

## 1. INTRODUCTION

processor has to work out how to reformat the document every time you type something. If you have a large document with a great many inserted objects (such as figures and equations) the response to keyboard input can become very slow. You may find that after typing a few words you will have to wait until the computer catches up before you can see what you have typed. With  $\text{\LaTeX}$  you type in your code using an ordinary text editor. The document doesn't get formatted until you pass it to  $\text{\LaTeX}$ , which means that you are not slowed down by constant reformatting.

Lastly, there's the fact that  $\text{\TeX}$  follows certain typographical rules, so you can leave most of the typesetting to  $\text{\TeX}$ . For example, if any of the following combination of letters are found: **fl**, **ffl**, **ff**, **fi**, **ffi**, they will automatically be converted into the corresponding ligatures: fl, ffl, ff, fi, ffi. Note the difference between fluffier (2 ligatures) and fluffier (no ligatures).  $\text{\TeX}$  also has good justification and hyphenation rules, and can help prevent widows and orphans.<sup>2</sup>

Some of these points may seem minor but they all contribute towards the impact of the entire document. When writing technical documents, the presentation as well as the content is important. All too often ex-

[[Typography tutorials](#)]

---

<sup>2</sup>For those of you unfamiliar with typesetting terms, a widow is where the last line of a paragraph occurs on the top of a page and an orphan is where the first line of a paragraph occurs on the bottom of a page. This is generally considered sloppy typography.  $\text{\TeX}$  provides some parameters that can be modified to reduce such occurrences.

## 1. INTRODUCTION

aminers or referees are put off reading a document because it is badly formatted. This provokes an immediate negative reaction and provides little desire to look favourably upon your work. The same is true in the publishing world: if your submission looks sloppy, you will be marked as an amateur and not worth their time. Would you spend time on preparing a presentation for an important job interview, only to turn up wearing jeans and a T-shirt? It's not enough to have a good idea, you need to be able to engage a reader's interest, and maintain that interest in order to disseminate your idea.

To give you an idea of what you can do with L<sup>A</sup>T<sub>E</sub>X, this document was written in L<sup>A</sup>T<sub>E</sub>X.<sup>3</sup> The PDF versions were generated using PDFL<sup>A</sup>T<sub>E</sub>X and `makeindex` and the HTML version was generated using the L<sup>A</sup>T<sub>E</sub>X2HTML<sup>4</sup> converter.

For more good reasons as to why you might want to use L<sup>A</sup>T<sub>E</sub>X instead of a word processor, have a look at [http://www.ctan.org/what\\_is\\_tex.html#whytex](http://www.ctan.org/what_is_tex.html#whytex).

[Conversion  
from (La)TeX  
to HTML]

---

<sup>3</sup>The source code is available at <http://theoval.cmp.uea.ac.uk/~nlct/latex/>, but it really is *not* the place to start if you are a beginner, as it contains L<sup>A</sup>T<sub>E</sub>X and Perl code beyond the scope of this tutorial.

<sup>4</sup><http://www.latex2html.org/>

## 1.1 Overview

This document is structured as follows:

**Chapter 2** defines terms that will be used throughout this document.

I strongly suggest that you look through this chapter before you start so that you understand the terminology used in this document.

At the very least, you should read the first part that details how input code and corresponding typeset output are displayed in this document—you need to understand the difference between input (source code) and output (how the source code will appear in the typeset document).

**Chapter 3** details the software that you will need to use  $\text{\LaTeX}$  and describes how to use the software.

**Chapter 4** shows you how to create a very basic document.

**Chapter 5** shows you how to create chapters and other sectional units so that you end up with a fully structured document.

**Chapter 6** shows you how to load packages, and also how to download and install additional packages that weren't installed with your  $\text{\LaTeX}$  distribution.

## 1. INTRODUCTION

**Chapter 7** describes how to create figures and tables.

**Chapter 8** describes how to define your own commands, and redefine existing commands.

**Chapter 9** describes how to typeset mathematics.

**Chapter 10** describes how to define new environments.

**Chapter 11** discusses how numbers are stored in counters, how to change their values, and how to define your own counter.

**Chapter 12** discusses how dimensions are stored, allowed units and how to change a dimension.

**Chapter 13** documents possible errors you may encounter, and gives advice on how to fix them.

Throughout this document there are pointers to related topics in the UK TUG List of Frequently Asked Questions [2]. These are displayed in the margin in square brackets, as illustrated on the right. You may find these links useful in answering related questions that are not covered in this document.

[What is  
LaTeX?]

## 1. INTRODUCTION

This document and associated files are available on-line at: <http://theoval.cmp.uea.ac.uk/~nlct/latex/novices/>. The links in this document are colour-coded: internal links are blue, external links are magenta.

### 1.2 Recommended Reading

This document is designed as an introductory text, not a comprehensive guide. For further reading try some of the following:

*L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System* [3] is the user's guide and reference manual for L<sup>A</sup>T<sub>E</sub>X, and is a good basic text for anyone starting out, however it doesn't cover AMST<sub>E</sub>X, so anyone who needs to typeset more than basic mathematics may prefer either *A Guide to L<sup>A</sup>T<sub>E</sub>X* [4] or *The L<sup>A</sup>T<sub>E</sub>X Companion* [5]. Both these books cover AMST<sub>E</sub>X, BIB<sub>T</sub>E<sub>X</sub> and `makeindex`. *A Guide to L<sup>A</sup>T<sub>E</sub>X* also has an appendix that contains a brief summary of all commands described in the book for a quick and easy reference which is quite useful.

In the same series as *The L<sup>A</sup>T<sub>E</sub>X Companion*, there is also *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion* [6] which details how to illustrate documents with L<sup>A</sup>T<sub>E</sub>X and PostScript, including a chapter on colour (coloured text, background, tables and slides). This is recommended to anyone who is contemplating heavy use of graphics, but you do need a basic knowledge of

[Books on TeX and its relations]

[What are the AMS packages?]



## 1. INTRODUCTION

L<sup>A</sup>T<sub>E</sub>X before delving into it.

The final book in the Companion series which you may find useful is *The L<sup>A</sup>T<sub>E</sub>X Web Companion* [7]. This is recommended for those interested in creating documents for the web, either as HTML or PDF. This book details how to convert L<sup>A</sup>T<sub>E</sub>X documents into HTML using various applications such as **LaTeX2HTML** and **TeX4ht**, and how to create PDF documents using PDFL<sup>A</sup>T<sub>E</sub>X, including how to create active links within your document using the **hyperref** package.

[Drawing with TeX]

[What is PDF<sup>A</sup>TeX?]

There is also a wealth of L<sup>A</sup>T<sub>E</sub>X-related information on the world wide web. The Comprehensive T<sub>E</sub>X Archive Network (CTAN) [8] is a good place to start. In the UK, the UK T<sub>E</sub>X Archive [9] is closer. You can check the **on-line catalogue** for information about available software, and there is also the list of **frequently asked questions** which I recommend you try if you have any queries.

[How to get help]

You can also try using a search engine, such as **Google**. For example, if you get an error message you don't understand, try copying and pasting the message into a search engine.

If that still doesn't answer your question, try the **comp.text.tex** [10] newsgroup or the **texhax** [11] archives, however if you want to post a query to these newsgroups, make sure you structure your question clearly and concisely with an informative subject line.

[Specialist mailing lists]

[How to ask a question]

# Chapter 2

## Some Definitions

As mentioned in [chapter 1](#), L<sup>A</sup>T<sub>E</sub>X is a language, so you can't simply start typing and expect to see your document appear before your very eyes. You need to know a few things before you can get started, so it's best to define a few terms first. Don't worry if there seems a lot to take in, there will be some practical examples later, which should hopefully make things a little clearer.

[Why is TeX not a WYSIWYG system?]

Throughout this document, [source code](#) is illustrated by a typewriter font with the word `Input` placed in the margin, and the corresponding output is typeset with the word `Output` in the margin. For example:

Sample Code:

```
This is an \textbf{example}.
```

Input

Resulting `output`:

This is an **example**.

Output

## 2. SOME DEFINITIONS

Segments of code that are longer than one line are bounded above and below by a horizontal line, illustrated as follows:

```
Line one\par
Line two\par
Line three.
```

↑Input

↓Input

with corresponding output:

```
Line one
Line two
Line three.
```

↑Output

↓Output

**Command** definitions are shown in a typewriter font in the form:

```
\documentclass[<options>]{<class file>}
```

Definition

In this case the command being defined is called `\documentclass` and text typed *<like this>* (e.g. *<options>* and *<class file>*) indicates the type

## 2. SOME DEFINITIONS

of thing you need to substitute. For example, if you want the `article` class file you would substitute `<class file>` with `article` and if you want the `a4paper` option you would substitute `<options>` with `a4paper`, like this:

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

But more on that later.

One other thing to mention is the comment character `%` (the percent symbol). Anything from the percent symbol up to, and including, the end of line character is ignored by `LATEX`. Thus

```
A simple % next comes a command to make some bold text  
\textbf{example}
```

↑ Input

↓ Input

will produce the output

A simple **example**

Output

The percent symbol is often used to suppress unwanted space resulting from line breaks<sup>1</sup> in the [source code](#). For example, the following code

↑ Input

---

<sup>1</sup>`LATEX` treats the end-of-line character as a space.

## 2. SOME DEFINITIONS

Foo%

Bar

\_\_\_\_\_

↓ Input

will produce the output:

FooBar

Output

as opposed to

\_\_\_\_\_

↑ Input

Foo

Bar

\_\_\_\_\_

↓ Input

which will produce the output:

Foo Bar

Output

## 2.1 Source Code

The source code is all the text and  $\text{\LaTeX}$  [commands](#) that make up an entire document. The source code is typed in using a text editor, and saved with the file extension `.tex`. The source code may be contained in just one file, or it might be split across several files.

[TeX-friendly editors and shells]

## 2.2 DVI File (or Output File)

The  $\text{\LaTeX}$  application will convert your [source code](#) into typeset output which will be written to a device independent (DVI) file. This file can then be viewed using a DVI viewer. [MiKTeX](#) comes with the DVI viewer called `YAP`. If you are using the X Window System, the DVI viewer is called `xdvi` (there are also other variants such as `kdvi`).

[What is a DVI file?]

[DVI previewers]

Many people these days use  $\text{PDF}\text{\LaTeX}$  rather than  $\text{\LaTeX}$ , which produces a PDF file, instead of a DVI file. Where this document refers to “the output file”, it means the DVI file if you are using  $\text{\LaTeX}$  and the PDF file if you are using  $\text{PDF}\text{\LaTeX}$ .

[What is PDFTeX?]

## 2.3 Commands (also called macros or control sequences)

A command is used to tell L<sup>A</sup>T<sub>E</sub>X to do a particular thing at that point in the document. There are four basic forms a command can take:

1. A backslash followed by letters. There can be no non-alphabetical characters in the command, apart from the initial backslash. For example `\today` will print the current date, `\twocolumn` will start a new page, and change to a two column format, `\LaTeX` will print the LaTeX logo: L<sup>A</sup>T<sub>E</sub>X. Most L<sup>A</sup>T<sub>E</sub>X commands have fairly self-explanatory names (for example, `\rightarrow` prints an arrow pointing to the right, `\chapter` starts a new chapter). All commands are case-sensitive, so `\gamma` and `\Gamma` have different meanings.

This is the most common form of command. Any spaces immediately following a command of this type are ignored, so for example

```
\TeX nician
```

will produce

T<sub>E</sub>Xnician

[Commands gobble following space]

Input

Output

## 2. SOME DEFINITIONS

whereas

```
\TeX{} nician
```

Input

will produce

```
TEX nician
```

Output

2. A backslash followed by a single non-alphabetical character. For example `\%` will print a percent symbol. Spaces are not ignored after this type of command, for example

```
17.5\% VAT
```

Input

will produce

```
17.5% VAT
```

Output



## 2. SOME DEFINITIONS

3. A special sequence of characters. For example `ffi` is the command to produce the ffi ligature, and the sequence of symbols `!` is the command to produce the upside down exclamation mark `;`
4. An internal command. This is like the first type, but the `@` character appears in the command name (for example `\c@section`) *however* internal commands should only be used in [class files](#) or [packages](#). The `@` symbol takes on a special meaning when a file is included using `\documentclass` (a class file) or `\usepackage` (a package).

[`\@` and `@` in macro names]

For example, in a class file or package `\c@section` is an internal representation of the section [counter](#), whereas in a `.tex` file `\c@section` is interpreted as the command `\c` (the cedilla [accent command](#)) that takes the character `@` as its argument, followed by `section`, which produces the rather odd looking `@section`.

[`\spacefactor` complaints]

Don't be tempted to use internal commands until you have first grasped the basics. You have been warned!

There is one command that you must use in every document you create, and that is the `\documentclass` command. This command must be placed at the very start of your document, and indicates what type of document you are creating. This command takes an [argument](#), and is described in more detail in [chapter 4](#).

## 2.4 Grouping

A segment of code may be grouped by placing it within `{` and `}` (curly braces). Most [commands](#) that occur within a group will be local to that group. For example, `\bfseries` changes the font weight to bold, so the following segment of code:

```
Here is some text. {This text \bfseries is in a
group.} Here is some more text.
```

↑ Input

↓ Input

will appear in the typeset document looking like:

Here is some text. This text **is in a group**. Here is some more text.

Output

As can be seen, the font change only stays in effect until it reaches the end of the group (signified by the closing curly brace `}`).

## 2.5 Arguments (also called parameters)

Some [commands](#) take one or more arguments. This allows you to give L<sup>A</sup>T<sub>E</sub>X additional information, so that it is able to carry out the command. There are two types of arguments: [mandatory](#) and [optional](#).

### 2.5.1 Mandatory Arguments

Mandatory (or compulsory) arguments are arguments that *have* to be specified. Examples:

1. If you want to start a new chapter, you need to use the `\chapter` command, but you also need to tell L<sup>A</sup>T<sub>E</sub>X the title of this new chapter. So the `\chapter` command takes one mandatory argument that specifies the title. For example, the following code:

```
\chapter{Some Definitions}
```

Input

was used to generate the heading for [chapter 2](#) of this document.

2. The command `\textbf` typesets its argument in a bold font (as opposed to the [declaration](#) `\bfseries` which switches to a bold font). For example, the following code:

## 2. SOME DEFINITIONS

```
\textbf{Some bold text.}
```

Input

produces the output

**Some bold text.**

Output

### Note 1:

1.  $\text{\LaTeX}$  takes the first non-space object following the command name as the argument, which is why in the above examples the arguments have to be [grouped](#). Suppose the last example above didn't have a group, so instead the code was:

```
\textbf S some bold text.
```

Input

then only the 'S' would be the argument because it's the first object following the command, in which case the output would look like:

## 2. SOME DEFINITIONS

Some bold text.

Output

2. If you want the argument to be blank, use an empty group: `{}`. For example, suppose you want to have a chapter without a title<sup>2</sup> you would need to do:

```
\chapter{}
```

Input

### 2.5.2 Optional Arguments

Some [commands](#) may have one or more optional arguments. Unlike [mandatory arguments](#), optional arguments must always be enclosed in square brackets [ ]. For example, the command `\` ends a line. So the following segment of code:

```
Line one\\Line two.
```

Input

will produce the following output:

---

<sup>2</sup>The numbers for chapters, sections etc are automatically inserted by L<sup>A</sup>T<sub>E</sub>X, so this example would produce a numbered chapter without a title.

## 2. SOME DEFINITIONS

```
Line one  
Line two.
```

↑Output

↓Output

However the `\` command also has an optional argument that allows you to specify how big the gap between the two lines should be. So the following segment of code:

```
Line one\\[1cm]Line two.
```

Input

will produce the following output:

```
Line one
```

↑Output

```
Line two.
```

↓Output

## 2. SOME DEFINITIONS

Incidentally, note the difference between the previous example, and the following example:

Code:

[Start of line  
goes awry]

```
Line one\\{[1cm]} Line two.
```

Input

Output:

```
Line one  
[1cm] Line two.
```

↑Output

↓Output

In this example the `[1cm]` has been placed inside a group, so it is no longer considered to be an optional argument, and since the command `\\` does not take a mandatory argument, the `[1cm]` is simply interpreted as ordinary text.

Here's another example: The command `\framebox` takes a **mandatory argument** and an optional argument. `\framebox` puts a frame around the contents of its mandatory argument:

Code:

```
\framebox{Some Text}
```

Input

## 2. SOME DEFINITIONS

Output:

Some Text

Output

The optional argument can be used to make the box a specified width:

Code:

```
\framebox[4cm]{Some Text}
```

Input

Output:

Some Text

Output

And there's a second optional argument that specifies the justification of the text (left, right or centred) within the box:

Code:

```
\framebox[4cm][r]{Some Text}
```

Input

Output:

Some Text

Output



## 2. SOME DEFINITIONS

In general, if a command has both optional and mandatory arguments, the optional arguments are usually specified first (although there are a few exceptions).

### 2.6 Moving Arguments and Fragile Commands

Certain types of [commands](#), called fragile commands, can really mess things up when they are used in what is termed a moving argument. These types of [argument](#) are generally those whose contents are copied to another part of the document. For example, section headings appear at the start of a section, but they can also appear in the table of contents. The `\footnote` command is a fragile command, so

```
\section{A heading\footnote{with a footnote}}
```

Input

will cause an error.

[An extra '']??

If there is no other command to use in its place, you should use `\protect` immediately before the fragile command:

```
\section{A heading\protect\footnote{with a footnote}}
```

Input

## 2. SOME DEFINITIONS

This, however, is a contrived example, as it isn't a good idea to have a footnote in a section heading, as it will also end up in the table of contents, and possibly in page headings.

[Footnotes in  
LaTeX section  
headings]

### 2.7 Robust Commands

A robust command is a command that is not a [fragile command](#).

### 2.8 Short and Long Commands

A short [command](#) is a command whose [argument](#) may not contain a paragraph break (either as a blank line or using `\par`). This is the standard behaviour for commands defined using TeX's `\def` command.

Conversely, a long [command](#) is a command whose [argument](#) may contain a paragraph break. Using short commands helps to test for forgotten braces, so it is recommended that when you [define a new command](#), you should always make the command a short command, unless there is a chance that the argument may need to contain a paragraph break.

## 2.9 Declarations

The term declaration is used to refer to a [command](#) that affects the document from that point onwards. The declaration itself does not produce any text, and its effect can be localised by placing the declaration within a [group](#). For example, `\bfseries` is a declaration that switches the current font weight to bold:

```
Here is some normal text.  
\bfseries Here is some bold text.
```

↑Input

↓Input

will appear in the typeset document looking like:

Here is some normal text. **Here is some bold text.**

Output

## 2.10 Environments

An environment is a block of code contained within the [commands](#)

## 2. SOME DEFINITIONS

```
\begin{<env-name>}
```

Definition

and

```
\end{<env-name>}
```

Definition

where *<env-name>* is the name of the environment. The block of code is then formatted in a method specific to that environment. For example, the `bfseries`<sup>3</sup> environment will typeset the contents of the environment in a bold font. The following code:

```
\begin{bfseries}
Here is some bold text.
\end{bfseries}
```

↑ Input

↓ Input

will appear in the typeset document looking like:

**Here is some bold text.**

Output

---

<sup>3</sup>note there is no backslash in the environment name

## 2. SOME DEFINITIONS

Some environments also supply **commands** that may only be used within that environment. For example, the `itemize` environment provides a command called `\item` so that you can specify individual items within an unordered list. Example:

```
Shopping List:  
\begin{itemize}  
\item Cabbages  
\item Bananas  
\item Apples  
\end{itemize}
```

↑Input

↓Input

will produce the following output:

- ```
Shopping List:  
• Cabbages  
• Bananas
```

↑Output

## 2. SOME DEFINITIONS

- Apples

[↓Output](#)

### 2.11 Preamble

The preamble is the part of the [source code](#) that comes between the `\documentclass` [command](#) and `\begin{document}` (the start of the [document environment](#)). Only a few special commands may be placed in the preamble, and there are a few special commands that may only go in the preamble.

```
\documentclass{...}
```

← This bit in here is the preamble.

```
\begin{document}
```

### 2.12 Class File

The class file (`.cls`) defines the page layout, heading styles and various [commands](#) and [environments](#) needed for a particular style of document.

## 2. SOME DEFINITIONS

The class file is specified using the command

```
\documentclass[<options>]{<class-name>}
```

Definition

where *<class-name>* is the name of the file without the `.cls` extension. All  $\text{\LaTeX}$  documents must start with this command. The basic class files are: `article`, `report`, `book` and `letter`, but there are many others available.

[Replacing the standard classes]

# Chapter 3

## From Source Code to Typeset Output

Every time you want to create or edit a  $\text{\LaTeX}$  document, there are three basic steps you will always need to follow:

1. Write or edit the [source code](#)
2. Pass the source code to the  $\text{\LaTeX}$  application (“ $\text{\LaTeX}$  the document”)
  - If there are any error messages, return to [step 1](#)
  - If there are no error messages, a [DVI file](#) is created.
3. View the [DVI file](#) to check the result. If you need to modify your document, go back to [step 1](#).

You will therefore need:

1. A text editor or front-end (to perform [step 1](#)), see below.



### 3. FROM SOURCE CODE TO TYPESET OUTPUT

2. The  $\text{T}_{\text{E}}\text{X}/\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  installation (to perform [step 2](#)). If you don't already have  $\text{T}_{\text{E}}\text{X}$  installed on your machine, you will need to download and install the relevant software. This will depend on which operating system you are using. See <http://www.ctan.org/starter.html> for more information, and for up-to-date links. At the time of writing, the main distributions are:

[(La)TeX for different machines]

[The TeX collection]

**Windows:** [proTeXt](#) is easy to install and is based on the [MiKTeX](#) distribution. It also includes [TeXnicCenter](#), [ghostscript](#) and [GSview](#) (see below).

**UNIX-type systems:** a popular choice is [teTeX](#), however it is likely that the  $\text{T}_{\text{E}}\text{X}$  distribution may already be installed. Some UNIX systems that have  $\text{T}_{\text{E}}\text{X}$  installed may require an additional file to be placed in your home directory or designated hidden directory for you to be able to use the software. You can check to see if  $\text{T}_{\text{E}}\text{X}$  is installed using the command:

```
which tex
```

If you get the response `tex: Command not found`, then contact your system administrator.

**Machintosh:** [MacTeX](#) is a complete TeX system for Mac OS X, see <http://tug.org/mactex> for further details.

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

For other operating systems, look in the [tex-archive/systems](#) directory of the UK T<sub>E</sub>X Archive [9].

3. A DVI viewer (to perform [step 3](#)). The T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X installation should come with a DVI viewer. It is also possible to convert your DVI file into Adobe's [Portable Document Format](#) (PDF) or [PostScript](#), in which case you will need an application that can read those formats.

[DVI  
previewers]

[DVI to  
PostScript  
conversion  
programs]

**PostScript:** PostScript files can be viewed using `ghostscript` (and related applications `ghostview`, `GSview` and `MacGSview`). These are available on a number of operating systems, and can be obtained from <http://www.cs.wisc.edu/~ghost/>.

**PDF:** PDF files can be viewed using [Adobe Reader](#). There are also other PDF viewers such as `xpdf` and `kpdf`. The `ghostscript` family can also view PDF files, but any links in the document will be inactive.

By converting your output to PostScript or PDF, you can enhance the functionality of L<sup>A</sup>T<sub>E</sub>X allowing you to perform operations such as rotating text (see [section 6.1.1](#) for further details). If you use PDFL<sup>A</sup>T<sub>E</sub>X to generate a PDF document, you can also create active

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

links (see *The L<sup>A</sup>T<sub>E</sub>X Web Companion* [7] for more information, or if you'd rather a brief on-line introduction you can try *Creating a PDF Document using PDFLaTeX*).

Documented below are instructions of how to use L<sup>A</sup>T<sub>E</sub>X using a terminal+text editor approach and two different front-end approaches. If you are using Windows, I strongly recommend that you use a front-end. If you have used **proTeXt** to install the T<sub>E</sub>X distribution, you should already have a copy of **TeXnicCenter** installed.

## 3.1 Text Editor and Terminal Approach

Creating a L<sup>A</sup>T<sub>E</sub>X document using a text editor and a terminal is an approach often favoured by UNIX-type users. If you have never used a terminal (i.e. you have only ever used point-and-click menu driven applications) then you will be better off using a front-end, in which case I suggest you turn to Sections 3.2 and 3.3 which describe **TeXnicCenter** and **WinEdt**, respectively.

To begin, you will first need a text editor. There are a number available that are suited to using with L<sup>A</sup>T<sub>E</sub>X, some people advocate **Emacs**, others advocate **vim**, and there are various others such as **NEdit**. I prefer to use **vim**—I'm not overly keen on using the mouse, and I prefer being able to

[TeX-friendly editors and shells]

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

issue all commands via the keyboard (although there is a GUI version of `vim`). As with some other editors, it comes with syntax highlighting, regular expression search and replace, auto-insertion, and a brace matching mechanism which I find useful. If you are using version 7 of `vim`, there is an integrated spell checker, otherwise there is a spell checker plug-in called `vimspell`, so you can check your spelling as you type. If there is already a text editor that you are comfortable with, then stick with that, otherwise try out available editors, and decide which one you prefer.

When using the terminal and text editor approach, some people like to have at least two terminals open: one to run the editor, the other to run `LATEX`. This means that you don't have to keep quitting the editor every time you want to [L<sup>A</sup>T<sub>E</sub>X your document](#). Some editors allow you to run commands, but personally I don't like to use this approach. If your editor has a GUI interface, then you'll probably only need one terminal open.

Let's get started: start up your text editor. This is usually done by entering the name of the editor at the command prompt in your terminal. With most editors you can also specify the filename as well. If the file doesn't exist, a new one will be created when you save your document. [Figure 3.1](#) shows my terminal. The command prompt looks like `[nlct@nlct1tpc examples]$`. It will be different for your system. I have typed `vim sample1.tex` at the command prompt. This will start `vim` with

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

a new file called `sample1.tex`. (In this section, I will be using `vim` as the text editor, if you are not using `vim`, then substitute the editor of your choice.)

Once I have pressed the return key, my terminal looks like [figure 3.2](#). Normally `vim` starts in visual command mode, which means that when you start typing text, it will be interpreted as part of a command. In order to type text into your file, you will need to enter input mode. There are a number of ways of doing this, but pressing `i` will do for now.<sup>1</sup> [Figure 3.3](#) shows how my terminal looks when I am in input mode. I can now go ahead and type in my text ([figure 3.4](#)). To go back to the visual command mode, press the escape key (`Esc`). Now that you are back in the visual command mode, you can save your document, either using the command `:w` if you have already given your file a name, or `:w <filename>` (e.g. `:w sample1.tex`) if you started `vim` without specifying a file, see [figure 3.5](#). When you want to quit `vim` you can do `:wq` to save and quit or `:q!` to quit without saving, but I suggest you don't do this just yet if you have another terminal available.

[Step 1](#) is now complete, and you are now ready to move on to [step 2](#): using `LATEX`. Go to your other terminal (or quit your editor if you only have access to one terminal) and make sure that you are in the same

---

<sup>1</sup>For a complete set of available commands, see the `vim` manual at <http://vimdoc.sourceforge.net/>.

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

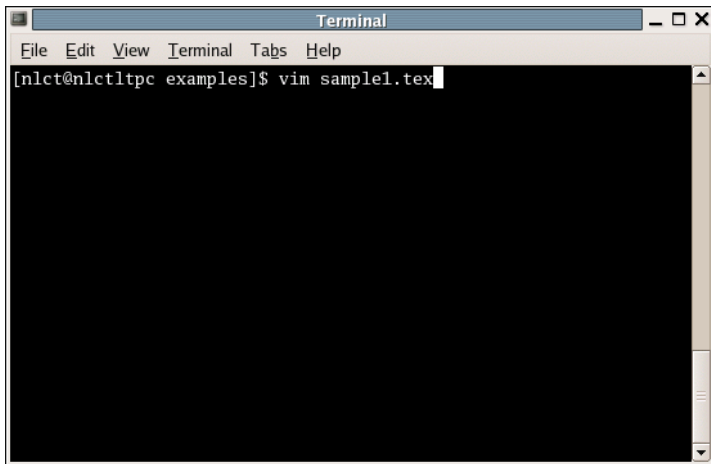


Figure 3.1: Starting vim from a terminal

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

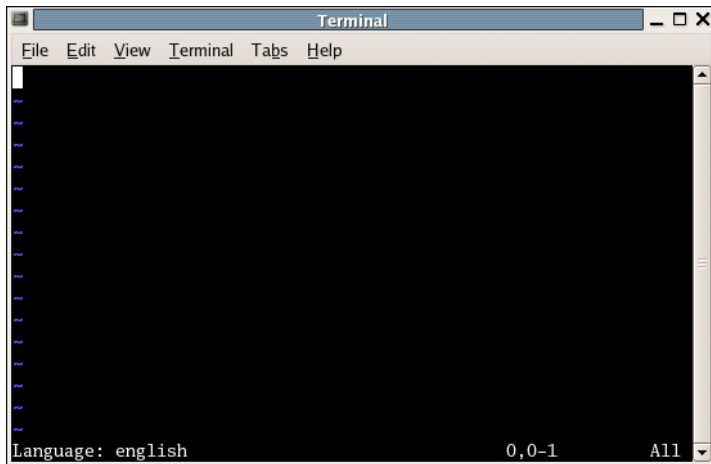


Figure 3.2: Starting a new file in vim

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

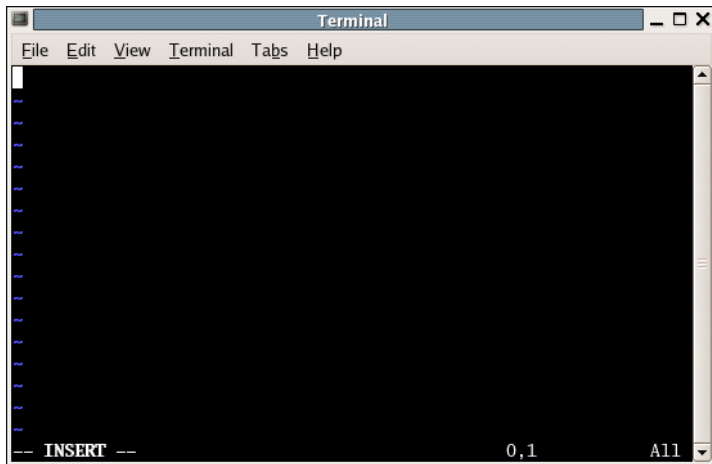
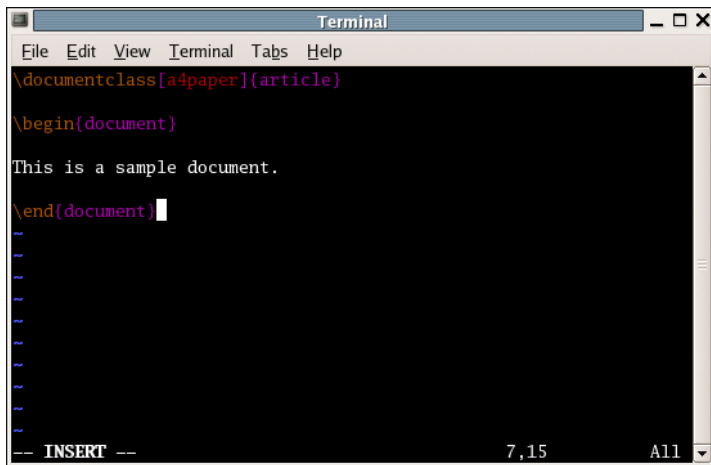


Figure 3.3: Input mode in vim



### 3. FROM SOURCE CODE TO TYPESET OUTPUT



The image shows a terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content is as follows:

```
\documentclass[a4paper]{article}

\begin{document}

This is a sample document.

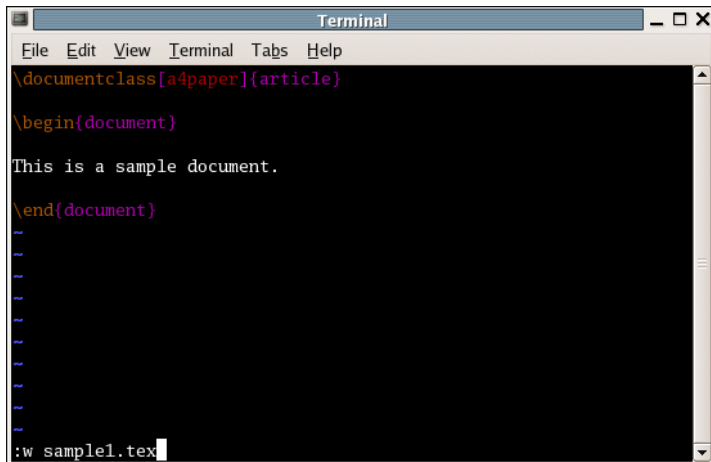
\end{document}

```

At the bottom of the terminal, the status bar displays "INSERT" on the left, "7,15" in the center, and "All" on the right.

Figure 3.4: Creating a sample document in vim

### 3. FROM SOURCE CODE TO TYPESET OUTPUT



```
Terminal
File Edit View Terminal Tabs Help
\documentclass[a4paper]{article}

\begin{document}

This is a sample document.

\end{document}

:w sample1.tex
```

Figure 3.5: Saving your document in vim (the file name should be omitted if the file already has a name)

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

directory as the file you just created. Typing `ls` at the command prompt will list the contents of your current directory. If you do this, you should see the file that you have just created (see [figure 3.6](#)). At the command prompt type:

```
latex sample1.tex
```

as shown in [figure 3.7](#). You can omit the `.tex` extension if you like,  $\text{\LaTeX}$  will automatically add this if it has been omitted. If you prefer to use  $\text{\PDFLaTeX}$ , type

```
pdflatex sample1.tex
```

instead (again the `.tex` extension may be omitted). You should now see something like [figure 3.8](#).

Numbers appearing in square brackets, e.g. [1], indicate which page  $\text{\LaTeX}$  is currently processing. In this case, there is only one page. The last line to appear on screen indicates that information about this  $\text{\LaTeX}$  run has been written to the log file `sample1.log`, which you can look at using your text editor.

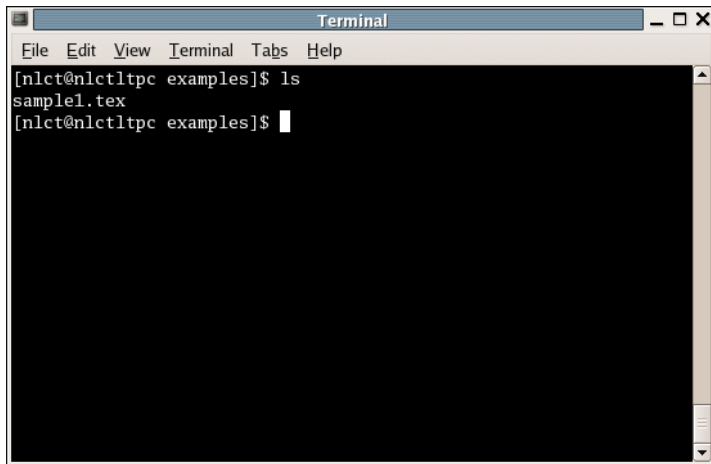
The most important thing to note is the penultimate line:<sup>2</sup>

Output written on `sample1.dvi` (1 page, 248 bytes).

---

<sup>2</sup>if you are using  $\text{\PDFLaTeX}$ , it will have `sample1.pdf` instead of `sample1.dvi`

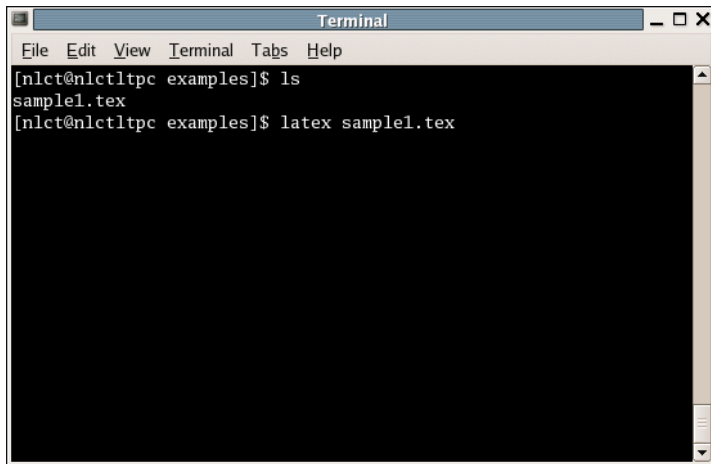
### 3. FROM SOURCE CODE TO TYPESET OUTPUT

A screenshot of a terminal window titled "Terminal". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows a prompt "[nlct@nlctltpc examples]" followed by the command "ls". The output of the command is "sample1.tex". The prompt is followed by a cursor. The terminal background is black, and the text is white. There are standard window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
[nlct@nlctltpc examples]$ ls
sample1.tex
[nlct@nlctltpc examples]$
```

Figure 3.6: Listing the contents of the current directory

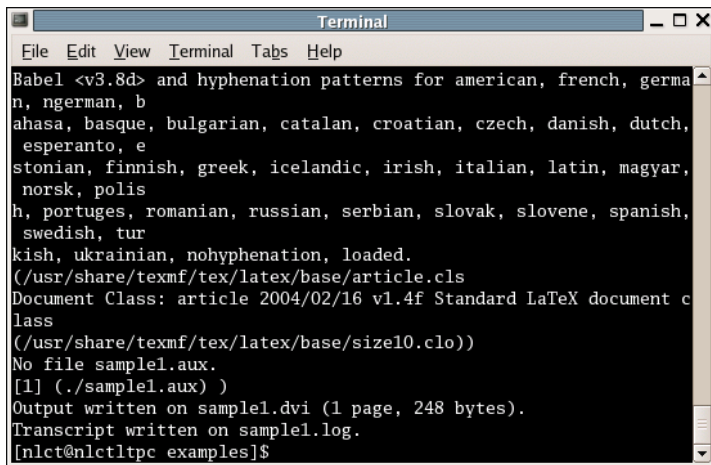
### 3. FROM SOURCE CODE TO TYPESET OUTPUT

A terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows a shell prompt "[nlct@nlctltpc examples]" followed by the command "ls" and its output "sample1.tex". A second shell prompt "[nlct@nlctltpc examples]" is followed by the command "latex sample1.tex".

```
[nlct@nlctltpc examples]$ ls
sample1.tex
[nlct@nlctltpc examples]$ latex sample1.tex
```

Figure 3.7: Running L<sup>A</sup>T<sub>E</sub>X

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

A terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal output shows the execution of a LaTeX document class, listing supported languages and hyphenation patterns, and reporting the successful compilation of a document into a DVI file.

```
Terminal
File Edit View Terminal Tabs Help
Babel <v3.8d> and hyphenation patterns for american, french, german,
n, ngerman, b
ahasa, basque, bulgarian, catalan, croatian, czech, danish, dutch,
esperanto, e
stonian, finnish, greek, icelandic, irish, italian, latin, magyar,
norsk, polis
h, portuges, romanian, russian, serbian, slovak, slovene, spanish,
swedish, tur
kish, ukrainian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document c
lass
(/usr/share/texmf/tex/latex/base/size10.clo))
No file sample1.aux.
[1] (./sample1.aux) )
Output written on sample1.dvi (1 page, 248 bytes).
Transcript written on sample1.log.
[nlct@nlctltpc examples]$
```

Figure 3.8: Running  $\text{\LaTeX}$

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

This means that the document has been successfully created, and is one page long.

If you have made a mistake in your [source code](#), for example suppose you have missed the starting backslash in `\documentclass`, then the output will look something like:

```
! LaTeX Error: Missing \begin{document}.
```

See the LaTeX manual or LaTeX Companion for explanation.

```
Type H <return> for immediate help.
```

```
...
```

```
1.1 d
```

```
documentclass[a4paper]{article}
```

```
?
```

There are several things you can do at this point, but the easiest thing to do is to exit L<sup>A</sup>T<sub>E</sub>X by typing `X` followed by the return key. Go back to your editor, fix the mistake, save the document, and then try again. (If you do get an error message, check the list of common errors in [chapter 13](#).) Note that it is important to always save your document before running L<sup>A</sup>T<sub>E</sub>X.

You can view the typeset output by loading the file `sample1.dvi` into

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

a DVI viewer, such as `xdvi` or `kdvi`. To do this type

```
xdvi sample1.dvi
```

or

```
kdvi sample1.dvi
```

at the command prompt (see [figure 3.9](#)). You will then see the final output, as shown in [figure 3.10](#).

If you have used PDF $\LaTeX$  instead of  $\LaTeX$ , you should have a file called `sample1.pdf` instead of `sample1.dvi`. You can view this using a PDF viewer, such as `acroread` or `kpdf`.

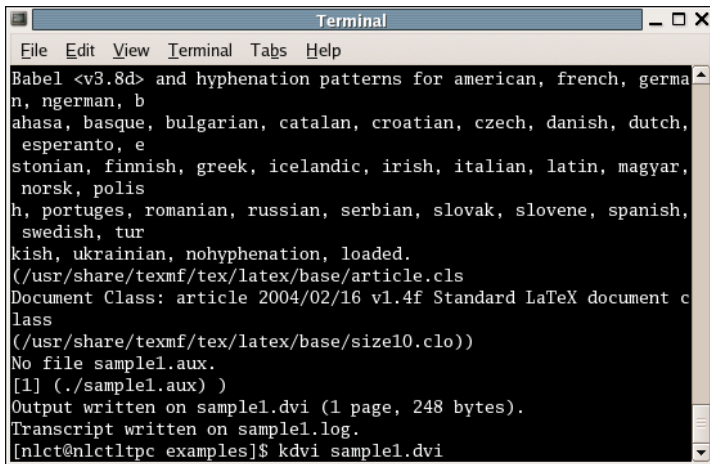
Some viewers, such as `kdvi` and `kpdf` will automatically reload the file whenever it is modified, in which case you may like to keep the viewer open, and as you keep editing and  $\LaTeX$ ing your document, the viewer will automatically reload the new versions. Some viewers, such as `xpdf` don't automatically reload, but have a reload facility, which you can use whenever you  [\$\LaTeX\$  your document](#).

If you like, you can convert your DVI file to PostScript using `dvips`. To do this, type the following at the command prompt in your terminal:

```
dvips -o sample1.ps sample1.dvi
```



### 3. FROM SOURCE CODE TO TYPESET OUTPUT

A terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal text shows the output of a LaTeX compilation process. It lists various languages supported by Babel, including American, French, German, and many others. It then shows the loading of the article class and size10 font, followed by the execution of the 'kdvi' command to view the sample1.dvi file.

```
Terminal
File Edit View Terminal Tabs Help
Babel <v3.8d> and hyphenation patterns for american, french, germa
n, ngerman, b
ahasa, basque, bulgarian, catalan, croatian, czech, danish, dutch,
esperanto, e
stonian, finnish, greek, icelandic, irish, italian, latin, magyar,
norsk, polis
h, portuges, romanian, russian, serbian, slovak, slovene, spanish,
swedish, tur
kish, ukrainian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document c
lass
(/usr/share/texmf/tex/latex/base/size10.clo))
No file sample1.aux.
[1] (./sample1.aux) )
Output written on sample1.dvi (1 page, 248 bytes).
Transcript written on sample1.log.
[nlct@nlctltpc examples]$ kdvi sample1.dvi
```

Figure 3.9: Load a DVI file into a DVI viewer

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

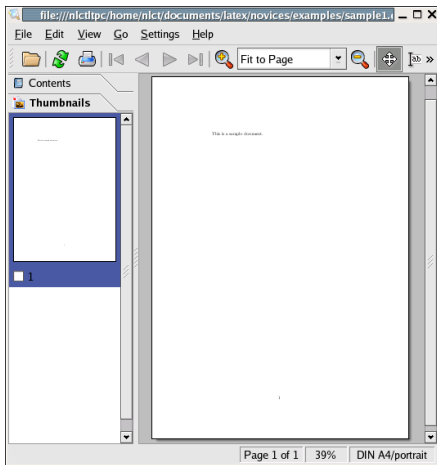


Figure 3.10: Viewing a DVI file in kdvi

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

(The `.dvi` extension may be omitted.) You can then view the PostScript file using `ghostscript` or one of its associated applications, such as `ghostview`. I have `kghostview` installed on my laptop, so to view the PostScript file, `sample1.ps`, I would need to type:

```
kghostview sample1.ps
```

(See [figure 3.11](#).)

## 3.2 TeXnicCenter

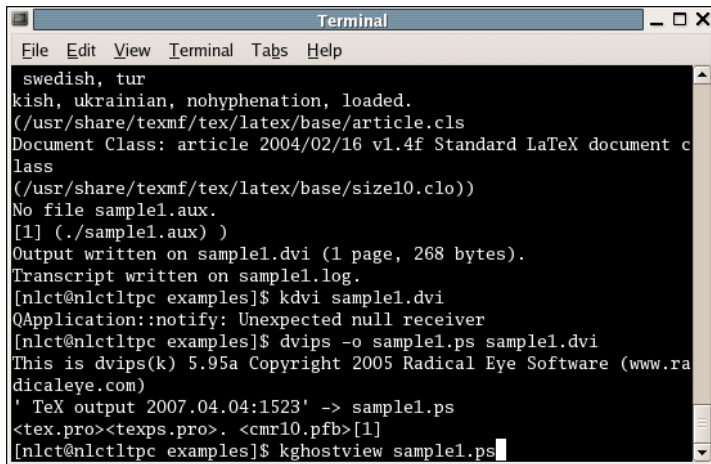
`TeXnicCenter` is an application that enables you to edit  $\text{\LaTeX}$  source code, and simply click on a button to pass the source code to  $\text{\LaTeX}$ , and then click on another button to view the resulting typeset document. Many people prefer this approach to the text editor and terminal approach described in the previous section. This section gives a brief overview of `TeXnicCenter`, however it has been several years since I last used it,<sup>3</sup> so this information may be dated.

`TeXnicCenter` is free and can be downloaded from the UK  $\text{\TeX}$  Archive [9] in the [systems/win32/TeXnicCenter/](#) directory or from <http://www.toolscenter>.

---

<sup>3</sup>I used to use `TeXnicCenter` and `MiKTeX` when I was teaching  $\text{\LaTeX}$ , but that was my limit of using  $\text{\TeX}$  under Windows

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

A terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal output shows the compilation of a LaTeX document. It starts with "swedish, turkish, ukrainian, nohyphenation, loaded." followed by the loading of "article.cls" and "size10.clo". It reports "Document Class: article 2004/02/16 v1.4f Standard LaTeX document class" and "No file sample1.aux.". Then it shows "[1] (./sample1.aux) )", "Output written on sample1.dvi (1 page, 268 bytes).", and "Transcript written on sample1.log.". The user enters "\$ kdvi sample1.dvi" and receives a notification: "Application:notify: Unexpected null receiver". Then the user enters "\$ dvips -o sample1.ps sample1.dvi" and receives the output: "This is dvips(k) 5.95a Copyright 2005 Radical Eye Software (www.radicaleye.com)", "' TeX output 2007.04.04:1523' -> sample1.ps", and "<tex.pro><texps.pro>. <cmr10.pfb>[1]". Finally, the user enters "\$ kghostview sample1.ps" and the cursor is at the end of the line.

```
swedish, tur
kish, ukrainian, nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 2004/02/16 v1.4f Standard LaTeX document c
lass
(/usr/share/texmf/tex/latex/base/size10.clo))
No file sample1.aux.
[1] (./sample1.aux) )
Output written on sample1.dvi (1 page, 268 bytes).
Transcript written on sample1.log.
[nlct@nlctltpc examples]$ kdvi sample1.dvi
Application:notify: Unexpected null receiver
[nlct@nlctltpc examples]$ dvips -o sample1.ps sample1.dvi
This is dvips(k) 5.95a Copyright 2005 Radical Eye Software (www.ra
dicaleye.com)
' TeX output 2007.04.04:1523' -> sample1.ps
<tex.pro><texps.pro>. <cmr10.pfb>[1]
[nlct@nlctltpc examples]$ kghostview sample1.ps
```

Figure 3.11: Loading a PostScript file

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

org/. Note that you must have a [T<sub>E</sub>X/L<sub>A</sub>T<sub>E</sub>X distribution](#) installed before you install **TeXnicCenter**. If you installed **proTeXt**, you should already have **TeXnicCenter** installed. If you have any problems with installing or running **TeXnicCenter**, go to their help page at <http://www.texniccenter.org/help.html>.

Once the installation is complete, you can then run **TeXnicCenter** from the Start Menu:

Start → Programs → TeXnicCenter → TeXnicCenter

Firstly you should see the tip of the day window ([figure 3.12](#)).

You can close this window, and then, if this is the first time you are using **TeXnicCenter** you will have to use the configuration wizard to set up **TeXnicCenter** correctly. I recommend that you choose the default settings. (Select Next, Next and then Finish.)

Now you are ready to use **TeXnicCenter**. It should look like [figure 3.16](#).

To start a new project select **File** → **New Project**. This will open the window shown in [figure 3.17](#).

Enter a name for your project, and specify the directory where you want to save your work. For example, I shall call my project “example” and I want to save it in `c:\My Documents\Nicky\example` (see [figure 3.18](#)).

Select the empty project icon, and click on the button labelled “OK”. You should now see something like [figure 3.19](#).

### 3. FROM SOURCE CODE TO TYPESET OUTPUT



Figure 3.12: TeXnicCenter Tip of the Day Window

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

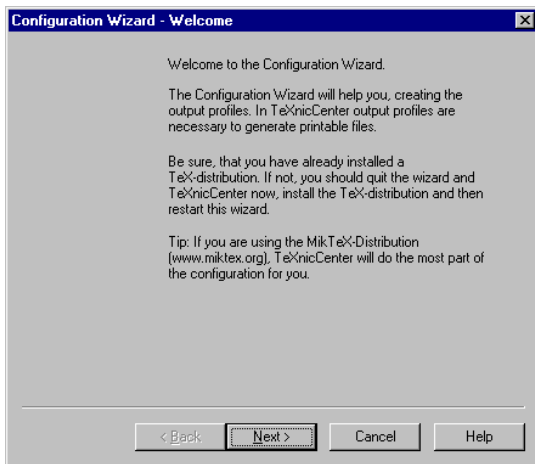


Figure 3.13: TeXnicCenter Configuration Wizard

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

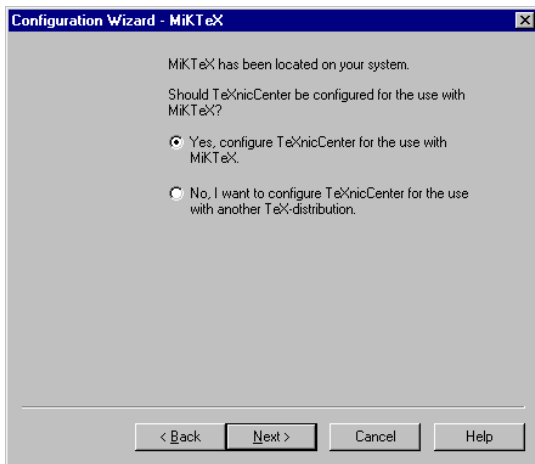


Figure 3.14: TeXnicCenter Configuration Wizard



### 3. FROM SOURCE CODE TO TYPESET OUTPUT



Figure 3.15: TeXnicCenter Configuration Wizard

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

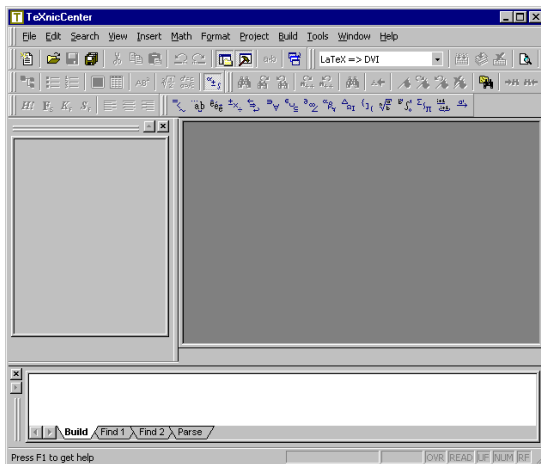


Figure 3.16: TeXnicCenter

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

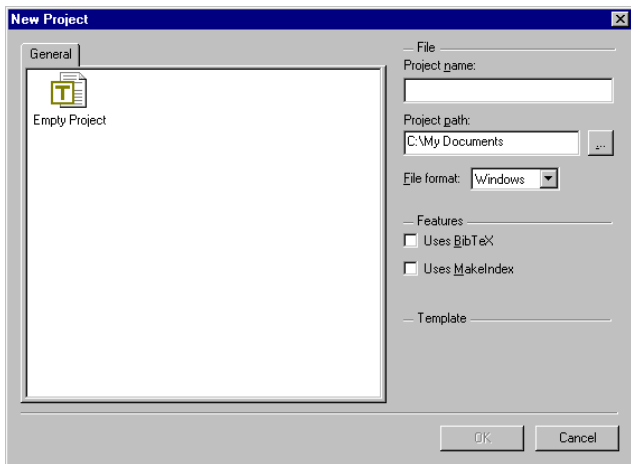


Figure 3.17: New Project Dialog Box

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

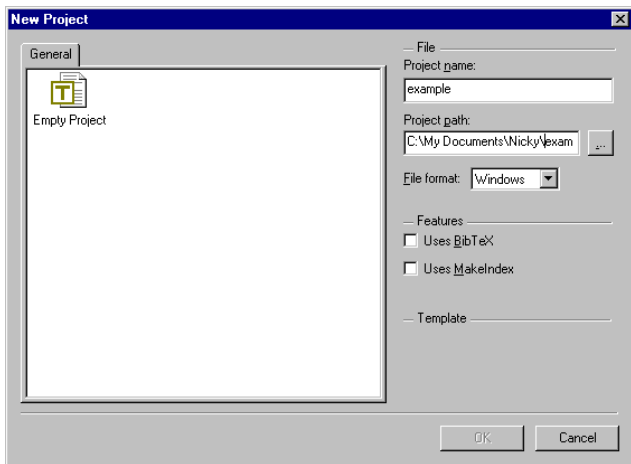


Figure 3.18: New Project Dialog Box

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

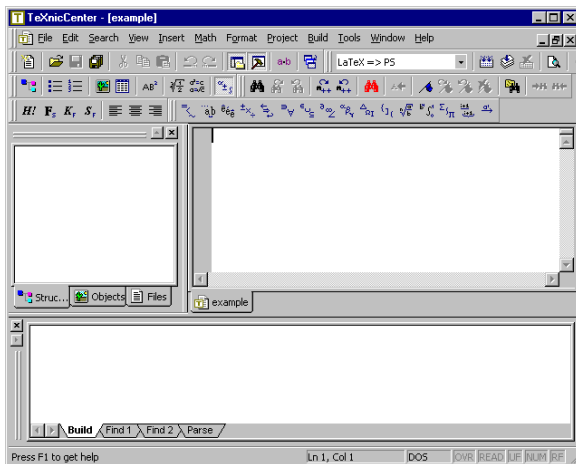


Figure 3.19: TeXnicCenter — New Project Started

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

You can now start typing the source code (we'll cover this [later](#)). See [figure 3.20](#).



Save it by either clicking on the save icon or select File → Save

Now select what type of output you want (DVI, PDF or PostScript) see [figure 3.21](#). If this box is blank, then it's possible that you didn't complete all the steps in the configuration wizard described above.



Now click on the build output icon or select Build → Build Output. The transcript will be written in the window at the bottom (see [figure 3.22](#)). This transcript should be the same as described on page [44](#) onwards. If you have selected LaTeX => PDF, then TeXnicCenter will use PDFL<sup>A</sup>T<sub>E</sub>X instead of L<sup>A</sup>T<sub>E</sub>X. If you have selected LaTeX => PS, then TeXnicCenter will use L<sup>A</sup>T<sub>E</sub>X followed by dvips (as in [figure 3.22](#)). The dvips messages will follow on from the L<sup>A</sup>T<sub>E</sub>X messages. (If you selected the BibTeX or MakeIndex features when you initialised the project, [figure 3.18](#), then TeXnicCenter will also use the BIB<sub>T</sub>E<sub>X</sub> and MakeIndex applications.)



To view the document, click the view output button. (Note that if you have selected LaTeX => PDF or LaTeX => PS you will need Adobe

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

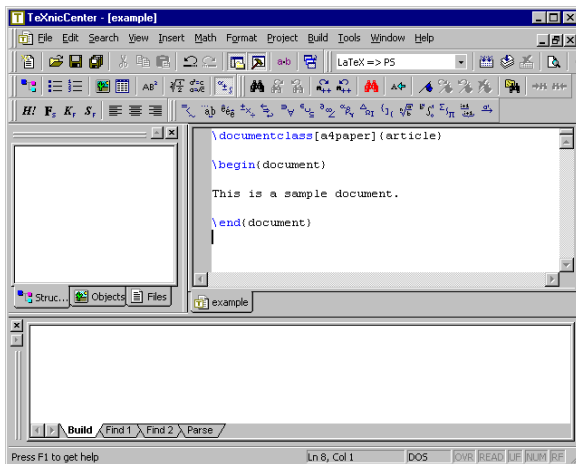


Figure 3.20: TeXnicCenter — Typing in Source Code

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

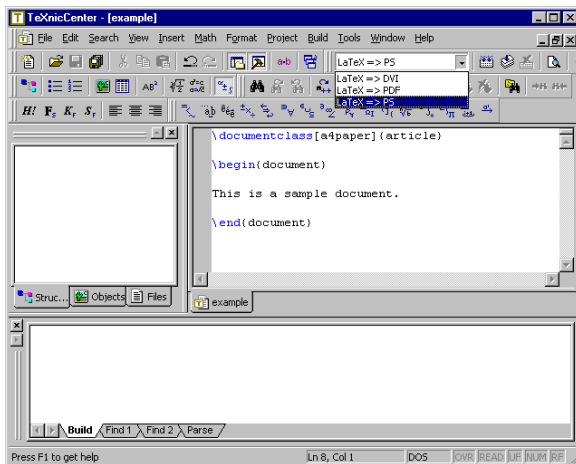


Figure 3.21: TeXnicCenter — Selecting Output Type



### 3. FROM SOURCE CODE TO TYPESET OUTPUT

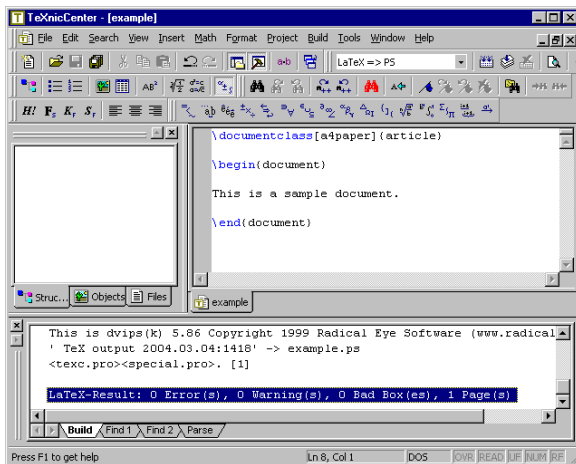


Figure 3.22: TeXnicCenter (using L<sup>A</sup>T<sub>E</sub>X and dvips)

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

Reader or GSview, respectively, to view the output file.)

If there are any errors, you can select **Build** → **Next Error** and it will show you where the error has occurred (see [figure 3.23](#)). If you do have any errors, check [chapter 13](#).

## 3.3 WinEdt

WinEdt (not to be confused with WinEdit which is a completely different application) is an application that enables you to edit L<sup>A</sup>T<sub>E</sub>X source code, and simply click on a button to pass the source code to L<sup>A</sup>T<sub>E</sub>X, and then click on another button to view the resulting typeset document. This section gives a brief overview of WinEdt, however it has been several years since I tried it, so this information may be dated.

WinEdt is shareware: it can be downloaded from the UK T<sub>E</sub>X Archive [9] in the [systems/win32/winedt](#) directory or from <http://www.winedt.com/> and evaluated for a trial period of 31 days, after which, if you want to continue to use it, you must pay the registration fee. Details of prices and types of licence available can be found at <http://www.winedt.com/>.

Again, you must have a T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X distribution installed before you start. WinEdt is fairly easy to install. First unpack all the files, and then run the `setup.exe` application. I recommend that you use the default settings. If you have any problems installing or using WinEdt, go to [http:](#)

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

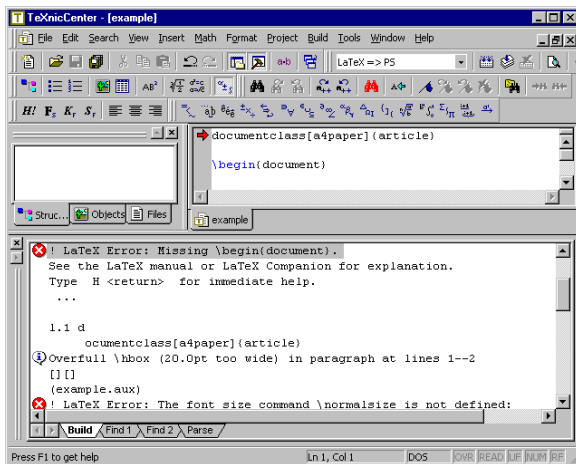


Figure 3.23: TeXnicCenter — Showing Error

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

`//www.winedt.com/support.html`.


To run WinEdt, select WinEdt from the start menu:


Start → Programs → WinEdt → WinEdt

It should look something like [figure 3.24](#).

Click on the new document button or select File → New. You can now start typing your [source code](#) into the WinEdt window, as shown in [figure 3.25](#).

You can now save your document using the File → Save as menu. Select the file type to be TeX, and type in the name of your file, e.g. `sample1.tex` (see [figure 3.26](#)).

To L<sup>A</sup>T<sub>E</sub>X your document, simply click on the L<sup>A</sup>T<sub>E</sub>X button . The output will appear in an MSDOS Prompt window (see [figure 3.27](#)).

To view your typeset document, click on the view DVI button . You can convert your DVI file to PostScript by clicking the dvips but-

ton . If you have GSview installed, you can then view the PostScript

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

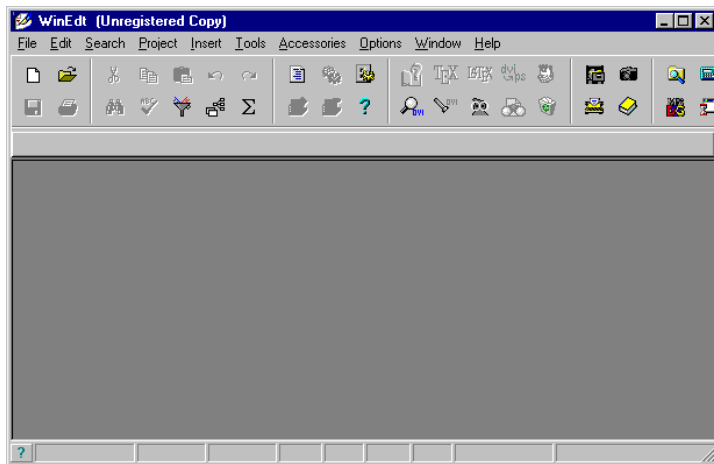


Figure 3.24: WinEdt

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

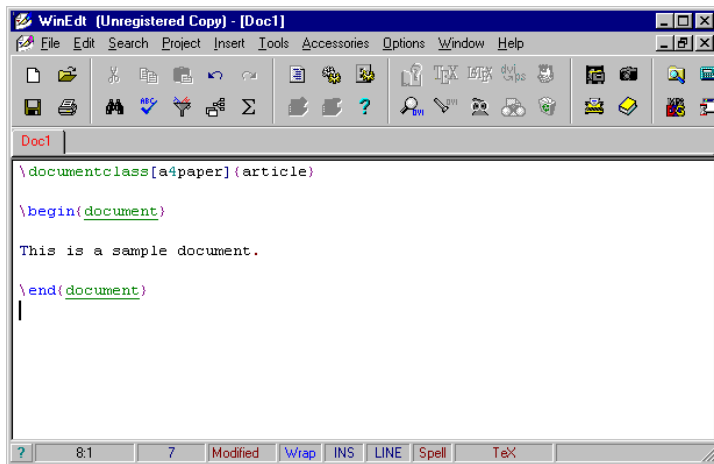


Figure 3.25: WinEdt

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

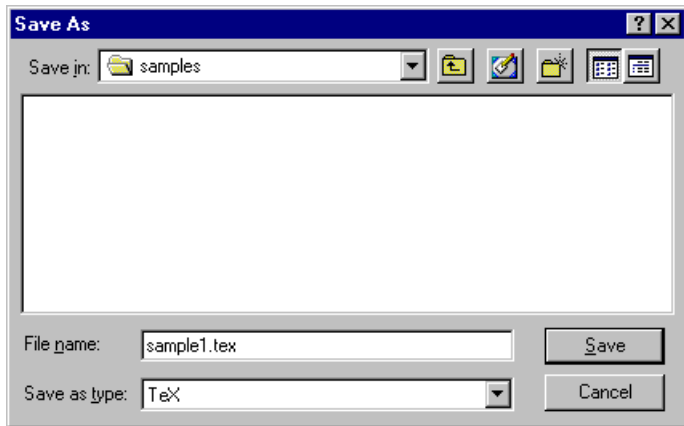
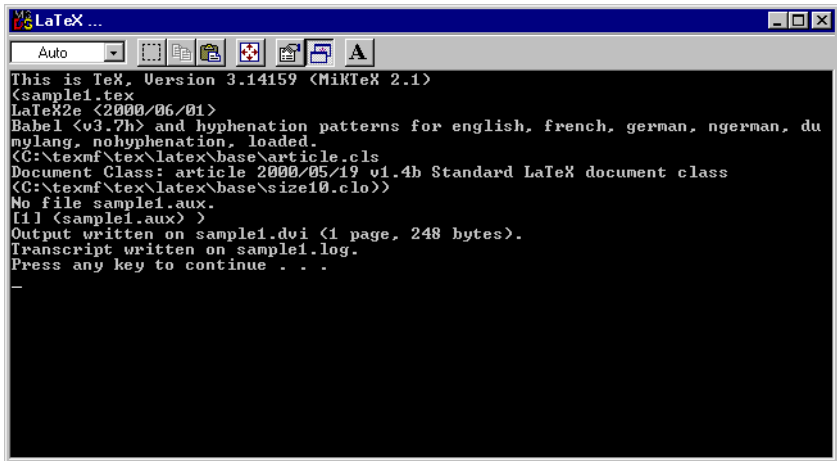


Figure 3.26: WinEdt — Saving the File

### 3. FROM SOURCE CODE TO TYPESET OUTPUT

A screenshot of the WinEdt software interface. The window title is "LaTeX ...". The menu bar includes "Auto" and several icons for file operations. The main text area displays the following LaTeX output:

```
This is TeX, Version 3.14159 (MiKTeX 2.1)
(sample1.tex
LaTeX2e <2000/06/01>
Babel <v3.7h> and hyphenation patterns for english, french, german, ngerman, du
mylang, nohyphenation, loaded.
(C:\texmf\tex\latex\base\article.cls
Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
(C:\texmf\tex\latex\base\size10.clo))
No file sample1.aux.
[1] (sample1.aux) >
Output written on sample1.dvi (1 page, 248 bytes).
Transcript written on sample1.log.
Press any key to continue . . .
```

Figure 3.27: WinEdt —  $\text{\LaTeX}$  Output



### 3. FROM SOURCE CODE TO TYPESET OUTPUT



file by clicking on the GSview button.

Depending on which version of WinEdt you have installed, there may also be a PDF $\LaTeX$  button which you can click on to create a Portable Document Format (`.pdf`) document. If not, you can click on the MSDOS



button to open up an MS-DOS Prompt window, and type `pdflatex` followed by the filename. For example:

```
pdflatex sample1.tex
```

Note that if the filename contains a space, you will need to use double quotes:

```
pdflatex "my file.tex"
```

# Chapter 4

## Creating a Simple Document

Having installed and tested the software, let's now look at how to actually write the [source code](#). The very first line of any document that you create must have the [command](#):

```
\documentclass[<option-list>]{<class-name>}
```

Definition

This tells L<sup>A</sup>T<sub>E</sub>X what type of document you want to create (e.g. an article, a technical report, correspondence). The `\documentclass` command takes one [mandatory argument](#) `<class-name>` that specifies the [class file](#). There are a great many available, but the basic ones are: **article** (short documents without chapters), **report** (longer technical documents containing chapters), **book** (for writing books), **letter** (for correspondence) and **slides** (for creating slides for use with OHP or data projectors).

[Replacing the standard classes]

We'll be starting with a very simple document, so let's use the **article** class file. In this case the very first line of the [source code](#) should be:

```
\documentclass{article}
```

#### 4. CREATING A SIMPLE DOCUMENT

The `\documentclass` command also takes an [optional argument](#) *<option-list>* which should be a comma separated list of options to be passed to the class file. This allows you to override the class file defaults. For example, the `article` class file by default uses US letter paper, but in the UK we would want to use A4. This can be achieved using the option `a4paper`. So you would need to edit the above line to:

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

Let's change another option. The normal font size is `10pt` by default, but we have the option to change it to `11pt` or `12pt`, so let's change it to `11pt`:

```
\documentclass[a4paper,11pt]{article}
```

You can also change your document so that it is in a two column format using the `twocolumn` option:

```
\documentclass[a4paper,11pt,twocolumn]{article}
```

Note that there must not be any spaces between the options.

After deciding what type of document we want, we now need to specify the contents of the document. We do this inside the [document environment](#). The document is started with the command:

```
\begin{document}
```

#### 4. CREATING A SIMPLE DOCUMENT

and ended with

```
\end{document}
```

So our [source code](#) now looks like:

```
\documentclass[a4paper,11pt]{article}

\begin{document}

\end{document}
```

[↑Code](#)

[↓Code](#)

Every document you create must have this form. You can't simply start typing the contents of the document. You must firstly specify your class file, and then place the contents of the document inside the `document` environment. It is a common mistake when first starting out to miss out one or more of these three lines.

So far so good, but at the moment we have an empty document, so we won't get any output. Let's now put some text into our document:

## 4. CREATING A SIMPLE DOCUMENT

```
\documentclass[a4paper,11pt]{article}

\begin{document}

This is a simple document.
Here is the first paragraph.

Here is the second paragraph. As you
can see it's
a very
short document.

\end{document}
```

[↑Code](#)

[↓Code](#)

### Exercise 1 (Simple Document)

Try typing the above code into your editor (see [chapter 3](#) if you can't

#### 4. CREATING A SIMPLE DOCUMENT

remember what to do). You can also [download](#) a copy of this file, but I recommend that you try typing it in to give yourself some practice. If you are using `TeXnicCenter`, start a new project as detailed on page [54](#). Call your project, say, `sample1`.

Things to note while you are typing: Firstly, when you press the return character at the end of the line this end of line character is converted into a space in the [output file](#). So the fact that I have some very ragged lines in my source code has no effect on the final result. (Note that some front-ends will reformat your lines as you type.)

Secondly, multiple spaces are converted into a single space, so the large gap between the words `can` and `see` is no different from having a single space.

Thirdly, a completely blank line will be converted into a paragraph break, but that doesn't mean that you'll have a blank line between your paragraphs in the output. In fact, by default you won't with most class files, although you can override this.

Fourthly, you don't need to worry about the indentation at the start of new paragraphs as this is done automatically (again it is possible to override paragraph indentation, or change the indentation length).

Once you have typed up your source code, save your file as, say, `sample1.tex`, and then pass it to `LATEX` using the methods described in [chapter 3](#). If all goes well, something like the following should be displayed

#### 4. CREATING A SIMPLE DOCUMENT

on the screen:

```
This is TeX, Version 3.14159 (MikTeX 2.1)
(sample1.tex
LaTeX2e <2000/06/01>
Babel <v3.7h> and hyphenation patterns for american, french, german,
ngerman, italian, nohyphenation, loaded.
(C:\texmf\tex\latex\base\article.cls
Document Class: article 2000/05/19 v1.4b Standard LaTeX document class
(C:\texmf\tex\latex\base\size11.clo))
No file sample1.aux.
[1] (sample1.aux) )
Output written on sample1.dvi (1 page, 376 bytes).
Transcript written on sample1.log.
```

This indicates that your source code has successfully been converted into the typeset output contained in the new file `sample1.dvi`. You can now view this document either by typing `xdvi sample1.dvi` in the [terminal](#), or by clicking on the view output button in [TeXnicCenter](#) or the view DVI button in [WinEdt](#).

If you have made a mistake in the source code, an error message will be displayed on screen, and the question mark prompt will appear. At this point you can either type `h` for a help message, or type `x` to exit  $\text{\LaTeX}$

## 4. CREATING A SIMPLE DOCUMENT

and go back to your source code and fix the problem.<sup>1</sup> If you do have an error, consult [chapter 13](#) for guidance.

---

### 4.1 Using Simple Commands

Now let's try adding a few simple [commands](#) to our document. The command `\LaTeX` produces the logo L<sup>A</sup>T<sub>E</sub>X and the command `\today` prints the current date. L<sup>A</sup>T<sub>E</sub>X always ignores any spaces that follow a command name that consists of letters, as it uses the space to indicate the end of the command name. This means that if we want a space to occur immediately after the command, we would need to explicitly say so using the command `\_` where `_` indicates a space character. So, for example:

[Typesetting all those TeX-related logos]

```
\LaTeX\ logo
```

Input

produces the output:

```
LATEX logo
```

Output

---

<sup>1</sup>TeXnicCenter is non-interactive, it will carry on going until it gets to the end. Once it has finished you can locate each error as described on page [67](#).



## 4. CREATING A SIMPLE DOCUMENT

Some people when starting out can get a bit confused by this and read it as the entity “`\LaTeX`” whereas it is in fact two commands: “`\LaTeX`” (print the L<sup>A</sup>T<sub>E</sub>X logo) followed “`\` ” (print a space).

Let’s also try using a command that takes an [argument](#). The command

```
\footnote{<text>}
```

Definition

takes one argument that specifies the text that should appear in the footnote. This command should be placed where you want the footnote marker to appear.

### Exercise 2 (Using Simple Commands)

Try editing the document you created in [exercise 1](#), so that it looks like the following: (You can [download](#) it if you like, but again it is better if you try typing it in yourself)

```
\documentclass[a4paper,11pt]{article}

\begin{document}
```

[↑Code](#)

#### 4. CREATING A SIMPLE DOCUMENT

```
This is a simple \LaTeX\ document.
```

```
Here is the first paragraph.
```

```
Here is the second paragraph. As you  
can see it's
```

```
a very
```

```
short document.\footnote{with a footnote.}
```

```
This document was created on: \today.
```

```
\end{document}
```

[↓ Code](#)

Now [L<sup>A</sup>T<sub>E</sub>X your document](#) and view the result. (Remember to check [chapter 13](#) if you have a problem.) You should see the L<sup>A</sup>T<sub>E</sub>X logo, the footnote marker and the current date. If you scroll down to the bottom of the page, you should see the footnote.

---

## 4.2 Special Characters and Symbols

You can use any of the standard characters that you find on your keyboard, except the following 10 symbols:

{ } % & \$ # \_ ^ ~ \

These symbols may only occur in L<sup>A</sup>T<sub>E</sub>X commands. We have already used the curly braces { and }. The percent symbol % is a comment character. Everything from the percent symbol up to the end of line is ignored by L<sup>A</sup>T<sub>E</sub>X. This means you can have comments in your [source code](#) to remind you what a particular part of your code is doing. You have also used the backslash symbol \ which indicates that you are using a L<sup>A</sup>T<sub>E</sub>X command, as in `\LaTeX` or `\today`. The meaning of the other special characters will be covered later.

So what do you do if you want one of these symbols to actually appear in your document? [Table 4.1](#) lists commands that produce these and other symbols. Note that some of the commands have shortcuts, such as `---` instead of `\textemdash` and `?‘` instead of `\textquestiondown`.

The symbol ‘ is the backtick symbol, as opposed to the apostrophe symbol ’. The backtick symbol usually looks like ` on a keyboard, and on most UK keyboards it is situated to the left of the 1 key. The opening double quote is created using two adjacent backtick symbols, and the

[Where can I find the symbol for ...?]

[How to get copyright, trademark, etc]

#### 4. CREATING A SIMPLE DOCUMENT

closing double quote is created using two adjacent apostrophe symbols, this gives 66 and 99 style quotes, which you wouldn't get using the double quote character.

Note that the symbols | < and > have to be created using `\textbar`, `\textless` and `\textgreater` when in normal text mode. If you try to enter them using the corresponding keyboard characters you will get — ; and ¡. (They do however work if you are in [maths mode](#).) The slash character / may be used directly, as in `and/or`, but no line break will be permitted at the slash, whereas `\slash` (as in `and\slash{ }or`) will allow a line break at that point.

Ligatures and special symbols are shown in [table 4.2](#). A ligature is where two or more letters are combined as a single glyph. In English, the most common ligatures are the ff, fl, ffl, fi and ffi ligatures, mentioned in the [introduction](#). Without the ligature, the two letters collide and appear ugly. This usually only occurs with serif fonts, not for san-serif fonts. Some fonts may provide additional glyphs.

As already mentioned, the f-ligatures are converted automatically<sup>2</sup> without the need for commands, but this may not always be desirable, for example, where the sequence of letters cross a boundary in a composite word. This doesn't happen very often in English, but when it does, there are various methods you can use to break the ligature. *The*

[What are encodings?]

---

<sup>2</sup>depending on the font.

#### 4. CREATING A SIMPLE DOCUMENT

Table 4.1: Symbols

|                                                   |                |                     |                    |                                                    |                   |
|---------------------------------------------------|----------------|---------------------|--------------------|----------------------------------------------------|-------------------|
| <code>\textbackslash</code>                       | <code>\</code> | <code>\_</code>     | <code>_</code>     | <code>\textgreater</code>                          | <code>&gt;</code> |
| <code>\textasciicircum</code>                     | <code>^</code> | <code>\\$</code>    | <code>\$</code>    | <code>\textbar</code>                              | <code> </code>    |
| <code>\textasciitilde</code>                      | <code>~</code> | <code>\{</code>     | <code>{</code>     | <code>\textless</code>                             | <code>&lt;</code> |
| <code>\pounds</code>                              | £              | <code>\}</code>     | <code>}</code>     | <code>\dag</code>                                  | †                 |
| <code>\textregistered</code>                      | ®              | <code>\#</code>     | <code>#</code>     | <code>\ddag</code>                                 | ‡                 |
| <code>\texttrademark</code>                       | ™              | <code>\%</code>     | <code>%</code>     | <code>'</code> or <code>\textquoteright</code>     | '                 |
| <code>\copyright</code>                           | ©              | <code>\&amp;</code> | <code>&amp;</code> | <code>'</code> or <code>\textquoteleft</code>      | '                 |
| <code>\yen</code>                                 | ¥              | <code>\i</code>     | <code>i</code>     | <code>''</code> or <code>\textquotedblright</code> | ”                 |
| <code>?'</code> or <code>\textquestiondown</code> | ¿              | <code>\j</code>     | <code>j</code>     | <code>''</code> or <code>\textquotedblleft</code>  | “                 |
| <code>!'</code> or <code>\textexclamdown</code>   | ¡              | <code>-</code>      | <code>-</code>     | <code>--</code> or <code>\textendash</code>        | –                 |
| <code>---</code> or <code>\textemdash</code>      | —              | <code>\S</code>     | <code>§</code>     | <code>\textperiodcentered</code>                   | ·                 |
| <code>\ldots</code>                               | ...            | <code>\P</code>     | ¶                  | <code>\slash</code>                                | /                 |

#### 4. CREATING A SIMPLE DOCUMENT

*TEXbook* [12] uses the example “shelfful”, and suggests various solutions, including `shelf{}ful` and `shelf\/ful`. The latter uses an italic correction `\/` which will be discussed in [section 4.4.1](#).

When using a command in the middle of a word, take care that the command doesn’t run into the rest of the word. For example, the British spelling of the word manœuvre has an œ-ligature in the middle of it. There are several ways to code this in L<sup>A</sup>T<sub>E</sub>X:

1. Group the command:

```
man{\oe}uvre
```

Input

2. Place a space after the command:

```
man\oe uvre
```

Input

3. Place an empty brace after the command:

```
man\oe{}uvre
```

Input

#### 4. CREATING A SIMPLE DOCUMENT

Each of these three methods produce the same result, but I personally prefer the first method. It is important to make your [source code](#) as easy to read as possible, as you may need to edit your document at some later date; the first of the above three examples retains the look of a complete word, whereas the second example fragments the word, so although the word is whole in the output, it doesn't read well when you're editing your code. The third example, like the first example, maintains the word's cohesion, but it gives the misleading impression that the command `\oe` has an argument. However, as I mentioned, this is my personal preference, you should use whichever method you feel most comfortable with, just as long as you don't do the following:

```
man\oeuvre
```

This is incorrect, as  $\text{\LaTeX}$  will interpret it as the command `\oeuvre` which doesn't exist.

English speakers are by and large very lackadaisical when it comes to accents, but accents affect pronunciation, and so are just as important as the correct spelling. There is a very big difference between putting your knife into someone's pâté (meat paste), and putting your knife into someone's pate (head)!

Accented letters are created by specifying which accent you want, and the letter on which to put the accent. The accent commands are listed in

## 4. CREATING A SIMPLE DOCUMENT

Table 4.2: Ligatures and Special Symbols

|                  |    |                  |     |                  |    |                  |     |
|------------------|----|------------------|-----|------------------|----|------------------|-----|
| <code>\AE</code> | Æ  | <code>\ae</code> | æ   | <code>\OE</code> | Œ  | <code>\oe</code> | œ   |
| <code>fi</code>  | fi | <code>ffi</code> | ffi | <code>fl</code>  | fl | <code>ffl</code> | ffl |
| <code>\AA</code> | Å  | <code>\aa</code> | å   | <code>\L</code>  | Ł  | <code>\l</code>  | ł   |
| <code>\O</code>  | Ø  | <code>\o</code>  | ø   | <code>\SS</code> | Š  | <code>\ss</code> | š   |

[table 4.3](#), and each command takes one [mandatory argument](#). The command indicates what accent to use, the argument indicates the letter on which to put the accent. You may have noticed in [table 4.1](#) the commands `\i` and `\j` which produce a dotless i and j (ı and j̄). You should use these instead of `i` and `j` as the argument to an accent command, since `i` and `j` should lose their dot when they have an accent over them.<sup>3</sup> Example:

---

```
It's na{"\i}ve to think that eating mouldy p^at\'e
won't result in food poisoning.
```

<sup>3</sup>Note that in some font encodings `na{"i}ve` works.



#### 4. CREATING A SIMPLE DOCUMENT

↓ Input

Result:

It's naïve to think that eating mouldy pâté won't result in food poisoning.

↑ Output

↓ Output

**Note 2:** Recall that if the [mandatory argument](#) only consists of a single letter, no grouping is required, thus `p\^at\'e` is equivalent to `p\^{a}t\'{e}`. However, in the case of `na\{i}ve`, grouping is required (or a space is required after the `\i`) otherwise the `\i` will run into the rest of the word: `na\live`. In this case you will get an error, as the command `\ive` doesn't exist.

### Exercise 3 (Using Special Characters)

Start a new file (or project if using `TeXnicCenter`), and see if you can write the source code to create the following output:

#### 4. CREATING A SIMPLE DOCUMENT

Table 4.3: Accent Commands

| Definition                       | Example              |            | Definition                       | Example             |          |
|----------------------------------|----------------------|------------|----------------------------------|---------------------|----------|
|                                  | Input                | Output     |                                  | Input               | Output   |
| <code>\' {&lt;object&gt;}</code> | <code>\' {c}</code>  | ć          | <code>\={&lt;object&gt;}</code>  | <code>\={c}</code>  | ċ        |
| <code>\' {&lt;object&gt;}</code> | <code>\' {c}</code>  | ĉ          | <code>\. {&lt;object&gt;}</code> | <code>\. {c}</code> | ċ        |
| <code>\^ {&lt;object&gt;}</code> | <code>\^ {c}</code>  | ĉ          | <code>\~ {&lt;object&gt;}</code> | <code>\~ {c}</code> | č        |
| <code>\" {&lt;object&gt;}</code> | <code>\" {c}</code>  | č          | <code>\v {&lt;object&gt;}</code> | <code>\v {c}</code> | č        |
| <code>\u {&lt;object&gt;}</code> | <code>\u {c}</code>  | č          | <code>\H {&lt;object&gt;}</code> | <code>\H {c}</code> | č        |
| <code>\t {&lt;object&gt;}</code> | <code>\t {xy}</code> | $\bar{xy}$ | <code>\c {&lt;object&gt;}</code> | <code>\c {c}</code> | ç        |
| <code>\d {&lt;object&gt;}</code> | <code>\d {c}</code>  | ç          | <code>\b {&lt;object&gt;}</code> | <code>\b {c}</code> | <u>c</u> |
| <code>\r {&lt;object&gt;}</code> | <code>\r {c}</code>  | ċ          |                                  |                     |          |

## 4. CREATING A SIMPLE DOCUMENT

Item #1: Our travel expenditure came to \$2000.00 & our equipment expenditure came to £100.00 plus VAT @ 17.5%.

↑Output

Chloë collected Zoë from the crèche. They stopped to admire the façade of a new café, and then went to a matinée.

↓Output

You can [download](#) or [view](#) the source code if you can't work out how to do it, and remember to check [chapter 13](#) if you have a problem.

---

### 4.3 Lists

Now you've had a go at using some [commands](#), let's use some [environments](#). A good example of environments are the list making environments. There are three basic list making environments: `itemize` (for unordered lists), `enumerate` (for ordered lists) and `description` (for lists where you want to specify your own label).

In each of these environments, there is a command

```
\item[<marker>]
```

Definition

## 4. CREATING A SIMPLE DOCUMENT

which you need to use to specify each item of the list. The optional argument `<marker>` can be used to override the default marker for that particular item (for example, to replace the bullet point for an individual item in an [unordered list](#) to make that item stand out from all the other items). We will be looking at how to change the default marker in [section 8.2](#).

Related UK TUG FAQ [2] topics:

- [Perhaps a missing `\item`?](#)
- [Fancy enumeration lists](#)
- [How to adjust list spacing](#)
- [Interrupting enumerated lists](#)
- [“Too deeply nested”](#)

### 4.3.1 Unordered Lists

Unordered lists are created using the `itemize` environment. For example, the following code:

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{itemize}
\item Animal
\item Vegetable
\item Mineral
\end{itemize}
```

↓ Input

will produce the following output:

- Animal
- Vegetable
- Mineral

↑ Output

↓ Output

It is also possible to nest `itemize` environments. For example, the following code:

↑ Input

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{itemize}
\item Animal
\begin{itemize}
\item Mammals
\item Birds
\item Reptiles. For example:
\begin{itemize}
\item dinosaurs
\item crocodiles
\end{itemize}
\end{itemize}
\item Vegetable
\begin{itemize}
\item Cultivated
\item Wild
\end{itemize}
\item Mineral
\end{itemize}
```

[↓ Input](#)

will produce the following output:

[↑ Output](#)

#### 4. CREATING A SIMPLE DOCUMENT

- Animal
  - Mammals
  - Birds
  - Reptiles. For example:
    - \* dinosaurs
    - \* crocodiles
- Vegetable
  - Cultivated
  - Wild
- Mineral

↓ Output

That looks good, but our code is a bit cramped and a little difficult to read. Blank lines between list items are ignored by  $\text{\LaTeX}$ , and multiple spaces are treated as a single space, so we could make the code a bit more readable, without affecting the final result:

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{itemize}

  \item Animal

  \begin{itemize}

    \item Mammals

    \item Birds

    \item Reptiles. For example:
    \begin{itemize}

      \item dinosaurs

      \item crocodiles

    \end{itemize}

  \end{itemize}

\end{itemize}

\item Vegetable
```



## 4. CREATING A SIMPLE DOCUMENT

```
\begin{itemize}

  \item Cultivated

  \item Wild

\end{itemize}

\item Mineral

\end{itemize}
```

[↓ Input](#)

It's now a little easier to see which `\begin{itemize}` matches up with the corresponding `\end{itemize}`.

### 4.3.2 Ordered Lists

Ordered lists are created using the `enumerate` environment. It has exactly the same format as the `itemize` environment described in [the previous section](#).

[Fancy enumeration lists]

#### 4. CREATING A SIMPLE DOCUMENT

We can use the same example as before, only this time use `enumerate` instead of `itemize`.

```
\begin{enumerate}
\item Animal
\item Vegetable
\item Mineral
\end{enumerate}
```

↑Input

↓Input

The above input will produce the following output:

1. Animal
2. Vegetable
3. Mineral

↑Output

↓Output

#### 4. CREATING A SIMPLE DOCUMENT

Again, the environments can be nested:

```
\begin{enumerate}

  \item Animal

  \begin{enumerate}

    \item Mammals

    \item Birds

    \item Reptiles. For example:
    \begin{enumerate}

      \item dinosaurs

      \item crocodiles

    \end{enumerate}

  \end{enumerate}
```

↑Input

#### 4. CREATING A SIMPLE DOCUMENT

```
\end{enumerate}
```

```
\item Vegetable
```

```
\begin{enumerate}
```

```
  \item Cultivated
```

```
  \item Wild
```

```
\end{enumerate}
```

```
\item Mineral
```

```
\end{enumerate}
```

↓ Input

The above input will produce the following output:

```
1. Animal
```

```
  (a) Mammals
```

↑ Output

#### 4. CREATING A SIMPLE DOCUMENT

(b) Birds

(c) Reptiles. For example:

i. dinosaurs

ii. crocodiles

2. Vegetable

(a) Cultivated

(b) Wild

3. Mineral

[↓Output](#)

### 4.3.3 Description Environment

The **description** environment has exactly the same format as the **itemize** environment described in [section 4.3.1](#), only this time you need to specify a marker as an [optional argument](#) to the `\item` command, since there is no default marker for this environment. For example, the following code:

[↑Input](#)

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{description}  
\item[Animal] Living being  
\item[Vegetable] Plant  
\item[Mineral] Natural inorganic substance  
\end{description}
```

↓ Input

will produce the following output:

**Animal** Living being

**Vegetable** Plant

**Mineral** Natural inorganic substance

↑ Output

↓ Output

It is possible to nest all the listing environments:

↑ Input

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{description}

  \item[Animal] Living being

  \begin{itemize}

    \item Mammals

    \item Birds

    \item Reptiles. For example:
    \begin{enumerate}

      \item dinosaurs

      \item crocodiles

    \end{enumerate}

  \end{itemize}

  \item[Vegetable] Plant
```

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{itemize}
```

```
  \item Cultivated
```

```
  \item Wild
```

```
\end{itemize}
```

```
\item[Mineral] Natural inorganic substance
```

```
\end{description}
```

↓ Input

The above input will produce the output:

↑ Output

**Animal** Living being

- Mammals
- Birds
- Reptiles. For example:



#### 4. CREATING A SIMPLE DOCUMENT

1. dinosaurs
2. crocodiles

#### **Vegetable** Plant

- Cultivated
- Wild

#### **Mineral** Natural inorganic substance

↓Output

### Exercise 4 (Lists)

Try writing the [source code](#) that will create the following output:

↑Output

#### **Village** A small collection of dwelling places. Examples:

1. Marlingford

## 4. CREATING A SIMPLE DOCUMENT

### 2. Saxlingham Nethergate

**Town** A large collection of dwelling places. Examples:

1. Great Yarmouth
2. Beccles

**City** A large town, usually containing a cathedral. Examples:

1. Norwich
2. Birmingham
3. London

[↓Output](#)

---

You can [download](#) or [view](#) the answer if you can't work out how to do it.

---

## 4.4 Simple font changing commands

L<sup>A</sup>T<sub>E</sub>X uses Donald Knuth's Computer Modern fonts by default. This supplies three font families: serif, sans-serif and a typewriter (or monospaced)

[[Using PostScript fonts with TeX](#)]

## 4. CREATING A SIMPLE DOCUMENT

font (as well as the [maths fonts](#) which are discussed in [section 9.3.1](#)). With each font family, you can change the shape and weight, as well as the size. It is possible to use different font families, but that isn't covered here.

[Installing a new font]

### 4.4.1 Changing the Font Style

There are two basic ways of changing fonts: you can either change the font for a small selection of text, for example, if you want to *emphasize* a word, or you may wish to change the font “from this point onwards”. The [commands](#) shown in [table 4.4](#) are of the first type (text-block commands), whereas those shown in [table 4.5](#) are of the second type—a [declaration](#) (or modal command).

[What's wrong with `\bf`, `\it` etc.??]

If you use an italic or slanted font declaration, you will need to add an italic correction `\/` at the end of the block of text, when the last letter of the sloping text leans too far over. For example, compare

```
{\itshape Some italic text} back to upright.
```

Input

which yields

*Some italic text* back to upright.

Output

to

#### 4. CREATING A SIMPLE DOCUMENT

`{\itshape Some italic text\}` back to upright.

Input

which yields

*Some italic text* back to upright.

Output

In the first example, the final letter “t” in the word “text” leans too far over the space. In the second example, extra space (known as italic correction) is inserted.

If you use one of the text-block commands, such as `\textit`, the italic correction is dealt with by the command, so the above example would be equivalent to:

`\textit{Some italic text}` back to upright.

Input

which again yields

*Some italic text* back to upright.

Output

The effect is more noticeable when part of a word is stressed. For example:

#### 4. CREATING A SIMPLE DOCUMENT

```
{\itshape repeated}ly \textit{repeated}ly
```

Input

produces

*repeatedly repeatedly*

Output

[Environments](#) can be used instead. Each environment has the same name as its corresponding declaration, but *without* the preceding backslash. For example:

```
\begin{sffamily}Some sans-serif text.\end{sffamily}
```

Input

yields:

Some sans-serif text.

Output

You can combine a font family with a given shape and weight using a variety of methods, such as:

1. Declarations:

## 4. CREATING A SIMPLE DOCUMENT

Table 4.4: Font changing commands

| Command                                | Example Input                          | Corresponding output   |
|----------------------------------------|----------------------------------------|------------------------|
| <code>\textrm{&lt;text&gt;}</code>     | <code>\textrm{roman} text</code>       | roman text             |
| <code>\textsf{&lt;text&gt;}</code>     | <code>\textsf{sans serif} text</code>  | sans serif text        |
| <code>\texttt{&lt;text&gt;}</code>     | <code>\texttt{typewriter} text</code>  | typewriter text        |
| <code>\textmd{&lt;text&gt;}</code>     | <code>\textmd{medium} text</code>      | medium text            |
| <code>\textbf{&lt;text&gt;}</code>     | <code>\textbf{bold} text</code>        | <b>bold</b> text       |
| <code>\textup{&lt;text&gt;}</code>     | <code>\textup{upright} text</code>     | upright text           |
| <code>\textit{&lt;text&gt;}</code>     | <code>\textit{italic} text</code>      | <i>italic</i> text     |
| <code>\textsl{&lt;text&gt;}</code>     | <code>\textsl{slanted} text</code>     | <i>slanted</i> text    |
| <code>\textsc{&lt;text&gt;}</code>     | <code>\textsc{Small Caps} text</code>  | SMALL CAPS text        |
| <code>\emph{&lt;text&gt;}</code>       | <code>\emph{emphasized} text</code>    | <i>emphasized</i> text |
| <code>\textnormal{&lt;text&gt;}</code> | <code>\textnormal{default} text</code> | default text           |

#### 4. CREATING A SIMPLE DOCUMENT

Table 4.5: Font changing declarations

| Declaration              | Example Input                          | Corresponding output   |
|--------------------------|----------------------------------------|------------------------|
| <code>\rmfamily</code>   | <code>\rmfamily roman text</code>      | roman text             |
| <code>\sffamily</code>   | <code>\sffamily sans serif text</code> | sans serif text        |
| <code>\ttfamily</code>   | <code>\ttfamily typewriter text</code> | typewriter text        |
| <code>\mdseries</code>   | <code>\mdseries medium text</code>     | medium text            |
| <code>\bfseries</code>   | <code>\bfseries bold text</code>       | <b>bold text</b>       |
| <code>\upshape</code>    | <code>\upshape upright text</code>     | upright text           |
| <code>\itshape</code>    | <code>\itshape italic text</code>      | <i>italic text</i>     |
| <code>\slshape</code>    | <code>\slshape slanted text</code>     | <i>slanted text</i>    |
| <code>\scshape</code>    | <code>\scshape Small Caps text</code>  | SMALL CAPS TEXT        |
| <code>\em</code>         | <code>\em emphasized text</code>       | <i>emphasized text</i> |
| <code>\normalfont</code> | <code>\normalfont default text</code>  | default text           |

#### 4. CREATING A SIMPLE DOCUMENT

```
{\sffamily\slshape Some slanted sans-serif text\/}
```

Input

2. Mixing commands and declarations:

```
\textsl{\sffamily Some slanted sans-serif text}
```

Input

3. Nested commands

```
\textsf{\textsl{Some slanted sans-serif text}}
```

Input

4. Mixing environments and declarations:

```
_____
```

↑ Input



#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{sffamily}\slshape Some slanted sans-serif  
text\end{sffamily}
```

[↓ Input](#)

All of the above produce the same output:

*Some slanted sans-serif text*

[Output](#)

Note that some combinations are not available, in which case L<sup>A</sup>T<sub>E</sub>X will give a warning message, and will substitute the font for what it considers to be the closest available match.

[Warning:  
“Font shape ...  
not available”]

Note also that the [command](#) `\emph`, the [declaration](#) `\em` and the [environment](#) `em` behave slightly differently to the corresponding `\textit` command, `\itshape` declaration and `itshape` environment. The latter simply use an italic font, whereas the former will toggle between sloping and upright. So if the surrounding font is upright then `\emph`, `\em` and `em` will use the sloping font, but if the surrounding font is italic or slanted, `\emph`, `\em` and `em` will use an upright font. This is particularly useful in abstracts where the abstract font varies between [class files](#). It is recommended that if your intention is to emphasize something, you should use `\emph` etc. rather than `\textit` etc. Examples:

[How to do  
bold-tt or  
bold-sc]

#### 4. CREATING A SIMPLE DOCUMENT

1. Emphasized text in upright surrounding:

Some `\emph{emphasized}` text.

Input

yields

Some *emphasized* text.

Output

2. Emphasized text in italic surrounding:

`{\itshape Some \emph{emphasized} text.}`

Input

yields

*Some emphasized text.*

Output

## 4. CREATING A SIMPLE DOCUMENT

3. Emphasized text in upright sans-serif surrounding:

```
{\sffamily Some \emph{emphasized} text.}
```

Input

yields

Some *emphasized* text.

Output

### 4.4.2 Changing the Font Size

The size of the font is changed using one of the declarations shown in [table 4.6](#). The sizes are all relative to the size of the normal font. So if you decide to change the normal font from, say, 11pt to 12pt (by changing the class file option as mentioned on [page 76](#)), all the font sizes will be changed relative to the new size. There are no equivalent text-block commands.

Again, [environments](#) can be used instead, where each environment has the same name as its corresponding declaration, but *without* the preceding backslash. Font environments may be nested, for example:

---

↑Input

#### 4. CREATING A SIMPLE DOCUMENT

Table 4.6: Font size changing declarations

| Declaration                | Example Input                                  | Corresponding output |
|----------------------------|------------------------------------------------|----------------------|
| <code>\tiny</code>         | <code>\tiny tiny text</code>                   | tiny text            |
| <code>\scriptsize</code>   | <code>\scriptsize script sized text</code>     | script sized text    |
| <code>\footnotesize</code> | <code>\footnotesize footnote sized text</code> | footnote sized text  |
| <code>\small</code>        | <code>\small small text</code>                 | small text           |
| <code>\normalsize</code>   | <code>\normalsize normal sized text</code>     | normal sized text    |
| <code>\large</code>        | <code>\large large text</code>                 | large text           |
| <code>\Large</code>        | <code>\Large even larger</code>                | even larger          |
| <code>\LARGE</code>        | <code>\LARGE larger still</code>               | larger still         |
| <code>\huge</code>         | <code>\huge huge</code>                        | huge                 |
| <code>\Huge</code>         | <code>\Huge really huge</code>                 | really huge          |

```
\begin{itshape} Some italic text.  
\begin{Large}This text is large.\end{Large}  
\end{itshape} Back to normal.
```

[↓ Input](#)

Output:

## 4. CREATING A SIMPLE DOCUMENT

*Some italic text. This text is large.* Back to normal.

Output

Note also that the original Computer Modern fonts are not scalable, so if you use a class file with a large normal font (such as the `a0poster` class file or one of the class files in the `exsizes` distribution), some of the font sizes may not be available, and font substitutions will occur.

For more information on using fonts, including using fonts not covered in this document, see *A Guide to L<sup>A</sup>T<sub>E</sub>X* [4] or *The L<sup>A</sup>T<sub>E</sub>X Companion* [5].

[Choice of  
scalable outline  
fonts]

### Exercise 5 (Fonts)

Go back to the document you created in [exercise 1](#) and change the first paragraph to a large bold font and the second paragraph to normal size italic. Emphasize the words “simple” and “short”. (Again, you can [download](#) or [view](#) the solution.)

---

## 4.5 Aligning Material in Rows and Columns

Text can be aligned in rows and columns using the `tabular` environment.

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{tabular}[<pos>]{<column specifiers>}
```

Definition

This **environment** has a **mandatory argument** `<column specifiers>` that specifies how to align each column. There are three basic specifiers: `r` (right aligned), `l` (left aligned) and `c` (centred). For example, suppose we want three columns with the first column left justified and the last two columns centred we would do:

```
\begin{tabular}{lcc}
```

Input

(Make sure you don't confuse `l` (the letter ell) with `1` (one).)

The ampersand character `&` is used to separate column entries and `\\` is used to separate rows. For example, let's have two columns, the first left justified and the second right justified:

[Alignment tab changed to `\cr`]

```
\begin{tabular}{lr}
Video & 8.99\\
CD & 9.99\\
DVD & 15.00\\
Total & 33.98
\end{tabular}
```

Input

#### 4. CREATING A SIMPLE DOCUMENT

↓Input

Output:

|       |       |
|-------|-------|
| Video | 8.99  |
| CD    | 9.99  |
| DVD   | 15.00 |
| Total | 33.98 |

↑Output

↓Output

Remember that L<sup>A</sup>T<sub>E</sub>X ignores multiple spaces, so we could just have easily done:

↑Input

```
\begin{tabular}{lr}
Video & 8.99\\
CD & 9.99\\
DVD & 15.00\\
Total & 33.98
\end{tabular}
```

↓Input

#### 4. CREATING A SIMPLE DOCUMENT

and we would still have got the same result.

Entries form implicit [grouping](#), so [declarations](#) made within a [tabular](#) environment only have an effect up to the next `&` or `\\`. Example:

```
\begin{tabular}{lr}
Video & 8.99\\
CD    & 9.99\\
DVD   & 15.00\\
\bfseries Total & 33.98
\end{tabular}
```

↑Input

↓Input

Output:

```
Video    8.99
CD       9.99
DVD      15.00
Total 33.98
```

↑Output

↓Output

Let's add an extra column and a header row:



#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{tabular}{lrr}
Item & ex VAT & inc VAT\\
Video & 8.99 & 10.56\\
CD & 9.99 & 11.74\\
DVD & 15.00 & 17.63\\
\bfseries Total & 33.98 & 39.93
\end{tabular}
```

↑Input

↓Input

Output:

| Item         | ex VAT | inc VAT |
|--------------|--------|---------|
| Video        | 8.99   | 10.56   |
| CD           | 9.99   | 11.74   |
| DVD          | 15.00  | 17.63   |
| <b>Total</b> | 33.98  | 39.93   |

↑Output

↓Output

The command

```
\multicolumn{<cols spanned>}{<col specifier>}{<text>}
```

Definition

#### 4. CREATING A SIMPLE DOCUMENT

can be used to span several columns. The first `argument` `<cols spanned>` is the number of columns you want to span, the second argument `<col specifier>` indicates how to align this column spanning entry, the third argument `<text>` indicates what should go in this entry. We can use `\multicolumn` to modify the previous example as follows:

[Merging cells in a column of a table]

```
\begin{tabular}{lrr}
    & & & \multicolumn{2}{c}{Price (\pounds)}\\
Item & ex VAT & inc VAT\\
Video & 8.99 & & 10.56\\
CD & 9.99 & & 11.74\\
DVD & 15.00 & & 17.63\\
\bfseries Total & 33.98 & & 39.93
\end{tabular}
```

↑Input

↓Input

Output:

↑Output



#### 4. CREATING A SIMPLE DOCUMENT

|           | Year1   | Year2   |
|-----------|---------|---------|
| Travel    | 100,000 | 110,000 |
| Equipment | 50,000  | 60,000  |

↑Output

↓Output

In this example, the headers “Year1” and “Year2” would look better centred, but the rest of the entries in the second and third columns look best right aligned. We can use `\multicolumn` to span just one column, and use the second argument of `\multicolumn` to override the column specification:

```
\begin{tabular}{lrr}
      & \multicolumn{1}{c}{Year1}
      & \multicolumn{1}{c}{Year2} \\
Travel & 100,000 & 110,000 \\
Equipment & 50,000 & 60,000
\end{tabular}
```

↑Input

↓Input

Output:

#### 4. CREATING A SIMPLE DOCUMENT

|           | Year1   | Year2   |
|-----------|---------|---------|
| Travel    | 100,000 | 110,000 |
| Equipment | 50,000  | 60,000  |

↑Output

↓Output

### Exercise 6 (Aligning Material)

Go back to the document you created in [exercise 2](#), and add the following to your document:

|                  | <b>Expenditure</b> |         |
|------------------|--------------------|---------|
|                  | Year1              | Year2   |
| <b>Travel</b>    | 100,000            | 110,000 |
| <b>Equipment</b> | 50,000             | 60,000  |

↑Output

↓Output

#### 4. CREATING A SIMPLE DOCUMENT

Note that the `tabular` environment doesn't create a caption, all it does is arrange its contents in rows and columns. You will find out how to turn your `tabular` environment into a table in [section 7.2](#).

You can [download](#) or [view](#) the result.

---

For more information about using the `tabular` environment, including how to add vertical and horizontal lines, see *L<sup>A</sup>T<sub>E</sub>X: A Document Preparation System* [3], *A Guide to L<sup>A</sup>T<sub>E</sub>X* [4] or *The L<sup>A</sup>T<sub>E</sub>X Companion* [5]. The latter reference also describes how to span rows using the `multirow` package. For information on how to create coloured tables using the `colortbl` package, see *The L<sup>A</sup>T<sub>E</sub>X Graphics Companion* [6]. Related UK TUG FAQ [2] topics:

- [How to change a whole row of a table](#)
- [Merging cells in a column of a table](#)
- [Fixed width tables](#)
- [Variable-width columns in tables](#)
- [Spacing lines in tables](#)
- [The thickness of rules in LaTeX tables](#)

## 4.6 Boxes and Mini-Pages

$\text{\TeX}$  views everything on a page as a form of box. Each box has an associated width, height and depth, and the boxes are placed together on the page with glue. This is reminiscent of the days of manual typesetting, where each letter or symbol was on a wooden block, and the wooden blocks were glued in place. The simplest form of box is a single letter. Some letters, such as “a” only have a height and width, whereas other letters, such as “y” have a height, width and depth (see [figure 4.1](#)).

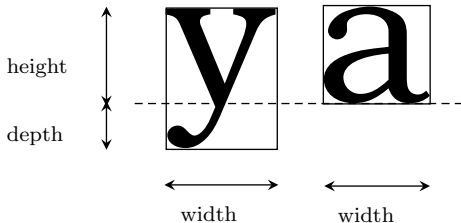


Figure 4.1:  $\text{\TeX}$  views each letter as a box

#### 4. CREATING A SIMPLE DOCUMENT

For example, the phrase “cabbages and peas” is made up of 15 boxes:

cabbages and peas

whereas the word “cauliflower” consists of 10 boxes:<sup>4</sup>

cauliflower

More complicated boxes are made up of smaller boxes. We have already encountered one of these more complicated boxes: the `tabular` environment, discussed in [the previous section](#). This type of box is called a horizontal box, which means that it can go in a line of text. For example:

```
Here is some text.  
\begin{tabular}{cc}  
A & B\\  
C & D  
\end{tabular}  
The rest of the line.
```

↑Input

↓Input

produces:

↑Output

---

<sup>4</sup>The fl-ligature is a single character, and so is a single box, not two.



#### 4. CREATING A SIMPLE DOCUMENT

Here is some text. 

|   |   |
|---|---|
| A | B |
| C | D |

 The rest of the line.

↓Output

In fact, you may have noticed in [the previous section](#), that the `tabular` environment had an optional argument `<pos>`. This governs the vertical alignment when the `tabular` environment occurs within a line of text. This can be one of `c` (centred—the default, as illustrated above), `t` (top) and `b` (bottom). For example,

```
Here is some text.  
\begin{tabular}[b]{cc}  
A & B\\  
C & D  
\end{tabular}  
The rest of the line.
```

↑Input

↓Input

produces:

↑Output

#### 4. CREATING A SIMPLE DOCUMENT

```
                A   B  
Here is some text. C   D   The rest of the line.
```

[↓ Output](#)

Another type of box which can again be placed in a line of text, is the `minipage` environment.

```
\begin{minipage}[<pos>][<height>]{<width>}
```

Definition

As the name suggests, this environment creates a “mini-page” of the given width. For example:

```
Some text.  
\begin{minipage}{2in}  
This is a mini-page. The text inside  
it is formatted as usual.
```

[↑ Input](#)

```
Paragraph breaks can also be used, but  
there is no indentation by default\footnote{and  
this is how a footnote appears}.  
\end{minipage}
```

#### 4. CREATING A SIMPLE DOCUMENT

The rest of the line.

↓Input

which produces

This is a mini-page. The text inside it  
is formatted as usual.

↑Output

Some text. Paragraph breaks can also be used, but The rest of the line.  
there is no indentation by default<sup>a</sup>.

<sup>a</sup>and this is how a footnote appears

↓Output

You can optionally specify a height, and how the mini-page is aligned with the rest of the text. As with the `tabular` environment, the alignment option `<pos>` can be one of `t` (top), `c` (centred) or `b` (bottom). The default is `c`, which is why the above example has the mini-page centred vertically. This can be changed, for example:

Some text.

↑Input

#### 4. CREATING A SIMPLE DOCUMENT

```
\begin{minipage}[t]{2in}
```

```
This is a mini-page. The text inside  
it is formatted as usual.
```

```
Paragraph breaks can also be used, but  
there is no indentation by default\footnote{and  
this is how a footnote appears}.
```

```
\end{minipage}
```

```
The rest of the line.
```

↓ Input

which produces

```
Some text. This is a mini-page. The text inside it The rest of the line.  
is formatted as usual.
```

↑ Output

```
Paragraph breaks can also be used, but  
there is no indentation by defaulta.
```

---

<sup>a</sup>and this is how a footnote appears

↓ Output

#### 4. CREATING A SIMPLE DOCUMENT

Note that the width can be specified relative to the current line width, using `\linewidth`. For example,

```
\begin{minipage}{0.5\linewidth}
```

will start a mini-page that is half the width of the current line.

There is also a corresponding command

```
\parbox[<pos>][<height>]{<width>}{<text>}
```

Definition

which behaves in a similar way. So the above example, can be rewritten using a `\parbox`:

---

Some text.

```
\parbox[t]{2in}{This is a parbox.
```

```
The text inside
```

```
it is formatted as usual.
```

```
Paragraph breaks can also be used, but  
there is no indentation by default.}
```

```
The rest of the line.
```

↑ Input

#### 4. CREATING A SIMPLE DOCUMENT

↓ Input

which produces

Some text. This is a parbox. The text inside it is The rest of the line.  
formatted as usual.  
Paragraph breaks can also be used, but  
there is no indentation by default.

↑ Output

↓ Output

You may have noticed that the `\footnote` command has not been used in the above example. The `\parbox` is designed for a small amount of text, and is more restricted than the `minipage` environment, so you can't use the `\footnote` command in it. There are also certain environments, such as the [list-making environments](#), that can be used in a `minipage`, but not in a `\parbox`.

Related UK TUG FAQ [2] topics:

- [Automatic sizing of minipage](#)
- [Float\(s\) lost](#)
- [Perhaps a missing \item?](#)

# Chapter 5

## Structuring Your Document

Let's go back to the document we modified in [exercise 6](#). In this chapter we shall edit this document step by step until we have a fully fledged document with title, abstract, table of contents, sections etc.

### 5.1 Author and title information

The term “title page” is used to indicate the author, title and date information that can either appear on the front cover by itself or along the top of the first page of text. In order to do this, you must first specify the information. Once this information has been specified it can then be displayed.

The author, title and date are entered using the [commands](#):

```
\author{<author names>}  
\title{<title text>}  
\date{<document date>}
```

Definition

## 5. STRUCTURING YOUR DOCUMENT

These commands only *store* information, they don't actually display anything. Once you have used these commands, you can then display the information using the command:

[The style of document titles]

```
\maketitle
```

Definition

Note that if you don't use the `\date` command, the current date will be inserted. If you want no date to appear, you need to specify an empty argument:

```
\date{}
```

Input

Multiple authors should be separated by the command `\and`, for example:

```
\author{A. Jones\\University of Somewhere \and  
B. Smith\\University of Somewhere Else}
```

↑Input

↓Input



## 5. STRUCTURING YOUR DOCUMENT

Within these titling fields, you can also use the command:

```
\thanks{text}
```

Definition

which produces a special type of footnote. For example:

```
\title{A Great Project\thanks{funded by XYZ}}
```

Input

Note that the footnote marker produced using `\thanks` is considered to have zero width, so if it occurs in the middle of a line, rather than the end, you will need to insert some extra space using `\_` (backslash space). The argument of `\thanks` is a [moving argument](#).

### Exercise 7 (Creating Title Pages)

Try editing the document you modified in [exercise 6](#) to include title information. Modifications are illustrated [like this](#):

```
\documentclass[a4paper,11pt]{article}
```

Code

## 5. STRUCTURING YOUR DOCUMENT

```
\begin{document}
```

```
\title{A Simple Document}
```

```
\author{Me}
```

```
\maketitle
```

This is a simple \LaTeX\ document.

Here is the first paragraph.

Here is the second paragraph. As you can see it's a very short document\footnote{with a footnote}.

This document was created on: \today.

```
\begin{tabular}{lrr}
```

```
& \multicolumn{2}{c}{\bfseries Expenditure}\\
```

```
& \multicolumn{1}{c}{Year1} & \multicolumn{1}{c}{Year2}\\
```

```
\bfseries Travel & 100,000 & 110,000\\
```

```
\bfseries Equipment & 50,000 & 60,000
```

```
\end{tabular}
```

```
\end{document}
```

You can [download](#) this document.

---

### 5.2 Abstract

The abstract `environment` is used to create an abstract for the document. The way in which the abstract is formatted depends on the class file. The `report` class file will put the abstract on a page by itself, some class files will indent the abstract and some will typeset the abstract in italic. Note also that some class files (such as `book` and `letter`) don't have an `abstract` environment. Abstracts traditionally go at the start of the document after the title, so the `abstract` environment should go after the `\maketitle` command.

[1-column  
abstract in  
2-column  
document]

#### Exercise 8 (Creating an Abstract)

Try editing your document so that it has an abstract: Modifications are illustrated [like this](#):

## 5. STRUCTURING YOUR DOCUMENT

Code

```
\documentclass[a4paper,11pt]{article}

\begin{document}

\title{A Simple Document}
\author{Me}

\maketitle

\begin{abstract}
A brief document to illustrate how to use \LaTeX.
\end{abstract}

This is a simple \LaTeX\ document.
Here is the first paragraph.

Here is the second paragraph. As you can see it's a very
short document\footnote{with a footnote}.
This document was created on: \today.

\begin{tabular}{lrr}
```

## 5. STRUCTURING YOUR DOCUMENT

```
& \multicolumn{2}{c}{\bfseries Expenditure}\\
& \multicolumn{1}{c}{Year1} & \multicolumn{1}{c}{Year2}\\
\bfseries Travel & 100,000 & 110,000\\
\bfseries Equipment & 50,000 & 60,000
\end{tabular}

\end{document}
```

[↓ Code](#)

You can [download](#) this document.

---

### 5.3 Chapters, Sections, Subsections ...

Chapters, sections, subsections etc can be inserted using the commands:

## 5. STRUCTURING YOUR DOCUMENT

```
\part [<short title>]{<title>}
\chapter [<short title>]{<title>}
\section [<short title>]{<title>}
\subsection [<short title>]{<title>}
\subsubsection [<short title>]{<title>}
\paragraph [<short title>]{<title>}
\subparagraph [<short title>]{<title>}
```

Definition

(All these commands have a [moving argument](#), so fragile commands will need to be protected using `\protect`.)

**Note 3:** The availability of these commands depends on the class file you are using. For example, the `article` class file that we have been using is designed for short articles, so the `\chapter` command is not defined in the `article` class file, whereas it is defined in the `report` class file.

Each of the commands above have a [mandatory argument](#) `<title>` and an [optional argument](#) `<short title>`. The mandatory argument `<title>` is simply the title of the chapter/section/subsection etc. For example:

[How to create  
a  
`\subsubsection`  
Input

```
\section{Introduction}
```

If you are using the `article` class file, the output will look like:

## 1 Introduction

Output

Note that you don't specify the section number as L<sup>A</sup>T<sub>E</sub>X does this automatically. This means that you can insert a new section or chapter or swap sections around or even change a section to a subsection etc, without having to worry about updating all the section numbers.

[The style of section headings]

If you are using a class file that contains chapters as well as sections, the section number will depend on the chapter. So, for example, the current section is the 3<sup>rd</sup> section of chapter 5, so the section number is 5.3 (note that if you are using a class file where the section number depends on the chapter number, you must have a `\chapter` command before your first `\section` command, otherwise your section numbers will come out as 0.1, 0.2 etc).

[Why are my sections numbered 0.1 ...?]

Unnumbered chapters/sections etc are produced by placing an asterisk `*` after the command name. For example:

```
\chapter*{Acknowledgements}
```

Input

You can switch to appendices using the command

[Appendixes]

```
\appendix
```

Definition

## 5. STRUCTURING YOUR DOCUMENT

then continue using `\chapter`, `\section` etc. For example (using the `report` class file):

```
\appendix
\chapter{Derivations}
Some derivations.
```

```
\chapter{Tables}
Some tables.
```

↑Input

↓Input

### Exercise 9 (Creating Chapters, Sections etc)

Let's try editing our document so that it now has chapters, sections and an appendix. Since the `article` class file doesn't have chapters, let's change to the `report` class. Changes from our previous document are shown like this.

↑Code



## 5. STRUCTURING YOUR DOCUMENT

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

\title{A Simple Document}
\author{Me}

\maketitle

\begin{abstract}
A brief document to illustrate how to use \LaTeX.
\end{abstract}

\chapter{Introduction}
\section{The First Section}

This is a simple \LaTeX\ document.
Here is the first paragraph.

\section{The Next Section}

Here is the second paragraph. As you can see it's a very
short document\footnote{with a footnote}.
This document was created on: \today.
```

## 5. STRUCTURING YOUR DOCUMENT

```
\chapter{Another Chapter}
```

Here's another very interesting chapter.  
We're going to put a picture here later.

```
\chapter*{Acknowledgements}
```

I would like to acknowledge all those  
very helpful people who have assisted me in my work.

```
\appendix
```

```
\chapter{Tables}
```

We will turn this tabular environment into a table later.

```
\begin{tabular}{lrr}  
  & \multicolumn{2}{c}{\bfseries Expenditure}\\  
  & \multicolumn{1}{c}{Year1} & \multicolumn{1}{c}{Year2}\\  
\bfseries Travel & 100,000 & 110,000\\  
\bfseries Equipment & 50,000 & 60,000  
\end{tabular}
```

```
\end{document}
```

(You can [download](#) a copy of this file if you like, but I recommend that you try editing the file yourself to give you practice.)

---

### 5.4 Creating a Table of Contents

Once you have all your `\chapter`, `\section` etc commands, you can create a table of contents with the command

```
\tableofcontents
```

This command should go where you want your table of contents to appear (usually after `\maketitle`).

You may recall from [the previous section](#) that the sectioning commands all had an optional argument `<short title>`. If your chapter or section title is particularly long, you can use `<short title>` to specify a shorter title that should go in the table of contents.<sup>1</sup> The longer title (given by the

[The format of the Table of Contents, etc]

Definition

[My section title is too wide for the page header]

---

<sup>1</sup>and in the page header, depending on the page style.

## 5. STRUCTURING YOUR DOCUMENT

other argument `<title>`) will still appear in the section heading in the main part of the document.

L<sup>A</sup>T<sub>E</sub>X processes all source code sequentially, so when it first encounters the `\tableofcontents` command, it doesn't yet know anything about the chapters, sections etc. So the first time the document is L<sup>A</sup>T<sub>E</sub>Xed the necessary information is written to the table of contents (`.toc`) file. The subsequent pass reads the information in from the `.toc` file, and generates the table of contents. You will therefore need to L<sup>A</sup>T<sub>E</sub>X your document twice to make sure that the table of contents is up-to-date.

[Numbers too large in table of contents, etc]

### Exercise 10 (Creating a Table of Contents)

Try modifying your document so that it has a table of contents. Modifications from the previous exercise are illustrated like this:

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

\begin{document}

\title{A Simple Document}
```

[↑ Code](#)

## 5. STRUCTURING YOUR DOCUMENT

```
\author{Me}
```

```
\maketitle
```

```
\tableofcontents
```

```
\begin{abstract}
```

```
A brief document to illustrate how to use \LaTeX.
```

```
\end{abstract}
```

```
\chapter{Introduction}
```

```
\section{The First Section}
```

```
This is a simple \LaTeX\ document. Here is the first paragraph.
```

```
\section{The Next Section}
```

```
Here is the second paragraph. As you can see it's a very  
short document\footnote{with a footnote}.
```

```
This document was created on: \today.
```

```
\chapter{Another Chapter}
```

## 5. STRUCTURING YOUR DOCUMENT

Here's another very interesting chapter.  
We're going to put a picture here later.

```
\chapter*{Acknowledgements}
```

I would like to acknowledge all those  
very helpful people who have assisted  
me in my work.

```
\appendix  
\chapter{Tables}
```

We will turn this tabular environment into a table later.

```
\begin{tabular}{lrr}  
  & \multicolumn{2}{c}{\bfseries Expenditure}\\  & \multicolumn{1}{c}{Year1} & \multicolumn{1}{c}{Year2}\\ \bfseries Travel & 100,000 & 110,000\\ \bfseries Equipment & 50,000 & 60,000  
\end{tabular}
```

## 5. STRUCTURING YOUR DOCUMENT

```
\end{document}
```

[↓ Code](#)

If your table of contents doesn't come out right, try  $\LaTeX$ ing it again. (Again, you can [download](#) this file.)

---

### 5.5 Cross-Referencing

We have already seen that  $\LaTeX$  takes care of all the numbering for the chapters etc, but what happens if you want to refer to a chapter or section? There's no point leaving  $\LaTeX$  to automatically generate the section numbers if you have to keep a track of them all, and change all your cross-references every time you add a new section. Fortunately  $\LaTeX$  provides a way to generate the correct number, all you have to do is label the part of the document you want to reference, and then refer to this label when you want to cross-reference it.  $\LaTeX$  will then determine the correct number that needs to be inserted at that point.

[Referring to labels in other documents]

The first part, labelling the place you want to reference, is done using the command:

## 5. STRUCTURING YOUR DOCUMENT

```
\label{<string>}
```

Definition

The **argument** *<string>* should be a unique textual label. This label can be anything you like as long as it is unique, but it's a good idea to make it something obvious so that, firstly, you can remember the label when you want to use it, and secondly, when you read through your code at some later date, it's immediately apparent to you to which part of the document you are referring. People tend to have their own conventions for labelling. I usually start the label with two or three letters that signify what type of thing I'm labelling. For example, if I'm labelling a chapter I'll start with **ch**, if I'm labelling a section I'll start with **sec**. Example:

```
\chapter{Introduction}  
\label{ch:intro}
```

↑Input

↓Input

Another example:

```
\section{Technical Details}  
\label{sec:details}
```

↑Input



## 5. STRUCTURING YOUR DOCUMENT

↓Input

Note that the `\label` command doesn't produce any text, it simply assigns a label. You can now refer to that object using the command:

`\ref{<string>}`

Definition

which will produce the relevant number.

Example:

[Referring to things by their name]

↑Input

See Section `\ref{sec:results}` for an analysis of the results.

↓Input

It is a typographical convention that you should never start a new line with a number. For example, if you have the text “Chapter 1” the “1” must be on the same line as the “Chapter”. We can do this by using an unbreakable space, which will put a space but won't allow L<sup>A</sup>T<sub>E</sub>X to break the line at that point. This is done using the ~ [special character](#), so the example above should actually be:

↑Input

## 5. STRUCTURING YOUR DOCUMENT

See Section~\ref{sec:results} for an analysis of the results.

[↓Input](#)

There is a similar command:

```
\pageref{<string>}
```

Definition

which will insert the page number that the label appeared on. Example:

[↑Input](#)

```
See Chapter~\ref{ch:def} on  
page~\pageref{ch:def} for a list of definitions.
```

[↓Input](#)

The label `ch:def` obviously needs to be defined somewhere:

[↑Input](#)

```
\chapter{Definitions}  
\label{ch:def}
```

## 5. STRUCTURING YOUR DOCUMENT

[↓ Input](#)

In fact, I have done this in my source code for [chapter 2](#) of this document, so the above example would look like:

See Chapter 2 on page 11 for a list of definitions.

[Output](#)

It's not just chapters and sections that you can reference, most of the numbers that L<sup>A</sup>T<sub>E</sub>X automatically generates can be cross-referenced. The `enumerate` environment automatically numbers the items within an ordered list, so it's possible to label list items. For example:

[↑ Input](#)

```
\begin{enumerate}

  \item\label{itm:edit} Write or edit source code.

  \item Pass source code to the \LaTeX\ application
    (''\LaTeX\ the document'').

\begin{itemize}
```

## 5. STRUCTURING YOUR DOCUMENT

```
\item If there are any error messages,  
      return to Step~\ref{itm:edit}.
```

```
\item If there are no error messages, a DVI file  
      is created, go to Step~\ref{itm:view}.
```

```
\end{itemize}
```

```
\item\label{itm:view} View DVI file to check the result.
```

```
\end{enumerate}
```

[↓ Input](#)

Output:

[↑ Output](#)

1. Write or edit source code.
2. Pass source code to the L<sup>A</sup>T<sub>E</sub>X application (“L<sup>A</sup>T<sub>E</sub>X the document”).
  - If there are any error messages, return to Step 1.
  - If there are no error messages, a DVI file is created, go to Step 3.

## 5. STRUCTURING YOUR DOCUMENT

### 3. View DVI file to check the result.

[↓ Output](#)

The `\ref` and `\pageref` commands may come before or after the corresponding `\label` command. As with the table of contents, L<sup>A</sup>T<sub>E</sub>X first writes out all the cross-referencing information to another file (the auxiliary `.aux` file) and then reads it in the next time, so you will need to L<sup>A</sup>T<sub>E</sub>X your document twice to get everything up-to-date.

If the references aren't up-to-date, you will see the following message at the end of the L<sup>A</sup>T<sub>E</sub>X run:

```
LaTeX Warning: Label(s) may have changed.  
Rerun to get cross-references right.
```

The following warning

```
LaTeX Warning: There were undefined references.
```

means that L<sup>A</sup>T<sub>E</sub>X found a reference to a label that does not appear in the auxiliary file. This could mean that it's a new label, and the warning will go away the next time you L<sup>A</sup>T<sub>E</sub>X your document, or it could mean that either you've forgotten to define your label with the `\label` command, or you've simply misspelt the label.

## 5. STRUCTURING YOUR DOCUMENT

Very occasionally, if you have cross-references and a table of contents, you might have to  $\LaTeX$  your document three times to get everything up to date. Just check to see if the `Label(s) may have changed` warning appears.

If you have an undefined reference,  $\LaTeX$  will replace the reference number with two question marks ?? in the output. If this happens, check to see if the above warnings have occurred.

[“Rerun”  
messages won’t  
go away]

[LaTeX gets  
cross-references  
wrong]

### Exercise 11 (Cross-Referencing)

Try modifying your code so that it has cross-references. Again, changes made from the previous document are illustrated like this:

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

\begin{document}

\title{A Simple Document}
\author{Me}
```

↑Code

## 5. STRUCTURING YOUR DOCUMENT

```
\maketitle
```

```
\tableofcontents
```

```
\begin{abstract}
```

A brief document to illustrate how to use \LaTeX.

```
\end{abstract}
```

```
\chapter{Introduction}
```

```
\label{ch:intro}
```

```
\section{The First Section}
```

This is a simple \LaTeX\ document. Here is the first paragraph.

The next chapter is Chapter~\ref{ch:another}

and is on page~\pageref{ch:another}.

The next section is Section~\ref{sec:next}.

```
\section{The Next Section}
```

```
\label{sec:next}
```

Here is the second paragraph. As you can see it's a very

## 5. STRUCTURING YOUR DOCUMENT

short document\footnote{with a footnote}.  
This document was created on: \today.

```
\chapter{Another Chapter}  
\label{ch:another}
```

Here's another very interesting chapter.  
We're going to put a picture here later.  
See Chapter~\ref{ch:intro} for an  
introduction.

```
\chapter*{Acknowledgements}
```

I would like to acknowledge all those  
very helpful people who have assisted  
me in my work.

```
\appendix  
\chapter{Tables}
```

We will turn this tabular environment into a table later.

```
\begin{tabular}{lrr}
```



## 5. STRUCTURING YOUR DOCUMENT

```
& \multicolumn{2}{c}{\bfseries Expenditure}\\
& \multicolumn{1}{c}{Year1} & \multicolumn{1}{c}{Year2}\\
\bfseries Travel & 100,000 & 110,000\\
\bfseries Equipment & 50,000 & 60,000
\end{tabular}
```

[↓ Code](#)

(You can [download](#) a copy of this file.)

---

## 5.6 Creating a Bibliography

Bibliographies can be created using the `thebibliography` environment. This environment is very similar to the list making environments described in [section 4.3](#), but instead of `\item` use

```
\bibitem[<label>]{<key>}
```

Definition

where `<key>` is a unique keyword that identifies this item. Your keyword can be anything you like, but as with `\label` I recommend that you use

## 5. STRUCTURING YOUR DOCUMENT

a short memorable keyword. I tend to use the first author's surname followed by the year of publication. Example:

```
\begin{thebibliography}{1}
\bibitem{lampport94} ‘‘\LaTeX\ : a document preparation
system’’, Leslie Lamport, 2nd edition (updated for
\LaTeXe), Addison-Wesley (1994).

\bibitem{kopka95} ‘‘A Guide to \LaTeX2e: document
preparation for beginners and advanced users’’,
Helmut Kopka and Patrick W. Daly, Addison-Wesley (1995).

\bibitem{goossens94} ‘‘The \LaTeX\ Companion’’,
Michel Goossens, Frank Mittelbach and
Alexander Samarin, Addison-Wesley, (1994).

\end{thebibliography}
```

↑Input

↓Input

Output:

↑Output

## References

- [1] “ $\text{\LaTeX}$  : a document preparation system”, Leslie Lamport, 2nd edition (updated for  $\text{\LaTeX} 2\epsilon$ ), Addison-Wesley (1994).
- [2] “A Guide to  $\text{\LaTeX}2\epsilon$ : document preparation for beginners and advanced users”, Helmut Kopka and Patrick W. Daly, Addison-Wesley (1995).
- [3] “The  $\text{\LaTeX}$  Companion”, Michel Goossens, Frank Mittelbach and Alexander Samarin, Addison-Wesley, (1994).

↓ Output

You can cite an item in your bibliography with the command

```
\cite[<text>]{<key list>}
```

Definition

Example:

```
Goossens \emph{et al.}\cite{goossens94}.
```

↑ Input

## 5. STRUCTURING YOUR DOCUMENT

↓Input

Output:

For more information about writing bibliographies see Goossens *et al.* [3].

Output

If you want to cite multiple works, use a comma-separated list: Example:

↑Input

```
For more information about writing bibliographies  
see~\cite{kopka95,goossens94}.
```

↓Input

Output:

For more information about writing bibliographies see [2, 3].

Output

The [optional argument](#) `<text>` to the `\cite` command can be used to add text to the citation. Example:

↑Input

## 5. STRUCTURING YOUR DOCUMENT

For more information about writing bibliographies see Goossens `\emph{et al.}`~`\cite[Chapter~13]{goossens94}`.

↓ Input

Output:

For more information about writing bibliographies see Goossens *et al.* [3, Chapter 13].

↑ Output

↓ Output

The `thebibliography` environment has a **mandatory argument**:

```
\begin{thebibliography}{<widest entry>}
```

Definition

The argument `<widest entry>` is the widest label in the list of entries. This helps L<sup>A</sup>T<sub>E</sub>X to align the references correctly. In the example above, the labels appeared as: [1], [2] and [3], but they can be changed using the optional argument to the `\bibitem` command. In the above example, the labels were all approximately the same width so the argument `{1}` was used (although `{2}` and `{3}` could just have easily been used—in fact `{3}` is strictly speaking the widest). Consider the following example:

## 5. STRUCTURING YOUR DOCUMENT

↑ Input

```
\begin{thebibliography}{Goossens 1994}
\bibitem[Lamport 1994]{lamport94} ‘‘\LaTeX\ : a document
preparation system’’, Leslie Lamport, 2nd edition
(updated for \LaTeX2e), Addison-Wesley (1994).

\bibitem[Kopka 1995]{kopka95} ‘‘A Guide to \LaTeX2e: document
preparation for beginners and advanced users’’, Helmut Kopka
and Patrick W. Daly, Addison-Wesley (1995).

\bibitem[Goossens 1994]{goossens94} ‘‘The \LaTeX\ Companion’’,
Michel Goossens, Frank Mittelbach and
Alexander Samarin, Addison-Wesley, (1994).

\end{thebibliography}
```

↓ Input

Output:

**References**

↑ Output

## 5. STRUCTURING YOUR DOCUMENT

- [Lamport 1994] “ $\text{\LaTeX}$  : a document preparation system”, Leslie Lamport, 2nd edition (updated for  $\text{\LaTeX}$  2 $\epsilon$ ), Addison-Wesley (1994).
- [Kopka 1995] “A Guide to  $\text{\LaTeX}$ 2 $\epsilon$ : document preparation for beginners and advanced users”, Helmut Kopka and Patrick W. Daly, Addison-Wesley (1995).
- [Goossens 1994] “The  $\text{\LaTeX}$  Companion”, Michel Goossens, Frank Mittelbach and Alexander Samarin, Addison-Wesley, (1994).

↓ Output

In this example, the widest label is [Goossens 1994] so it is chosen to be the argument of the `thebibliography` environment:

```
\begin{thebibliography}{Goossens 1994}
```

Input

There is an application called  $\text{\BibTeX}$  that can be used in conjunction with  $\text{\LaTeX}$  to help generate bibliographies. This document does not cover  $\text{\BibTeX}$ , but if you are interested I recommend reading *A Guide to  $\text{\LaTeX}$*  [4] or *The  $\text{\LaTeX}$  Companion* [5]. For those of you who want a

[Creating a  
BibTeX  
bibliography  
file]

## 5. STRUCTURING YOUR DOCUMENT

quick look on-line, the document *Using L<sup>A</sup>T<sub>E</sub>X to Write a PhD Thesis* has a section containing a brief introduction to B<sup>I</sup>B<sub>T</sub>E<sub>X</sub>.

### Exercise 12 (Creating a Bibliography)

Try adding the following chapter to your document:

```
\chapter{Recommended Reading}
```

↑ Input

```
For a basic introduction to \LaTeX\ see Lamport~\cite{lampport94}.  
For more detailed information about \LaTeX\ and  
associated applications, consult Kopka and Daly~\cite{kopka95}  
or Goossens \emph{et al}~\cite{goossens94}.
```

↓ Input

and also add the bibliography shown above to the end of your document. You can [download](#) or [view](#) the solution, but have a go by yourself first. Remember that, as before, you will need to L<sup>A</sup>T<sub>E</sub>X the document twice to get the references up-to-date.



## 5.7 Page Styles and Page Numbering

You may have noticed that the documents you have created have all had their page numbers automatically inserted at the foot of most of the pages. If you have created the document that has gradually been modified over the previous few sections, you may have noticed that the title page has no header or footer, the table of contents is page 1, the abstract page has no page number, and the page after the abstract starts at page 1 and continues incrementally onwards from that point. All the page numbers are Arabic numbers. This can be changed using the command:

[Page numbering “<n> of <m>”]

```
\pagenumbering{<style>}
```

Definition

where <style> can be one of:

**arabic** Arabic page numbers (1, 2, 3, ...)

**roman** Lowercase Roman numerals (i, ii, iii, ...)

**Roman** Uppercase Roman numerals (I, II, III, ...)

**alph** Lower case alphabetical characters (a, b, c, ...)

**Alph** Upper case alphabetical characters (A, B, C, ...)

## 5. STRUCTURING YOUR DOCUMENT

Traditionally, the front matter (table of contents, list of figures etc) should have lowercase Roman numeral page numbering, while the main matter should be in Arabic numerals. Example (using `report` class file):

[Page numbering by chapter]

```
\author{Me}
\title{A Simple Document}
\maketitle

\pagenumbering{roman}
\tableofcontents

\begin{abstract}
This is the abstract.
\end{abstract}

\pagenumbering{arabic}
\chapter{Introduction}
```

↑Input

↓Input

Note that if you don't have an `abstract` environment, you will need to do `\clearpage` before doing `\pagenumbering{arabic}`:

## 5. STRUCTURING YOUR DOCUMENT

```
\author{Me}
\title{A Simple Document}
\maketitle

\pagenumbering{roman}
\tableofcontents

\clearpage\pagenumbering{arabic}
\chapter{Introduction}
```

↑ Input

↓ Input

The headers and footers can be changed using the command

```
\pagestyle{<style>}
```

Individual pages can be changed using

```
\thispagestyle{<style>}
```

Standard styles are:

[Alternative  
head- and  
footlines in  
LaTeX]

Definition

Definition

## 5. STRUCTURING YOUR DOCUMENT

- empty** No header or footer.
- plain** Header empty, page number in footer.
- headings** Header contains page number and various information, footer empty.
- myheadings** Header specified by user, footer empty.

If the **myheadings** style is used, the header information can be specified using:

`\markboth{<left head>}{<right head>}` Definition

if the **twoside** option has been passed to the [class file](#), or

`\markright{<right head>}` Definition

if the **oneside** option has been passed to the [class file](#) (default for **article** and **report**).

The **report** class file uses the **empty** style for the title and abstract pages and **plain** for the first page of each new chapter. By default the remaining pages are also **plain**, but these can be changed using the `\pagestyle`

## 5. STRUCTURING YOUR DOCUMENT

command. The A4 version of this document uses the `headings` page style, whereas this version uses a page style I defined myself that incorporates a navigation bar in the footer. (For information on how to do this, see *Creating a PDF Document using PDFLaTeX*.)

### Exercise 13 (Page Styles and Page Numbering)

Try editing your document so that the page numbering is lowercase Roman for the table of contents but Arabic for the main matter. You can try changing the page style as well, but since the chapters are less than a page each, you won't see any effect until we make our chapters a bit bigger. (You can [download](#) or [view](#) the edited document.)

---

# Chapter 6

## Packages

Packages are files with the extension `.sty` that either define new [commands](#) or redefine existing commands. We shall first look at how to [use](#) packages already installed on your system, and then we shall look at how to [download and install new packages](#).

[[What are LaTeX classes and packages](#)]

### 6.1 Using Packages

L<sup>A</sup>T<sub>E</sub>X has a great many useful commands, but it doesn't have a command to do absolutely everything, so if additional commands are required, they can be supplied in files called packages. If you want to use any [commands](#) or [environments](#) that are defined in a package, you first need to specify the name of the package with the command:

[[Documentation of packages](#)]

```
\usepackage[<options>]{<package name>}
```

Definition

## 6. PACKAGES

where  $\langle package\ name \rangle$  is the name of the package without the `.sty` extension, and  $\langle options \rangle$  is a comma separated list of options to be passed to the package (just as you can do with class files using the `\documentclass` command). Note that the `\usepackage` command must *always* go in the [preamble](#).

Let's look at a few examples.

### 6.1.1 The `graphicx` Package

It is possible to generate images using  $\text{\LaTeX}$  commands (see *The  $\text{\LaTeX}$  Graphics Companion* [6]) however most people find it easier to create a picture in some other application, and include that file into their  $\text{\LaTeX}$  document.

Some applications have an option that allows you to save an image as an Encapsulated PostScript (EPS) file. Alternatively, there are utilities available that will convert other file types to EPS such as: `pdftops`, `tiff2ps`, `pnmtops`. The `graphicx` package provides a command that enables you to include this EPS file into your document.<sup>1</sup>

Firstly, you need to specify that you want to use the `graphicx` package. So you will need to place the following command in the [preamble](#):

---

<sup>1</sup> $\text{\PDF\LaTeX}$  doesn't load EPS files, but instead can load PNG or Encapsulated PDF images.

[Drawing with TeX]

[How to import graphics into (La)TeX documents]

[What is "Encapsulated PostScript" ("EPS")]

## 6. PACKAGES

```
\usepackage{graphicx}
```

Input

The EPS file can then be included in your document using the command

```
\includegraphics[<key vals>]{<filename>}
```

Definition

where *<filename>* is the name of your EPS file, and *<key vals>* is a comma-separated list of options that can be used to manipulate the image.

Example: suppose you had a file called `shapes.ps`, then to include it in your document you would do:

```
\includegraphics{shapes.ps}
```

Input

Output:



## 6. PACKAGES



Output

If you omit the file extension,  $\LaTeX$  will search for a file with the default extension. If you are using ordinary  $\LaTeX$ , this will usually be `.ps` or `.eps`, however if you are using  $\text{PDF}\LaTeX$ , this will usually be `.pdf` or `.png`. Modifying the above example, we could do:

[Portable  
imported  
graphics]

```
\includegraphics{shapes}
```

Input

If we use  $\LaTeX$ , the file `shapes.ps` will be used, and if we use  $\text{PDF}\LaTeX$ ,

## 6. PACKAGES

the file `shapes.pdf` will be used. So, if you sometimes use  $\LaTeX$  and sometimes use `PDF $\LaTeX$` , you may find it easier to omit the extension, and have two copies of the image in both EPS and PDF format.

You can specify a full or relative pathname, but you must use a forward slash as the directory divider, even if you are using Windows. For example:

```
\includegraphics{pictures/shapes.ps}
```

means the file `pictures/shapes.ps` on Unix-type systems, and it means the file `pictures\shapes.ps` on Windows. This is mainly because the backslash character is a  $\LaTeX$  special character indicating a command, but it also helps portability between platforms.

You can specify which file types to look for with the command

```
\DeclareGraphicsExtensions{<ext-list>}
```

Definition

where `<ext-list>` is a comma-separated list of extensions. For example, if you are using `PDF $\LaTeX$` , you might want to search first for PDF files, and then for PNG files:

```
\DeclareGraphicsExtensions{.pdf,.png}
```

Input

or if you are using  $\LaTeX$  and `dvips`, you might want to first search for Encapsulated PostScript (EPS) files and then for PostScript (PS) files:

## 6. PACKAGES

`\DeclareGraphicsExtensions{.eps,.ps}`

Input

The **optional argument** `<key vals>` should be a comma separated list of `<key>=<label>` pairs. Common options are:

|                                                           |                                                                                                                                                                                                                                                                     |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>angle=x</code>                                      | rotate the picture by $x^\circ$                                                                                                                                                                                                                                     |
| <code>width=&lt;len&gt;</code>                            | scale the picture so that the width is <code>&lt;len&gt;</code> .<br>(Remember to specify the units)                                                                                                                                                                |
| <code>height=&lt;len&gt;</code>                           | scale the picture so that the height is <code>&lt;len&gt;</code> .<br>(Remember to specify the units)                                                                                                                                                               |
| <code>scale=&lt;value&gt;</code>                          | Scale the picture by <code>&lt;value&gt;</code>                                                                                                                                                                                                                     |
| <code>trim=&lt;l&gt; &lt;b&gt; &lt;r&gt; &lt;t&gt;</code> | Specifies the amount to remove from each side.<br>E.g. <code>trim=1 2 3 4</code> crops the picture by 1bp from the left, 2bp from the bottom, 3bp from the right and 4bp from the top. (The unit <code>bp</code> is a PostScript point $72\text{bp} = 1\text{in}$ ) |
| <code>draft</code>                                        | Don't actually print the image, just draw a box of the same size and print the filename inside it.                                                                                                                                                                  |

## 6. PACKAGES

Let's try rotating and scaling our picture:

```
\includegraphics[angle=45,width=1in]{shapes}
```

Input

Output:



Output

### Graphical Transformations

The `graphicx` package also provides commands to rotate, resize, reflect and scale text. They are as follows:

- `\rotatebox{<angle>}{<text>}`

Example:

## 6. PACKAGES

```
\rotatebox{45}{Some text}
```

Input

Output:

Some text

Output

- `\scalebox{<h scale>}[<v scale>]{<text>}`

Example:

```
\scalebox{0.8}{Some text}
```

Input

Output:

Some text

Output

## 6. PACKAGES

- `\reflectbox{<text>}`

Example:

```
\reflectbox{Some text}
```

Input

Output:

txət əmoɹ

Output

- `\resizebox{<h length>}{<v length>}{<text>}`

Example:

```
\resizebox{12mm}{1cm}{Some text}
```

Input

Output:

Some text

Output

## 6. PACKAGES

The `graphicx` package can have the following options passed to it:

**draft** Don't actually display the images, just print the filename in a box of the correct size. This is useful if you want to print out a draft copy of a document to check the text rather than the images. This option can also be passed to the class file.

**final** Opposite of **draft** (default). This option can also be passed to the class file.

**hiderotate** Don't show rotated text.

**hidescale** Don't show scaled text.

Example:

```
\usepackage[draft]{graphicx}
```

Input

## Exercise 14 (Using the `graphicx` Package)

Download the file `shapes.ps` from <http://theoval.cmp.uea.ac.uk/~nlct/latex/novices/exercises/>, and include it into your document. Alternatively, if you prefer to use PDF $\LaTeX$ , you can download the file `shapes.pdf` instead. Try experimenting with some of the options described above. (You can [download](#) or [view](#) an example solution.)

---

Some previewers may not be able to display EPS images or perform the scaling, rotating etc, in this case you can use `dvips` to convert your DVI file into a PostScript file either calling `dvips` in a terminal, or clicking on the appropriate button or setting in `WinEdt` or `TeXnicCenter`, and then view it using `GSview`.

For more information on the `graphicx` package see *The  $\LaTeX$  Graphics Companion* [6].

Related UK TUG FAQ [2] topics:

- [How to import graphics into \(La\)TeX documents](#)
- [Imported graphics in PDFLaTeX](#)
- [Imported graphics in dvips](#)



## 6. PACKAGES

- Imported graphics in dvipdfm
- Importing graphics from “somewhere else”
- Portable imported graphics
- Repeated graphics in a document
- Limit the width of imported graphics
- Top-aligning imported graphics
- Labelling graphics
- Graphics division by zero

### 6.1.2 Multi-Lingual Support: using the babel package

You may have noticed that the `\tableofcontents` and `\chapter` commands have produced English words like “Contents” and “Chapter”. If you are writing in another language, this is not appropriate. In this case, you should use the `babel` package, and specify which language you will be using, either as an option to the `babel` package, or as an option to the class file. If you are writing in more than one language, list all the

[How to change LaTeX’s “fixed names”]

[Using a new language with Babel]

## 6. PACKAGES

languages that you will be using, where the last named language, is the default language. For example:

[Parallel setting  
of text]

```
\usepackage[english,french]{babel}
```

or

```
\documentclass[english,french,a4paper]{report}  
\usepackage{babel}
```

You can then switch between the named languages either using the [declaration](#):

```
\selectlanguage{<language>}
```

Definition

or the `otherlanguage` [environment](#):

```
\begin{otherlanguage}{<language>}
```

Definition

These will affect all translations, including the date format and predefined names like “Chapter”. If you only want to set a short section of text in a different language, without affecting the date format or predefined names, then you can either use the command:

## 6. PACKAGES

```
\foreignlanguage{<language>}{<text>}
```

Definition

or the starred version of the `otherlanguage` environment:

```
\begin{otherlanguage*}{<language>}
```

Definition

You can determine if a given language is currently selected using:

```
\iflanguage{<language>}{<true text>}{<>false text>}
```

Definition

Example:

```
\documentclass{article}
```

```
\usepackage[english,french]{babel}
```

```
% french is the last named option, so that's the current language
```

```
\begin{document}
```

```
Ce texte est en fran\c{c}ais. La date aujourd'hui est: \today.
```

```
\selectlanguage{english}
```

```
This text is in English. Today's date is: \today.
```

```
\end{document}
```

### 6.1.3 Changing the format of `\today`

In the document we have been creating in the exercises, we have used the command `\today` to produce the current date. By default, this command displays the date in a US format, e.g. January 15, 2008, but if you live in the UK you might prefer a UK format. This can be done by loading a package that redefines the `\today` command. There are several packages available, amongst which are: `ukdate` and `datetime`. (If you are using the `babel` package, `\today` will display the date in the format for the currently selected language.)

For example, if you want to use the `ukdate` package, you would type the following in the [preamble](#):

```
\usepackage{ukdate}
```

Input

and the command `\today` will then display the date in the form: Tuesday 15<sup>th</sup> January, 2008

The `datetime` package has various options that can be used to change the format of `\today`. For example, by default the `datetime` package redefines `\today` to display the date in the form: Tuesday 15<sup>th</sup> January, 2008. The option `short` will produce an abbreviated form, (e.g. Tue 15<sup>th</sup> Jan, 2008) and the option `nodayofweek` won't display the day of the week (e.g. 15<sup>th</sup> January, 2008). These can be passed as a comma separated list in

## 6. PACKAGES

the [optional argument](#) to the `\usepackage` command. It is also possible to use a declaration instead. For example, to redefine `\today` to display the date in the form 15/01/2008, you can either do

```
\usepackage[ddmmyyyy]{datetime}
```

[Input](#)

or

```
\usepackage{datetime}  
\ddmmyyyydate
```

[↑Input](#)

[↓Input](#)

The `datetime` package also defines the command `\currenttime` which displays the current time, where again the format can be changed by the package options. So the option `12hr` will cause `\currenttime` to display the date in 12 hour format (e.g. 11:15am) and the option `24hr` will cause `\currenttime` to display the date in 24 hour format (e.g. 11:15).

## Exercise 15 (Using the datetime package)

Edit your document so that it uses the `datetime` package.<sup>2</sup> Experiment with the different package options, e.g.

```
\usepackage[short,nodayofweek,12hr]{datetime}
```

and add the current time

This document was created on: `\today\` at `\currenttime`.

For a full list of package options, see the [datetime documentation](#).

(You can [download](#) or [view](#) an example.)

---

## 6.2 Downloading and Installing Packages

New L<sup>A</sup>T<sub>E</sub>X packages are being created all the time, so you may find that there are some packages that you don't have on your installation. In this case, if you don't have the package you want, you can download it from the UK T<sub>E</sub>X Archive [9]. Before discussing installing new packages, it is necessary for you to understand the T<sub>E</sub>X Directory Structure (TDS).

---

<sup>2</sup>Note that if you are using the `babel` package, you will need to load `babel` before `datetime`. Read the `datetime` documentation for further details.

[Installing things on a (La)TeX system]

[Installing MiKTeX “known packages”]

[What is the TDS?]

## 6. PACKAGES

All the files that make up the T<sub>E</sub>X distribution are stored in a standard hierarchical structure. The root directory of the main distribution is called `texmf`. Its location depends on your system. For example, if you are using `teTeX`, it will probably be located in `/usr/share/texmf` or if you are using `MiKTeX` it may be located in `c:\texmf` or `c:\Program Files\texmf`. Whichever system you are using, I shall refer to this directory as `<TEXMF>`. So, if you are using `teTeX`, `<TEXMF>/doc` refers to the directory `/usr/share/texmf/doc`, or if you are using `MiKTeX`, `<TEXMF>\doc` refers to the folder `c:\texmf\doc` or `c:\Program Files\texmf\doc`.

You should also have a local `texmf` tree. This is where you should put any new packages that you download. That way, if you update your T<sub>E</sub>X distribution, you won't need to reinstall all those extra packages. Again, the location of the local `texmf` tree depends on your system. If you are using `teTeX`, it may be `/usr/local/texmf/` or `/usr/share/local-texmf`. If you are using `MiKTeX`, it may be `c:\localtexmf` or `c:\Program Files\localtexmf`. Whichever system you are using, I shall refer to this directory as `<LOCAL-TEXMF>`. Both the `<TEXMF>` and `<LOCAL-TEXMF>` directories must have the same structure. The principle sub-directories relating to L<sup>A</sup>T<sub>E</sub>X are illustrated in [figure 6.1](#). It may be that your `<LOCAL-TEXMF>` directory doesn't contain some of these sub-directories, if so, you will need to create them.

The documentation for L<sup>A</sup>T<sub>E</sub>X class files and packages can be found in

## 6. PACKAGES

the sub-directories `<TEXMF>/doc/latex` and `<LOCAL-TEXMF>/doc/latex` (or `<TEXMF>\doc\latex` and `<LOCAL-TEXMF>\doc\latex` on Windows).

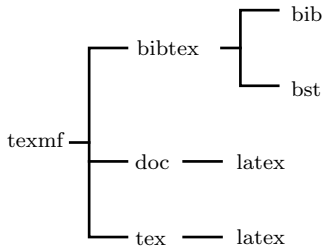


Figure 6.1: The TeX Directory Structure (TDS) showing the main L<sup>A</sup>T<sub>E</sub>X-related sub-directories.

Some packages are supplied in this format. For example, the package `pack`, may be distributed in a compressed file `pack.zip`, which contains the files



## 6. PACKAGES

```
texmf/doc/pack.pdf
texmf/tex/latex/pack/pack.sty
texmf/tex/latex/pack/pack-foo.sty
texmf/tex/latex/pack/pack-bar.sty
```

In this case all you need to do is decompress the contents of the `texmf` directory of the archive into the `<LOCAL-TEXMF>` directory. Then you must [refresh the T<sub>E</sub>X database](#) (described in [section 6.2.1](#)).

Many packages are supplied with the code and documentation all bundled together in one file. This file usually has the extension `.dtx`, and it usually comes with an installation script that has the extension `.ins`. Once you have downloaded the `.dtx` and `.ins` files, you will then have to extract the code before you can use it. Let's go back to the previous example. The package `pack` is now distributed in a DTX file, so the `pack.zip` archive now contains the files

[Documented  
LaTeX sources  
(.dtx files)]

```
pack.dtx
pack.ins
```

(with hopefully a `README` or `INSTALL` file!) Note that the archive no longer contains any `.sty` files, nor does it contain any sub-directories. The documentation (`pack.pdf` or `pack.dvi`) and the package code (`pack.sty`, `pack-foo.sty` and `pack-bar.sty`) are all contained in the file `pack.dtx`. This is how to extract them:

## 6. PACKAGES

1. Extract the contents of `pack.zip` to a temporary directory.
2. Run  $\text{\LaTeX}$  on the file `pack.ins`. If you are using a [terminal](#), you can type the following at the command prompt:

```
latex pack.ins
```

If you are using a front-end, load the file `pack.ins`, and click on whatever button you use to  $\text{\LaTeX}$  your documents.

This will create the files containing the package code. In this example it will create the files `pack.sty`, `pack-foo.sty` and `pack-bar.sty`.

3. Make a sub-directory of  $\langle LOCAL-TEXMF \rangle / \text{\textbackslash} \text{tex} / \text{\textbackslash} \text{latex}$ <sup>3</sup> in which to place these files. In this example, the package is called “pack”, so make a sub-directory called `pack`.
4. Move the files created in [step 2](#) into the new sub-directory you created in the previous step.
5. Run  $\text{\LaTeX}$  on the file `pack.dtx`. (The same as in [step 2](#), but use the file `pack.dtx` instead of `pack.ins`.) This will create a file called

---

<sup>3</sup>or  $\langle LOCAL-TEXMF \rangle \backslash \text{tex} \backslash \text{latex}$  on Windows

## 6. PACKAGES

`pack.dvi` if you used  $\text{\LaTeX}$ , or `pack.pdf` if you used  $\text{\PDFLaTeX}$ . You may need to repeat this step to ensure that the cross references are up-to-date. Check the `README` file or `INSTALL` file to see if there is anything else you need to do. (If you have downloaded the package from CTAN, it's possible that the documentation has already been supplied, as package authors are encouraged to supply a PDF version of the documentation for on-line viewing. If so, you can omit this step.)

6. Make a sub-directory of `<LOCAL-TEXMF>/doc/latex4` in which to place the documentation. In this example, the package is called “pack”, so make a sub-directory called `pack`.
7. Move the files created in [step 5](#) into the new sub-directory you created in the previous step.
8. [Refresh the database](#) (described below).

### 6.2.1 Refreshing the $\text{\TeX}$ Database

Whenever you install new class files or packages, you must update the  $\text{\TeX}$  database, otherwise the files will not be found. How to do this depends

---

<sup>4</sup>or `<LOCAL-TEXMF>\doc\latex` on Windows

## 6. PACKAGES

on the system you are using:

**teTeX** Use the command `texhash` (or `mktextlsr`).

**MiKTeX** If you are using an old MiKTeX distribution, you will need to run **MiKTeX Options** which will probably be in:

Start → Programs → MiKTeX → MiKTeX Options

and then click on the button labelled **Refresh Now** (see [figure 6.2](#)).

Recent versions of MiKTeX have an application called **MiKTeX Update Wizard** which can automatically download and install known packages, check the MiKTeX documentation for further details.

If you experience any problems, contact your system administrator for help.

Alternatively, you can leave the `.sty` file in the same directory as your  $\text{\LaTeX}$  document, but if you do this, you will only be able to use it with documents in that directory.

Related UK TUG FAQ [\[2\]](#) topics:

- [Installing things on a \(La\)TeX system](#)

## 6. PACKAGES

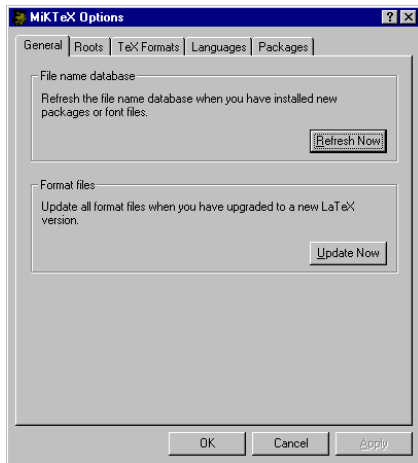


Figure 6.2: MiKTeX: Updating the database

## 6. *PACKAGES*

- Where to install packages
- Installing MiKTeX “known packages”
- “Temporary” installation of (La)TeX files
- “Private” installations of files

# Chapter 7

## Floats

Figures and tables are referred to as floats because they float to the nearest location. Floats have a caption and associated number. It is customary for figure captions to appear at the bottom of the figure and for table captions to appear at the top of the table. Figures and tables may not have page breaks within them.

[The style of captions]

For both figures and tables, the caption is generated using the command:

[Tables longer than a single page]

```
\caption[<short caption>]{<text>}
```

Definition

Note that the `\caption` command has a [moving argument](#), so fragile commands will need to be protected using `\protect`. The optional argument `<short caption>` is used to provide an alternative shorter caption for the list of figures or list of tables, akin to the optional argument to the [sectioning commands](#).

[Footnotes in captions]

## 7.1 Figures

Figures are created using the `figure` environment.

```
\begin{figure}[<placement specifiers>]
```

Definition

This environment may contain one or more captions (generated with the `\caption` command). The `figure` environment takes one optional argument which indicates permissible locations for the figure. This may be a combination of `h` (here), `t` (top), `b` (bottom) and `p` (page of floats). Note that this only gives a general guideline as to where the figure will end up. The final location is governed by other factors, such as space left on the page and the proportion of text to floats on the page. If you omit one or more of the placement specifiers, then you are prohibiting the figure from being placed in that location. A common mistake is to do

[Wide figures in two-column documents]

```
\begin{figure}[h]
```

which says “I want the figure here and it can’t go anywhere else!” If the figure *can’t* be placed exactly here (for example, there may not be enough room on the page), then you have given it no alternative location which can result in this and all subsequent figures being dumped at the end of the chapter or document, or can result in a fatal error when running



## 7. FLOATS

L<sup>A</sup>T<sub>E</sub>X. You may be able to manage with only one of the other options, for example,

[“Too many unprocessed floats”]

```
\begin{figure}[t]
```

however, if you have a large number of floats it is advisable to provide as many options as possible:

```
\begin{figure}[htbp]
```

If you are absolutely adamant that the figure must go “right here”, then it’s not a float, and you shouldn’t be using the `figure` environment (and you should also be prepared for the possibility of a large amount of white space at the end of the page if the image is too large to fit there).

Recall from [section 6.1.1](#) we can include an Encapsulated PostScript (EPS) file or PDF image in our document with the command `\includegraphics` defined in the `graphicx` [package](#). We can put our `shapes.ps` or `shapes.pdf` image into a figure as follows:

```
\begin{figure}[htbp]
\includegraphics{shapes}
\caption{Some shapes}
\end{figure}
```

↑Input

## 7. FLOATS

↓Input

So far so good, but our picture needs to be centred. This can be done using the `\centering` declaration:

↑Input

```
\begin{figure}[htbp]
\centering
\includegraphics{shapes}
\caption{Some shapes}
\end{figure}
```

↓Input

The `\caption` command generates a number, just like `\section`, so we can [cross-reference](#) it with `\ref` and `\label`. First, let's label the figure:

↑Input

```
\begin{figure}[htbp]
\centering
\includegraphics{shapes}
\caption{Some shapes}
\label{fig:shapes}
\end{figure}
```

## 7. FLOATS

[↓ Input](#)

Now we can reference it:

Figure`\ref{fig:shapes}` shows some shapes.

[Input](#)

(As [before](#) we use `~` to make an unbreakable space.) This produces the following output in the text:

Figure 7.1 shows some shapes.

[Output](#)

and produces [figure 7.1](#).

Just as we were able to generate a [table of contents](#) using `\tableofcontents`, we can also generate a list of figures using the command

`\listoffigures`

[Definition](#)

As before you will need to  $\text{\LaTeX}$  your document twice to get the list of figures up-to-date.

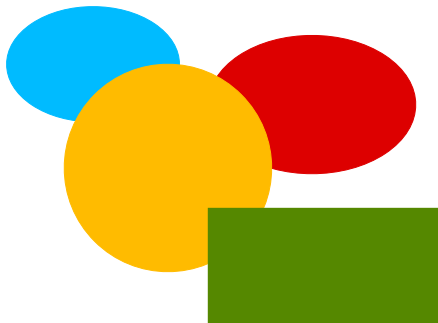


Figure 7.1: Some shapes

### Exercise 16 (Creating Figures)

If you did [exercise 14](#), you should have a document with the image [shapes.ps](#) (or [shapes.pdf](#)) in it. You now need to put this image into a `figure` environment. Remember to centre the image, and give the figure a caption. Next, try labelling the figure and referencing it in the text. You could also put in a list of figures after the table of contents.

(You can [download](#) or [view](#) an example.)

---

#### 7.1.1 Side-By-Side Figures

The `figure` environment, should really be called “figures” rather than “figure”, as you can have more than one `\caption` command within the environment, however, since the contents of the `figure` environment can’t have a page break, nor can the figures within a single `figure` environment float independently of each other, it is more usual to have a separate `figure` environment for each figure. As a result, people tend to forget that they can have more than one figure in a `figure` environment, which gives rise to the frequently asked question “how can I have side-by-side figures?”

The answer to this is to put the two figures in the same `figure` environment. To do this, we can use the `minipage` environment, which was covered

## 7. FLOATS

in [section 4.6](#). Recall that the `minipage` environment creates a horizontal box, which means that two mini-pages can be placed side-by-side on the same line. All you need to do now, is place one image and caption in one mini-page, and the other image and caption in the neighbouring mini-page:

```
\begin{figure}[htbp]
\begin{minipage}{0.5\linewidth}
\centering
\includegraphics{circle}
\caption{A Circle}
\label{fig:circle}
\end{minipage}%
\begin{minipage}{0.5\linewidth}
\centering
\includegraphics{rectangle}
\caption{A Rectangle}
\label{fig:rectangle}
\end{minipage}
\end{figure}
```

↑Input

↓Input

## 7. FLOATS

which produces [figure 7.2](#) and [figure 7.3](#). Note that each mini-page uses `\centering` to centre its contents, and the label is also placed in the same mini-page, after the `\caption` command. (Do you remember what effect is obtained by placing a [percent symbol](#) at the end of a line?)

A common mistake when trying to create side-by-side figures is to do:

```
\begin{figure}[htbp]
\begin{minipage}{0.5\linewidth}
\centering
\includegraphics{circle}
\caption{A Circle}
\label{fig:circle}
\end{minipage}

\begin{minipage}{0.5\linewidth}
\centering
\includegraphics{rectangle}
\caption{A Rectangle}
\label{fig:rectangle}
\end{minipage}
\end{figure}
```

↑Input

This produces one figure on top of the other, instead of side-by-side. Can you see why?<sup>1</sup>

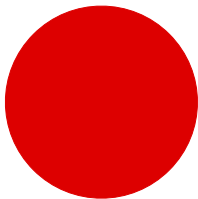


Figure 7.2: A Circle



Figure 7.3: A Rectangle

---

<sup>1</sup>The blank line indicates a paragraph break, so each minipage is in a separate paragraph, so it's not possible for them to be on the same line.



## 7.1.2 Sub-figures

Some figures have sub-figures within them. These can be generated using the `subfig2` package. Each sub-figure is specified using

```
\subfloat[<list entry>][<caption>]{<object>}
```

Definition

where `<list entry>` is the entry for the list of figures,<sup>3</sup> `<caption>` is the sub-caption and `<object>` is the code to create the image.

For example, suppose you have two files `circle.ps` and `rectangle.ps` (or `circle.pdf` and `rectangle.pdf`):

```
\begin{figure}[htbp]
\begin{center}
\subfloat[A Rectangle]{\includegraphics{rectangle}}
\hspace{0.5in}
\subfloat[A Circle]{\includegraphics{circle}}
\end{center}
\caption{Two Shapes: (a) A Rectangle and (b) A Circle}
```

↑ Input

<sup>2</sup>Note that the older `subfigure` package is obsolete.

<sup>3</sup>or tables or whatever type of float you are using.

## 7. FLOATS

```
\end{figure}
```

↓ Input

The whole figure is centred using the `center` environment, discussed earlier, and the two sub-figures are separated by a horizontal gap of half an inch using the command:

```
\hspace{<len>}
```

Definition

This ensures that the image doesn't look too cramped.

Again we can cross-reference the sub-figures. The `\label` command should go in the **mandatory argument** of the `\subfloat` command. The sub-figure can be referred to using `\ref` to produce, e.g. 1a, or can be referenced using:

```
\subref{<label>}
```

Definition

which will produce, e.g. (a).

```
\begin{figure}[htbp]
\begin{center}
```

↑ Input

## 7. FLOATS

```
\subfloat[A Rectangle]{%
\label{fig:rectangle}\includegraphics{rectangle}}
\hspace{1in}
\subfloat[A Circle]{%
\label{fig:circle}\includegraphics{circle}}
\end{center}
\caption{Two Shapes: \protect\subref{fig:rectangle} A Rectangle and
\protect\subref{fig:circle} A Circle}
\label{fig:shapes2}
\end{figure}
```

Figure~\ref{fig:shapes2} shows some shapes.  
Figure~\ref{fig:rectangle} shows a rectangle and  
Figure~\ref{fig:circle} shows a circle.

[↓ Input](#)

This produces the following text:

Figure 7.4 shows some shapes. Figure 7.4a shows a rectangle and  
Figure 7.4b shows a circle.

[↑ Output](#)

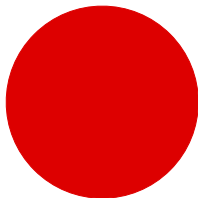
[↓ Output](#)

and produces [figure 7.4](#).

## 7. FLOATS



(a) A Rectangle



(b) A Circle

Figure 7.4: Two Shapes: (a) A Rectangle and (b) A Circle

## 7. FLOATS

Note that this only describes a small part of the capabilities of the `subfig` package. It can also be used to make continued figures, and can be applied to other types of floats as well, such as tables. The format for the sub-floats and their captions can also be modified. Check the `subfig` documentation for more details.

### Exercise 17 (Creating Sub-Figures)

Download [rectangle.ps](#) and [circle.ps](#) (or [rectangle.pdf](#) and [circle.pdf](#)) from <http://theoval.cmp.uea.ac.uk/~nlct/latex/novices/exercises/> and add [figure 7.4](#) to your document. You can [download](#) or [view](#) an example.

---

## 7.2 Tables

Tables are produced in much the same way as figures, except that the `table` environment is used instead. Tables typically have the caption at the top of the table (as opposed to figures, which have the caption at the bottom). Example:

## 7. FLOATS

```
\begin{table}
\caption{A Sample Table}
\label{tab:sample}
\centering
\begin{tabular}{lr}
Item & Cost\\
Video & 8.99\\
CD & 9.99\\
DVD & 15.00
\end{tabular}
\end{table}
```

[↓ Input](#)

This produces [table 7.1](#).

Table 7.1: A Sample Table

| Item  | Cost  |
|-------|-------|
| Video | 8.99  |
| CD    | 9.99  |
| DVD   | 15.00 |

Again, the `\centering` declaration is used to centre the `tabular` envi-

## 7. FLOATS

ronment, however I think that the table looks a little cramped,<sup>4</sup> so let's put in a bit of extra vertical space after the caption. This can be done using the command:

[Extra vertical space in floats]

```
\vspace{<length>}
```

Definition

Our code now looks like:

```
\begin{table}
\caption{A Sample Table}
\label{tab:sample}
\vspace{10pt}
\centering
\begin{tabular}{lr}
Item & Cost\\
Video & 8.99\\
CD & 9.99\\
DVD & 15.00
\end{tabular}
\end{table}
```

↑ Input

---

<sup>4</sup>The center environment would have put in some extra vertical space, thus dispensing with the `\vspace`, and I often use the center environment instead of the `\centering` declaration, however that seems to be a contentious issue.

## 7. FLOATS

```
\end{tabular}  
\end{table}
```

[↓ Input](#)

This produces [table 7.2](#).

Table 7.2: A Sample Table

| Item  | Cost  |
|-------|-------|
| Video | 8.99  |
| CD    | 9.99  |
| DVD   | 15.00 |

As with figures, you can create a list of tables using the command

```
\listoftables
```

[Definition](#)

### Exercise 18 (Creating Tables)

If you did [exercise 6](#), you should have a `tabular` environment in your



## 7. *FLOATS*

document. Try turning this into a table, and also add [table 7.2](#). You could also try adding a list of tables. You can [download](#) or [view](#) the document.

---

# Chapter 8

## Defining Commands

It is possible to define your own [commands](#) or redefine existing ones. Be very careful about redefining existing commands; don't redefine a command simply because you want to use the name, only redefine it if you are making a modification. For example, if you want to change the format of the current date, you would redefine `\today`, but if you want to define a command to display a specific date, you should define a new command with a different name.

There are several reasons why you might want to define a new command:

1. Reduce typing:

Suppose you have a series of commands or text that you find yourself frequently using, then you could define a command to do all these other commands for you.

Example: Suppose you want a lot of large bold slanted sans-serif portions of text within your document. Every time you type those portions of text, you will have to do something like:

## 8. DEFINING COMMANDS

```
\textsf{\large\bfseries\slshape Some text}
```

Input

It would be much easier if you could use just one command to do all that, called, say, `\largeboldsfsl`:

```
\largeboldsfsl{Some text}
```

Input

or we could call it, say, `\lbfsl` which is shorter, but slightly less memorable:

```
\lbfsl{Some text}
```

Input

### 2. Ensure consistency:

You may find that you want to format an object a certain way. For example, your document may have a lot of keywords in it, and you may want to format these keywords in a different font, say sans-serif, so that they stand out. You could just do:

## 8. DEFINING COMMANDS

A `\textsf{command}` usually begins with a backslash.

Input

however, it is better to define a new command called, say, `\keyword` that will typeset its argument in a sans-serif font. That way it becomes a lot easier to change the format at some later date. For example, you may decide to splash out and have your keywords typed in a particular colour. In which case, all you need to do is simply change the definition of the command `\keyword`, otherwise you'll have to go through your entire document looking for keywords and changing each one which could be very time consuming if you have a large document. You might also decide at some later date to make an index for your document. Indexing all the keywords then becomes very simple, as again all you'll need to do is modify the `\keyword` command.

New commands are defined using the command:

```
\newcommand{<cmd>}[<n-args>][<default>]{<text>}
```

Definition

The first [mandatory argument](#) `<cmd>` is the name of your new command, which must start with a backslash. The [optional argument](#) `<n-args>` specifies how many arguments your new command must take. The next

## 8. DEFINING COMMANDS

optional argument  $\langle default \rangle$  will be discussed later. The final mandatory argument  $\langle text \rangle$  specifies what L<sup>A</sup>T<sub>E</sub>X should do every time it encounters this command.

Let's begin with a trivial example. Suppose I wanted to write a document about a particular course, say “Programming — Languages and Software Construction”, and I had to keep writing the course title, then I might decide to define a command that prints the course title rather than having to laboriously type it out every time. Let's call our new command `\coursetitle`. We want the following code:

```
The course \emph{\coursetitle} is an undergraduate course.
```

Input

to produce the following output:

```
The course Programming — Languages and Software Construction is  
an undergraduate course.
```

↑Output

↓Output

Clearly this command doesn't need any arguments, so we don't need to worry about the optional argument  $\langle n\text{-args} \rangle$  to `\newcommand`, and the only thing our new command needs to do is print:

## 8. DEFINING COMMANDS

Programming --- Languages and Software Construction  
so we would define our new command as follows:

```
\newcommand{\coursetitle}{Programming --- Languages  
and Software Construction}
```

↑Input

↓Input

Commands must always be defined before they are used. The best place to define commands is in the [preamble](#):

```
\documentclass[a4paper]{article}  
  
\newcommand{\coursetitle}{Programming --- Languages  
and Software Construction}  
  
\begin{document}  
  
\section{\coursetitle}
```

↑Code

The course `\emph{\coursetitle}` is an undergraduate course.

## 8. DEFINING COMMANDS

```
\end{document}
```

[↓ Code](#)

Now let's try defining a command that takes an [argument](#) (or parameter). Let's go back to our `\keyword` example. This command needs to take one argument that is the keyword. Let's suppose we want keywords to come out in [sans-serif](#), then we could do:

```
\newcommand{\keyword}[1]{\textsf{#1}}
```

[Input](#)

In this case we have used the optional argument  $\langle n\text{-args} \rangle$  to `\newcommand`. We want our command `\keyword` to have one argument, so we have `[1]`. In `\textsf{#1}` the `#1` represents the first argument. (If we had more than one argument, `#2` would represent the second argument, `#3` would represent the third argument etc. up to a maximum of 9.) So

[\[How to break the 9-argument limit\]](#)

```
\keyword{commands}
```

will be equivalent to

```
\textsf{commands}
```

## 8. DEFINING COMMANDS

and

```
\keyword{environment}
```

will be equivalent to

```
\textsf{environment}
```

and so on.

Again, the line

```
\newcommand{\keyword}[1]{\textsf{#1}}
```

should go in the preamble. That way you can ensure the command won't be used before it's defined:

```
\documentclass[a4paper]{article}

\newcommand{\keyword}[1]{\textsf{#1}}

\begin{document}
```

A `\keyword{command}` usually begins with a backslash.

[Code](#)



## 8. DEFINING COMMANDS

```
\end{document}
```

[↓ Code](#)

Now if we want to change the way the keywords are formatted, we can simply change the definition of `\keyword`. Let's modify our code so that the keyword is now in a slanted sans-serif font:

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

```
\newcommand{\keyword}[1]{\textsf{\slshape #1}}
```

```
\begin{document}
```

A `\keyword{command}` usually begins with a backslash.

```
\end{document}
```

[↓ Code](#)

## 8. DEFINING COMMANDS

Let's go one stage further. The `color package` enables the use of colour, so let's make our keywords blue:

```
\documentclass[a4paper]{article}

\usepackage{color}

\newcommand{\keyword}[1]{\textsf{\slshape\color{blue}#1}}

\begin{document}

A \keyword{command} usually begins with a backslash.

\end{document}
```

[↑ Code](#)

[↓ Code](#)

Or we could index the keywords. To do this we need the `makeidx package` and the commands `\makeindex`, `\index{<text>}` and `\printindex`:

[↑ Code](#)

## 8. DEFINING COMMANDS

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

```
\usepackage{makeidx}
```

```
\makeindex
```

```
\newcommand{\keyword}[1]{\textsf{\slshape #1}\index{#1}}
```

```
\begin{document}
```

A `\keyword{command}` usually begins with a backslash.

```
\printindex
```

```
\end{document}
```

[↓ Code](#)

For further information about how to create an index, see *A Guide to L<sup>A</sup>T<sub>E</sub>X* [4] or *The L<sup>A</sup>T<sub>E</sub>X Companion* [5]. Alternatively, if you want a brief overview on-line, try *Using L<sup>A</sup>T<sub>E</sub>X to Write a PhD Thesis*.

Since it is unlikely that the keyword will contain a paragraph break, we should indicate that this is a [short command](#) using the starred form:

## 8. DEFINING COMMANDS

```
\newcommand*{\keyword}[1]{\textsf{\slshape #1}\index{#1}}
```

↑Input

↓Input

Now if you forget to add the closing brace, for example, `\keyword{command`, then TeX's error checking mechanism will pick up the error sooner. This will give an error message that looks like:

```
! Paragraph ended before \keyword was complete.
```

```
<to be read again>
```

```
    \par
```

```
1.604
```

This at least gives you the line number (604 in this example) of the end of the paragraph where the error has occurred.

If you don't use the starred form of `\newcommand`, then you will get the somewhat less than helpful error:

```
! File ended while scanning use of \keyword.
```

If you have a very large document, it may take a while to track down where exactly you have missed a brace.

## 8. DEFINING COMMANDS

**Note 4:** When you define a command using `\newcommand` you can't use a command name that already exists, and you can't use a name obtained by placing a backslash in front of an existing [environment](#) name. For example, since the `itemize` environment exists, you can't define a command called `\itemize`. In addition, you can't define a command that starts with `\end`. For example, you can't define a command called, say, `\endkeyword`. For further details, see [note 7](#) in [chapter 10](#).

### Exercise 19 (Defining a New Command)

Try typing up the following code:

```
\documentclass[a4paper]{article}

\newcommand*{\keyword}[1]{\textsf{#1}}

\begin{document}
```

A `\keyword{command}` usually begins with a backslash.

 Code

## 8. DEFINING COMMANDS

Segments of code may be `\keyword{grouped}`.

Some `\keyword{commands}` take one or more `\keyword{arguments}`.

```
\end{document}
```

[↓ Code](#)

Then modify your code so that the keywords are in a slanted sans-serif font, and then modify your code so that the keywords come out in blue. (You may need to convert your DVI file to PostScript in order to see the colour, either using `dvips` [in a terminal](#), in [WinEdt](#) or in [TeXnicCenter](#) as described in [chapter 3](#), or use PDF $\LaTeX$  instead of  $\LaTeX$ .) Again you can [download](#) or [view](#) the result.

**For the more adventurous:** If you want to create an index as in the previous example, you will need to use the application `makeindex`. Suppose your source code is saved as `exercise19.tex`, then if you are using a [terminal](#) you will need to do:

```
latex exercise19.tex  
makeindex exercise19.idx
```

## 8. DEFINING COMMANDS

```
latex exercise19.tex
```

If you are using [WinEdt](#) click the  $\text{\LaTeX}$  button, then select Makeindex from the menu, then click on the  $\text{\LaTeX}$  button again. If you are using [TeXnicCenter](#), if you select the checkbox labelled “uses MakeIndex” when you [create your project](#), [TeXnicCenter](#) will automatically call `makeindex` when you click on the build icon. If you have already created the project, you can modify its settings using the [Project](#) menu.

---

### 8.1 Defining Commands with an Optional Argument

As mentioned earlier, the `\newcommand` command has a second optional argument `<default>`. This allows you to define a command with an optional argument. For example, suppose we want a command called, say, `\price`. Suppose we want the following code:

```
\price{100}
```

to produce the following output:

[More than one optional argument]

Input

## 8. DEFINING COMMANDS

£100 excl VAT @ 17.5%

Output

and let's suppose we want an optional argument so that we can change the VAT. That is, we would want the following code:

```
\price[0]{30}
```

Input

to produce the following output:

£30 excl VAT @ 0%

Output

Therefore we want to define a command such that if the optional argument is absent we will have 17.5, and if it is present the optional argument will be substituted instead. This command can be defined as follows:

```
\newcommand{\price}[2][17.5]{\pounds #2 excl VAT @ #1\%}
```

Input

Here, #1 represents the optional argument (by default 17.5) and #2 represents the mandatory argument (the second argument if the optional argument is present, or the only argument if the optional argument is absent).



## 8. DEFINING COMMANDS

As before, since the argument is unlikely to contain a paragraph break, we should indicate that it is a [short command](#) using the starred form:

```
\newcommand*{\price}[2][17.5]{\pounds #2 excl VAT @ #1\%}
```

Input

### Exercise 20 (Defining Commands with an Optional Argument)

In this exercise, you will need to define a slightly modified version of the above example. Try defining a command called, say, `\cost`. It should take one optional argument and one mandatory argument. Without the optional argument, it behaves in the same way as the `\price` example above, so that, say,

```
\cost{50}
```

Input

will produce

```
£50 excl VAT @ 17.5%
```

Output

## 8. DEFINING COMMANDS

but with the optional argument, you can change the `excl VAT @ 17.5\%` bit. So that, say,

```
\cost[inc VAT]{50}
```

Input

will produce

```
£50 inc VAT
```

Output

You can [download](#) or [view](#) the solution.

**For the more adventurous:** If you did [exercise 19](#) and you modified `\keyword` so that it indexed the keyword, you may have noticed that `\keyword{command}` and `\keyword{commands}` produced separate entries in the index. It would be better to have an optional argument to override the indexing mechanism. For example, `\keyword{command}` should print and index the word “command”, whereas `\keyword[command]{commands}` should print “commands” and index “command”. In other words, we need an optional argument that defaults to the mandatory argument if it is not present. This is how to achieve that type of effect:

[Optional arguments like `\section`]

↑ Input

## 8. DEFINING COMMANDS

```
\newcommand*{\keyword}[2] [\keywordentry]{%  
\def\keywordentry{#2}%  
\textsf{#2}%  
\index{#1}}
```

[↓ Input](#)

In this example, the default value for the optional argument is the command `\keywordentry`. At the start of `\keyword` this is defined to be the mandatory argument (as specified by `#2`) using T<sub>E</sub>X's `\def` command:

```
\def\keywordentry{#2}%
```

(The percent symbol discards the space resulting from the end of line character.) Then typeset the keyword (given in the mandatory argument `#2`) in a sans-serif font:

```
\textsf{#2}%
```

Now index the term using the optional argument (`#1`):

```
\index{#1}
```

If an optional argument is specified, `#1` will be the given argument, but if the optional argument is missing, `#1` will be `\keywordentry`, which has earlier been set to the mandatory argument `#2`.

## 8.2 Redefining Commands

Commands can be redefined using the command:

```
\renewcommand{<cmd>}[<n-args>][<default>]{<text>}
```

Definition

This has exactly the same format as `\newcommand` but is used for redefining existing commands. **Caveat:** never redefine a command whose existing function is unknown to you. Again there is a starred version to indicate that the command is a [short command](#).

Recall the `itemize` environment discussed in [section 4.3.1](#). You may have up to four nested `itemize` environments, the labels for the outer environment are specified by the command `\labelitemi`, the labels for the second level are specified by `\labelitemii`, the third by `\labelitemiii` and the fourth by `\labelitemiv`. By default, `\labelitemi` is a bullet point, `\labelitemii` is an en dash, `\labelitemiii` is an asterisk and `\labelitemiv` is a dot (`• – * ·`). These can be changed by redefining `\labelitemi` etc.

Example: Recall from [table 4.1](#) that the command `\dag` produces a dagger symbol, we can use this symbol instead of a bullet point:

↑ Input

## 8. DEFINING COMMANDS

```
\renewcommand*{\labelitemi}{\dag}
```

```
\begin{itemize}
```

```
\item Animal
```

```
\item Mineral
```

```
\item Vegetable
```

```
\end{itemize}
```

↓ Input

Output:

```
† Animal
```

```
† Mineral
```

```
† Vegetable
```

↑ Output

## 8. DEFINING COMMANDS

↓Output

Here's another example, it uses the PostScript font ZapfDingbats via the `pifont` package:

↑Input

```
\renewcommand*{\labelitemi}{\ding{43}}  
  
\begin{itemize}  
  
\item Animal  
  
\item Mineral  
  
\item Vegetable  
  
\end{itemize}
```

↓Input

Output:

↑Output

## 8. DEFINING COMMANDS

- ☞ Animal
- ☞ Mineral
- ☞ Vegetable

↓Output

In the above example, it would actually be better to use the `dinglist` environment defined in the `pifont` package. See *The L<sup>A</sup>T<sub>E</sub>X Companion* [5] for more details.

You may have noticed that L<sup>A</sup>T<sub>E</sub>X automatically generates pieces of text such as “Chapter”, “Figure”, “Bibliography”. These are generated by the commands shown in [table 8.1](#).

You can change the defaults using `\renewcommand`. For example, suppose you want the table of contents to be labelled “Table of Contents”, instead of the default “Contents”, you would need to do:

```
\renewcommand*{\contentsname}{Table of Contents}
```

Input

## 8. DEFINING COMMANDS

Table 8.1: Predefined Names (<sup>†</sup>report class file, <sup>‡</sup>article class file, remainder both report and article)

<b>Command</b>	<b>Default Text</b>
<code>\contentsname</code>	Contents
<code>\listfigurename</code>	List of Figures
<code>\listtablename</code>	List of Tables
<code>\bibname<sup>†</sup></code>	Bibliography
<code>\refname<sup>‡</sup></code>	References
<code>\indexname</code>	Index
<code>\figurename</code>	Figure
<code>\tablename</code>	Table
<code>\partname</code>	Part
<code>\chaptername<sup>†</sup></code>	Chapter
<code>\appendixname</code>	Appendix
<code>\abstractname</code>	Abstract



## Exercise 21 (Renewing Commands)

If you did exercises [16](#) and [18](#), go back to that document and changed the figures and tables so that they are labelled “Fig” and “Tab” instead of “Figure” and “Table”.

You can [download](#) or [view](#) the solution.

---

# Chapter 9

## Mathematics

As mentioned in the [introduction](#), L<sup>A</sup>T<sub>E</sub>X is particularly good at typesetting mathematics. In order to use any of the maths commands we need to be in one of the mathematics [environments](#). There are two basic types of mathematics: in-line maths and displayed maths. In-line maths is mathematics that occurs within a line of text, for example:

The variable  $x$  is transformed by the function  $f(x)$ .

Output

Displayed maths is mathematics that occurs on a line of its own. For example:

A polynomial is a function of the form

↑Output

$$f(x) = \sum_{i=0}^n a_i x^i$$

This document only describes the basic mathematical commands and environments available to L<sup>A</sup>T<sub>E</sub>X users. For a more detailed discussion, try [13].

## 9.1 In-Line Mathematics

In-line mathematics is created using the `math` environment. (Note U.S. spelling — “math” not “maths”). Example:

The variable `\begin{math}x\end{math}` is transformed by the function `\begin{math}f(x)\end{math}`.

It’s somewhat cumbersome having to type `\begin{math}` and `\end{math}` and it also makes the [source code](#) a little difficult to read so there are shorthand notations that can be used instead: `\(` is equivalent to `\begin{math}` and `\)` is equivalent to `\end{math}`. So the example above can be rewritten:

## 9. MATHEMATICS

The variable  $\backslash(x\backslash)$  is transformed by the function  $\backslash(f(x)\backslash)$ .

Input

There is an even shorter notation: The [special character](#)  $\$$  is equivalent to both  $\backslashbegin\{math\}$  and  $\backslashend\{math\}$ :

The variable  $\$x\$$  is transformed by the function  $\$f(x)\$$ .

Input

This is considerably easier to type and to read, but you need to make sure that all your  $\$$  symbols have matching pairs. The above code will look like:

The variable  $x$  is transformed by the function  $f(x)$ .

Output

The other advantage in using  $\$$  over  $\backslash($  and  $\backslash)$  is that  $\$$  is a [robust command](#), whereas  $\backslash($  and  $\backslash)$  are [fragile commands](#) and will need to be protected if they occur in a [moving argument](#).

Note: you should always make sure you are in maths mode to typeset any variables (such as  $x$ ,  $y$ ,  $z$ ), as this will ensure that the correct maths fonts are used, as well as the appropriate spacing. For example, the following:

---

↑Input

## 9. MATHEMATICS

Notice the difference between  $(x, y, z)$  and  $\textit{(x, y, z)}$ .

[↓ Input](#)

produces:

Notice the difference between  $(x, y, z)$  and  $\textit{(x, y, z)}$ .

[Output](#)

## 9.2 Displayed Mathematics

Displayed mathematics can be created using either the `displaymath` or the `equation` environments. Example:

```
A linear function is a function of the form
\begin{displaymath}
y = mx + c
\end{displaymath}
```

[↑ Input](#)

[↓ Input](#)

Output:

## 9. MATHEMATICS

A linear function is a function of the form

$$y = mx + c$$

↑Output

↓Output

The `equation` environment is the same as the `displaymath` environment, except that the equation is numbered. Substituting `equation` for `displaymath` in the above example:

A linear function is a function of the form

```
\begin{equation}
```

```
y = mx + c
```

```
\end{equation}
```

↑Input

↓Input

results in the following output:

↑Output

## 9. MATHEMATICS

A linear function is a function of the form

$$y = mx + c \tag{9.1}$$

↓Output

Recall from [section 5.5](#) that we can [cross-reference](#) most things that  $\text{\LaTeX}$  automatically numbers using `\ref` and `\label`. Equations can be cross-referenced in the same way:

[Re-using an equation]

Equation~\ref{eqn:linear} is a linear function.

```
\begin{equation}
\label{eqn:linear}
f(x) = mx + c
\end{equation}
```

↑Input

↓Input

Equation 9.2 is a linear function.

↑Output

$$f(x) = mx + c \tag{9.2}$$

**Note 5:** Both the `equation` and the `displaymath` environments are only designed for one line of maths. Therefore you must not have any line breaks or paragraph breaks within them. If you want several aligned equations, you need to use another environment, such as `align`. This document does not cover these environments, but if you are interested see *The L<sup>A</sup>T<sub>E</sub>X Companion* [5] or *A Guide to L<sup>A</sup>T<sub>E</sub>X* [4].

[Why not use `eqnarray`?]

### 9.3 Mathematical Commands

Most of the `commands` described in this section may only be used in one of the mathematics environments. If you try to use a mathematics command outside a maths environment you will get a “Missing \$ inserted” error message.



### 9.3.1 Maths Fonts

Just as we are able to [change text fonts](#) using the commands `\textrm`, `\textbf` etc, we can also use commands to change the maths font. Basic maths font changing commands are shown in [table 9.1](#).

Table 9.1: Maths Font Changing Commands

Command	Example Input	Corresponding Output
<code>\mathrm{&lt;maths&gt;}</code>	<code>\$\$\mathrm{xyz}\$\$</code>	$xyz$
<code>\mathsf{&lt;maths&gt;}</code>	<code>\$\$\mathsf{xyz}\$\$</code>	$xyz$
<code>\mathtt{&lt;maths&gt;}</code>	<code>\$\$\mathtt{xyz}\$\$</code>	$xyz$
<code>\mathit{&lt;maths&gt;}</code>	<code>\$\$\mathit{xyz}\$\$</code>	$xyz$
<code>\mathbf{&lt;maths&gt;}</code>	<code>\$\$\mathbf{xyz}\$\$</code>	$\mathbf{xyz}$
<code>\mathcal{&lt;maths&gt;}</code>	<code>\$\$\mathcal{XYZ}\$\$</code>	$\mathcal{XYZ}$

The calligraphic fonts are only available for upper-case characters. Note that if you want actual text to appear in a maths environment you need to either use `\mbox{<text>}`:

[Better script fonts for maths]

[Text inside maths]

## 9. MATHEMATICS

```
\begin{displaymath}
x > y \mbox{ and } y < z
\end{displaymath}
```

↓ Input

which produces

$$x > y \text{ and } y < z$$

Output

or the command `\text{<text>}` which is defined in the `amsmath` package:

```
\begin{displaymath}
x > y \text{ and } y < z
\end{displaymath}
```

↑ Input

↓ Input

which again produces

$$x > y \text{ and } y < z$$

Output

The advantage of using `\text` rather than `\mbox` is that `\text` adjusts the font size if it occurs in a subscript or superscript. For example, the following code

```
\begin{displaymath}
x^{\mbox{new}} = x^{\text{old}} + b
\end{displaymath}
```

↑ Input

↓ Input

produces:

$$x^{\text{new}} = x^{\text{old}} + b$$

Output

The word “new” (typeset using `\mbox`) is in the normal sized font, whereas the word “old” (typeset using `\text`) is in the appropriate superscript sized font.

[Table 9.2](#) lists additional font commands supplied with the `amsmath` and `amsfonts` packages. Note that there are several types of bold commands in the table:

- `\mathbb` (blackboard bold) is typically used to denote the set of naturals, integers, real or complex numbers. For example `\mathbb{N}`.
- `\boldsymbol` produces italic bold for letters (unlike `\mathbf` which produces upright bold) and produces upright bold for symbols (including Greek letters).

## 9. MATHEMATICS

- `\pmb` (poor man's bold) produces a bold effect by overlaying multiple copies of the symbol each slightly offset from the previous.

See the `amsmath` and `amsfonts` user manuals for further details regarding these commands, as well as other commands not covered here.

Table 9.2: The `amsfonts`<sup>‡</sup> and `amsmath`<sup>†</sup> Font Commands

Command	Example Input	Example Output
<code>‡\mathbb{&lt;maths&gt;}</code>	<code>‡\mathbb{A}</code>	$\mathbb{A}$
<code>‡\mathfrak{&lt;maths&gt;}</code>	<code>‡\mathfrak{A}</code>	$\mathfrak{A}$
<code>†\boldsymbol{&lt;maths&gt;}</code>	<code>†\boldsymbol{\geq}</code>	$\geq$
<code>†\pmb{&lt;symbol&gt;}</code>	<code>†\pmb{&gt;}</code>	$\mathbf{>}$

### 9.3.2 Greek Letters

Greek letters that differ from the corresponding Roman letter are obtained by placing a backslash in front of the name.<sup>1</sup> Lower case Greek letters are

---

<sup>1</sup>so, for example, there is no omicron since it looks the same as a Roman o.

## 9. MATHEMATICS

shown in [table 9.3](#) and upper case Greek letters are shown in [table 9.4](#).

Table 9.3: Lower Case Greek Letters

<code>\alpha</code>	$\alpha$	<code>\beta</code>	$\beta$	<code>\gamma</code>	$\gamma$
<code>\delta</code>	$\delta$	<code>\epsilon</code>	$\epsilon$	<code>\varepsilon</code>	$\varepsilon$
<code>\zeta</code>	$\zeta$	<code>\eta</code>	$\eta$	<code>\theta</code>	$\theta$
<code>\vartheta</code>	$\vartheta$	<code>\iota</code>	$\iota$	<code>\kappa</code>	$\kappa$
<code>\lambda</code>	$\lambda$	<code>\mu</code>	$\mu$	<code>\nu</code>	$\nu$
<code>\xi</code>	$\xi$	<code>\pi</code>	$\pi$	<code>\varpi</code>	$\varpi$
<code>\rho</code>	$\rho$	<code>\varrho</code>	$\varrho$	<code>\sigma</code>	$\sigma$
<code>\varsigma</code>	$\varsigma$	<code>\tau</code>	$\tau$	<code>\upsilon</code>	$\upsilon$
<code>\phi</code>	$\phi$	<code>\varphi</code>	$\varphi$	<code>\chi</code>	$\chi$
<code>\psi</code>	$\psi$	<code>\omega</code>	$\omega$		

There are also some variants of certain symbols, such as `\vartheta` as opposed to `\theta`.

### 9.3.3 Subscripts and Superscripts

Subscripts are obtained either by the command

## 9. MATHEMATICS

Table 9.4: Upper Case Greek Letters

<code>\Gamma</code>	$\Gamma$	<code>\Delta</code>	$\Delta$	<code>\Theta</code>	$\Theta$
<code>\Lambda</code>	$\Lambda$	<code>\Xi</code>	$\Xi$	<code>\Pi</code>	$\Pi$
<code>\Sigma</code>	$\Sigma$	<code>\Upsilon</code>	$\Upsilon$	<code>\Phi</code>	$\Phi$
<code>\Psi</code>	$\Psi$	<code>\Omega</code>	$\Omega$		

`\sb{<maths>}`

Definition

or by the [special character](#):

`_ {<maths>}`

Definition

Superscripts are obtained either by the command

`\sp{<maths>}`

Definition

or by the special character:

`^ {<maths>}`

Definition

## 9. MATHEMATICS

Examples:

1. This example uses `\sb` and `\sp`:

```
\begin{displaymath}
y = x\sb{1}\sp{2} + x\sb{2}\sp{2}
\end{displaymath}
```

↑ Input

↓ Input

2. This example uses `_` and `^`

```
\begin{displaymath}
y = x_{1}^{2} + x_{2}^{2}
\end{displaymath}
```

↑ Input

↓ Input

## 9. MATHEMATICS

- Recall from page 21 that [mandatory arguments](#) only consisting of one character don't need to be grouped, so the above code can also be written as:

```
\begin{displaymath}
y = x_1^2 + x_2^2
\end{displaymath}
```

↑ Input

↓ Input

This is simpler than the first two examples. All three of the above examples produce the same output:

$$y = x_1^2 + x_2^2$$

Output

- Subscripts and superscripts can also be nested (note that it is now necessary to group the argument to the superscript command):

↑ Input



## 9. MATHEMATICS

```
\begin{displaymath}
f(x) = e^{x_1}
\end{displaymath}
```

↓ Input

which produces

$$f(x) = e^{x_1}$$

Output

This example is slightly incorrect as  $e$  isn't actually a variable and shouldn't be typeset in italic. The correct way to do this is:

```
\begin{displaymath}
f(x) = \mathrm{e}^{x_1}
\end{displaymath}
```

↑ Input

↓ Input

which results in:

## 9. MATHEMATICS

$$f(x) = e^{x^1}$$

Output

If you are going to use  $e$  a lot, it will be simpler to [define a new command](#) to do this. The definition should go in the [preamble](#):

```
\newcommand{\e}{\mathrm{e}}
```

↑Input

↓Input

Then it can be used in the document:

```
\begin{displaymath}
f(x_1, x_2) = \e^{x_1^2} + \e^{x_2^2}
\end{displaymath}
```

↑Input

↓Input

$$f(x_1, x_2) = e^{x_1^2} + e^{x_2^2}$$

Output

### 9.3.4 Functional Names

Functions such as `log` and `tan` can't simply be typed in as `log` or `tan` otherwise they will come out looking like the variables *l* times *o* times *g* (*log*) or *t* times *a* times *n* (*tan*). Instead you should use one of the commands listed in [table 9.5](#).

Table 9.5: Function Names

<code>\arccos</code>	<code>\arcsin</code>	<code>\arctan</code>	<code>\arg</code>	<code>\cos</code>	<code>\cosh</code>
<code>\cot</code>	<code>\coth</code>	<code>\csc</code>	<code>\deg</code>	<code>\det</code>	<code>\dim</code>
<code>\exp</code>	<code>\gcd</code>	<code>\hom</code>	<code>\inf</code>	<code>\ker</code>	<code>\lg</code>
<code>\lim</code>	<code>\liminf</code>	<code>\limsup</code>	<code>\ln</code>	<code>\log</code>	<code>\max</code>
<code>\min</code>	<code>\Pr</code>	<code>\sec</code>	<code>\sin</code>	<code>\sinh</code>	<code>\sup</code>
<code>\tan</code>	<code>\tanh</code>				

Of these functions, the following functions can have limits by using the subscript command `_` or the superscript command `^`:

## 9. MATHEMATICS

`\det`    `\gcd`    `\inf`    `\lim`    `\liminf`  
`\limsup`    `\max`    `\min`    `\Pr`    `\sup`

Examples:

1. This example uses the cos and sin functions and also the [Greek letter theta](#).

```
\begin{displaymath}
z = r(\cos\theta + i\sin\theta)
\end{displaymath}
```

↑ Input

↓ Input

$$z = r(\cos \theta + i \sin \theta)$$

Output

2. This example has a limit. The command `\infty` is the infinity symbol  $\infty$ , and the command `\to` displays an arrow pointing to the right. Note the use of `_` since the limit is a subscript.

```
\begin{displaymath}
\lim_{x\to\infty} f(x)
\end{displaymath}
```

↑Input

↓Input

$$\lim_{x \rightarrow \infty} f(x)$$

Output

The operators with limits behave differently depending on whether they are in [displayed](#) or [in-line](#) maths. Notice the difference when the same code appears in-line:

[Sub- and superscript positioning for operators]

In a line of text `\lim_{x\to\infty} f(x)`

Input

which now displays as:

In a line of text `\lim_{x\to\infty} f(x)`

Output

## 9. MATHEMATICS

3. This is another example of a functional name using a subscript:

```
\begin{displaymath}
\min_x f(x)
\end{displaymath}
```

↑ Input

↓ Input

$$\min_x f(x)$$

Output

Again, notice the difference when it is used in-line:

In a line of text `\min_x f(x)`

Input

In a line of text  $\min_x f(x)$

Output

## 9. MATHEMATICS

In addition, the following commands are also available:

Command	Example Input	Example Output
<code>\bmod</code>	<code>\$m \bmod n\$</code>	$m \bmod n$
<code>\pmod{&lt;maths&gt;}</code>	<code>\$m \pmod{n}\$</code>	$m \pmod{n}$

If you want a function that isn't specified in [table 9.5](#), you can define a new function using the command

```
\DeclareMathOperator{<cmd>}{<operator name>}
```

Definition

which is defined in the `amsopn` package. There is also a starred version of this command which will define a new function that can take limits (like `\lim` and `\min` described above)

```
\DeclareMathOperator*{<cmd>}{<operator name>}
```

Definition

Note that both forms of `\DeclareMathOperator` can only be used in the preamble. The first argument `<cmd>` is the name of the new command to produce this function (so it must start with a backslash), and the second argument is the name of the function.

[Defining a new log-like function in LaTeX]

Examples

## 9. MATHEMATICS

1. Suppose we want a function called `card`, which represents the cardinality of a set  $\mathcal{S}$ , we first define the new function (which I'm going to call `\card`):

```
\DeclareMathOperator{\card}{card}
```

Input

(Remember the above must be done in the preamble.) Now I can use this new function:

```
\begin{displaymath}
n = \card(\mathcal{S})
\end{displaymath}
```

↑ Input

↓ Input

$$n = \text{card}(\mathcal{S})$$

Output

In this example `\mathcal` is used as sets are usually represented in a calligraphic font.



## 9. MATHEMATICS

2. Let's have an example of an operator that takes a limit. Firstly, the following line needs to go in the preamble:

```
\DeclareMathOperator*{\mode}{mode}
```

Input

Then the following can go in the document:

```
\begin{displaymath}
x_m = \mode_{x \in \mathcal{S}}(x)
\end{displaymath}
```

↑ Input

↓ Input

which results in:

$$x_m = \operatorname{mode}_{x \in \mathcal{S}}(x)$$

Output

### 9.3.5 Fractions

Fractions are created using the command

`\frac{<numerator>}{<denominator>}`

Definition

Examples:

1. A simple fraction:

```
\begin{displaymath}
\frac{1}{1+x}
\end{displaymath}
```

↑ Input

↓ Input

$$\frac{1}{1+x}$$

Output

## 9. MATHEMATICS

2. A nested fraction:

```
\begin{displaymath}
\frac{1+\frac{1}{x}}{1+x+x^2}
\end{displaymath}
```

↑ Input

↓ Input

$$\frac{1 + \frac{1}{x}}{1 + x + x^2}$$

Output

3. A derivative:

```
\begin{displaymath}
f'(x) = \frac{df}{dx}
\end{displaymath}
```

↑ Input

## 9. MATHEMATICS

↓ Input

$$f'(x) = \frac{df}{dx}$$

Output

Again, as with e, the differential operator ‘d’ should be in an upright font as it is not a variable:

↑ Input

```
\begin{displaymath}
f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}
\end{displaymath}
```

↓ Input

$$f'(x) = \frac{df}{dx}$$

Output

## 9. MATHEMATICS

4. The above example is rather cumbersome, particularly if you have a lot of derivatives, so it might be easier to [define a new command](#). In the [preamble](#) define:

```
\newcommand{\deriv}[2]{\frac{\mathrm{d}#1}{\mathrm{d}#2}}
```

Input

Then in the document:

```
\begin{displaymath}
f'(x) = \deriv{f}{x}
\end{displaymath}
```

↑ Input

↓ Input

$$f'(x) = \frac{df}{dx}$$
 Output

## 9. MATHEMATICS

5. Partial derivatives can be obtained similarly using the command `\partial` to display the partial derivative symbol. As in the previous example, first define a new command to format a partial derivative in the [preamble](#):

```
\newcommand{\pderiv}[2]{\frac{\partial #1}{\partial #2}}
```

Input

Then in the document:

```
\begin{displaymath}
f_x = \pderiv{f}{x}
\end{displaymath}
```

InputInput

$$f_x = \frac{\partial f}{\partial x}$$
 Output

## 9. MATHEMATICS

6. A double partial derivative:

```
\begin{displaymath}
f_{xy} = \frac{\partial^2 f}{\partial x \partial y}
\end{displaymath}
```

↑ Input

↓ Input

$$f_{xy} = \frac{\partial^2 f}{\partial x \partial y}$$

Output

### 9.3.6 Roots

Roots are obtained using the command

```
\sqrt[<order>]{<maths>}
```

Definition

## 9. MATHEMATICS

without the optional argument `<order>` it will produce a simple square root. Cubic roots etc can be obtained using the optional argument.

Examples:

1. A square root:

```
\begin{displaymath}
\sqrt{a+b}
\end{displaymath}
```

↑ Input

↓ Input

$$\sqrt{a+b}$$

Output

2. A cubic root:

↑ Input



## 9. MATHEMATICS

```
\begin{displaymath}
\sqrt[3]{a+b}
\end{displaymath}
```

↓ Input

$$\sqrt[3]{a+b}$$

Output

### 3. An $n$ th root:

```
\begin{displaymath}
\sqrt[n]{a+b}
\end{displaymath}
```

↑ Input

↓ Input

$$\sqrt[n]{a+b}$$

Output

### 9.3.7 Mathematical Symbols

Relational symbols are shown in [table 9.6](#). If you want a negation that is not shown, you can obtain it by preceding the symbol with the command `\not`. For example: `\not\subset` produces the symbol  $\not\subset$ .

[Where can I find the symbol for ...]

Table 9.6: Relational Symbols

<code>\approx</code>	$\approx$	<code>\asymp</code>	$\asymp$	<code>\bowtie</code>	$\bowtie$
<code>\cong</code>	$\cong$	<code>\dashv</code>	$\dashv$	<code>\doteq</code>	$\doteq$
<code>\equiv</code>	$\equiv$	<code>\frown</code>	$\frown$	<code>\ge or \geq</code>	$\geq$
<code>\gg</code>	$\gg$	<code>\in</code>	$\in$	<code>\le or \leq</code>	$\leq$
<code>\ll</code>	$\ll$	<code>\mid or  </code>	$ $	<code>\models</code>	$\models$
<code>\neq</code>	$\neq$	<code>\ni</code>	$\ni$	<code>\notin</code>	$\notin$
<code>\parallel</code>	$\parallel$	<code>\prec</code>	$\prec$	<code>\preceq</code>	$\preceq$
<code>\perp</code>	$\perp$	<code>\propto</code>	$\propto$	<code>\sim</code>	$\sim$
<code>\simeq</code>	$\simeq$	<code>\smile</code>	$\smile$	<code>\sqsubseteq</code>	$\sqsubseteq$
<code>\sqsupseteq</code>	$\sqsupseteq$	<code>\subset</code>	$\subset$	<code>\subseteq</code>	$\subseteq$
<code>\succ</code>	$\succ$	<code>\succeq</code>	$\succeq$	<code>\supseteq</code>	$\supseteq$
<code>\supseteq</code>	$\supseteq$	<code>\vdash</code>	$\vdash$		

## 9. MATHEMATICS

Binary operator signals are shown in [table 9.7](#), and arrow symbols are shown in [table 9.8](#).

Table 9.7: Binary Operator Symbols

<code>\amalg</code>	$\amalg$	<code>\ast</code>	$*$	<code>\bullet</code>	$\bullet$
<code>\bigcirc</code>	$\bigcirc$	<code>\bigtriangledown</code>	$\bigtriangledown$	<code>\bigtriangleup</code>	$\bigtriangleup$
<code>\cap</code>	$\cap$	<code>\cdot</code>	$\cdot$	<code>\circ</code>	$\circ$
<code>\cup</code>	$\cup$	<code>\dagger</code>	$\dagger$	<code>\ddagger</code>	$\ddagger$
<code>\diamond</code>	$\diamond$	<code>\div</code>	$\div$	<code>\mp</code>	$\mp$
<code>\odot</code>	$\odot$	<code>\ominus</code>	$\ominus$	<code>\oplus</code>	$\oplus$
<code>\oslash</code>	$\oslash$	<code>\otimes</code>	$\otimes$	<code>\pm</code>	$\pm$
<code>\setminus</code>	$\setminus$	<code>\sqcap</code>	$\sqcap$	<code>\sqcup</code>	$\sqcup$
<code>\star</code>	$\star$	<code>\times</code>	$\times$	<code>\triangleleft</code>	$\triangleleft$
<code>\triangleright</code>	$\triangleright$	<code>\uplus</code>	$\uplus$	<code>\vee</code>	$\vee$
<code>\wedge</code>	$\wedge$	<code>\wr</code>	$\wr$		

Symbols that can have limits are shown in [table 9.9](#). The size of these symbols depends on whether they are in displayed maths or in-line maths. Examples:

Table 9.8: Arrow Symbols

<code>\downarrow</code>	$\downarrow$	<code>\Downarrow</code>	$\Downarrow$
<code>\hookleftarrow</code>	$\hookleftarrow$	<code>\hookrightarrow</code>	$\hookrightarrow$
<code>\leftarrow</code> or <code>\gets</code>	$\leftarrow$	<code>\Leftarrow</code>	$\Leftarrow$
<code>\leftharpoondown</code>	$\leftharpoondown$	<code>\leftharpoonup</code>	$\leftharpoonup$
<code>\leftrightarrow</code>	$\leftrightarrow$	<code>\Leftrightarrow</code>	$\Leftrightarrow$
<code>\longleftarrow</code>	$\longleftarrow$	<code>\Longleftarrow</code>	$\Longleftarrow$
<code>\longlefttrightarrow</code>	$\longleftrightarrow$	<code>\Longlefttrightarrow</code>	$\longleftrightarrow$
<code>\longmapsto</code>	$\longmapsto$	<code>\longrightarrow</code>	$\longrightarrow$
<code>\Longrightarrow</code>	$\Longrightarrow$	<code>\mapsto</code>	$\mapsto$
<code>\nearrow</code>	$\nearrow$	<code>\nrightarrow</code>	$\nrightarrow$
<code>\rightarrow</code> or <code>\to</code>	$\rightarrow$	<code>\Rightarrow</code>	$\Rightarrow$
<code>\rightharpoondown</code>	$\rightharpoondown$	<code>\rightharpoonup</code>	$\rightarrow$
<code>\rightleftharpoons</code>	$\rightleftharpoons$	<code>\searrow</code>	$\searrow$
<code>\swarrow</code>	$\swarrow$	<code>\uparrow</code>	$\uparrow$
<code>\Uparrow</code>	$\Uparrow$	<code>\updownarrow</code>	$\updownarrow$
<code>\Updownarrow</code>	$\Updownarrow$		

Table 9.9: Symbols with Limits

<code>\sum</code>	$\Sigma$	<code>\int</code>	$\int$	<code>\oint</code>	$\oint$
<code>\prod</code>	$\Pi$	<code>\coprod</code>	$\coprod$	<code>\bigcap</code>	$\bigcap$
<code>\bigcup</code>	$\bigcup$	<code>\bigsqcup</code>	$\bigsqcup$	<code>\bigvee</code>	$\bigvee$
<code>\bigwedge</code>	$\bigwedge$	<code>\bigodot</code>	$\bigodot$	<code>\bigotimes</code>	$\bigotimes$
<code>\bigoplus</code>	$\bigoplus$	<code>\biguplus</code>	$\biguplus$		

## 1. Displayed Maths

```

\begin{displaymath}
f(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i
\end{displaymath}

```

↑Input

↓Input

$$f(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i$$

Output

## 2. In-line Maths

```
\begin{math}
f(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i
\end{math}
```

↑ Input

↓ Input

$$f(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i$$

Output

It is possible to stack subscripts or superscripts using

```
{above} \atop {below}
```

Definition

## 9. MATHEMATICS

For example:

```
\begin{displaymath}
\sum_{i \in \mathcal{I} \ \text{atop} \ i \neq 0} x_i
\end{displaymath}
```

↑Input

↓Input

which produces:

$$\sum_{\substack{i \in \mathcal{I} \\ i \neq 0}} x_i$$

↑Output

↓Output

Ellipsis commands (omission marks) are shown in [table 9.10](#).

Examples:

1. Low ellipsis: This example uses the command `\forall` to produce the “for all” symbol  $\forall$ , and it also uses `\_` (backslash space) to make a space before the for all symbol:

Table 9.10: Ellipses

<code>\ldots</code>	<code>...</code>	<code>\cdots</code>	<code>\dots</code>
<code>\vdots</code>	<code>\vdots</code>	<code>\ddots</code>	<code>\ddots</code>

```
\begin{displaymath}
a_{ix_i} = b_i \ \forall i = 1, \dots, n
\end{displaymath}
```

↑ Input

↓ Input

$$a_i x_i = b_i \ \forall i = 1, \dots, n$$

Output



## 9. MATHEMATICS

### 2. Centred ellipsis:

```
\begin{displaymath}
y = a_1 + a_2 + \cdots + a_n
\end{displaymath}
```

↑Input

↓Input

$$y = a_1 + a_2 + \cdots + a_n$$

Output

For other common symbol commands, see *A Guide to L<sup>A</sup>T<sub>E</sub>X* [4] or *The L<sup>A</sup>T<sub>E</sub>X Companion* [5]. For a comprehensive list of symbols see *The Comprehensive L<sup>A</sup>T<sub>E</sub>X Symbol List* by Scott Pakin *et al.* which is available from CTAN [8].

### Exercise 22 (Maths: Fractions and Symbols)

This exercise uses a fraction, a square root, subscripts, superscripts and symbols. Try reproducing the following output:

The quadratic equation

$$\sum_{i=0}^2 a_i x^i = 0$$

has solutions given by

$$x = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2}$$

Again you can [download](#) or [view](#) the solution.

---

### 9.3.8 Delimiters

Placing brackets around a tall object in maths mode, such as fractions, does not look right if you use normal sized brackets. For example:

## 9. MATHEMATICS

```
\begin{displaymath}
(\frac{1}{1+x})
\end{displaymath}
```

[↓ Input](#)

results in:

$$\left(\frac{1}{1+x}\right)$$

[Output](#)

Under such circumstances, it is better to use the commands:

```
\left<delimiter>
```

[Definition](#)

and

```
\right<delimiter>
```

[Definition](#)

Note that you must always have matching `\left` and `\right` commands, although the delimiters used may be different. If you want one of the delimiters to be invisible, use a `.` (full stop) as the delimiter. Available delimiters are shown in [table 9.11](#).

Examples:

Table 9.11: Delimiters

(	(	)	)	[	[	]	]
\{	{	\}	}			\	\
/	/	\backslash	\	\langle	\langle	\rangle	\rangle
\lfloor		\rfloor		\lceil		\rceil	
\uparrow	↑	\downarrow	↓	\Uparrow	↑	\Downarrow	↓
\updownarrow	↕	\Updownarrow	↕				

## 1. Round bracket delimiters:

```

\begin{displaymath}
\left(
\frac{1}{1+x}
\right)
\end{displaymath}

```

↑Input

↓Input

$$\left(\frac{1}{1+x}\right)$$

Output

2. Vertical bar delimiters:

```
\begin{displaymath}
\left|
\frac{1}{1+x}
\right|
\end{displaymath}
```

↑ Input

↓ Input

$$\left|\frac{1}{1+x}\right|$$

Output

## 9. MATHEMATICS

### 3. Delimiters don't have to match:

```
\begin{displaymath}
\left[\frac{1}{1+x}\right\rangle
\end{displaymath}
```

↑ Input

↓ Input

$$\left[\frac{1}{1+x}\right\rangle$$

Output

### 4. An invisible delimiter:

```
\begin{displaymath}
\left.
\frac{\partial f}{\partial x}
\right.
```

↑ Input

## 9. MATHEMATICS

```
\right|_{x=0}
\end{displaymath}
```

↓ Input

$$\left. \frac{\partial f}{\partial x} \right|_{x=0}$$

Output

We have now learnt enough to reproduce the equation shown in [chapter 1](#):

```
\newcommand{\pderiv}[2]{\frac{\partial #1}{\partial #2}}
\newcommand{\e}{\mathrm{e}}

\begin{displaymath}
\pderiv{^2\mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
```

↑ Input

## 9. MATHEMATICS

```
\pderiv{v_i}{\rho_i} \frac{e^{v_i}}{1-e^{v_i}}
+ v_i \frac{e^{v_i} \pderiv{v_i}{\rho_i} (1-e^{v_i})
      + e^{2v_i} \pderiv{v_i}{\rho_i} (1-e^{v_i})^2}
\right)
\end{displaymath}
```

↓ Input

↑ Output

$$\frac{\partial^2 \mathcal{L}}{\partial z_i^{\rho^2}} = -\frac{\partial \rho_i}{\partial z_i^{\rho}} \left( \frac{\partial v_i}{\partial \rho_i} \frac{e^{v_i}}{1-e^{v_i}} + v_i \frac{e^{v_i} \frac{\partial v_i}{\partial \rho_i} (1-e^{v_i}) + e^{2v_i} \frac{\partial v_i}{\partial \rho_i}}{(1-e^{v_i})^2} \right)$$

↓ Output

**Note 6:** The above code looks a bit complicated, and there are so many braces that it can be easy to lose track, so here are some ways of making it a little easier to type:



## 9. MATHEMATICS

1. Whenever you start a new environment type in the `\begin` and `\end` bits first, and then insert whatever goes inside the environment. This ensures that you always have a matching `\begin` and `\end`.
2. Whenever you type any braces, always type the opening and closing braces first, and then insert whatever goes in between. This will ensure that your braces always match up.

So keeping these notes in mind, let's try typing in the code in a methodical manner:

1. Start the `displaymath` environment:

```
\begin{displaymath}  
\end{displaymath}
```

↑ Input

↓ Input

## 9. MATHEMATICS

- We now need a partial derivative (remember to define `\pderiv` in the preamble as we did [earlier](#)):

```
\begin{displaymath}
\pderiv{}{}
\end{displaymath}
```

↑Input

↓Input

- Let's do the first argument. This partial derivative is actually a double derivative, which means we need a squared bit on the top along with a calligraphic L:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{}
\end{displaymath}
```

↑Input

↓Input

## 9. MATHEMATICS

4. The second argument is the  $z_i^\rho$  squared bit. This is a nested superscript  $\{z_i^\rho\}^2$ :

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2}
\end{displaymath}
```

↑Input

↓Input

5. We can do the next partial derivative in the same way. This one is slightly easier to do:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\end{displaymath}
```

↑Input

↓Input

## 9. MATHEMATICS

6. Delimiters also need to occur in pairs, like curly braces and `\begin` and `\end`, so let's do them next:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
\right)
\end{displaymath}
```

↑Input

↓Input

7. Now we need to do the bits inside the brackets. First of all we have yet another partial derivative:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
```

↑Input

## 9. MATHEMATICS

```
\pderiv{v_i}{\rho_i}
\right)
\end{displaymath}
```

[↓Input](#)

8. Now we have a fraction:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
\pderiv{v_i}{\rho_i} \frac{e^{v_i}}{1-e^{v_i}}
\right)
\end{displaymath}
```

[↑Input](#)

[↓Input](#)

9. This is followed by  $v_i$  times another fraction:

## 9. MATHEMATICS

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
\pderiv{v_i}{\rho_i} \frac{e^{v_i}}{1-e^{v_i}}
+ v_i \frac{\quad}{\quad}
\right)
\end{displaymath}
```

↑ Input

↓ Input

10. This fraction is quite complicated. The bottom part of the fraction is easier than the top, so let's do that first:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
```

↑ Input

## 9. MATHEMATICS

```
\pderiv{v_i}{\rho_i} \frac{e^{v_i}}{1-e^{v_i}}
+ v_i \frac{}{(1-e^{v_i})^2}
\right)
\end{displaymath}
```

[Input](#)

11. Now it's time for the top part of the fraction. It's a bit complicated, so let's break it down:

(a) The first term is:

```
e^{v_i}
```

(b) The next term is another partial derivative:

```
\pderiv{v_i}{\rho_i}
```

(c) Then we have:

```
(1-e^{v_i})
```

(d) Next we have to add on:

## 9. MATHEMATICS

$$+e^{2v_i}$$

(e) And finally we have the last term:

$$\pderiv{v_i}{\rho_i}$$

12. Putting it all together, we have:

```
\begin{displaymath}
\pderiv{^2\mathcal{L}}{\{z_i^\rho\}^2} =
-\pderiv{\rho_i}{z_i^\rho}
\left(
\pderiv{v_i}{\rho_i} \frac{e^{v_i}}{1-e^{v_i}}
+ v_i \frac{e^{v_i} \pderiv{v_i}{\rho_i} (1-e^{v_i})
+ e^{2v_i} \pderiv{v_i}{\rho_i}}{(1-e^{v_i})^2}
\right)
\end{displaymath}
```

↑Input

↓Input



13. And remember that if you haven't already defined `\pderiv` and `\e`, you will need to do that in the [preamble](#)

```
\newcommand{\pderiv}[2]{\frac{\partial #1}{\partial #2}}
\newcommand{\e}{\mathrm{e}}
```

↑ Input

↓ Input

(Note that if we hadn't defined these two commands, the code would have had to have been far more complicated.)

### 9.3.9 Arrays

Mathematical structures such as matrices and vectors require elements to be arranged in rows and columns. Just as we can align material in rows and columns in text mode using the [tabular](#) environment, we can do the same in maths mode using the `array` environment. The `array` environment has the same format as the `tabular` environment, however it must be in maths mode. Examples:

1. A simple array, all three columns are right justified:

## 9. MATHEMATICS

```
\begin{displaymath}
\begin{array}{rrr}
0 & 1 & 19 \\
-6 & 10 & 200
\end{array}
\end{displaymath}
```

↑ Input

↓ Input

$$\begin{array}{rrr} 0 & 1 & 19 \\ -6 & 10 & 200 \end{array}$$

Output

2. Let's add some [delimiters](#):

```
\begin{displaymath}
```

↑ Input

## 9. MATHEMATICS

```
\left(  
\begin{array}{rrr}  
0 & 1 & 19\\-6 & 10 & 200  
\end{array}  
\right)  
\end{displaymath}
```

↓ Input

$$\left( \begin{array}{ccc} 0 & 1 & 19 \\ -6 & 10 & 200 \end{array} \right)$$

Output

3. This example uses an [invisible delimiter](#):

```
\begin{displaymath}  
f(x) =  
\left\{
```

↑ Input

## 9. MATHEMATICS

```
\begin{array}{rl}
-1 & x < 0 \\
0 & x = 0 \\
+1 & x > 0
\end{array}
\right.
\end{displaymath}
```

[↓ Input](#)

$$f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ +1 & x > 0 \end{cases}$$

[Output](#)

### 9.3.10 Vectors

Vectors can be produced using the command:

```
\vec{<variable>}
```

[Definition](#)

## 9. MATHEMATICS

Example:

```
\begin{displaymath}
\vec{x}
\end{displaymath}
```

↑Input

↓Input

$\vec{x}$

Output

These days it is customary to typeset vectors in bold. This can be done by [redefining](#) the `\vec` command. You could use `\mathbf`, for example:

```
\renewcommand{\vec}[1]{\mathbf{#1}}
\begin{displaymath}
\vec{x}\cdot\vec{\xi} = z
\end{displaymath}
```

↑Input

## 9. MATHEMATICS

↓ Input

$$\mathbf{x} \cdot \xi = z$$

Output

however, as you may have noticed, the Greek letter  $\xi$  has not come out in bold. Here's an alternative (using `\boldsymbol` defined in the `amsfonts` package):

↑ Input

```
\renewcommand{\vec}[1]{\boldsymbol{#1}}
\begin{displaymath}
\vec{x}\cdot\vec{\xi} = z
\end{displaymath}
```

↓ Input

$$\mathbf{x} \cdot \boldsymbol{\xi} = z$$

Output

**Exercise 23 (Maths: Vectors and Arrays)**

Try to produce the following:

$$\mathbf{Ax} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \end{pmatrix} = \mathbf{y}$$

↑Output

↓Output

As before, you can [download](#) or [view](#) the solution.

**9.3.11 Mathematical Spacing**

L<sup>A</sup>T<sub>E</sub>X deals with mathematical spacing fairly well, but sometimes you may find you want to adjust the spacing yourself. Available spacing commands are listed in [table 9.12](#).

**Exercise 24 (More Mathematics)**

This exercise uses the spacing command `\quad`. It also has a [func-](#)

Table 9.12: Mathematical Spacing Commands

Command	Example Input	Example Output
	<code>\$AB\$</code>	$AB$
<code>\thinspace</code> or <code>\,</code>	<code>\$A\,B\$</code>	$A B$
<code>\medspace</code> or <code>\:</code>	<code>\$A\:B\$</code>	$A B$
<code>\thickspace</code> or <code>\;</code>	<code>\$A\;B\$</code>	$A B$
<code>\quad</code>	<code>\$A\quad B\$</code>	$A \quad B$
<code>\qquad</code>	<code>\$A\qquad B\$</code>	$A \qquad B$
<code>\negthinspace</code> or <code>\!</code>	<code>\$A\!B\$</code>	$AB$
<code>\negmedspace</code>	<code>\$A\negmedspace B\$</code>	$AB$
<code>\negthickspace</code>	<code>\$A\negthickspace B\$</code>	$AB$



## 9. MATHEMATICS

tion name, `diag`, and it uses the `\forall` and `\ellipses` symbols. It also redefines the `\vec` command, as was done in the previous section, and it uses `delimiters` and the `array` environment, as well as using `subscripts` and `superscripts`.

Try to reproduce the following output:

The set of linear equations:

$$a_i x_i = b_i \quad \forall i = 1, \dots, n$$

can be written as a matrix equation:

$$\text{diag}(\mathbf{a})\mathbf{x} = \mathbf{b}$$

where  $\mathbf{x} = (x_1, \dots, x_n)^T$ ,  $\mathbf{b} = (b_1, \dots, b_n)^T$  and

$$\text{diag}(\mathbf{a}) = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix}$$

↑Output

## 9. MATHEMATICS

↓Output

Again, you can [download](#) or [view](#) the solution.

---

# Chapter 10

## Defining Environments

Just as you can [define new commands](#), you can also define new [environments](#). The command

```
\newenvironment{<env-name>}[<n-args>][<default>]{<begin-code>}{<end-code>}
```

Definition

is used to define a new environment. As with new commands, you can use the optional argument `<n-args>` to define an environment with arguments, and `<default>` to define an environment with an optional argument.

The first argument `<env-name>` is the name of your new environment. Remember that the environment name must not have a backslash. The mandatory arguments `<begin-code>` and `<end-code>` indicate what L<sup>A</sup>T<sub>E</sub>X should do at the beginning and end of the environment. Let's first consider an example of an environment without any arguments. Let's make an environment called, say, `exercise` that prints **Exercise** in bold and typesets the contents of the environment in italic. In other words, we

## 10. DEFINING ENVIRONMENTS

want the following code:

```
\begin{exercise}  
This is a sample.  
\end{exercise}
```

↑Input

↓Input

to produce the following output:

```
Exercise  
This is a sample.
```

↑Output

↓Output

Let's first consider what we want this environment to do: we can get the word “Exercise” in bold by simply doing `\textbf{Exercise}`, and the italic font can be obtained by using the [itshape environment](#). So, at the start of our new environment we need to do `\textbf{Exercise}` and we need to begin the `itshape` environment, and at the end of our new environment we need to end the `itshape` environment:

↑Input

## 10. DEFINING ENVIRONMENTS

```
\newenvironment{exercise}%           environment name
{\textbf{Exercise}\begin{itshape}}%  begin code
{\end{itshape}}%                    end code
```

↓ Input

Let's try it out:

```
\begin{exercise}
This is a sample.
\end{exercise}
```

↑ Input

↓ Input

**Exercise** *This is a sample.*

Output

Not quite right. Let's put a paragraph break after **Exercise**, and put one before it as well. The command `\par` can be used to make a paragraph break:

```
\newenvironment{exercise}%           environment name
{\par\textbf{Exercise}\begin{itshape}\par}%  begin code
{\end{itshape}}%                    end code
```

↑ Input

↓Input

Let's have a look at the output now:

### **Exercise**

*This is a sample.*

↑Output

↓Output

One more thing, we need to remove the paragraph indentation. This can be done using the command `\noindent`:

↑Input

```
\newenvironment{exercise}
{\par\noindent\textbf{Exercise}\begin{itshape}\par\noindent}
{\end{itshape}}
```

↓Input

The exercise environment now appears as:

### **Exercise**

*This is a sample.*

↑Output

↓Output

Now let's modify our code so that the environment takes an argument. The argument should indicate the exercise topic. For example, the following code:

```
\begin{exercise}{An Example}
This is a sample.
\end{exercise}
```

↑Input

↓Input

should produce the following result:

```
Exercise (An Example)
This is a sample.
```

↑Output

↓Output

As with `\newcommand`, `#1` is used to indicate the first argument. We can now modify the code as follows (modifications are indicated like this):

```
\newenvironment{exercise}[1]{%
```

↑Input

## 10. DEFINING ENVIRONMENTS

```
{\par\noindent\textbf{Exercise #1}\begin{itshape}\par\noindent}%  
\end{itshape}}
```

[↓ Input](#)

**Note 7:** When you define a new environment called  $\langle name \rangle$ , `\newenvironment` creates the commands `\langle name \rangle` and `\end\langle name \rangle`, so when you define a new environment, make sure that you don't give it the same name as an existing command. Likewise, when you define a new command, you need to make sure that there is no environment of the same name. When  $\text{\LaTeX}$  encounters `\begin\langle name \rangle` it implements `\langle name \rangle` (as well as some other stuff, such as starting a [group](#)) and when  $\text{\LaTeX}$  encounters `\end\langle name \rangle`, it implements the command `\end\langle name \rangle` if it exists (and also does some other stuff, such as ending the group). This is why [declarations](#) such as `\bfseries` can also be used as an environment of the same name without the backslash. You can't however use a text-block command, such as `\textbf`, as an environment.<sup>1</sup>

---

<sup>1</sup>Actually, it is possible to do, say, `\begin{textbf}Some text\end{textbf}`, but only the S would appear in bold, since it is approximately the same as `{\textbf Some text}`. Whilst you can do `\begin{textbf}\{Some text\}\end{textbf}`, it is much more efficient to type `\textbf\{Some text\}`.



So the above example, creates two commands: `\exercise` (which takes an argument) and `\endexercise` (which doesn't take an argument). However, *don't* be tempted to use `\exercise` and `\endexercise` explicitly as you will miss out on L<sup>A</sup>T<sub>E</sub>X's `\begin/\end` checking mechanism, and you will not be able to benefit from the implicit grouping resulting from using an environment. If you are not sure if something is supposed to be used as a command or as an environment then check your reference manuals, or package documentation if it is defined in a package.

### Exercise 25 (Defining a New Environment)

If you did any of the exercises from [exercise 9](#) to [exercise 18](#), go back to the document you created and define the `exercise` environment as in the example above. Then try creating some exercises using this environment. You could, maybe, put an exercise in the first chapter, and then another one in the second chapter.

Then try modifying the environment so that it puts a bit of vertical space before and after the environment using `\vspace{<length>}`. Again you can [download](#) or [view](#) an example.

## 10. *DEFINING ENVIRONMENTS*

---

# Chapter 11

## Counters

As we have seen,  $\text{\LaTeX}$  automatically generates numbers for chapters, sections, equations etc. These numbers are stored in counters. The names of these counters are usually the same as the name of the object with which it is associated but without any backslash. For example, the `\chapter command` has an associated counter called `chapter`, the `\footnote command` has an associated counter called `footnote`, the `equation environment` has an associated counter called `equation`, the `figure environment` has an associated counter called `figure` and the `table environment` has an associated counter called `table`. There is also a counter called `page` that keeps track of the current page number.

The value of a counter can be displayed using the command

```
\the<counter>
```

Definition

where `<counter>` is the name of the associated counter. Note that `<counter>` does not go in curly braces and adjoins `\the` (e.g. `\thepage`, `\thesection`, `\thechapter`). Example:

[Page number  
is wrong at  
start of page]

## 11. COUNTERS

```
This page is Page~\thepage.  
The current chapter is Chapter~\thechapter.
```

↑ Input

↓ Input

produces:

This page is Page 317. The current chapter is Chapter 11.

Output

New counters can be created using the command:

```
\newcounter{<counter-name>}[<outer-counter>]
```

Definition

The [mandatory argument](#) `<counter-name>` is the name of your new counter (no backslash in the name). For example, let's define a counter called `exercise` to keep track of each exercise.

```
\newcounter{exercise}
```

Input

We can now display the value of the counter using the command `\theexercise`. At the moment the counter has the value zero, the value can be changed using one of the following commands:

## 11. COUNTERS

<code>\stepcounter{&lt;counter&gt;}</code>	Increments <i>&lt;counter&gt;</i> by 1
<code>\refstepcounter{&lt;counter&gt;}</code>	As above, but allows you to cross-reference the counter using <code>\label</code> and <code>\ref</code>
<code>\setcounter{&lt;counter&gt;}{&lt;num&gt;}</code>	Sets the counter to <i>&lt;num&gt;</i>
<code>\addtocounter{&lt;counter&gt;}{&lt;num&gt;}</code>	Adds <i>&lt;num&gt;</i> to <i>&lt;counter&gt;</i>

A couple of the commands above take a number *<num>* as one of the arguments. If you want to use another counter for this argument, you need to use

`\value{<counter>}`

Definition

For example, if you want to set our new `exercise` counter to the same value as the `page` counter, you would do

`\setcounter{exercise}{\value{page}}`

Input

Let's go back to the `exercise` environment you created in [exercise 25](#). The exercises really ought to have an associated number, and this number

## 11. COUNTERS

should be incremented each time we use the exercise environment. So let's modify our code to do this. Modifications are illustrated like this:

```
\newcounter{exercise}

\newenvironment{exercise}[1]%
{\refstepcounter{exercise}\vspace{10pt}\par\noindent
\textbf{Exercise \theexercise\ (#1)}
\begin{itshape}\par\noindent\vspace{10pt}}%
{\end{itshape}\vspace{10pt}\par}
```

↑ Input

↓ Input

Note that the counter needs to be incremented before it is used. Since we've used `\refstepcounter` instead of `\stepcounter` we can cross-reference our exercise environment:

```
Exercise~\ref{ex:simple} is a simple exercise.
```

```
\begin{exercise}{Simple Exercise}%
\label{ex:simple}%
```

↑ Input

## 11. COUNTERS

This is a simple exercise.

```
\end{exercise}
```

↓Input

(Why are there [percent symbols](#) at the end of some of the lines? What would happen without them?) This produces the following output:

[There's a space added after my environment]

Exercise 1 is a simple exercise.

↑Output

### Exercise 1 (Simple Exercise)

*This is a simple exercise.*

↓Output

The counter representation can be changed by redefining `\theexercise` using the command `\renewcommand`. The following commands can be used to display the counter:

[Redefining counters' \the-commands]

`\arabic{<counter>}` Arabic number (1, 2, 3, ...)

`\Roman{<counter>}` Uppercase Roman numeral (I, II, III, ...)

## 11. COUNTERS

`\roman{<counter>}` Lowercase Roman numeral (i, ii, iii, ...)

`\alph{<counter>}` Lowercase letter (a, b, c, ..., z)

`\Alph{<counter>}` Uppercase letter (A, B, C, ..., Z)

`\fnsymbol{<counter>}` A footnote symbol (\* † ‡ § ¶ || \*\* †† ‡‡)

For example, to make the chapter numbers appear as uppercase Roman numerals you would do:

```
\renewcommand{\thechapter}{\Roman{chapter}}
```

Input

You may have noticed that `\newcounter` has an optional argument `<outer-counter>`. This is for use if you require the new counter to be reset every time `<outer-counter>` is incremented. For example, the section numbers in the `report` class are dependent on the chapter numbers. Each time a new chapter is started, the section numbers are reset. Suppose we want our `exercise` counter to be dependent on the `chapter` counter, we would do

[Master and slave counters]

```
\newcounter{exercise}[chapter]
```

Input



## 11. COUNTERS

We now need to modify `\theexercise` so that it includes the chapter number:

```
\renewcommand{\theexercise}{\thechapter.\arabic{exercise}}
```

Input

Notice the use of `\thechapter` instead of, say, `\arabic{chapter}`. By using `\thechapter` we don't need to keep track of the chapter counter format.

### Exercise 26 (Using Counters)

Modify the document from [exercise 25](#) so that the exercise environment has a counter. Make the counter dependent on the chapter. You can [download](#) or [view](#) an example.

---

# Chapter 12

## Lengths

Lengths are commands that store dimensions (such as 1in, 5cm, 8.25mm). These are used to determine page layouts etc. For example, the page width is given by the length `\pagewidth`, and the height of the main body of text is given by `\textheight`. The paragraph indentation is given by `\parindent` and the gap between paragraphs is given by `\parskip`. Acceptable units of measurement are shown in [table 12.1](#).

Example: The default paragraph indentation `\parindent` is usually around 15pt, but we can change this if we like. To change a length you need to use the command:

```
\setlength{<cmd>}{<length>}
```

Definition

where `<cmd>` is the particular length command (e.g. `\parindent`) and `<length>` is the new length. To display the value of a length do:

```
\the<cmd>
```

Definition

Table 12.1: Units of Measurement

pt	point: $72.27\text{pt} = 1\text{in}$
in	inch: $1\text{in} = 25.4\text{mm}$
mm	millimetre: $1\text{mm} = 2.845\text{pt}$
cm	centimetre: $1\text{cm} = 10\text{mm}$
ex	height of the letter x in the current font
em	width of the letter M in the current font
sp	scaled point: $1\text{sp} = 65536\text{pt}$
bp	big point (or PostScript point): $72\text{bp} = 1\text{in}$
dd	didot point: $1\text{dd} = 0.376\text{mm}$
pc	pica: $1\text{pc} = 12\text{pt}$
cc	cicero: $1\text{cc} = 12\text{dd}$
mu	math unit: $18\text{mu} = 1\text{em}$

## 12. LENGTHS

For example, the following code:

```
\setlength{\parindent}{0pt}
```

This is the first paragraph.

This is the second paragraph.

The paragraph indentation is `\the\parindent`.

will produce the following output:

```
This is the first paragraph.
```

```
This is the second paragraph. The paragraph indentation is 0.0pt.
```

A rubber length is a length that has a certain amount of elasticity. This enables you to specify your desired length with a certain amount of flexibility that will allow  $\text{\LaTeX}$  to stretch or contract the space to get the body of text as flushed with the margins as possible.

## 12. LENGTHS

For example, the paragraph gap `\parskip` is usually set to 0pt plus 1pt. This means that the preferred gap is 0pt but L<sup>A</sup>T<sub>E</sub>X can stretch it up to 1pt to help prevent the page from having a ragged bottom. Let's further modify the above example:

```
\setlength{\parindent}{0pt}
\setlength{\parskip}{10pt plus 1pt minus 1pt}
```

This is the first paragraph.

This is the second paragraph.

The paragraph indentation is `\the\parindent`.

The paragraph skip is `\the\parskip`.

This now produces:

This is the first paragraph.

↑Input

↓Input

↑Output

## 12. LENGTHS

This is the second paragraph. The paragraph indentation is 0.0pt. The paragraph skip is 10.0pt plus 1.0pt minus 1.0pt.

[↓Output](#)

In this example, the preferred paragraph gap is 10pt but it will allow for a deviation of up to plus or minus 1pt.

If you want to change any of the page layout lengths (such as `\textwidth`), the easiest way to do it is to use the `geometry` package. This package should have been installed when you installed your  $\text{T}_{\text{E}}\text{X}$  distribution. Using an example from the `geometry` documentation: suppose you want the total text area to be 6.5in wide and 8.75in high, with a left margin 0.4in from the left edge, no header, and the first line of the page to be 1.2in from the top of the paper, then you would do:

```
\usepackage[body={6.5in,8.75in},top=1.2in,left=0.4in,nohead]{geometry}
```

# Chapter 13

## Common Errors

- If the only message that gets printed to the terminal is:

```
latex: Command not found.
```

or

```
Bad command or file name
```

then you have either mistyped the command name, or you don't have L<sup>A</sup>T<sub>E</sub>X installed on your computer, or your path hasn't been set up correctly. First check that you have typed the command correctly, then check to see if you have T<sub>E</sub>X installed. Failing that, contact your system administrator for help.


- If you get the message (or something similar):

```
This is TeX, Version 3.14159 (Web2C 7.3.1)
```

### 13. COMMON ERRORS

```
! I can't find file 'sample'.  
<*> sample
```

Please type another input file name:

then you have either misspelt the filename or you are in the wrong directory. If you have misspelt the filename, simply type in the correct name at the prompt. If you are in the wrong directory or you want to quit, type X followed by the return character . This is an error that you may encounter if you are using a [terminal](#), as typing errors may occur, or you may forget to change to the correct directory. To check you are in the right directory, you can type:

```
ls
```

if you are using a Unix-type system. This will list the contents of the directory. If you are certain that you have spelt the filename correctly and that you are in the right directory, there may be something wrong with your path, in which case contact your system administrator. You are unlikely to get this error with [WinEdt](#) or [TeXnicCenter](#) as they set the directory, and you only need to click a button, so you won't get any typing errors.



### 13. COMMON ERRORS

- Error messages will usually look something like:

```
! Undefined control sequence.  
1.1 \documentclass  
           [a4paper,11pt]{article}  
?
```

The first line is the error message. In this example I have misspelt the command `\documentclass`. The next line begins with `1.` followed by a number. This is the line number in the source code where the error occurred. In this case the error occurred on line 1. Following the line number is the input line `LATEX` has processed so far, and staggered on the next line is the remainder of the input line.

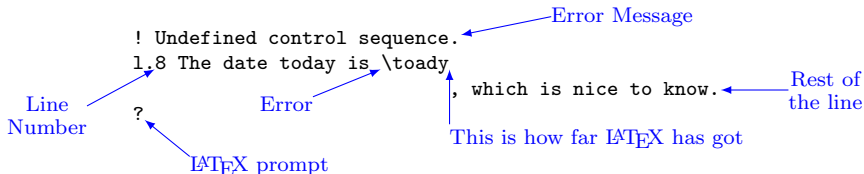
Here's another example. Suppose line 8 of my [source code](#) looks like:

```
The date today is: \toady, which is nice to know.
```

The error in this case is the misspelling of the command `\today`. The error message will appear as follows:

[The structure  
of TeX error  
messages]

## 13. COMMON ERRORS



At the L<sup>A</sup>T<sub>E</sub>X prompt, you can either type `h` for a help message, or type `x` to exit L<sup>A</sup>T<sub>E</sub>X and go back to your source code and fix the problem.

There follows below a list of common error messages. If your problem isn't listed there, try the UK TUG FAQ [2].

[How to approach errors]

### 13.1 \* (No message, just an asterisk prompt)

You've gone into T<sub>E</sub>X! This is probably because you've forgotten the `\end{document}`. The asterisk is the T<sub>E</sub>X prompt. At this point the best thing to do is type `\end{document}` at the prompt (followed by the return character  $\leftarrow$ ) and hope for the best.

["Please type a command or say \end"]

## 13.2 Argument of `\cline` has an extra }

If this error occurred on the first line in your tabular [environment](#), you may have forgotten the [argument](#) to the tabular environment.

## 13.3 Argument of `\multicolumn` has an extra }

If this error occurred on the first line in your tabular [environment](#), you may have forgotten the [argument](#) to the tabular environment.

## 13.4 `\begin{...}` ended by `\end{...}`

The beginning of your environment doesn't have a matching end.

- Check to make sure you have spelt the name of the environment correctly.

You will get this error message if you do, say,

```
\end{docment} (incorrect)
```

instead of

## 13. COMMON ERRORS

`\end{document}` (correct)

- Check that for every `\begin` you have a corresponding `\end` with the same name.

### 13.5 Bad math environment delimiter

Only a certain type of character may be used as a [delimiter](#) (e.g. ( ) [ ] \{ \} | \| . ), check which one you have specified. This error may also occur if you have forgotten a `\right` (Remember to use a `.` if you want an invisible delimiter) or you may have forgotten to end your array environment with `\end{array}`.

### 13.6 Can only be used in preamble.

Some commands, such as `\usepackage` may only appear in the [preamble](#). Check to see where you have put it. For example, this error will be caused by doing:

### 13. COMMON ERRORS

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

```
\begin{document}
```

(incorrect)

```
\usepackage{graphicx}
```

instead of

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

```
\usepackage{graphicx}
```

(correct)

```
\begin{document}
```

## 13.7 Command ... already defined

You have tried to define a [command](#) which already exists. Try giving it a different name. Remember never to redefine a command if you don't know what the command originally does.

Alternatively, you have tried to define an [environment](#) which already exists. Give the new environment a different name. Again, never redefine an environment where you don't know what the original environment does.

## 13.8 Display math should end with \$\$

You may have a dollar sign (\$) in a `displaymath` or equation `environment`. Remember that `$` is short hand for `\begin{math}` or `\end{math}`, so you can't end one of the other environments with a `$`. (This error message is in fact a bit confusing, as it seems to be suggesting that you end a displayed maths environment with `$$` which you also shouldn't do.)

[Why use `\[...\]` in place of `$$...$$`]

## 13.9 Environment ... undefined.

`LATEX` doesn't recognise the environment you have specified.

- Check you have spelt the environment name correctly.

You will get this error if you do, say,

```
\begin{docment} (incorrect)
```

instead of

```
\begin{document} (correct)
```

## 13. COMMON ERRORS

- If it's your own environment, check you have defined the environment before using it.
- If the environment is defined in a package, check you have included the package using the `\usepackage` command.

### 13.10 Extra alignment tab has been changed to `\cr`

You have too many ampersands (&) in one row. The most probable cause is that you have forgotten the end of row command `\\` on the previous row. Remember also that if you have a `\multicolumn` command to span more than one column, you should have fewer &s in that row.

[Alignment tab changed to `\cr`]

### 13.11 Extra `\right`

There are a number of possible causes. The most probable is that you have a `\right` that doesn't have a matching `\left`. (Remember left comes before right.) Another possible cause is that you have missed out `\end{array}`. (Remember that `environments` provide implicit `grouping`, and `\left` and its matching `\right` must appear within the same group level.)

## 13.12 File ended while scanning use of ...

The most usual cause of this error is a missing closing brace.

You will get this error if you do, say,

```
\end{document} (incorrect)
```

instead of

```
\end{document} (correct)
```

## 13.13 File not found.

L<sup>A</sup>T<sub>E</sub>X can't find the file you have specified. You will be given the opportunity to type in the correct filename at the prompt. If you want to quit, simply type **X** followed by the return character  $\leftarrow$ .

- Make sure that you have spelt the filename correctly.

This error will be caused by, say,

```
\documentclass[a4paper]{article} (incorrect)
```



### 13. COMMON ERRORS

instead of

```
\documentclass[a4paper]{article} (incorrect)
```

If this is the case, simply type in the correct name at the prompt (followed by the return character  $\leftarrow$ )

- Make sure that the file is in the same directory as your document or in the  $\LaTeX$  path. If the file is in another directory (not in the  $\LaTeX$  path), you will need to specify the pathname, but remember that when using  $\LaTeX$  under Windows, you need to use a forward slash (/) as the directory divider, as a backslash would be interpreted as a command. For example, if you have a file called `shapes.ps` in the subdirectory `pictures` then you would get a “file not found” error message if you did

```
\includegraphics{shapes.ps} (incorrect)
```

instead of `\includegraphics{pictures/shapes.ps}` (correct)

- If the file is a [package](#) or [class file](#), it's possible that you don't have that file installed on your computer. If this is the case you will need to download and install it as described in [section 6.2](#). Remember

## 13. COMMON ERRORS

that you need to refresh the database after installing a new package or class file.

### 13.14 Illegal character in array arg

You have used a character in the [argument](#) of a tabular or array [environment](#) that is not allowed. The standard available characters are: `r` (right justified), `l` (left justified) and `c` (centred). This error will also occur if you have forgotten the argument to the `tabular` or `array` environment.

### 13.15 Illegal parameter number in definition

You have referred to a [parameter \(argument\) number](#) that is greater than the number of parameters you have specified. For example, suppose you defined the command to have only one parameter, then you can't use `#2` which refers to the second, non-existent, parameter. Remember that you need to specify how many parameters you want in the [optional argument](#) to `\newcommand`, otherwise it will be assumed that the command has no arguments.

See also the FAQ entry: [Illegal parameter number in definition](#).

## 13.16 Illegal unit of measure (pt inserted).

You have either not specified a unit when giving a length (even zero lengths must have a unit) or you have specified an invalid unit or you have misspelt the unit. Available units are listed in [table 12.1](#).

## 13.17 Lonely `\item`

The command `\item` may only appear in one of the list making environments (such as `itemize`). Make sure you haven't forgotten your environment.

## 13.18 Misplaced alignment tab character `&`

You have used the special character `&` where you shouldn't have. Recall from [section 4.2](#) that if you want an `&` sign to appear you need to do `\&` not just `&`.

You would have got this error message if you had done, say,

```
& our equipment (wrong)
```

instead of

### 13. COMMON ERRORS

`\& our equipment (correct)`

#### 13.19 Missing } inserted

You have missed a closing curly brace, or you may have missed out an argument.

Example: if the following line occurs in a tabular environment:

```
& \multicolumn{2}{c}\
```

this will produce the error. (The third argument to `\multicolumn` has been omitted.)

#### 13.20 Missing \$ inserted

This message can be caused by a number of errors:

- You may have typed `$` instead of `\$` (you actually want a dollar symbol to appear). Recall from [section 4.2](#) that if you want a \$ sign to appear you need to do `\$` not just `$`.

You would have got this error message if you had done, say,

```
expenditure came to $2000.00 (wrong)
```

### 13. COMMON ERRORS

instead of

expenditure came to  $\$2000.00$  (**correct**)

- You might have missed the beginning of one of the mathematics [environments](#) (that is, you've used a [command](#) that must only appear in maths mode).
- You may have missed the end of a mathematics environment, or you may have a paragraph break within a `math`, `displaymath` or `equation` environment, which is not permitted. Make sure you don't have any blank lines within the environment. If you want a blank line in your code to make it easier to edit, try having a percent sign at the start of an empty line to ensure that the line is ignored by L<sup>A</sup>T<sub>E</sub>X. For example:

```
\begin{equation}
%
E = mc^2
%
\end{equation}
```

See also the FAQ entry [“Missing \\$ inserted”](#).

## 13.21 Missing `\begin{document}`

You have put some text outside of the document `environment`. Check the following: [Missing `\begin{document}`]

- You have remembered `\begin{document}`

This error would be caused by, say,

```
\documentclass[a4paper]{article}
This is a simple document
```

(incorrect)

instead of

```
\documentclass[a4paper]{article}
\begin{document}
This is a simple document
```

(correct)

- You haven't placed any text before `\begin{document}`. For example:

```
\documentclass[a4paper]{article}
This is a simple document
\begin{document}
```

(incorrect)

instead of

### 13. COMMON ERRORS

```
\documentclass[a4paper]{article}
\begin{document}                                     (correct)
This is a simple document
```

- You haven't missed out a backslash from either `\documentclass` or `\begin{document}`

This error would be caused by, say,

```
documentclass[a4paper]{article} (incorrect)
```

instead of

```
\documentclass[a4paper]{article} (incorrect)
```

## 13.22 Missing delimiter

You have forgotten to specify the type of delimiter you want (e.g. ( ) [ ] \{ \} | \l . ) (Remember to use a . if you want an invisible delimiter, and remember that if you want a curly brace, you must have a backslash followed by the curly brace.)

Example:

### 13. COMMON ERRORS

```
f(x) = \left{
\begin{array}{l}
0 & \& x \leq 0 \\
1 & \& x > 1
\end{array}
\right. (incorrect)
```

instead of

```
f(x) = \left\{
\begin{array}{l}
0 & \& x \leq 0 \\
1 & \& x > 1
\end{array}
\right. (correct)
```

#### 13.23 Missing `\endcsname` inserted

This is a  $\TeX$  error rather than a  $\LaTeX$  error which makes it harder to determine the cause, however it can be caused by placing a backslash in front of the name of an [environment](#). (Remember that [environment](#) names do not contain a backslash.)

This error will be caused by, say,



### 13. COMMON ERRORS

`\begin{\sffamily}` (**incorrect**)

instead of

`\begin{sffamily}` (**correct**)

## 13.24 Missing `\endgroup` inserted

A number of things could have caused this. You may have missed out the end of an [environment](#), or you may have an environment inside of another environment it's not allowed to be in. For example, this error can be caused by placing an `eqnarray` environment inside a `displaymath` environment, which is not allowed.

## 13.25 Missing number, treated as zero

$\LaTeX$  is expecting a number. If your command takes more than one [argument](#), check to make sure the arguments are in the correct order. For example, if you are using a `minipage` [environment](#), you might have omitted the [mandatory argument](#) which specifies the width of the `minipage`, or you may have the [optional arguments](#) the wrong way round. The placement

### 13. COMMON ERRORS

specifier should come first, followed by the height.

If you are using `\addtocounter` or `\setcounter` remember that the second argument must be a number, so if you want the value of a counter as the argument you must use `\value`. This error can be caused by, say,

```
\setcounter{exercise}{chapter} (incorrect)
```

instead of

```
\setcounter{exercise}{\value{chapter}} (correct)
```

See also the FAQ entry “[Missing number, treated as zero](#)”.

## 13.26 Paragraph ended before `\begin` was complete

You’ve probably missed a closing brace at the end of the argument to `\begin`. This error will be caused by, say,

```
\begin{document} (incorrect)
```

instead of

`\begin{document}` (correct)

## 13.27 Runaway argument

There are a number of possible causes of this error:

- Paragraph breaks are not permitted in the [arguments](#) of [short commands](#). If there is a corresponding [environment](#) then you should use that instead. For example, this error message will be generated by doing, say,

```
\textbf{This is a simple document.  
Here is the first paragraph.
```

(incorrect)

```
Here is the second paragraph.}
```

instead of

### 13. COMMON ERRORS

```
\begin{bfseries}
```

This is a simple document.

Here is the first paragraph.

(correct)

Here is the second paragraph.

```
\end{bfseries}
```

- The closing brace of a [mandatory argument](#) is missing: This error will be caused by, say,

```
\title{A Simple Document (incorrect)}
```

instead of

```
\title{A Simple Document} (correct)
```

- This error can also be caused by omitting the [mandatory argument](#) of an [environment](#). For example:

```
\begin{thebibliography}
```

```
\bibitem{kopka95} A Guide to \LaTeX2e: document
```

(incorrect)

instead of

### 13. COMMON ERRORS

```
\begin{thebibliography}{1}
\bibitem{kopka95} A Guide to \LaTeX2e: document
```

 (correct)

## 13.28 Something's wrong—perhaps a missing `\item`.

You may have missed an `\item` command. The first object in a list environment must either be an `\item` command, or another list environment.

[Perhaps a missing `\item`?]

This error will be caused by, say,

```
\begin{itemize}
Animal
\item Vegetable
\item Mineral
\end{itemize}
```

 (incorrect)

instead of

```
\begin{itemize}
\item Animal
\item Vegetable
\item Mineral
\end{itemize}
```

 (correct)

### 13. COMMON ERRORS

This error can also be caused by a missing `\bibitem` in the [bibliography](#).  
For example:

```
\begin{thebibliography}{1}                                     (incorrect)
A Guide to \LaTeX2e: document
```

instead of

```
\begin{thebibliography}{1}                                     (correct)
\bibitem{kopka95} A Guide to \LaTeX2e: document
  See also the FAQ entry Perhaps a missing \item?.
```

#### 13.29 There's no line here to end.

You have placed a line breaking command (`\`, `\newline` or `\linebreak`) where it doesn't make sense to have one.

#### 13.30 Undefined control sequence

$\LaTeX$  doesn't understand the [command](#) you have used.

- Check to see if you have misspelt the command name. (Remember that all  $\LaTeX$  command names are case-sensitive.)

### 13. COMMON ERRORS

You will get this error if you do, say,

```
This is a simple \Latex\ document (incorrect)
```

instead of

```
This is a simple \LaTeX\ document (correct)
```

- Check that you have remembered the space when typing `\`. For example:

```
This is a simple \LaTeX\document (incorrect)
```

instead of

```
This is a simple \LaTeX\ document (correct)
```

- If you are using a command that is defined in a [package](#) make sure you have included the package using `\usepackage`.
- Check that your command name hasn't run into the next piece of text. For example, you can do

### 13. COMMON ERRORS

`man{\oe}uvre`

or

`man\oe uvre`

or

`man\oe{ }uvre`

but not

`man\oeuvre`

- You have used a backslash instead of a forward slash as a directory divider. (Remember that when using L<sup>A</sup>T<sub>E</sub>X under Windows, you need to use a forward slash (/) as the directory divider, as a backslash would be interpreted as a command. )

For example, suppose you have a file called `shapes.ps` in a subdirectory called `pictures`, then you would get an error if you did

`\includegraphics{pictures\shapes.ps}` (**Incorrect**)



## 13. COMMON ERRORS

instead of

```
\includegraphics{pictures/shapes.ps} (Correct)
```

### 13.31 You can't use 'macro parameter character #' in horizontal mode

You have used the special character # where you shouldn't have. Recall from [section 4.2](#) that if you want a # sign to appear you need to do \# not just #.

This error message will be caused by doing, say,

Item #1 (Incorrect)

instead of

Item \#1 (Correct)

# Chapter 14

## Need More Help?

Try some of the following:

- <http://www.tex.ac.uk/faq>
- [comp.text.tex newsgroup](#) (check the previous link before posting a query).
- [texhax archives](#)
- <http://www.ctan.org/>
- <http://www.tug.org/>
- If you live in the U.K., considering joining the UK T<sub>E</sub>X User Group, and take advantage of the circulating library and book discount scheme. See <http://uk.tug.org/> for more details.

# Bibliography

- [1] The T<sub>E</sub>X catalogue topic index. <http://www.tex.ac.uk/tex-archive/help/Catalogue/bytopic.html>.
- [2] The UK list of frequently asked questions. <http://www.tex.ac.uk/faq>.
- [3] Leslie Lamport. *L<sup>A</sup>T<sub>E</sub>X : a document preparation system*. Addison-Wesley, 2nd edition (updated for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>) edition, 1994.
- [4] Helmut Kopka and Patrick W. Daly. *A guide to L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>: document preparation for beginners and advanced users*. Addison-Wesley, 1995.
- [5] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The L<sup>A</sup>T<sub>E</sub>X companion*. Addison-Wesley, 1994.
- [6] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach. *The L<sup>A</sup>T<sub>E</sub>X graphics companion*. Addison-Wesley, 1997.
- [7] Michel Goossens, Sebastian Rahtz, et al. *The L<sup>A</sup>T<sub>E</sub>X web companion*. Addison-Wesley, 1999.

## BIBLIOGRAPHY

- [8] The comprehensive T<sub>E</sub>X archive network (CTAN). <http://www.ctan.org/>.
- [9] The UK T<sub>E</sub>X archive. <http://www.tex.ac.uk/>.
- [10] The `comp.text.tex` news group. <http://groups.google.com/group/comp.text.tex>.
- [11] The `texhax` archives. <http://tug.org/pipermail/texhax/>.
- [12] Donald E. Knuth. *The T<sub>E</sub>Xbook*. Addison-Wesley, 1986.
- [13] Herbert Voß. Math mode. <ftp://cam.ctan.org/tex-archive/info/math/voss/mathmode/Mathmode.pdf>.

I strongly recommend that you have a look at the UK T<sub>E</sub>X Archive [9], particularly UK TUG FAQ [2] and the [On-Line Catalogue](#). It's also a good idea to look at the [documentation](#) that was installed with your T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X distribution. If you are using MiKTeX you can access the on-line help via the Start Menu:

Start → Programs → MiKTeX → Help

# History

**15 Jan 2008** Version 1.3 released. The main reason behind this change was to increase accessibility and conform to W3C guidelines. If you are experiencing problems relating to accessibility, please let me know (clearly stating the problem).

- Corrected error in the university's post code on the title page
- Added alternative text tags to more of the images, and made some of the images hyperlinks to a more detailed description of the image.
- Added information on how to break ligatures.
- Moved information on TeX to the introduction, and removed section on TeX that was in the “Some Definitions” chapter.
- Document nodes now have permanent names instead of the generic node<n>.html which L<sup>A</sup>T<sub>E</sub>X2HTML generates by default.
- Went back to using straight double quotes in the HTML document as the fancy typographic double quotes are nonstandard.

**8 May 2007** Version 1.2 released.

## HISTORY

- Links to UK TUG FAQ [2] added.
- Overview made into a separate section, and tidied up a bit.
- Added some extra definitions: moving arguments and fragile commands, robust commands, short and long commands.
- Changed “Text editor and Terminal approach” to deal with Unix-type systems rather than MS-DOS.
- Moved section on `tabular` environment.
- Added section on boxes and mini-pages.
- Segmented section on font changing commands.
- Segmented section describing `graphicx`.
- Added section on the `babel` package.
- Updated and segmented section on downloading and installing new packages.
- Added section on side-by-side figures.
- Updated section on sub-figures to use the new `subfloat` package instead of the obsolete `subfigure` package.
- Added “Need More Help?” chapter.

**6 September 2004** Version 1.1 released.

# GNU Free Documentation License

Version 1.2, November 2002

Copyright © 2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

## Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It comple-

ments 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.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a worldwide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “**Document**”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “**you**”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “**Modified Version**” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.



A “**Secondary Section**” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “**Invariant Sections**” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “**Cover Texts**” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “**Transparent**” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly

with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “**Opaque**”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “**Title Page**” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which

do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “**Entitled XYZ**” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “**Acknowledgements**”, “**Dedications**”, “**Endorsements**”, or “**History**”.) To “**Preserve the Title**” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions

whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

### **3. COPYING IN QUANTITY**

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the

actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## **4. MODIFICATIONS**

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modifica-

tion 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.

## **5. COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as

Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and 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”.

## **6. COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## **10. FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

### **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 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with ... Texts.” line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# Index

## Symbols

		%	13, 84
i	18, 86	\&	86, 340
!'	18, 86	&	84, 119, 121, 336, 340
'	86	\'	91
''	86	\(	244
(	285	\)	244
)	285	\,	305
-	86	\.	91
--	86	\:	305
---	84, 86	\;	305
.	284	\=	91
\/	87, 108	\\	22–24, 119, 121, 336, 351
/	285	\^	91
¿	84, 86	^	84, 255, 256, 260
?'	84, 86	\-	86
\"	91	-	84, 255, 256, 260, 261
\#	86, 354	\'	91
#	84, 224, 312, 354	\	84
\\$	86, 341	\	285
\$	84, 245, 335, 341		285
\%	17, 86	[	22, 285

## Symbols

A N  
B O  
C P  
D Q  
E R  
F S  
G T  
H U  
I V  
J W  
K X  
L Y  
M Z

# INDEX

]	22, 285	amsmath package	251–253
\{	86, 285	amsopn package	264
{	19	\and	137
\}	86, 285	\appendix	144
}	19	\appendixname	241
\~	91	\approx	275
~	84, 154, 204	\arabic	320
‘	86	\arccos	260
“	86	\arcsin	260
<b>A</b>		\arctan	260
\AA	89	\arg	260
\aa	89	argument	20
abstract	114, 140, 173	mandatory.....	20, 75
abstract environment	140	optional.....	22, 76
\abstractname	241	array environment	298, 306
acread	49	\ast	276
\addtocounter	318, 347	\asympt	275
Adobe Reader	35, 67	\atop	279
\AE	89	\author	136
\ae	89	auxiliary file (.aux)	158
\Alph	321	<b>B</b>	
\alph	321	\b	91
\alpha	254	babel package	186, 189, 191
\amalg	276	\backslash	285
amfonts package	252, 253, 303	\begin	29, 290, 313, 347

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z



# INDEX

<code>\beta</code>		254	<code>\cdots</code>	281
<code>\bfseries</code>	19, 20, 28,	112	center environment	211, 216
<code>\bibitem</code>	162, 166,	351	<code>\centering</code>	203, 208, 215, 216
<code>\bibname</code>		241	<code>\chapter</code>	16, 20, 143, 143, 144, 145,
<code>\bigcap</code>		278		148, 186, 316
<code>\bigcirc</code>		276	<code>\chaptername</code>	241
<code>\bigcup</code>		278	<code>\chi</code>	254
<code>\bigodot</code>		278	<code>\circ</code>	276
<code>\bigoplus</code>		278	<code>\cite</code>	164, 165
<code>\bigotimes</code>		278	class file options	
<code>\bigsqcup</code>		278	10pt .....	76
<code>\bigtriangledown</code>		276	11pt .....	76
<code>\bigtriangleup</code>		276	12pt .....	76
<code>\biguplus</code>		278	a4paper .....	13, 76
<code>\bigvee</code>		278	oneside.....	173
<code>\bigwedge</code>		278	twocolumn.....	76
<code>\bmod</code>		264	twoside.....	173
<code>\boldsymbol</code>	253, 303		class files (.cls)	4, 13,
<code>\bowtie</code>		275		31, 75, 77, 114, 118, 140,
<code>\bullet</code>		276		143, 144, 173, 184
<b>C</b>			a0poster .....	118
<code>\c</code>		91	article . 13, 32, 75, 76, 143, 145,	
<code>\cap</code>		276		173
<code>\caption</code>	200, 203, 206,	208	book .....	32, 75, 140
<code>\cdot</code>		276	letter.....	32, 75, 140
			report ....	32, 75, 140, 143, 145,

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

	171, 173, 321	<code>\currenttime</code>	190
slides.....	75	<b>D</b>	
<code>\clearpage</code>	171	<code>\d</code>	91
color package	227	<code>\dag</code>	86, 237
colortbl package	127	<code>\dagger</code>	276
command	16, 20, 22, 28, 31	<code>\dashv</code>	275
long.....	<i>see</i> long command	<code>\date</code>	136, 137
short.....	<i>see</i> short command	datetime package	189–191
<code>\cong</code>	275	12hr.....	190
<code>\contentsname</code>	241	24hr.....	190
<code>\coprod</code>	278	nodayofweek.....	189
<code>\copyright</code>	86	short.....	189
<code>\cos</code>	260	<code>\ddag</code>	86
<code>\cosh</code>	260	<code>\ddagger</code>	276
<code>\cot</code>	260	<code>\ddots</code>	281
<code>\coth</code>	260	declaration	28, 121
counters	316	<code>\DeclareGraphicsExtensions</code>	179
chapter.....	316, 321, 322	<code>\DeclareMathOperator</code>	264
equation.....	316	<code>\DeclareMathOperator*</code>	264
figure.....	316	<code>\def</code>	27
footnote.....	316	<code>\deg</code>	260
page.....	316	<code>\Delta</code>	255
table.....	316	<code>\delta</code>	254
<code>\csc</code>	260	description environment	102
CTAN	10, 282	<code>\det</code>	260, 261
<code>\cup</code>	276		

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

# INDEX

<code>\diamond</code>	276	<code>\end</code>	29, 290, 313
<code>\dim</code>	260	enumerate environment	98
dinglist environment	240	environment	28, 31, 110, 116
directory divider	179	<code>\epsilon</code>	254
displayed maths	243	eqnarray environment	346
displaymath environment	246, 335, 342, 346	equation environment	246, 316, 335, 342
<code>\div</code>	276	<code>\equiv</code>	275
document environment	76	<code>\eta</code>	254
<code>\documentclass</code>	18, 31, 32, 75, 75, 76, 176	<code>\exp</code>	260
<code>\doteq</code>	275	<b>F</b>	
<code>\Downarrow</code>	277, 285	<code>ffi</code>	89
<code>\downarrow</code>	277, 285	<code>ffl</code>	89
DVI file ( <code>.dvi</code> )	15, 33, 35, 48, 49, 69, 80, 185, 231	<code>fi</code>	89
<code>dvips</code>	49, 63, 185, 231	figure environment	201, 206, 316
<b>E</b>		<code>\figurename</code>	241
ellipses (omission marks)	280	<code>fl</code>	89
<code>\em</code>	112, 114	floats	200
em environment	114	<code>\fnsymbol</code>	321
Emacs	36	<code>\footnote</code>	26, 82, 135, 316
<code>\emph</code>	111, 114	<code>\footnotesize</code>	117
Encapsulated PostScript (EPS) file	176, 177, 185, 202	<code>\forall</code>	280, 306
		<code>\foreignlanguage</code>	188
		<code>\frac</code>	267
		fragile command	26, 245

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

# INDEX

<code>\framebox</code>	24	<code>\hom</code>	260
<code>\frown</code>	275	<code>\hookleftarrow</code>	277
<b>G</b>		<code>\hookrightarrow</code>	277
<code>\Gamma</code>	16, 255	horizontal box	129, 207
<code>\gamma</code>	16, 254	<code>\hspace</code>	211
<code>\gcd</code>	260, 261	<code>\Huge</code>	117
<code>\ge</code>	275	<code>\huge</code>	117
geometry package	327	hyperref package	10
<code>\geq</code>	275	<b>I</b>	
<code>\gets</code>	277	<code>\i</code>	86, 89
<code>\gg</code>	275	<code>\in</code>	275
ghostscript	34, 35, 52	in-line maths	243
ghostview	35, 52	<code>\includegraphics</code>	177, 202
glue	128	<code>\index</code>	227
graphicx package	176, 181, 184, 185, 202	<code>\indexname</code>	241
draft.....	184	<code>\inf</code>	260, 261
final.....	184	<code>\infty</code>	261
hiderotate .....	184	input	7
hidescale.....	184	<code>\int</code>	278
grouping	19, 21, 28	<code>\iota</code>	254
GSview	34, 35, 67, 69, 185	italic correction	109
<b>H</b>		<code>\item</code>	30, 92, 102, 162, 340, 350
<code>\H</code>	91	itemize environment	93, 237, 340
		<code>\itshape</code>	112, 114
		itshape environment	114, 116, 309

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

## INDEX

<b>J</b>		<code>\LaTeX</code>	16, 81, 84	Symbols A N B O C P D Q E R F S G T H U I V J W K X L Y M Z
<code>\j</code>	86, 89	<code>\lceil</code>	285	
<b>K</b>		<code>\ldots</code>	86, 281	
<code>\kappa</code>	254	<code>\le</code>	275	
<code>kdvi</code>	15, 49	<code>\left</code>	284, 284, 336	
<code>\ker</code>	260	<code>\Leftarrow</code>	277	
<code>kghostview</code>	52	<code>\leftarrow</code>	277	
<code>kpdf</code>	35, 49	<code>\leftharpoondown</code>	277	
		<code>\leftharpoonup</code>	277	
		<code>\Leftrightarrow</code>	277	
<b>L</b>		<code>\leftrightharrow</code>	277	
<code>\L</code>	89	<code>lengths</code>	323	
<code>\l</code>	89	<code>\leq</code>	275	
<code>\label</code>	153, 154, 158, 162, 203, 211, 248, 318	<code>\lfloor</code>	285	
<code>\labelitemi</code>	237	<code>\lg</code>	260	
<code>\labelitemii</code>	237	<code>\lim</code>	260, 261, 264	
<code>\labelitemiii</code>	237	<code>\liminf</code>	260, 261	
<code>\labelitemiv</code>	237	<code>\limsup</code>	260, 261	
<code>\Lambda</code>	255	<code>\linebreak</code>	351	
<code>\lambda</code>	254	<code>\linewidth</code>	134	
<code>\langle</code>	285	<code>\listfigurename</code>	241	
<code>\LARGE</code>	117	<code>\listoffigures</code>	204	
<code>\Large</code>	117	<code>\listoftables</code>	217	
<code>Large environment</code>	116	<code>\listtablename</code>	241	
<code>\large</code>	117	<code>\ll</code>	275	
		<code>\ln</code>	260	

## INDEX

<code>\log</code>	260	<code>\mathit</code>	250
long command	27	<code>\mathrm</code>	250
<code>\Longleftarrow</code>	277	<code>\mathsf</code>	250
<code>\longleftarrow</code>	277	<code>\mathtt</code>	250
<code>\Longleftrightarrow</code>	277	<code>\max</code>	260, 261
<code>\longleftrightarrow</code>	277	<code>\mbox</code>	250
<code>\longmapsto</code>	277	<code>\mdseries</code>	112
<code>\Longrightarrow</code>	277	<code>\medspace</code>	305
<code>\longrightarrow</code>	277	<code>\mid</code>	275
<b>M</b>		MiKTeX	15, 34, 192, 197
MacGSview	35	MiKTeX Options	197
makeidx package	227	<code>\min</code>	260, 261, 264
<code>\makeindex</code>	227	minipage environment	135, 207, 346
makeindex	231, 232	mktxlsrc	197
<code>\maketitle</code>	137, 140, 148	<code>\models</code>	275
mandatory argument	<i>see</i> argument,	moving argument	26, 143, 200
mandatory		<code>\mp</code>	276
<code>\mapsto</code>	277	<code>\mu</code>	254
<code>\markboth</code>	173	<code>\multicolumn</code>	122, 123–125, 336, 341
<code>\markright</code>	173	multirow package	127
math environment	244, 342	<b>N</b>	
<code>\mathbb</code>	253	<code>\nearrow</code>	277
<code>\mathbf</code>	250, 252, 302	NEdit	36
<code>\mathcal</code>	250, 265	<code>\negmedspace</code>	305
<code>\mathfrak</code>	253	<code>\negthickspace</code>	305

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

# INDEX

`\negthinspace` 305  
`\neq` 275  
`\newcommand` 221, 222, 224, 229, 232, 237, 312  
`\newcounter` 317, 321  
`\newenvironment` 308  
`\newline` 351  
`\ni` 275  
`\noindent` 311  
`\normalfont` 112  
`\normalsize` 117  
`\not` 275  
`\notin` 275  
`\nu` 254  
`\nwarrow` 277  
  
**O**  
`\O` 89  
`\o` 89  
`\odot` 276  
`\OE` 89  
`\oe` 88, 89  
`\oint` 278  
`\Omega` 255  
`\omega` 254  
`\ominus` 276

`\oplus` 276  
 optional argument *see* argument, optional  
`\oslash` 276  
`\otimes` 276  
 output 7  
 output file *see* DVI file (`.dvi`)  
  
**P**  
`\P` 86  
 packages (`.sty`) 175  
 page numbering  
   Alph.....170  
   alph.....170  
   arabic.....170  
   Roman.....170  
   roman.....170  
 page style  
   empty.....173  
   headings.....173, 174  
   myheadings.....173  
   plain.....173  
`\pagenumbering` 170  
`\pageref` 155, 158  
`\pagestyle` 172, 173  
`\pagewidth` 323

## Symbols

A N  
 B O  
 C P  
 D Q  
 E R  
 F S  
 G T  
 H U  
 I V  
 J W  
 K X  
 L Y  
 M Z

## INDEX

<code>\par</code>	27, 310	PostScript	35, 63, 69, 180, 185, 231, 239, 324
<code>\paragraph</code>	143	<code>\pounds</code>	86, 124
paragraph break	79	<code>\Pr</code>	260, 261
paragraph indentation	79	preamble	31, 189
<code>\parallel</code>	275	<code>\prec</code>	275
parameter	<i>see</i> argument	<code>\preceq</code>	275
<code>\parbox</code>	134	<code>\printindex</code>	227
<code>\parindent</code>	323	<code>\prod</code>	278
<code>\parskip</code>	323, 326	<code>\propto</code>	275
<code>\part</code>	143	<code>\protect</code>	26, 143, 200
<code>\partial</code>	271	proTeXt	34, 36, 54
<code>\partname</code>	241	<code>\Psi</code>	255
PDF	35, 202	<code>\psi</code>	254
PDFL <sup>A</sup> T <sub>E</sub> X	10, 15, 35, 44, 49, 63, 74, 176, 178, 179, 185, 196	<b>Q</b>	
pdftops	176	<code>\qqquad</code>	304, 305
<code>\perp</code>	275	<code>\quad</code>	305
<code>\Phi</code>	255	<b>R</b>	
<code>\phi</code>	254	<code>\r</code>	91
<code>\Pi</code>	255	<code>\rangle</code>	285
<code>\pi</code>	254	<code>\rceil</code>	285
pifont package	239, 240	<code>\ref</code>	154, 158, 203, 211, 248, 318
<code>\pm</code>	276	<code>\reflectbox</code>	183
<code>\pmb</code>	253	<code>\refname</code>	241
<code>\pmod</code>	264		
pnmtops	176		

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z



# INDEX

<code>\refstepcounter</code>	318, 319	<code>\section</code>	143, 144, 145, 148
<code>\renewcommand</code>	237, 240, 320	<code>\selectlanguage</code>	187
<code>\resizebox</code>	183	<code>\setcounter</code>	318, 347
<code>\rfloor</code>	285	<code>\setlength</code>	323
<code>\rho</code>	254	<code>\setminus</code>	276
<code>\right</code>	284, 284, 333, 336	<code>\sffamily</code>	112
<code>\Rightarrow</code>	277	short command	27, 228, 234, 237
<code>\rightarrow</code>	16, 277	<code>\Sigma</code>	255
<code>\rightharpoondown</code>	277	<code>\sigma</code>	254
<code>\rightharpoonup</code>	277	<code>\sim</code>	275
<code>\rightleftharpoons</code>	277	<code>\simeq</code>	275
<code>\rmfamily</code>	112	<code>\sin</code>	260
robust command	27, 245	<code>\sinh</code>	260
<code>\Roman</code>	320	<code>\slash</code>	86
<code>\roman</code>	321	<code>\slshape</code>	112
<code>\rotatebox</code>	181	<code>\small</code>	117
rubber length	325	<code>\smile</code>	275
<b>S</b>		source code	15, 33, 75, 77, 79, 80
<code>\S</code>	86	<code>\sp</code>	255, 256
<code>\sb</code>	255, 256	spaces	79
<code>\scalebox</code>	182	<code>\sqcap</code>	276
<code>\scriptsize</code>	117	<code>\sqcup</code>	276
<code>\scshape</code>	112	<code>\sqrt</code>	272
<code>\searrow</code>	277	<code>\sqsubsetq</code>	275
<code>\sec</code>	260	<code>\sqsupsetq</code>	275
		<code>\SS</code>	89

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

# INDEX

<code>\ss</code>	89	tabular environment	118, 127, 129,
<code>\star</code>	276		130, 132, 332
<code>\stepcounter</code>	318, 319	<code>\tan</code>	260
subfig package	210, 214	<code>\tanh</code>	260
subfigure package	210	<code>\tau</code>	254
<code>\subfloat</code>	210, 211	TeX	34, 192, 197
<code>\subparagraph</code>	143	texhash	197
<code>\subref</code>	211	TeXnicCenter	34, 36, 52, 63, 79, 90,
<code>\subsection</code>	143		185, 231, 232
<code>\subset</code>	275	<code>\text</code>	251
<code>\subsetq</code>	275	<code>\textasciicircum</code>	86
<code>\subsubsection</code>	143	<code>\textasciitilde</code>	86
<code>\succ</code>	275	<code>\textbackslash</code>	86
<code>\succeq</code>	275	<code>\textbar</code>	86
<code>\sum</code>	278	<code>\textbf</code>	20, 111, 250, 309
<code>\sup</code>	260, 261	<code>\textemdash</code>	86
<code>\supset</code>	275	<code>\textendash</code>	86
<code>\supseteq</code>	275	<code>\textexclamdown</code>	86
<code>\swarrow</code>	277	<code>\textgreater</code>	86
<b>T</b>		<code>\textheight</code>	323
<code>\t</code>	91	<code>\textit</code>	111, 114
table environment	214, 316	<code>\textless</code>	86
table of contents file (.toc)	149	<code>\textmd</code>	111
<code>\tablename</code>	241	<code>\textnormal</code>	111
<code>\tableofcontents</code>	148, 149, 186, 204	<code>\textperiodcentered</code>	86
		<code>\textquestiondown</code>	86

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

# INDEX

<code>\textquotedblleft</code>	86	<code>\tiny</code>	117
<code>\textquotedblright</code>	86	<code>title</code>	173
<code>\textquoteleft</code>	86	<code>\title</code>	136
<code>\textquoteright</code>	86	<code>\to</code>	261, 277
<code>\textregistered</code>	86	<code>\today</code>	16, 81, 84, 189, 190, 219
<code>\textrm</code>	111, 250	<code>\triangleleft</code>	276
<code>\textsc</code>	111	<code>\triangleright</code>	276
<code>\textsf</code>	111	<code>\ttfamily</code>	112
<code>\textsl</code>	111	<code>\twocolumn</code>	16
<code>\texttrademark</code>	86	<b>U</b>	
<code>\texttt</code>	111	<code>\u</code>	91
<code>\textup</code>	111	<code>ukdate package</code>	189
<code>\textwidth</code>	327	<code>unbreakable space</code>	154
<code>\thanks</code>	138	<code>\Uparrow</code>	277, 285
<code>\the</code>	316, 323	<code>\uparrow</code>	277, 285
<code>thebibliography environment</code>	166	<code>\Updownarrow</code>	277, 285
<code>\thechapter</code>	316, 322	<code>\updownarrow</code>	277, 285
<code>\thepage</code>	316	<code>\uplus</code>	276
<code>\thesection</code>	316	<code>\upshape</code>	112
<code>\Theta</code>	255	<code>\Upsilon</code>	255
<code>\theta</code>	254, 254	<code>\upsilon</code>	254
<code>\thickspace</code>	305	<code>\usepackage</code>	18, 175, 176, 190, 333, 352
<code>\thinspace</code>	305		
<code>\thispagestyle</code>	172		
<code>tiff2ps</code>	176		
<code>\times</code>	276		

## Symbols

A	N
B	O
C	P
D	Q
E	R
F	S
G	T
H	U
I	V
J	W
K	X
L	Y
M	Z

# INDEX

<b>V</b>		<b>Y</b>		<b>Symbols</b> A N B O C P D Q E R F S G T H U I V J W K X L Y M Z
<code>\v</code>	91	YAP	15	
<code>\value</code>	318, 347	<code>\yen</code>	86	
<code>\varepsilon</code>	254			
<code>\varphi</code>	254	<b>Z</b>		
<code>\varpi</code>	254	<code>\zeta</code>	254	
<code>\varrho</code>	254			
<code>\varsigma</code>	254			
<code>\vartheta</code>	254, 254			
<code>\vdash</code>	275			
<code>\vdots</code>	281			
<code>\vec</code>	301, 302, 306			
<code>\vee</code>	276			
<code>vim</code>	36–38			
<code>\vspace</code>	216, 314			
<b>W</b>				
<code>\wedge</code>	276			
<code>WinEdt</code>	67, 69, 74, 185, 231, 232			
<code>\wr</code>	276			
<b>X</b>				
<code>xdvi</code>	15, 49			
<code>\Xi</code>	255			
<code>\xi</code>	254			
<code>xpdf</code>	35, 49			