\LaTeX{} for Complete Novices

Nicola Talbot

Monday 27\textsuperscript{th} September, 2004
## Contents

1 **Introduction** .......................................................... 1  
  1.1 Recommended Reading  .............................................. 2

2 **Some Definitions** .................................................. 4  
  2.1 Source Code  ........................................................... 5  
  2.2 DVI File (or Output File)  ......................................... 5  
  2.3 Commands (also called “Macros” or “Control Sequences”) ........ 5  
  2.4 Grouping  .............................................................. 5  
  2.5 Arguments (also called “Parameters”)  ............................ 6  
    2.5.1 Mandatory Arguments  .......................................... 6  
    2.5.2 Optional Arguments  ........................................... 7  
  2.6 Declarations  .......................................................... 8  
  2.7 Environments  ....................................................... 9  
  2.8 Preamble  .............................................................. 9  
  2.9 Class File  ........................................................... 10  
  2.10 \TeX  .................................................................. 10

3 **From Source Code to Typeset Output** ................................ 11  
  3.1 Notepad, MS-DOS Prompt, YAP .................................... 12  
  3.2 TeXnicCenter  ......................................................... 19  
  3.3 WinEdt  ................................................................. 28

4 **Creating a Simple Document** ........................................ 31  
  4.1 Using Simple Commands  ............................................. 33  
  4.2 Special Characters and Symbols  ................................... 34  
  4.3 Lists  ................................................................... 37  
    4.3.1 Unordered Lists  ............................................... 37  
    4.3.2 Ordered Lists  ................................................. 39  
    4.3.3 Description Environment  ..................................... 41  
  4.4 Simple font changing commands  .................................. 43

5 **Creating Chapters, Sections etc** .................................... 45  
  5.1 Author and title information  ....................................... 45  
  5.2 Abstract  .................................................................. 46  
  5.3 Sections, Subsections  ................................................ 47  
  5.4 Creating a Table of Contents  ....................................... 49  
  5.5 Cross-Referencing  .................................................... 50  
  5.6 Creating a Bibliography  ............................................. 54  
  5.7 Page Styles and Page Numbering  .................................. 57  
  5.8 Aligning Material in Rows and Columns  ......................... 59
CONTENTS

6 Packages 64
6.1 Using Packages ........................................ 64
   6.1.1 graphicx Package ............................... 64
   6.1.2 Changing the format of \today .................. 68
6.2 Downloading and Installing Packages ................. 68

7 Figures and Tables 71
7.1 Figures .............................................. 71
   7.1.1 Subfigures .................................... 73
7.2 Tables ............................................... 74

8 Defining Commands 76
8.1 Defining Commands with an Optional Argument ....... 81
8.2 Redefining Commands ................................. 82

9 Mathematics 85
9.1 In-Line Mathematics ................................ 85
9.2 Displayed Mathematics ................................. 86
9.3 Mathematical Commands ............................... 87
   9.3.1 Maths Fonts .................................. 88
   9.3.2 Greek Letters ................................ 88
   9.3.3 Subscripts and Superscripts .................. 89
   9.3.4 Functional Names .............................. 92
   9.3.5 Fractions ..................................... 95
   9.3.6 Roots .......................................... 97
   9.3.7 Mathematical Symbols ......................... 99
   9.3.8 Delimiters .................................... 102
   9.3.9 Arrays ........................................ 108
   9.3.10 Vectors ....................................... 109
   9.3.11 Mathematical Spacing ......................... 110

10 Defining Environments 112

11 Counters 115

12 Lengths 118

13 Common Errors 120
   13.1 * (No message, just an asterisk prompt!) ....... 121
   13.2 Argument of \cline has an extra } .............. 121
   13.3 Argument of \multicolumn has an extra } ........ 121
   13.4 \begin{…} ended by \end{…} ....................... 121
   13.5 Bad math environment delimiter .................. 122
   13.6 Can only be used in preamble ..................... 122
   13.7 Command … already defined ....................... 122
   13.8 Display math should end with $$ .................. 122
   13.9 Environment … undefined ......................... 122
   13.10 Extra alignment tab has been changed to \cr .... 123
   13.11 Extra \right ...................................... 123
   13.12 File ended while scanning use of … .......... 123
   13.13 File not found ................................... 123
   13.14 Illegal character in array arg .................. 124
   13.15 Illegal parameter number in definition ........ 124
   13.16 Illegal unit of measure (pt inserted) ......... 124
   13.17 Lonely \item ...................................... 124
13.18 Misplaced alignment tab character & .......................... 125
13.19 Missing \} inserted ........................................... 125
13.20 Missing $ inserted ............................................. 125
13.21 Missing \begin{document} ...................................... 126
13.22 Missing delimiter .............................................. 126
13.23 Missing \endcsname inserted ................................. 127
13.24 Missing \endgroup inserted ................................. 127
13.25 Missing number, treated as zero ............................ 127
13.26 Paragraph ended before \begin was complete .............. 127
13.27 Runaway argument ............................................. 128
13.28 Something’s wrong—perhaps a missing \item .................. 128
13.29 There’s no line here to end. .................................. 129
13.30 Undefined control sequence ................................. 129
13.31 You can’t use ‘macro parameter character #’ in horizontal mode . 130

Bibliography .................................................. 131

Index ...................................................... 132
# List of Figures

3.1 Using notepad ......................................................... 13  
3.2 Saving your document .................................................. 14  
3.3 File saved correctly .................................................. 14  
3.4 File saved incorrectly ............................................... 15  
3.5 MS-DOS Prompt ....................................................... 15  
3.6 Changing Directory .................................................. 16  
3.7 Using \LaTeX .......................................................... 16  
3.8 \LaTeX output ......................................................... 17  
3.9 Three New Files Created by \LaTeX .................................. 17  
3.10 Loading output file into YAP ........................................ 18  
3.11 Viewing typeset document ......................................... 18  
3.12 Using PDF\LaTeX ..................................................... 19  
3.13 PDF\LaTeX Output .................................................. 20  
3.14 Viewing PDF file in Acrobat ....................................... 20  
3.15 TeXnicCenter Tip of the Day Window .............................. 21  
3.16 TeXnicCenter Configuration Wizard ................................. 22  
3.17 TeXnicCenter Configuration Wizard ................................. 22  
3.18 TeXnicCenter Configuration Wizard ................................. 23  
3.19 TeXnicCenter ........................................................ 23  
3.20 New Project Dialog Box ............................................. 24  
3.21 New Project Dialog Box ............................................. 24  
3.22 TeXnicCenter — New Project Started .............................. 25  
3.23 TeXnicCenter — Typing in Source Code ............................ 25  
3.24 TeXnicCenter — Selecting Output Type ............................ 26  
3.25 TeXnicCenter (using \LaTeX and dvips) ............................ 27  
3.26 TeXnicCenter — Showing Error .................................... 27  
3.27 WinEdit ............................................................. 28  
3.28 WinEdit ............................................................. 29  
3.29 WinEdit — Saving the File ....................................... 29  
3.30 WinEdit — \LaTeX Output ......................................... 30  
6.1 Updating the database .............................................. 70  
7.1 Some shapes ........................................................ 72  
7.2 Two Shapes: (a) A Rectangle and (b) A Circle ..................... 74
# List of Tables

4.1 Symbols ................................................................. 35
4.2 Ligatures and Special Symbols ........................................ 36
4.3 Accent Commands ..................................................... 36
4.4 Font changing commands ............................................ 43
4.5 Font changing declarations ........................................... 43
4.6 Font size changing declarations ..................................... 44

7.1 A Sample Table ....................................................... 74
7.2 A Sample Table ....................................................... 75

8.1 Object Names .......................................................... 84

9.1 Maths Font Changing Commands ..................................... 88
9.2 The \texttt{amsfonts} and \texttt{amsmath} Font Commands ....... 89
9.3 Lower Case Greek Letters ............................................ 90
9.4 Upper Case Greek Letters ............................................ 90
9.5 Function Names ....................................................... 92
9.6 Relational Symbols .................................................. 99
9.7 Binary Operator Symbols ........................................... 100
9.8 Arrow Symbols ....................................................... 100
9.9 Symbols with Limits ................................................ 100
9.10 Ellipses ............................................................... 101
9.11 Delimiters ............................................................ 103
9.12 Mathematical Spacing Commands .................................. 111

12.1 Units of Measurement ............................................... 118
List of Exercises

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Simple Document</td>
<td>32</td>
</tr>
<tr>
<td>2 Using Simple Commands</td>
<td>33</td>
</tr>
<tr>
<td>3 Using Special Characters</td>
<td>36</td>
</tr>
<tr>
<td>4 Lists</td>
<td>42</td>
</tr>
<tr>
<td>5 Fonts</td>
<td>44</td>
</tr>
<tr>
<td>6 Creating Title Pages</td>
<td>45</td>
</tr>
<tr>
<td>7 Creating an Abstract</td>
<td>46</td>
</tr>
<tr>
<td>8 Creating Chapters, Sections etc</td>
<td>48</td>
</tr>
<tr>
<td>9 Creating a Table of Contents</td>
<td>49</td>
</tr>
<tr>
<td>10 Cross-Referencing</td>
<td>53</td>
</tr>
<tr>
<td>11 Creating a Bibliography</td>
<td>57</td>
</tr>
<tr>
<td>12 Page Styles and Page Numbering</td>
<td>59</td>
</tr>
<tr>
<td>13 Aligning Material</td>
<td>62</td>
</tr>
<tr>
<td>14 Using the graphicx Package</td>
<td>67</td>
</tr>
<tr>
<td>15 Downloading and Installing a New Package</td>
<td>70</td>
</tr>
<tr>
<td>16 Creating Figures</td>
<td>72</td>
</tr>
<tr>
<td>17 Creating Sub-Figures</td>
<td>74</td>
</tr>
<tr>
<td>18 Creating Tables</td>
<td>75</td>
</tr>
<tr>
<td>19 Defining a New Command</td>
<td>80</td>
</tr>
<tr>
<td>20 Defining Commands with an Optional Argument</td>
<td>82</td>
</tr>
<tr>
<td>21 Renewing Commands</td>
<td>84</td>
</tr>
<tr>
<td>22 Maths: Fractions and Symbols</td>
<td>102</td>
</tr>
<tr>
<td>23 Maths: Vectors and Arrays</td>
<td>110</td>
</tr>
<tr>
<td>24 More Mathematics</td>
<td>110</td>
</tr>
<tr>
<td>25 Defining a New Environment</td>
<td>114</td>
</tr>
<tr>
<td>26 Using Counters</td>
<td>117</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The aim of this document is to introduce \LaTeX{} to a non-technical person. \LaTeX{} 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.

\LaTeX{} does take a while to learn, so why should I use it? Firstly, \LaTeX{} is far better at typesetting mathematical equations than word processors. Compare the following equations:

1. Using equation editor in Microsoft Word:\footnote{I was unable to find a caligraphic font for the $L$. The font looks a little ragged because I had to convert it to bitmap to include it into the document.}:

$$\frac{\partial^2 L}{\partial \tau_2^2} = \frac{\partial \rho_{i}}{\partial \tau_2^2} \left( \frac{\partial v_{i}}{\partial \rho_{i}} e^v + \frac{v_{i}}{\partial \rho_{i}} \frac{1 - e^v}{1 - e^{\rho_{i}}} \frac{1}{(1 - e^v)^2} \right)$$

2. Using \LaTeX{}:

$$\frac{\partial^2 L}{\partial \tau_1^2} = -\frac{\partial \rho_{i}}{\partial \tau_1^2} \left( \frac{\partial v_{i}}{\partial \rho_{i}} e^v + \frac{v_{i}}{\partial \rho_{i}} \frac{1 - e^v}{1 - e^{\rho_{i}}} \frac{1}{(1 - e^v)^2} \right)$$

(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 104 in Section 9.3.8.)

Secondly, \LaTeX{} makes it very easy to cross-reference chapters, sections, equations, figures, tables etc, and it also makes it very 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 all the section numbering etc. \LaTeX{} can also ensure consistent formatting, and the style of the document can be completely changed simply by using a different class file, or loading additional packages.

Thirdly, when you are editing a document using a word processor, the word processor has to work out how to reformat the document everytime 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 \LaTeX you type your code in using an ordinary text editor. The document doesn’t get formatted until you pass it to \LaTeX, which means that you are not slowed down by constant reformatting.

Lastly, there’s the fact that \LaTeX follows certain typographical rules, so you can leave most of the typesetting to \LaTeX. You rarely need to worry about minor things such as remembering to put two spaces between sentences and only one space between words, as \LaTeX will do this automatically, and it will also automatically deal with f-ligatures. That is, if any of the following combination of letters are found: fl, ff, fi, ffi, they will automatically be converted into the corresponding ligatures: fl, ff, fi, ffi. Note the difference between fluffier (2 ligatures) and fluffier (no ligatures). 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 examiners, referees etc 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.

To give you an idea of what you can do with \LaTeX, this document was written in \LaTeX. The PostScript version was generated using \LaTeX, makeindex and dvips, the PDF versions were generated using PDF\LaTeX and makeindex and the HTML version was generated using the \LaTeX2HTML\footnote{http://www.latex2html.org/} converter. All versions were generated from the same source code with occasional switches for minor variations between formats\footnote{plus a small Perl script to generate the file size information that appears at the start of the HTML version}.

This document is structured as follows: Chapter 2 defines terms that will be used throughout this document. If you like, you can give this chapter a cursory glance to begin with and go back to it later. Chapter 3 details the software that you will need to use \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, sections etc 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 \LaTeX distribution. Chapter 7 describes how to create figures and tables. Chapter 8 describes how to define your own commands. Chapter 9 describes how to typeset mathematics. Chapter 10 describes how to define new environments. Chapter 11 discusses counters. Chapter 12 discusses lengths, and Chapter 13 documents possible errors you may encounter, and gives advice on how to fix them.

This document and associated files are available on-line at: \url{http://theoval.cmp.uea.ac.uk/~nlct/latex/novices/}. This document is also available in $6 \times 4$in PDF format for on-line viewing and HTML format. If you are viewing this document in Acrobat Reader, you can click on the bookmarks tab to help navigate your way around the document.

1.1 Recommended Reading

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

"\LaTeX : a document preparation system" by Leslie Lamport\cite{Lamport1993} is the user’s guide and reference manual for \LaTeX, and is a good basic text for anyone starting out, however it doesn’t cover AMSTeX, so anyone who needs to typeset more than basic mathematics may prefer either “A guide to \LaTeX” by Helmut Kopka and Patrick Daly\cite{Kopka1994} or “The \LaTeX companion” by Michel Goossens, Frank Mittelbach and Alexander Samarin\cite{Goossens1994}. Both these books cover AMSTeX, Bi\LaTeX and
makeindex. “A guide to \TeX" 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 \TeX companion”, there is also “The \TeX graphics companion” by Michel Goossens, Sebastian Rahtz and Frank Mittelbach [4] which details how to illustrate documents with \TeX 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 \TeX before delving into it.

The final book in the “Companion” series is “The \TeX web companion” by Michel Goossens, Sebastian Rahtz et al. [5] which is recommended for those interested in creating documents for the web, either as HTML or PDF. This book details how to convert \TeX documents into HTML using various applications such as \LaTeX2HTML and \TeX4ht, and how to create PDF documents using PDFLaTeX, including how to create active links within your document using the hyperref package.

There is also a wealth of \TeX-related information on the world wide web. The Comprehensive \TeX Archive Network\(^4\) (CTAN) is a good place to start. In the UK, the \TeX Archive at \url{http://www.tex.ac.uk/} is closer. You can check the on-line catalogue for information about available software, and there is also a list of frequently asked questions which I recommend you try if you have any queries. You can also try using a search engine, such as Google, but take care not to simply search for “latex” or you will end up with thousands of hits, most of which will be totally irrelevant. Search engines are unable to tell the difference between \LaTeX (the typesetting language) and \latex (the plant substance or synthetic product), so I would recommend that you use the advance search facility to exclude certain obvious words related to the latter (and you might also want to consider selecting the “filter using SafeSearch” option.) It would also be a good idea to specify a few extra words to help narrow down your search. For example, if you want a general introduction to \TeX, you could type latex into the box marked “with all of the words” and type introduction beginners guide novices into the box marked “with at least one of the words”. Alternatively, if you have an error message you don’t understand, you could try typing part of the error message into the box marked “with the exact phrase”.

\(^4\)\url{http://www.ctan.org/}
Chapter 2

Some Definitions

As mentioned in the previous chapter, \LaTeX{} 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.

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:

\begin{verbatim}
This is an \textbf{example}.
\end{verbatim}

Resulting output:

\begin{verbatim}
This is an example.
\end{verbatim}

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

\begin{verbatim}
Line one \par
Line two \par
Line three.
\end{verbatim}

with corresponding output:

\begin{verbatim}
Line one
Line two
Line three.
\end{verbatim}

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

\begin{verbatim}
\documentclass[options]{class file}
\end{verbatim}
In this case the command being defined is called $\texttt{\textbackslash documentclass}$ and text typed like this (e.g. $\texttt{options}$ and $\texttt{class file}$) indicates the type of thing you need to substitute. For example, if you want the $\texttt{article}$ class file you would substitute $\texttt{class file}$ with $\texttt{article}$ and if you want the $\texttt{a4paper}$ option you would substitute $\texttt{options}$ with $\texttt{a4paper}$, like this:

$\texttt{\documentclass[a4paper]{article}}$

But more on that later.

2.1 Source Code

The source code is all the text and $\LaTeX$ 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.

2.2 DVI File (or Output File)

The $\LaTeX$ 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. $\texttt{MiKTeX}$ comes with the DVI viewer $\texttt{YAP}$. If you are using the X Window System (under UNIX or Linux etc), the DVI viewer is called $\texttt{xdvi}$.

2.3 Commands (also called “Macros” or “Control Sequences”)

A command usually begins with a backslash, (e.g. $\texttt{\today}$) and is used to tell $\LaTeX$ to do a particular thing at that point in the document. For example, $\texttt{\today}$ will print the current date, $\texttt{\twocolumn}$ will start a new page, and change to a two column format, $\texttt{\LaTeX}$ will print the \LaTeX\ logo: $\LaTeX$. Most $\LaTeX$ commands have fairly self-explanatory names. (For example, $\texttt{\rightarrow}$ prints an arrow pointing to the right, $\texttt{\chapter}$ starts a new chapter.) All commands are case-sensitive, so $\texttt{\gamma}$ and $\texttt{\Gamma}$ have different meanings.

There is one command that you must use in every document you create, and that is the $\texttt{\documentclass}$ 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

Segments 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, $\texttt{\bfseries}$ changes the font weight to bold, so the following segment of code:

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

will appear in the typeset document looking like:
Here is some text. This text is in a group. Here is some more text.

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 LaTeX 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 LaTeX the title of this new chapter. So the `\chapter` command takes one mandatory argument that specifies the title. For example, the following code:

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

   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.) The following code:

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

   will look like:

   **Some bold text.**

### Notes

1. LaTeX takes the first object following the command name as the argument, which is why the argument has to be grouped. Suppose the last example above didn’t have a group, so instead the code was:

   ```latex
   \textbf Some bold text.
   ```

   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:

   Some bold text.
2. If you want the argument to be blank, use empty braces: \{\}. For example, suppose you want to have a chapter without a title\footnote{The numbers for chapters, sections etc are automatically inserted by \LaTeX, so this example would produce a numbered chapter without a title.} you would need to do:

\chapter{}

\section*{2.5.2 Optional Arguments}

Some \texttt{commands} may have one or more optional arguments. Unlike \texttt{mandatory arguments}, optional arguments must always be enclosed in square brackets \[\]. For example, the command \verb|\| \ starts a new line. So the following segment of code:

\begin{verbatim}
\verb|Line one|\linebreak \verb|Line two.|
\end{verbatim}

will produce the following output:

\begin{verbatim}
Line one
Line two.
\end{verbatim}

However the \verb|\| 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:

\begin{verbatim}
\verb|Line one|\[1cm]\verb|Line two.|
\end{verbatim}

will produce the following output:

\begin{verbatim}
Line one
[1cm] Line two.
\end{verbatim}

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

Code:

\begin{verbatim}
\verb|Line one|\{[1cm]\} \verb|Line two.|
\end{verbatim}

Output:

\begin{verbatim}
Line one
[1cm] Line two.
\end{verbatim}
CHAPTER 2. SOME DEFINITIONS

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}

Output:

Some Text

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

Code:

\framebox[4cm]{Some Text}

Output:

Some Text

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

Code:

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

Output:

Some Text

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 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:

\bfseries Here is some bold text.

will appear in the typeset document looking like:

Here is some normal text. Here is some bold text.
2.7 Environments

An environment is a block of code contained within the commands `\begin{env-name}` and `\end{env-name}`, 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` environment will typeset the contents of the environment in a bold font. The following code:

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

will appear in the typeset document looking like:

Here is some bold text.

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:

\begin{itemize}
\item Cabbages
\item Bananas
\item Apples
\end{itemize}

will produce the following output:

Shopping List:

• Cabbages
• Bananas
• Apples

2.8 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.

\footnote{note there is no backslash in the environment name}
2.9 Class File

The class file (.cls) defines the page layout, heading styles and various commands and environments needed for a particular style of document. The class file is specified using the command

\documentclass[options]{class-name}

where class-name is the name of the file without the .cls extension. All \LaTeX\ documents must start with this command.

2.10 \TeX\n
\TeX\ is the typesetting language written by Donald Knuth. Plain \TeX\ is a bit complicated to use, unless you want to write a very basic document, so Leslie Lamport wrote a format of \TeX\ called \LaTeX\ to make it a bit easier to use. You can think of \LaTeX\ as a go-between converting your instructions into \TeX\. This document mostly uses the term \LaTeX, even if the matter is more general to \TeX, to avoid complicating matters. Some error messages you may see will be \LaTeX\ messages, some will be \TeX\ messages. \LaTeX\ error messages tend to be a bit easier to understand than \TeX\ messages.
Chapter 3

From Source Code to Typeset Output

Every time you want to create or edit a \TeX or \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 \TeX application (“\TeX 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.
2. The \TeX or \LaTeX installation (to perform Step 2). There are a number available, however a popular choice for Windows is MiKTeX, which is free and can be downloaded from the \TeX Archive [6] in the systems/win32/miktex directory and is easy to install. Simply follow the installation instructions. Default values are provided if you are unsure what option to choose, but if you have any difficulties, contact your system administrator. Note that even if you are using a front-end, you must first install MiKTeX (or some other \TeX installation). If you are using UNIX or Linux, a popular choice is teTeX, this can be downloaded from the systems/unix or systems/linux directories.
3. A DVI viewer (to perform Step 3). The \TeX or \LaTeX installation should come with a DVI viewer. (MiKTeX comes with YAP.) It is also possible to convert your DVI file into PostScript (.ps) or Acrobat (.pdf) format, in which case you will need GSView or Acrobat, respectively, to view the files. By converting your output to PostScript or PDF, you can enhance the functionality of \LaTeX allowing you to perform operations such as rotating text (See Section 6.1.1 for further details). If you use PDFLaTeX to generate a PDF document, you can also create active links (see The \LaTeX Web Companion [5]

\footnote{GSview can also display PDF files, but any links in the document will be inactive}
for more information, or if you’d rather a brief on-line introduction you can try [Creating a PDF Document using PDFLaTeX](http://theoval.cmp.uea.ac.uk/~nlct/latex/pdfdoc/).

Documented below are instructions of how to use \LaTeX on Windows using three different methods:

1. Using **notepad** as a text editor, the **MS-DOS Prompt** to access \LaTeX, and **YAP** to view the DVI file.

2. Using the front-end **TeXnicCenter** (free) to perform all three steps.

3. Using the front-end **WinEdt** (shareware) to perform all three steps.

Using **notepad** and the **MS-DOS Prompt** is fiddly and prone to human error, however it is useful to know, just in case there is a situation that the front-end can’t handle (e.g. you want to \LaTeX a file that doesn’t have a .tex extension, which may happen if you want to install new packages on your system — some front-ends allow you to do this, others may not). I would therefore strongly recommend that you use one of the front-ends (TeXnicCenter or WinEdt) rather than using notepad. TeXnicCenter and WinEdt are both easy to use, although TeXnicCenter has the advantage of being free.

If you are using UNIX or Linux, follow the instructions for using notepad and the MS-DOS prompt, but substitute notepad for your favourite text editor (e.g. vim or emacs), use a terminal instead of the MS-DOS Prompt, and use xdvi instead of YAP.

### 3.1 Notepad, MS-DOS Prompt, YAP

**Notepad** is a very basic text editor that comes with Windows. It is usually found through the Start menu:

Start → Programs → Accessories → Notepad

Once you have opened up **notepad**, you can start to type in your source code. (See Figure 3.1.)

Care needs to be taken when saving the document, as notepad automatically tries to add the extension .txt to any file you save, whereas all \LaTeX files must have the extension .tex. You can force notepad to do this by placing the filename in double quotes in the “Save as” dialogue box. (See Figure 3.2.)

You can check to see if your file has been saved correctly by looking at the directory viewer. If the file has been saved correctly, it should look something like Figure 3.3.

If the file has the incorrect .txt extension added to it, it will probably look something like Figure 3.4. In this case the icon indicates that this is an ordinary text file. If you look at the file’s properties you will see that its name is actually sample1.tex.txt, which is incorrect.

Step 1 is now complete. Time to move on to Step 2: passing the source code to \LaTeX. To do this you will need to run the MS-DOS Prompt. This is usually found in:

Start → Programs → MS-DOS Prompt

or

Start → Programs → Accessories → MS-DOS Prompt

The command prompt should look something like Figure 3.5.

The first thing you need to do is to change to the directory where you saved your file (otherwise \LaTeX won’t know where the file is.) You can do this using the
cd command. I saved my file in the directory My Documents\Nicky\samples on the C drive. Since the directory name has a space in it, it will need to be enclosed in double quotes. At the command prompt, I would then have to type

```
  cd "c:\My Documents\Nicky\samples"
```

as shown in Figure 3.6.

You can now pass the source code to \LaTeX. I called my file sample1.tex so I would need to type

```
  latex sample1.tex
```

at the command prompt, as shown in Figure 3.7. (You don’t have to specify the .tex extension, as \LaTeX will automatically assume your file has the correct extension).

If there are no errors in the document, you should see something like the output shown in Figure 3.8. (If you do get an error message, check the list of common errors in Chapter 13.)

The second to last line

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

indicates that the resulting typeset document has been saved as the file sample1.dvi and it is one page long. Numbers appearing in square brackets, e.g. [1], indicate which page \LaTeX is currently processing. In this case, there is only one page. The last line to appear on screen indicates that information about this \LaTeX run has been written to the log file sample1.log, which you can look at using notepad. You can see these new files by having a look at the directory viewer, as shown in Figure 3.9.

You can view the typeset output by loading the file sample1.dvi into YAP. You can do this either by double clicking on its icon, or by typing

```
  yap sample1.dvi
```

at the command prompt. (See Figure 3.10.)

You will then see the final output, as shown in Figure 3.11.
Figure 3.2: Saving your document

Figure 3.3: File saved correctly
Figure 3.4: File saved incorrectly

Figure 3.5: MS-DOS Prompt
Figure 3.6: Changing Directory

Figure 3.7: Using \LaTeX
Figure 3.8: \LaTeX{} output

Figure 3.9: Three New Files Created by \LaTeX{}
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

Figure 3.10: Loading output file into YAP

Figure 3.11: Viewing typeset document
Figure 3.12: Using PDFLaTeX.

If you like, you can then convert your DVI file into a PostScript file by using dvips. To do this, type the following in the MS-DOS Prompt window:

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

You can then view the PostScript file using GSView\(^3\) which can be downloaded from the \TeX\ Archive [6].

Alternatively, you can create a PDF document instead by using PDFLaTeX instead of \LaTeX:

```
pdflatex sample1.tex
```

as shown in Figure 3.12. (Again the extension may be omitted.)

The output is shown in Figure 3.13.

The new PDF file `sample1.pdf` can now be loaded into Acrobat\(^4\), as shown in Figure 3.14.

Each time you want to edit the document, you will have to go back to Step 1 (although you shouldn’t need to worry about changing directory anymore, unless you exit the MS-DOS Prompt.) This method can be rather cumbersome, however life is made a lot easier by using a front-end, such as WinEdit or TeXnicCenter.

### 3.2 TeXnicCenter

TeXnicCenter is an application that enables you to edit \LaTeX source code, and simply click on a button to pass the source code to \LaTeX, and then click on another button to view the resulting typeset document. This alleviates the problems encountered using notepad and the MS-DOS Prompt detailed in Section 3.1.

TeXnicCenter is free and can be downloaded from the \TeX Archive [6] in the `systems/win32/TexnicCenter/` directory. Note that you must have a \TeX/\LaTeX distribution installed before you install TeXnicCenter. Once you have downloaded

---

\(^3\) \texttt{ghostview} or \texttt{ggv} if you are using UNIX or Linux

\(^4\) You can either use \texttt{xpdf} or \texttt{acroread} if you are using UNIX or Linux
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

Figure 3.13: PDF\LaTeX\ Output.

Figure 3.14: Viewing PDF file in Acrobat.
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

the TeXnicCenter setup file\(^5\), run it by double clicking on its icon in the directory viewer. I recommend that you use the default settings. If you have any problems installing TeXnicCenter, contact your systems administrator.

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 3.15.)

![Tip of the day window](image)

Did you know...

Welcome to TeXnicCenter

We have some tips for you, about how to use LaTeX and especially TeXnicCenter.

Leave this dialog enabled for some time, until you think, you know all features of TeXnicCenter. If you don't want to see the dialog anymore, you can uncheck the box in the lower left corner. You can popup this dialog every time about the menu item "Tip of the day..."

Figure 3.15: TeXnicCenter Tip of the Day Window

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 would 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 3.19.

To start a new project select File → New Project. This will open the window shown in Figure 3.20.

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 3.21.)

Select the "Empty Project" icon, and click on “Okay”. You should now see something like Figure 3.22.

You can now start typing the source code (we’ll cover this later). See Figure 3.23.

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 3.24. 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 3.25) This tran-

\(^5\)currently called TXCSetup_iBeta6_21.exe
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

Figure 3.16: TeXnicCenter Configuration Wizard

Figure 3.17: TeXnicCenter Configuration Wizard
Figure 3.18: TeXnicCenter Configuration Wizard

Figure 3.19: TeXnicCenter
Figure 3.20: New Project Dialog Box

Figure 3.21: New Project Dialog Box
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

Figure 3.22: TeXnicCenter — New Project Started

Figure 3.23: TeXnicCenter — Typing in Source Code
script should be the same as described on page 13 onwards. If you have selected LaTeX => PDF, then TeXnicCenter will use PDFLaTeX instead of B\TeXX. If you have selected LaTeX => PS, then TeXnicCenter will use B\TeXX followed by dvips (as in Figure 3.25). The dvips messages will follow on from the \LaTeXX messages. (If you selected the Bib\TeX or MakeIndex features when you initialised the project, Figure 3.21, then TeXnicCenter will also use the Bib\TeX and MakeIndex applications.)

To view the document, click the View Output button \act{View Output} (Note that if you have selected LaTeX => PDF or LaTeX => PS you will need Adobe Acrobat or GSView, respectively, to view the output file.)

If there are any errors, you can select Build \rightarrow Next Error and it will show you where the error has occurred (See Figure 3.26). If you do have any errors, check Chapter 13.
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

Figure 3.25: TeXnicCenter (using \LaTeX\ and dvips)

Figure 3.26: TeXnicCenter — Showing Error
3.3 WinEdt

WinEdt (not to be confused with WinEdit which is a completely different application) is an application that enables you to edit \LaTeX\ source code, and simply click on a button to pass the source code to \LaTeX, and then click on another button to view the resulting typeset document. This alleviates the problems encountered using notepad and the MS-DOS Prompt detailed in Section 3.1.

WinEdt is shareware: it can be downloaded from the \TeX\ Archive [6] in the systems/win32/winedt directory 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 \TeX/\LaTeX\ 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 WinEdt, contact your system administrator.

To run WinEdt, select WinEdt from the start menu:

Start → Programs → WinEdt → WinEdt

It should look like Figure 3.27

![Figure 3.27: WinEdt](image)

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 3.28.

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 3.29.

To \LaTeX\ your document, simply click on the \LaTeX\ button . The output will appear in an MSDOS Prompt window (see Figure 3.30).

To view your typeset document, click on the “view DVI” button .

You can convert your DVI file to PostScript by clicking the button. If
CHAPTER 3. FROM SOURCE CODE TO TYPESET OUTPUT

Figure 3.28: WinEdt

Figure 3.29: WinEdt — Saving the File
you have GSView installed, you can then view the PostScript file by clicking on the button.

Depending on which version of WinEdt you have installed, there may also be a PDFLaTeX button which you can click on to create a Portable Document Format (.pdf) document. If not, you can click on the button to open up an MS-DOS Prompt window⁶, and you can use the commands listed on page 19.

⁶WinEdt should automatically set the correct directory, so you shouldn’t need to worry about changing directory
Chapter 4

Creating a Simple Document

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}

This tells \LaTeX what type of document you want to create (e.g. an article, a technical report, correspondence). The \texttt{\documentclass} command takes one mandatory argument \texttt{class-name} that specifies the class file. There are a great many available, but the basic ones are: \texttt{article} (short documents without chapters), \texttt{report} (longer technical documents containing chapters), \texttt{book} (for writing books), \texttt{letter} (for writing correspondence) and \texttt{slides} (for creating slides for use with OHP or data projectors).

We’ll be starting with a very simple document, so let’s use the \texttt{article} class file. In this case the very first line of the source code should be:

\documentclass{article}

The \texttt{\documentclass} command also takes an optional argument \texttt{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 \texttt{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 \texttt{a4paper}. So you would need to edit the above line to:

\documentclass[a4paper]{article}

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

\documentclass[a4paper,11pt]{article}

You can also change your document so that it is in a two column format using the \texttt{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 in a \texttt{document} environment. The document is started with the command:

\begin{document}

and ended with

\end{document}

So our source code now looks like:
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:

```
\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}
```

Exercise 1 (Simple Document)

Try typing the above code into your editor (see Chapter 3 if you can’t remember what to do.) You can also download a copy of this file, but I would 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 21. 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.

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 (or just click on the save icon if you are using TeXnicCenter) and then pass it to \LaTeX (either by typing latex sample1.tex in the MS-DOS Prompt, or by clicking on the \LaTeX icon in WinEdt, or by clicking on the build icon in TeXnicCenter as detailed in Chapter 3.) If all goes well, you should see something that looks like the following displayed 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 yap sample1.dvi in the MS-DOS Prompt, 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 \LaTeX and go back to your source code and fix the problem\(^1\). If you do have an error, consult the list of common mistakes in 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 \LaTeX and the command \today prints the current date. \LaTeX always ignores any spaces that follow a command name, 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. 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)

\(^1\)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 26.
CHAPTER 4. CREATING A SIMPLE DOCUMENT

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

Now \LaTeX\ your document and view the result. (Remember to check the list of common errors in Chapter 13 if you have a problem.) You should see the \LaTeX\ 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 \LaTeX\ 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 \LaTeX. 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 \LaTeX\ 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. (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 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.)

Ligatures and special symbols are shown in Table 4.2. (Note that, as mentioned in the Introduction, the f-ligatures are automatically converted.) 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 “manoeuvre” has an œ-ligature in the middle of it. There are several ways to code this in \LaTeX:
1. Group the command:

\texttt{\textbackslash man\oeuvre}

2. Place a space after the command:

\texttt{\textbackslash man\oe uvre}

3. Place an empty brace after the command:

\texttt{\textbackslash man\oe{}uvre}

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; 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 right when you’re editing your code. The third example, like the first example, maintains the word’s cohesion, but it gives the incorrect 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:

\texttt{man\oeuvre}

This is incorrect, as \LaTeX will interpret it as the command \oeuvre which doesn’t exist.

Accented letters are created by specifying which accent you want, and what letter to put the accent on. The accent commands are listed in Table 4.3, and each command takes one mandatory argument. The command indicates what accent to use, the argument indicates what letter to put the accent on. You may have noticed in Table 4.1 the commands \i and \j which produce a dotless i and j (i and j). You should use these instead of i and j as the argument to an accent command, since i and j should loose their dot when they have an accent over them. Example:
CHAPTER 4. CREATING A SIMPLE DOCUMENT

Table 4.2: Ligatures and Special Symbols

<table>
<thead>
<tr>
<th>AE</th>
<th>Â</th>
<th>ae</th>
<th>æ</th>
<th>Æ</th>
<th>Æ</th>
<th>Æ</th>
<th>Æ</th>
</tr>
</thead>
<tbody>
<tr>
<td>fi</td>
<td>fi</td>
<td>ffi</td>
<td>ffi</td>
<td>fl</td>
<td>fl</td>
<td>ffl</td>
<td>ffl</td>
</tr>
<tr>
<td>ÂA</td>
<td>Â</td>
<td>aa</td>
<td>å</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Ø</td>
<td>Ø</td>
<td>o</td>
<td>ø</td>
<td>SS</td>
<td>SS</td>
<td>SS</td>
<td>β</td>
</tr>
</tbody>
</table>

Table 4.3: Accent Commands

<table>
<thead>
<tr>
<th>Definition</th>
<th>Example</th>
<th>Input</th>
<th>Output</th>
<th>Definition</th>
<th>Example</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>'{object}</td>
<td>'{c}</td>
<td>é</td>
<td></td>
<td>={object}</td>
<td>={c}</td>
<td>é</td>
<td></td>
</tr>
<tr>
<td>'{object}</td>
<td>'{c}</td>
<td>c</td>
<td></td>
<td>={object}</td>
<td>={c}</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>&quot;{object}</td>
<td>&quot;{c}</td>
<td>c</td>
<td></td>
<td>\v{object}</td>
<td>\v{c}</td>
<td>ˇc</td>
<td></td>
</tr>
<tr>
<td>{object}</td>
<td>{c}</td>
<td>c</td>
<td></td>
<td>\H{object}</td>
<td>\H{c}</td>
<td>ˇc</td>
<td></td>
</tr>
<tr>
<td>\t{object}</td>
<td>\t{cc}</td>
<td>c</td>
<td></td>
<td>\c{object}</td>
<td>\c{c}</td>
<td>ˇc</td>
<td></td>
</tr>
<tr>
<td>\d{object}</td>
<td>\d{c}</td>
<td>c</td>
<td></td>
<td>\b{object}</td>
<td>\b{c}</td>
<td>ˇc</td>
<td></td>
</tr>
</tbody>
</table>

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

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:

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

You can download or view the source code if you can’t work out how to do it, and remember to check the list of common errors in 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: \texttt{itemize} (for unordered lists), \texttt{enumerate} (for ordered lists) and \texttt{description} (for lists where you want to specify your own label.)

In each of these environments, there is a command
\begin{verbatim}
\item[\textit{label}]
\end{verbatim}

which you need to use to specify each item of the list.

4.3.1 Unordered Lists

Unordered lists are created using the \texttt{itemize} environment. For example, the following code:
\begin{verbatim}
\begin{itemize}
  \item Animal
  \item Vegetable
  \item Mineral
\end{itemize}
\end{verbatim}

will produce the following output:

\begin{verbatim}
• Animal
• Vegetable
• Mineral
\end{verbatim}

It is also possible to nest \texttt{itemize} environments. For example, the following code:
\begin{verbatim}
\begin{itemize}
  \item Animal
  \begin{itemize}
    \item Mammals
    \item Birds
  \end{itemize}
  \item Reptiles. For example:
  \begin{itemize}
    \item dinosaurs
    \item crocodiles
  \end{itemize}
  \item Vegetable
  \begin{itemize}
    \item Cultivated
  \end{itemize}
\end{itemize}
\end{verbatim}
\item Wild
\end{itemize}
\item Mineral
\end{itemize}

will produce the following output:

\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}

That looks good, but our code is a bit cramped and a little difficult to read. Blank lines between list items are ignored by \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:

\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}
CHAPTER 4. CREATING A SIMPLE DOCUMENT

\item Vegetable
\begin{itemize}
  \item Cultivated
  \item Wild
\end{itemize}
\item Mineral
\end{itemize}

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 \texttt{enumerate} environment. It has exactly the same format as the \texttt{itemize} environment described in the previous section.

We can use the same example as before, only this time use \texttt{enumerate} instead of \texttt{itemize}.

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

The above input will produce the following output:

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

Again, the environments can be nested:

\begin{enumerate}
  \item Animal
\begin{enumerate}
\end{enumerate}
\end{enumerate}
The above input will produce the following output:

1. Animal
   (a) Mammals
   (b) Birds
   (c) Reptiles. For example:
       i. dinosaurs
       ii. crocodiles

2. Vegetable
   (a) Cultivated
   (b) Wild

3. Mineral
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 label as an optional argument to the `\item` command. For example, the following code:

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

will produce the following output:

- **Animal** Living being
- **Vegetable** Plant
- **Mineral** Natural inorganic substance

It is possible to nest all the listing environments:

\[
\begin{description}
\item[Animal] Living being
\begin{itemize}
\item Mammals
\item Birds
\item Reptiles. For example:
\begin{enumerate}
\item dinosaurs
\item crocodiles
\end{enumerate}
\end{itemize}
\item[Vegetable] Plant
\begin{itemize}
\item Cultivated
\end{itemize}
\end{description}
\]
\item Wild
\end{itemize}
\item[Mineral] Natural inorganic substance
\end{description}
\begin{itemize}
\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
  \begin{itemize}
  \item Cultivated
  \item Wild
  \end{itemize}
\item[Mineral] Natural inorganic substance
\end{itemize}

Exercise 4 (Lists)

Try writing the source code that will create the following output:

\begin{itemize}
\item[Village] A small collection of dwelling places. Examples:
  \begin{enumerate}
  \item Marlingford
  \item Saxlingham
  \end{enumerate}
\item[Town] A large collection of dwelling places. Examples:
  \begin{enumerate}
  \item Great Yarmouth
  \item Beccles
  \end{enumerate}
\item[City] A large town, usually containing a cathedral. Examples:
  \begin{enumerate}
  \item Norwich
  \item Birmingham
  \item London
  \end{enumerate}
\end{itemize}

You can download or view the answer if you can’t work out how to do it.
4.4 Simple font changing commands

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, whereas those shown in Table 4.5 are of the second type — a declaration.

Table 4.4: Font changing commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Example Input</th>
<th>Corresponding output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textrm{text}</td>
<td>\textrm{roman} text</td>
<td>roman text</td>
</tr>
<tr>
<td>\textsf{text}</td>
<td>\textsf{sans serif} text</td>
<td>sans serif text</td>
</tr>
<tr>
<td>\texttt{text}</td>
<td>\texttt{typewriter} text</td>
<td>typewriter text</td>
</tr>
<tr>
<td>\textmd{text}</td>
<td>\textmd{medium} text</td>
<td>medium text</td>
</tr>
<tr>
<td>\textbf{text}</td>
<td>\textbf{bold} text</td>
<td>bold text</td>
</tr>
<tr>
<td>\textit{text}</td>
<td>\textit{italic} text</td>
<td>italic text</td>
</tr>
<tr>
<td>\textsl{text}</td>
<td>\textsl{slanted} text</td>
<td>slanted text</td>
</tr>
<tr>
<td>\textsc{text}</td>
<td>\textsc{Small Caps} text</td>
<td>Small Caps text</td>
</tr>
<tr>
<td>\textup{text}</td>
<td>\textup{upright} text</td>
<td>upright text</td>
</tr>
<tr>
<td>\textit{text}</td>
<td>\textit{italic} text</td>
<td>italic text</td>
</tr>
<tr>
<td>\textsl{text}</td>
<td>\textsl{slanted} text</td>
<td>slanted text</td>
</tr>
<tr>
<td>\textsc{text}</td>
<td>\textsc{Small Caps} text</td>
<td>Small Caps text</td>
</tr>
<tr>
<td>\emph{text}</td>
<td>\emph{emphasized} text</td>
<td>emphasized text</td>
</tr>
<tr>
<td>\textnormal{text}</td>
<td>\textnormal{default} text</td>
<td>default text</td>
</tr>
</tbody>
</table>

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 31), all the font sizes will be changed relative to the new size.

Environments can be used instead. Each environment has the same name as its corresponding declaration, but without the preceding backslash. Example:

\begin{itshape} Some italic text. \end{itshape}
\begin{Large} This text is large. \end{Large}
Table 4.6: Font size changing declarations

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Example Input</th>
<th>Corresponding output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tiny</td>
<td>\tiny tiny text</td>
<td>tiny text</td>
</tr>
<tr>
<td>\scriptsize</td>
<td>\scriptsize script sized text</td>
<td>script sized text</td>
</tr>
<tr>
<td>\footnotesize</td>
<td>\footnotesize footnote sized text</td>
<td>footnote sized text</td>
</tr>
<tr>
<td>\small</td>
<td>\small small text</td>
<td>small text</td>
</tr>
<tr>
<td>\normalsize</td>
<td>\normalsize normal sized text</td>
<td>normal sized text</td>
</tr>
<tr>
<td>\large</td>
<td>\large large text</td>
<td>large text</td>
</tr>
<tr>
<td>\LARGE</td>
<td>\LARGE even larger</td>
<td>even larger</td>
</tr>
<tr>
<td>\HUGE</td>
<td>\HUGE larger still</td>
<td>larger still</td>
</tr>
<tr>
<td>\huge</td>
<td>\huge huge</td>
<td>huge</td>
</tr>
<tr>
<td>\Huge</td>
<td>\Huge really huge</td>
<td>really huge</td>
</tr>
</tbody>
</table>

\end{Large}\end{itshape} Back to normal.

\begin{Large}\end{Large}\begin{itshape}\end{itshape} Back to normal.

Output:

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

Note that the command \textit, 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 italic and upright. So if the surrounding font is upright then \textit, \em and \em will use the italic font, but if the surrounding font is italic, \textit, \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 \textit etc rather than \textit etc.

For more information on using fonts, including using fonts not covered in this document, see A Guide to \LaTeX{} [2] or The \LaTeX{} Companion [3].

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.)
Chapter 5

Creating Chapters, Sections etc

Let’s go back to the document we created in Exercise 2. In this chapter we shall modify 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}

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:

\maketitle

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{}

Exercise 6 (Creating Title Pages)

Try editing the document you created in Exercise 2 to include title information. Modifications are illustrated like this:

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

\end{document}

You can download this document.

\section*{5.2 Abstract}

The \texttt{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 \texttt{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 \texttt{book} and \texttt{letter}) don’t have an \texttt{abstract} environment. Abstracts traditionally go at the start of the document after the title, so the \texttt{abstract} environment should go after the \texttt{maketitle} command.

\textbf{Exercise 7 (Creating an Abstract)}

Try editing your document so that it has an abstract: Modifications are illustrated like this:

\begin{verbatim}
\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}
\end{document}
\end{verbatim}

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. With a footnote.

This document was created on: \today.

\end{document}

You can download this document.

5.3 Sections, Subsections ...

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

\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>}

Note that 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 \textit{title} and an optional argument \textit{short title}. The mandatory argument \textit{title} is simply the title of the chapter/section/subsection etc. For example:

\section{Introduction}

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

1 Introduction

Note that you don’t specify the section number as \LaTeX 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.

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, if the current section is the 4th section of chapter 5, the section number will be 5.4 (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).

Unnumbered chapters/sections etc are produced by placing an asterisk \textastars after the command name. For example:
\chapter*{Acknowledgements}

You can switch to appendices using the command
\appendix
then continue using \chapter, \section etc. For example (using the report class file):

\appendix
\chapter{Derivations}
Some derivations.
\chapter{Tables}
Some tables.

Exercise 8 (Creating Chapters, Sections etc)

Let’s try editing our document so that it now has chapters, sections etc. 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.

\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.
\chapter{Another Chapter}
Here’s another very interesting chapter.
We’re going to put a picture here later.
I would like to acknowledge all those very helpful people who have assisted me in my work.

We’re going to put some tables here later.

\end{document}

\documentclass[a4paper,11pt]{report}
\begin{document}
\title{A Simple Document}
\author{Me}
\maketitle
\tableofcontents
\end{document}
\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}

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’re going to put some tables here later.

\end{document}

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

\section{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.

The first part, labelling the place you want to reference, is done using the command:
The **argument** *string* should be a unique textual label. This label can be anything you like as long as it is unique, but it is recommended that it isn’t too long or you may use up too much memory. 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:

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

Another example:

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

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:

```latex
\ref{string}
```

Example:

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

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 \LaTeX to break the line at that point. This is done using the ~ special character, so the example above should actually be:

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

The `\pageref{string}` command will insert the page number that the label appeared on. Example:

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

The label `ch:def` obviously needs to be defined somewhere:
\chapter{Definitions}
\label{ch:def}

In fact, I have done this in my source code for this document, so the above example would look like:

See Chapter 2 on page 4 for a list of definitions.

It’s not just chapters and sections that you can reference; most of the numbers that \LaTeX{} automatically generates can be cross-referenced. The \texttt{enumerate} environment described in Section \ref{sec:enumerate} automatically numbers the items within an ordered list, so it’s possible to label list items. For example:

\begin{enumerate}
\item\label{itm:edit} Write or edit source code.
\item Pass source code to the \LaTeX{} application ("\LaTeX{} the document").
\begin{itemize}
\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}

Output:

1. Write or edit source code.
2. Pass source code to the \LaTeX{} application ("\LaTeX{} the document").
   \begin{itemize}
   \item If there are any error messages, return to Step 1.
   \item If there are no error messages, a DVI file is created, go to Step 3.
   \end{itemize}
3. View DVI file to check the result.
\documentclass[a4paper,11pt]{report}
\begin{document}
\title{A Simple Document}
\author{Me}
\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}.
\end{document}
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 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’re going to put some tables here later.
\end{document}

\section*{5.6 Creating a Bibliography}

Bibliographies can be created using the \texttt{thebibliography} environment. This environment is very similar to the list making environments described in Section 4.3, but instead of \texttt{\item} use

\begin{verbatim}
\bibitem[label]{key}
\end{verbatim}

where \texttt{key} is a unique keyword that identifies this item. Your keyword can be anything you like, but as with \texttt{\label} I would recommend that you use a short memorable keyword. I tend to use the first author’s surname followed by the year of publication. Example:

\begin{verbatim}
\begin{thebibliography}{1}
\bibitem[lamport94] {``\LaTeX\ : a document preparation system''}, Leslie Lamport, 2nd edition (updated for \LaTeX2e), Addison–Wesley (1994).
\end{thebibliography}
\end{verbatim}
\bibitem{kopka95} ‘‘A Guide to \LaTeXe: 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}

You can cite an item in your bibliography with the command

\cite[text]{key list}

Example:

For more information about writing bibliographies see Goossens \emph{et al.} \cite{goossens94}.

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

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

The \textit{optional argument} \texttt{text} to the \texttt{cite} command can be used to add text to the citation. Example:
For more information about writing bibliographies see Goossens \emph{et al.}~\cite[Chapter~13]{goossens94}.

Output:

For more information about writing bibliographies see Goossens \textit{et al.} [3, Chapter 13].

The \texttt{thebibliography} environment has a \textbf{mandatory argument}:

\begin{verbatim}
\begin{thebibliography}{widest entry}
\end{verbatim}

The argument \texttt{widest entry} is the widest label in the list of entries. This helps \LaTeX{} 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 \texttt{\bibitem} command. In the above example, the labels were all the same width so the argument 1 was used (although 2 and 3 could just have easily been used). Consider the following example:

\begin{verbatim}
\begin{thebibliography}{Goossens 1994}
\bibitem[Lamport 1994]{lamport94} ''\LaTeX{}: a document preparation system'', Leslie Lamport, 2nd edition (updated for \LaTeX{}2e), Addison-Wesley (1994).
\end{thebibliography}
\end{verbatim}

Output:

\textbf{References}


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

\begin{thebibliography}{Goossens 1994}

There is an application called \texttt{Bib\TeX} that can be used in conjunction with \texttt{B\LaTeX} to help generate bibliographies. This document does not cover \texttt{Bib\TeX}, but if you are interested I would recommend reading \textit{A Guide to B\LaTeX} [2] or \textit{The B\LaTeX Companion} [3]. For those of you who want a quick look on-line, the document \textit{Using B\LaTeX to Write a PhD Thesis}\footnote{http://theoval.cmp.uea.ac.uk/~nlct/latex/thesis/thesis.html} has a section containing a brief introduction to \texttt{Bib\TeX}.

**Exercise 11 (Creating a Bibliography)**

Try added the following chapter to your document:

\chapter{Recommended Reading}

For a basic introduction to \texttt{\LaTeX} see Lamport\cite{lamport94}. For more detailed information about \texttt{\LaTeX} and associated applications, consult Kopka and Daly\cite{kopka95} or Goossens \texttt{et al}\cite{goossens94}.

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 \texttt{\LaTeX} 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:

\begin{verbatim}
\pagenumbering{style}
\end{verbatim}

where \texttt{style} can be one of:

- \texttt{arabic} Arabic page numbers (1, 2, 3, \ldots)
- \texttt{roman} Lowercase Roman numerals (i, ii, iii, \ldots)
- \texttt{Roman} Uppercase Roman numerals (I, II, III, \ldots)
- \texttt{alph} Lower case alphabetical characters (a, b, c, \ldots)
- \texttt{Alph} Upper case alphabetical characters (A, B, C, \ldots)
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):

```latex
\documentclass{report}
\usepackage{titlesec}
\titleformat*{\chapter}{\large\bfseries}
\author{Me}
\title{A Simple Document}
\maketitle
\tableofcontents
\begin{abstract}
This is the abstract.
\end{abstract}
\chapter{Introduction}
```

Note that if you don’t have an abstract environment, you will need to do \clearpage before doing \chapter{Introduction}:

```latex
\documentclass{report}
\usepackage{titlesec}
\titleformat*{\chapter}{\large\bfseries}
\author{Me}
\title{A Simple Document}
\maketitle
\tableofcontents
\clearpage
\chapter{Introduction}
```

The headers and footers can be changed using the command \pagestyle{style}.

Individual pages can be changed using \thispagestyle{style}.

Standard styles are:

- **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:\n
\markboth{left head}{right head}
5.8 Aligning Material in Rows and Columns

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

\begin{tabular}[placement specifier]{column specifiers}

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 (‘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:

\begin{tabular}{lr}

(Input)

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

\end{tabular}
CHAPTER 5. CREATING CHAPTERS, SECTIONS ETC

Output:

<table>
<thead>
<tr>
<th>Item</th>
<th>ex VAT</th>
<th>inc VAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>8.99</td>
<td>10.56</td>
</tr>
<tr>
<td>CD</td>
<td>9.99</td>
<td>11.74</td>
</tr>
<tr>
<td>DVD</td>
<td>15.00</td>
<td>17.63</td>
</tr>
<tr>
<td>Total</td>
<td>33.98</td>
<td>39.93</td>
</tr>
</tbody>
</table>

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

\begin{tabular}{lrr}
Item & ex VAT & inc VAT\[
Video & 8.99 & 10.56\[
CD & 9.99 & 11.74\[
DVD & 15.00 & 17.63\[
\textbf{Total} & 33.98 & 39.93
\end{tabular}
The command
\multicolumn{cols\ spanned}{col\ specifier}{text}

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

\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
\textbf{Total} & 33.98 & 39.93
\end{tabular}

In this example we are spanning two columns, so the first argument to \texttt{\multicolumn} is \{2\}, we want the entry centred, so the second argument is \{c\} and the text to go in this entry is simply \{Price (\pounds)\}.

The \texttt{\multicolumn} command can also be used to override the alignment of individual entries. Consider the following example:

\begin{tabular}{lrr}
\item Year1 & Year2
Travel & 100,000 & 110,000
\end{tabular}

\texttt{\begin{tabular}{lrr}
\item & Year1 & Year2
\item Travel & 100,000 & 110,000
\end{tabular}}

\url{http://theoval.cmp.uea.ac.uk/~nlct/latex/pdfdoc/}
Exercise 13 (Aligning Material)

You may have noticed that the document you have been creating throughout this chapter has an appendix entitled “Tables”. The tabular environment does not create a table, but later on in Section 7.2 we’ll see how to turn it into one. For now, try placing the following tabular environment into the appendix of your document:

\begin{tabular}{ll} 
\hline 
\textbf{Expenditure} & \\
\textbf{Year1} & \textbf{Year2} \\
Travel & 100,000 & 110,000 \\
Equipment & 50,000 & 60,000 \\
\hline 
\end{tabular}

You can download or view the result.

For more information about using the tabular environment, including how to add vertical and horizontal lines, see the \LaTeX{} user’s guide [1], A Guide to \LaTeX{} [2].
or *The \LaTeX Companion* [3]. The latter reference also describes how to span rows using the \texttt{multirow} package. For information on how to create coloured tables using the \texttt{colortbl} package, see *The \LaTeX Graphics Companion* [4].
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.

6.1 Using Packages

\LaTeX 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:

\usepackage[options]{package name}

where package name is the name of the package without the .sty extension, and options 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 couple of examples.

6.1.1 graphicx Package

It is possible to generate images using \LaTeX commands (See \textit{The \LaTeX Graphics Companion} \cite{4}) however most people find it easier to create a picture in some other application, and include that file into their \LaTeX document.

Some applications have an option that allows you to save an image as a PostScript file\footnote{If it doesn’t have this option you can use a PostScript printer driver and print to file. You can also convert other file types to PostScript using applications such as: pdftops, tiff2ps, pnmtofts}. The \texttt{graphicx} package provides a command that enables you to include this PostScript file into your document\footnote{You can also use other file types, such as .pdf or .png, depending on what system you are using}.

Firstly, you need to specify that you want to use the \texttt{graphicx} package. So you will need to place the following command in the \texttt{preamble}:
The PostScript file can then be included in your document using the command

\includegraphics[\textit{key vals}]{\textit{filename}}

where \textit{filename} is the name of your PostScript file, and \textit{key vals} is a comma-separated list of options that can be used to manipulate the image.

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

\includegraphics{shapes.ps}

Output:

\begin{center}
\includegraphics{shapes.ps}
\end{center}

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 \texttt{.ps}, however if you are using PDF\LaTeX{}, this will usually be \texttt{.pdf}. Modifying the above example, we could do:

\includegraphics{shapes}

If we use \LaTeX{}, the file \texttt{shapes.ps} will be used, and if we use PDF\LaTeX{}, the file \texttt{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 PostScript and PDF format.

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

\DeclareGraphicsExtensions{\textit{ext-list}}

where \textit{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}

or if you are using \LaTeX{} and \texttt{dvips}, you might want to first search for encapsulated PostScript (EPS) files and then for PostScript (PS) files:

\DeclareGraphicsExtensions{.eps,.ps}
The optional argument *key vals* should be a comma separated list of *key*=`label` pairs. Common options are:

- **angle=x**  
  rotate the picture by \( x^\circ \)

- **width=len**  
  scale the picture so that the width is *len*. (Remember to specify the units)

- **height=len**  
  scale the picture so that the height is *len*. (Remember to specify the units)

- **scale=value**  
  Scale the picture by *value*

- **trim=l b r t**  
  Specifies the amount to remove from each side. E.g. *trim=1 2 3 4* crops the picture by 1bp from the left, 2bp from the bottom, 3bp from the right and 4bp from the top. (The unit **bp** is a PostScript point 72bp = 1in)

- **draft**  
  Don’t actually print the image, just draw a box of the same size and print the filename inside it.

Let’s try rotating and scaling our picture:

\[
\text{\texttt{\includegraphics[angle=45,width=1in]{shapes}}} \quad \text{Input}
\]

Output:

\[
\text{Output}
\]

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

- **\rotatebox{angle}{text}**  
  Example:

  \[
  \text{\texttt{\rotatebox{45}{Some text}}} \quad \text{Input}
  \]

  Output:

  Some text

- **\scalebox{h scale}{v scale}{text}**  
  Example:

  \[
  \text{\texttt{\scalebox{0.8}{Some text}}} \quad \text{Input}
  \]

  Output:
Output:

Some text

• $\text{\reflectbox{\text{Some text}}}$

Example:

$\text{\reflectbox{\text{Some text}}}$

Output:

Some text

• $\text{\resizebox{\h length}{\v length}{\text{some text}}}$

Example:

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

Output:

Some text

The \texttt{graphicx} package can have the following options passed to it:

\texttt{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.

\texttt{final} Opposite of \texttt{draft} (default).

\texttt{hiderotate} Don’t show rotated text.

\texttt{hidescale} Don’t show scaled text.

Example:

$\text{\usepackage[draft]{graphicx}}$

Exercise 14 (Using the graphicx Package)

Download the file \texttt{shapes.ps}, and include it into your document. Alternatively, if you prefer to use PDFLaTeX, you can download the file \texttt{shapes.pdf} instead. (You can download or view an example solution.)
6.1.2 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., September 27, 2004, but 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.

For example, if you want to use the ukdate package, you would type the following in the preamble:

```
\usepackage{ukdate}
```

and the command \today will then display the date in the form: Monday 27th September, 2004.

The datetime package provides twelve different date formats, as well as providing commands for printing the current time. The required date format can either be set using a declaration, or by passing the relevant option to the package. For example, to redefine \today to display the date in the form 27/09/2004, you can either do

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

or

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

The datetime package also provides a command to define your own date format (and your own time format) if the available formats don’t meet your requirements.

6.2 Downloading and Installing Packages

New \textTeX 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 \textTeX Archive [6].

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 illustrate this with an example.
The **datetime** package comes with the files `datetime.dtx` and `datetime.ins`. You need to download both these files. You then need to **\LaTeX** the installation script to obtain the file `datetime.sty`. You do this in the **MS-DOS Prompt** by typing:

\texttt{latex datetime.ins}

This should create the file `datetime.sty`. Now you need to extract the documentation which details what commands are supplied with this package. You do this by **\LaTeX**ing the file `datetime.dtx`:

\texttt{latex datetime.dtx}

This should create the file `datetime.dvi`. Alternatively, you can use PDF\LaTeX:

\texttt{pdflatex datetime.dtx}

This will obtain the file `datetime.pdf`, and since the file `datetime.dtx` uses the `hyperref` package, all the cross-references in the PDF document will be active links.

The file `datetime.sty` needs to be put somewhere where \LaTeX\ can find it. \LaTeX\ usually searches subdirectories of `c:\texmf\tex\latex`, depending on how your system is configured. New files should be placed in the local texmf directory tree, which is usually `c:\localtexmf\tex\latex`. Put the file `datetime.sty` in a subdirectory of `c:\localtexmf\tex\latex` (e.g. `c:\localtexmf\tex\latex\datetime`), and place `datetime.dvi` or `datetime.pdf` in one of the documentation subdirectories (usually in `c:\localtexmf\doc\latex`). Once you have done this you will need to update the \TeX\ database. To do this you need to run **MiKTeX Options** which will probably be in:

Start → Programs → MiKTeX → MiKTeX Options

and then click on the Refresh Now button (See Figure 6.1). If you are using UNIX or Linux you need to use the command `texhash` or `mktexlsr`. 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 \LaTeX\ document, but if you do this, you will only be able to use it with documents in that directory.

As mentioned in the previous section, 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: Monday 27\textsuperscript{th} September, 2004. The option `short` will produce an abbreviated form, (e.g. Mon 27\textsuperscript{th} Sept, 2004) and the option `nodayofweek` won’t display the day of the week (e.g. 27\textsuperscript{th} September, 2004). These can be passed as a comma separated list in the optional argument to the `\usepackage` command. The **datetime** package (version 2.3 and above) also defines the command `\currenttime` which displays the current time. For example:

```latex
\documentclass[a4paper,11pt]{article}
\usepackage[short,nodayofweek]{datetime}
\begin{document}
\begin{verbatim}
In(put
\end{verbatim}
```

\footnote{you may have to create the relevant subdirectories}
Figure 6.1: Updating the database

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\ at \currenttime.

\end{document}

Exercise 15 (Downloading and Installing a New Package)

Try downloading the \texttt{datetime} package to give you practice extracting and installing packages. Then edit your document so that it uses the \texttt{datetime} package. (You can \texttt{download} or \texttt{view} an example.)
Figures and Tables

Figures and tables are referred to as “floats” because they are floated 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 (although there is a package called longtable that allows you to have a table that spans several pages, but that’s not covered here.)

7.1 Figures

Figures are created using the figure environment. The command:

\caption[short caption]{text}  

is used to generate the caption.

Recall from Section 6.1.1 we can include a PostScript 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}
\includegraphics{shapes}
\caption{Some shapes}
\end{figure}

So far so good, but our picture needs to be centred. This can be done using the command \centerline{object}:

\begin{figure}
\centerline{\includegraphics{shapes}}
\caption{Some shapes}
\end{figure}

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:
CHAPTER 7. FIGURES AND TABLES

\begin{figure}
\centerline{\includegraphics{shapes}}
\caption{Some shapes}
\label{fig:shapes}
\end{figure}

Now we can reference it:

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

(As before we use ~ to make an unbreakable space.) This produces the following output:

Figure 7.1 shows some shapes.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{shapes}
\caption{Some shapes}
\label{fig:shapes}
\end{figure}

Just as we were able to generate a \textit{table of contents} using \texttt{tableofcontents}, we can also generate a list of figures using the command \texttt{listoffigures}.

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

\textbf{Exercise 16 (Creating Figures)}

If you did Exercise 14, you should have a document with the image \texttt{shapes.ps} (or \texttt{shapes.pdf}) in it. You now need to put this image into a \texttt{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.)

Coming up next is a description of the \texttt{subfigure} package. If you're struggling a bit you can skip this bit and move on to Section 7.2 on page 74.
7.1.1 Subfigures

Some figures have subfigures within them. These can be generated using the `subfigure` package. Each subfigure is specified using

\subfigure[\textit{caption}] {\textit{object}} \quad \text{Definition}

For example, suppose you have two files `circle.ps` and `rectangle.ps`:

\begin{figure}
\begin{center}
\subfigure[A Rectangle] {\includegraphics{rectangle.ps}}
\hspace{1in}
\subfigure[A Circle] {\includegraphics{circle.ps}}
\end{center}
\caption{Two Shapes: (a) A Rectangle and (b) A Circle}
\end{figure}

A few notes:

- \texttt{\hspace{\textit{len}}} makes a horizontal space of length \textit{len}.
- The \texttt{center} environment centres its contents.

Again we can cross-reference the subfigures. The `\label` command should go in the mandatory argument of the `\subfigure` command.

\begin{figure}
\begin{center}
\subfigure[A Rectangle] {\label{fig:rectangle} \includegraphics{rectangle.ps}}
\hspace{1in}
\subfigure[A Circle] {\label{fig:circle} \includegraphics{circle.ps}}
\end{center}
\caption{Two Shapes: (a) A Rectangle and (b) 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.

This produces the following output:

Figure 7.2 shows some shapes. Figure 7.2(a) shows a rectangle and Figure 7.2(b) shows a circle.

and produces Figure 7.2.
Exercise 17 (Creating Sub-Figures)

Download rectangle.ps and circle.ps (or rectangle.pdf and circle.pdf) and add Figure 7.2 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 \texttt{table} environment is used instead. It is a typographical convention to have the caption at the top of the table (as opposed to figures, which have the caption at the bottom).

Example:

```
\begin{table}
  \caption{A Sample Table}
  \label{tab:sample}
  \centerline{
    \begin{tabular}{lr}
      \textbf{Item} & \textbf{Cost} \\
      Video & 8.99 \\
      CD & 9.99 \\
      DVD & 15.00 \\
    \end{tabular}
  }
\end{table}
```

This produces Table 7.1.

```
\begin{table}[h]
\centering
\begin{tabular}{ll}
\textbf{Item} & \textbf{Cost} \\
Video & 8.99 \\
CD & 9.99 \\
DVD & 15.00 \\
\end{tabular}
\caption{A Sample Table}
\label{tab:sample}
\end{table}
```

Again, \texttt{centerline} is used to centre the \texttt{tabular} environment, however the table is a little cramped, so let’s put in a bit of extra vertical space after the caption. This can be done using the command:
Table 7.2: A Sample Table

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>8.99</td>
</tr>
<tr>
<td>CD</td>
<td>9.99</td>
</tr>
<tr>
<td>DVD</td>
<td>15.00</td>
</tr>
</tbody>
</table>

\vspace{\text{length}}

Our code now looks like:

```
\begin{table}
\caption{A Sample Table}
\label{tab:sample}
\vspace{10pt}
\centerline{
\begin{tabular}{lr}
Item & Cost \\
Video & 8.99 \\
CD & 9.99 \\
DVD & 15.00 \\
\end{tabular}}
\end{table}
```

This produces Table 7.2.

As with figures, you can create a list of tables using the command

```
\listoftables
```

**Exercise 18 (Creating Tables)**

If you did Exercise 13, you should have a `tabular` environment in your document. Try turning this into a table, and 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:

   \textsf{\large\bfseries\slshape Some text}

   It would be much easier if you could use just one command to do all that, called, say, \largeboldsfsl:

   \largeboldsfsl{Some text}

   or we could call it, say, \lbsfl which is shorter, but slightly less memorable:

   \lbsfl{Some text}

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:
however, it is better to define a new command called, say, \texttt{\textsf{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 \texttt{\textsf{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 \texttt{\textsf{keyword}} command.

New commands are defined using the command:

\makeatletter
\newcommand{\texttt{cmd}}\texttt{[n-args]}\texttt{[default]}\texttt{[text]}
\makeatother

The first mandatory argument \texttt{cmd} is the name of your new command, which must start with a backslash. The optional argument \texttt{n-args} specifies how many arguments your new command must take. The next optional argument \texttt{default} will be discussed later. The final mandatory argument \texttt{text} specifies what \LaTeX{} should do everytime 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 \texttt{\textsf{coursetitle}}. We want the following code:

\makeatletter
\newcommand{\textsf{coursetitle}}{Programming --- Languages and Software Construction}
\makeatother

The course \texttt{\textsf{\textbf{coursetitle}}} is an undergraduate course.

Clearly this command doesn’t need any arguments, so we don’t need to worry about the optional argument \texttt{n-args} to \texttt{\textsf{\texttt{newcommand}}}, and the only thing our new command needs to do is print:

\texttt{Programming --- Languages and Software Construction}

so we would define our new command as follows:

\makeatletter
\newcommand{\textsf{coursetitle}}{Programming --- Languages and Software Construction}
\makeatother

Commands must always be defined before they are used. The best place to define commands is in the \texttt{preamble}:
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}}

In this case we have used the optional argument n-args 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

\keyword\{commands\}

will be equivalent to

\textsf{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:
A \texttt{command} usually begins with a backslash.

Now if we want to change the way the keywords are formatted, we can simply change the definition of \texttt{keyword}. Let’s modify our code so that the keyword is now in a slanted sans-serif font:

\begin{verbatim}
\documentclass[a4paper]{article}
\newcommand{\keyword}[1]{\textsf{\slshape #1}}
\begin{document}
A \texttt{command} usually begins with a backslash.
\end{document}
\end{verbatim}

Let’s go one stage further. The \texttt{color} package enables the use of colour, so let’s make our keywords blue:

\begin{verbatim}
\documentclass[a4paper]{article}
\usepackage{color}
\newcommand{\keyword}[1]{\textsf{\slshape\color{blue}#1}}
\begin{document}
A \texttt{command} usually begins with a backslash.
\end{document}
\end{verbatim}

Or we could index the keywords. To do this we need the \texttt{package makeidx} and the commands \texttt{makeindex}, \texttt{index\{text\}} and \texttt{printindex}:

\begin{verbatim}
\documentclass[a4paper]{article}
\end{verbatim}
\usepackage{makeidx}
\makeindex
\newcommand{\keyword}[1]{\textsf{\slshape #1}\index{#1}}
\begin{document}
A \keyword{command} usually begins with a backslash.
\printindex
\end{document}

For further information about how to create an index, see *A Guide to \TeX* \[2\] or *The \TeX\ Companion* \[3\]. Alternatively, if you want a brief overview on-line, try *Using \TeX\ to Write a PhD Thesis*\[1\].

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.

Segments of code may be \keyword{grouped}.

Some \keyword{commands} take one or more \keyword{arguments}.
\end{document}

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, using \texttt{dvips} as described in Chapter 3, or use PDF\LaTeX instead of \LaTeX.) Again you can \texttt{download} or \texttt{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 \texttt{makeindex}. Suppose your source code is saved as \texttt{exercise19.tex}, then if you are using the MS-DOS \texttt{Prompt} you will need to do:

\[^1\texttt{http://theoval.cmp.uea.ac.uk/~nlct/latex/thesis/thesis.html}\]
8.1 Defining Commands with an Optional Argument

As mentioned earlier, the \texttt{\textbackslash newcommand} command has an optional argument \texttt{default}. This allows you to define a command with an optional argument. For example, suppose we want a command called, say, \texttt{\textbackslash price}. Suppose we want the following code:

\begin{verbatim}
\texttt{\textbackslash price}\{100\}
\end{verbatim}

Input

to produce the following output:

\begin{verbatim}
£100 excl VAT @ 17.5\%
\end{verbatim}

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:

\begin{verbatim}
\texttt{\textbackslash price}[0]\{30\}
\end{verbatim}

Input

produce the following output:

\begin{verbatim}
£30 excl VAT @ 0\%
\end{verbatim}

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:

\begin{verbatim}
\texttt{\textbackslash newcommand}\{\textbackslash price\}[2][17.5]\{\textbackslash pounds \ #2\ \textbackslash excl\ \textbackslash VAT\ \textbackslash @\ \#1\%\}
\end{verbatim}

Input

Here, \texttt{#1} represents the optional argument (by default 17.5) and \texttt{#2} represents the mandatory argument (the second argument if the optional argument is present, or the only argument if the optional argument is absent.)
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, \texttt{\textbackslash cost}. It should take one optional argument and one mandatory argument. Without the optional argument, it behaves in the same way as the \texttt{\textbackslash price} example above, so that, say,

\texttt{\textbackslash cost\{50\}}

will produce

\texttt{£50 excl VAT @ 17.5\%}

but with the optional argument, you can change the \texttt{excl VAT @ 17.5\%} bit. So that, say,

\texttt{\textbackslash cost[inc VAT]\{50\}}

will produce

\texttt{£50 inc VAT}

You can download or view the solution.

8.2 Redefining Commands

Commands can be redefined using the command:

\texttt{\textbackslash renewcommand\{cmd\}\{n-args\}\{default\}\{text\}}

This has exactly the same format as \texttt{\textbackslash newcommand} but is used for redefining existing commands. \textbf{Caveat}: never redefine a command whose existing function is unknown to you.

Recall the \texttt{itemize} environment from Section 4.3.1. You may have up to four nested \texttt{itemize} environments, the labels for the outer environment are specified by the command \texttt{\textbackslash labelitemi}, the labels for the second level are specified by \texttt{\textbackslash labelitemii}, the third by \texttt{\textbackslash labelitemiii} and the fourth by \texttt{\textbackslash labelitemiv}. By default, \texttt{\textbackslash labelitemi} is a bullet point, \texttt{\textbackslash labelitemii} is an en dash, \texttt{\textbackslash labelitemiii} is an asterisk and \texttt{\textbackslash labelitemiv} is a dot (• – * ·). These can be changed by redefining \texttt{\textbackslash labelitemi} etc.

Example: Recall the command \texttt{\textbackslash dag} produces a dagger symbol, we can use this symbol instead of a bullet point:

\texttt{\textbackslash renewcommand\{\textbackslash labelitemi\}\{\textbackslash dag\}}

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

Here’s another example, it uses the PostScript font\textsuperscript{2} ZapfDingbats via the \texttt{pifont} package:

\begin{verbatim}
\renewcommand{\labelitemi}{\ding{43}}
\begin{itemize}
\item Animal
\item Mineral
\item Vegetable
\end{itemize}
\end{verbatim}

In the above example, it would actually be better to use the \texttt{dinglist} environment defined in the \texttt{pifont} package. See \textit{The \LaTeX\ Companion} \cite{Lamport1994} for more details.

You may have noticed that \LaTeX\ automatically generates pieces of text such as “Chapter”, “Figure”, “Bibliography”. These are generated by the commands shown in Table 8.1.

\textsuperscript{2}This font may not come out if you view the PDF version of your document in \texttt{xpdf}
Table 8.1: Object Names (†report class file, ‡article class file, remainder both report and article)

<table>
<thead>
<tr>
<th>Command</th>
<th>Default Text</th>
</tr>
</thead>
<tbody>
<tr>
<td>\contentsname</td>
<td>Contents</td>
</tr>
<tr>
<td>\listfigurename</td>
<td>List of Figures</td>
</tr>
<tr>
<td>\listtablename</td>
<td>List of Tables</td>
</tr>
<tr>
<td>\bibname†</td>
<td>Bibliography</td>
</tr>
<tr>
<td>\refname‡</td>
<td>References</td>
</tr>
<tr>
<td>\indexname</td>
<td>Index</td>
</tr>
<tr>
<td>\figurename</td>
<td>Figure</td>
</tr>
<tr>
<td>\tablename</td>
<td>Table</td>
</tr>
<tr>
<td>\partname</td>
<td>Part</td>
</tr>
<tr>
<td>\chaptername†</td>
<td>Chapter</td>
</tr>
<tr>
<td>\appendixname</td>
<td>Appendix</td>
</tr>
<tr>
<td>\abstractname</td>
<td>Abstract</td>
</tr>
</tbody>
</table>

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}

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, \LaTeX{} 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 $f(x)$.

Displayed maths is mathematics that occurs on a line of its own. For example:

A polynomial is a function of the form

$$f(x) = \sum_{i=0}^{n} a_i x^i$$

9.1 In-Line Mathematics

In-line mathematics is created using the \texttt{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: $\backslash($ is equivalent to $\begin{math}$ and $\backslash)$ is equivalent to $\end{math}$. So the example above can be rewritten:

The variable $(x)$ is transformed by the function $(f(x))$.
There is an even shorter notation: The special character $ is equivalent to both \begin{math} and \end{math}:

The variable $x$ is transformed by the function $f(x)$.

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 $f(x)$.

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.

Notice the difference between $(x, y, z)$ and \textit{(x, y, z)}.

Notice the difference between (x, y, z) and (x, y, z).

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}

Output:

A linear function is a function of the form 

$y = mx + c$

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}

results in the following output:

\begin{equation}
y = mx + c \tag{9.1}
\end{equation}

Recall from Section 5.5 that we can cross-reference most things that \LaTeX automatically numbers using \texttt{\textbackslash ref} and \texttt{\textbackslash label}. Equations can be cross-referenced in the same way:

Equation~\texttt{\textbackslash ref(eqn:linear)} is a linear function.
\begin{equation}
f(x) = mx + c
\end{equation}

Equation 9.2 is a linear function.
\begin{equation}
f(x) = mx + c \tag{9.2}
\end{equation}

\textbf{Note:} both the \texttt{equation} and the \texttt{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 \texttt{eqnarray} or \texttt{align}. This document does not cover these environments, but if you are interested see The \LaTeX Companion [3] or A Guide to \LaTeX [2].

### 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 "\texttt{Missing $ inserted}" error message.
Table 9.1: Maths Font Changing Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Example Input</th>
<th>Corresponding Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{maths}</td>
<td>$\text{maths}(xyz)$</td>
<td>$xyz$</td>
</tr>
<tr>
<td>\textsf{maths}</td>
<td>$\textsf{maths}(xyz)$</td>
<td>$xyz$</td>
</tr>
<tr>
<td>\texttt{maths}</td>
<td>$\texttt{maths}(xyz)$</td>
<td>$xyz$</td>
</tr>
<tr>
<td>\textit{maths}</td>
<td>$\textit{maths}(xyz)$</td>
<td>$xyz$</td>
</tr>
<tr>
<td>\textbf{maths}</td>
<td>$\textbf{maths}(xyz)$</td>
<td>$xyz$</td>
</tr>
<tr>
<td>\textcal{maths}</td>
<td>$\textcal{maths}(XYZ)$</td>
<td>$\mathcal{XYZ}$</td>
</tr>
</tbody>
</table>

9.3.1 Maths Fonts

Just as we are able to change text fonts using the commands \textit, \textbf etc, we can also use commands to change the maths font. Basic maths font changing commands are shown in Table 9.1.

The caligraphic 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}:

\begin{displaymath}
\begin{array}{ll}
\text{Input} & \text{Output} \\
\text{\begin{displaymath} x > y \mbox{ and } y < z \end{displaymath}} & \begin{array}{c}
\mbox{$x>y$ and $y < z$} \\
\text{\begin{displaymath} x > y \text{ and } y < z \end{displaymath}} & \begin{array}{c}
\text{\begin{displaymath} x > y \text{ and } y < z \end{displaymath}}
\end{array}
\end{array}
\end{array}
\end{displaymath}

Table 9.2 lists additional font commands supplied with the amsmath and amsfonts packages.

9.3.2 Greek Letters

Greek letters that differ from the corresponding Roman letters are obtained by placing a backslash in front of the name. Lower case Greek letters are shown in Table 9.3 and upper case Greek letters are shown in Table 9.4.
Note that there is no omicron as this is the same as a lower case o. 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

\[ \texttt{sb}\{\textit{maths}\} \]

or by the special character:

\[ _{\{\textit{maths}\}} \]

Superscripts are obtained either by the command

\[ \texttt{sp}\{\textit{maths}\} \]

or by the special character:

\[ ^{\{\textit{maths}\}} \]

Examples:

1. This example uses \(\texttt{sb}\) and \(\texttt{sp}\):

\[
\begin{displaymath}
  y = x_{1}^{2} + x_{2}^{2}
\end{displaymath}
\]

2. This example uses \(\_\) and \(^\)

<table>
<thead>
<tr>
<th>Command</th>
<th>Example Input</th>
<th>Example Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\texttt{mathbb}{\textit{maths}})</td>
<td>$\texttt{mathbb}A+B=C$</td>
<td>$A + B = C$</td>
</tr>
<tr>
<td>(\texttt{mathfrak}{\textit{maths}})</td>
<td>$\texttt{mathfrak}A+B=C$</td>
<td>$A + B = C$</td>
</tr>
<tr>
<td>(\texttt{boldsymbol}{\textit{maths}})</td>
<td>$\texttt{boldsymbol}A+B=C$</td>
<td>$A + B = C$</td>
</tr>
<tr>
<td>(\texttt{pmb}{\textit{symbol}})</td>
<td>$\texttt{pmb}+-=$</td>
<td>$+-=$</td>
</tr>
</tbody>
</table>
Table 9.3: Lower Case Greek Letters

\alpha \quad \alpha
\beta \quad \beta
\gamma \quad \gamma
\delta \quad \delta
\epsilon \quad \epsilon
\varepsilon \quad \varepsilon
\zeta \quad \zeta
\eta \quad \eta
\theta \quad \theta
\vartheta \quad \vartheta
\iota \quad \iota
\kappa \quad \kappa
\lambda \quad \lambda
\mu \quad \mu
\nu \quad \nu
\xi \quad \xi
\pi \quad \pi
\varpi \quad \varpi
\rho \quad \rho
\varrho \quad \varrho
\sigma \quad \sigma
\varsigma \quad \varsigma
\tau \quad \tau
\upsilon \quad \upsilon
\phi \quad \phi
\varphi \quad \varphi
\chi \quad \chi
\psi \quad \psi
\omega \quad \omega

Table 9.4: Upper Case Greek Letters

\Gamma \quad \Gamma
\Delta \quad \Delta
\Theta \quad \Theta
\Lambda \quad \Lambda
\Xi \quad \Xi
\Pi \quad \Pi
\Sigma \quad \Sigma
\Upsilon \quad \Upsilon
\Phi \quad \Phi
\Psi \quad \Psi
\Omega \quad \Omega

\begin{displaymath}
 y = x_1^2 + x_2^2
\end{displaymath}

3. Recall 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}
CHAPTER 9. MATHEMATICS

91

\begin{displaymath}
y = x_1^2 + x_2^2
\end{displaymath}

This is simpler than the above two examples.

4. Subscripts and superscripts can also be nested:

\begin{displaymath}
f(x) = e^{x_1}
\end{displaymath}

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) = \text{e}^{x_1}
\end{displaymath}

If you are going to use e a lot, it will be simpler to define a new command to do this:

\begin{verbatim}
\newcommand{\e}{\text{e}}
\end{verbatim}

\begin{displaymath}
f(x_1, x_2) = \e^{x_1^2} + \e^{x_2^2}
\end{displaymath}
CHAPTER 9. MATHEMATICS

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

| \arccos | \arcsin | \arctan | \arg | \cos | \cosh |
| \cot | \coth | \csc | \deg | \det | \dim |
| \exp | \gcd | \hom | \inf | \ker | \lg |
| \lim | \liminf | \limsup | \ln | \log | \max |
| \min | \Pr | \sec | \sin | \sinh | \sup |
| \tan | \tanh |

Of these functions, the following functions can have limits by using the subscript command _ or the superscript command ^:

| \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}
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 $\rightarrow$.

\begin{displaymath}
\lim_{x \to \infty} f(x)
\end{displaymath}

3. This is another example of a functional name using a subscript:

\begin{displaymath}
\min_x f(x)
\end{displaymath}

In addition, the following commands are also available:

<table>
<thead>
<tr>
<th>Command</th>
<th>Example Input</th>
<th>Example Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bmod$ $m \bmod n$</td>
<td>$m \bmod n$</td>
<td>$m \bmod n$</td>
</tr>
<tr>
<td>$\pmod{\text{maths}}$ $m \pmod{n}$</td>
<td>$m \pmod{n}$</td>
<td>$m \pmod{n}$</td>
</tr>
</tbody>
</table>

If you want a function that isn’t specified in Table 9.5, you can use the command

\texttt{$\operatorname{operator name}$}

or

\texttt{$\operatornamewithlimits{operator name}$}

both of which are defined in the \texttt{amsmath} package.

The second of these commands, $\operatornamewithlimits$, allows you to have a function that can take limits using the $\_\text{or}^\text{}$ commands, just like $\lim$, $\min$ etc. Examples
1. Suppose we want a function called card, which represents the cardinality of a set $\mathcal{S}$:

\begin{displaymath}
n = \operatorname{card}(\mathcal{S})
\end{displaymath}

In this example $\mathcal{S}$ is used as sets are usually represented in a calligraphic font.

2. It’s a bit cumbersome having to keep typing $\operatorname{card}$ everytime you want card, a better thing to do would be to define a new command called $\texttt{\textbackslash card}$. 

\begin{displaymath}
n = \texttt{\textbackslash card}(\mathcal{S})
\end{displaymath}

3. Let’s have an example of an operator that takes a limit:

\begin{displaymath}
x_m = \operatorname{mode}_{x \in \mathcal{S}}(x)
\end{displaymath}
9.3.5 Fractions

Fractions are created using the command

\[ \frac{\text{numerator}}{\text{denominator}} \]

Definition

Examples:

1. A simple fraction:

\[
\frac{1}{1+x}
\]

2. A nested fraction:

\[
\frac{1 + \frac{1}{x}}{1 + x + x^2}
\]
3. A derivative:

\begin{displaymath}
\frac{df}{dx}
\end{displaymath}

Again, as with e, the differential operator ‘d’ should be in an upright font as it is not a variable:

\begin{displaymath}
\frac{\mathrm{d}f}{\mathrm{d}x}
\end{displaymath}

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:

\newcommand{\deriv}[2]{\frac{\mathrm{d}#1}{\mathrm{d}#2}}

\begin{displaymath}
\deriv{f}{x}
\end{displaymath}
5. Partial derivatives can be obtained similarly using the command \( \texttt{\partial} \) to display the partial derivative symbol:

\begin{verbatim}
\newcommand{\pderiv}[2]{\frac{\partial #1}{\partial #2}}
\begin{displaymath}
f_x = \pderiv{f}{x}
\end{displaymath}
\end{verbatim}

6. A double partial derivative:

\begin{verbatim}
\begin{displaymath}
f_{xy} = \frac{\partial^2 f}{\partial x \partial y}
\end{displaymath}
\end{verbatim}

9.3.6 Roots

Roots are obtained using the command

\begin{verbatim}
\sqrt[\text{order}]{\text{maths}}
\end{verbatim}

without the optional argument \textit{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}

\sqrt{a+b}

2. A cubic root:

\begin{displaymath}
\sqrt[3]{a+b}
\end{displaymath}

\sqrt[3]{a+b}

3. An $n$th root:

\begin{displaymath}
\sqrt[n]{a+b}
\end{displaymath}

\sqrt[n]{a+b}
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 \nsubset.

Binary operator signals are shown in Table 9.7, and arrow symbols are shown in Table 9.8.

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:

1. Displayed Maths

\begin{displaymath}
f(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i
\end{displaymath}

2. In-line Maths

\begin{math}
f(x) = \sum_{i=1}^{n} x_i + \prod_{i=1}^{n} x_i
\end{math}
Table 9.7: Binary Operator Symbols

| \amalg   | II | \ast | * | \bullet | ⋅ |
| \bigcirc | ○ | \bigtriangledown | ▽ | \bigtriangleup | △ |
| \cap     | ☐ | \cdot | · | \circ | ◦ |
| \cup     | ☐ | \dagger | † | \ddagger | ‡ |
| \diamond | ○ | \div | ⊸ | \mp | ⊩ |
| \odot    | ○ | \ominus | ⊺ | \oplus | ⊕ |
| \oslash  | ○ | \otimes | ⊗ | \pm | ⊞ |
| \setminus | \setminus | \sqcap | ⊓ | \sqcup | ⊔ |
| \star    | × | \times | × | \triangleleft | ◄ |
| \triangleright | > | \uplus | ⊕ | \vee | ∨ |
| \wedge   | ^ | \wr | ? |

Table 9.8: Arrow Symbols

| \downarrow | ↓ | \Downarrow | ⇓ |
| \hookleftarrow | ← | \hookrightarrow | → |
| \leftarrow or \gets | ← | \Leftarrow | ⇐ |
| \leftharpoondown | ← | \leftharpoonup | ↩ |
| \leftrightharpoons | ⇐ | \uparrow | ↑ |
| \longleftarrow | ←− | \Longrightarrow | ⇒ |
| \longmapsto | →−− | \swarrow | ↘ |
| \nearrow | ↗ | \searrow | ↙ |
| \rightarrow or \to | → | \Rightarrow | ⇒ |
| \rightharpoondown | ← | \rightharpoonup | ↩ |
| \rightleftharpoons | ⇐ | \Leftarrow | ⇐ |
| \longmapsto | →−− | \downarrow | ⇓ |
| \Longleftarrow | ⇐ | \searrow | ↘ |
| \Uparrow | ↑ | \mapsto | ↩ |

Table 9.9: Symbols with Limits

| \sum | ∑ | \int | ∫ | \oint | ∮ |
| \prod | ∏ | \coprod | ∐ | \bigcap | ∩ |
| \bigcup | ∪ | \bigsqcup | ∪ | \bigvee | ∨ |
| \bigwedge | ∧ | \bigodot | ⊙ | \bigotimes | ⊗ |
| \bigoplus | ⊕ | \biguplus | ⊳ |
Ellipsis commands are shown in Table 9.10.

Table 9.10: Ellipses

\ldots \ \cdots \\
\vdots \ \ddots

Examples:

1. Low ellipsis: This example uses the command \texttt{\textbackslash for all} to produce the ‘for all’ symbol $\forall$, and it also uses $\texttt{\textbackslash_\_}$ (backslash space) to make a space before the for all symbol:

\begin{displaymath}
a_i x_i = b_i \ \forall i = 1, \ldots, n
\end{displaymath}

2. Centred ellipsis:
For other symbol commands, see *A Guide to \LaTeX* [2] or *The \LaTeX Companion* [3].

**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:

```
\begin{displaymath}
(\frac{1}{1+x})
\end{displaymath}
```

Under such circumstances, it is better to use the commands:

```
\left(delimiter
\right(delimiter)
```

Note that you must always have matching \left and \right commands, although the delimiters used may be different. Available delimiters are shown in Table 9.11.

If you want one of the delimiters to be invisible, use a . (full stop) as the delimiter.

Examples:

1. Round bracket delimiters:
Table 9.11: Delimiters

\begin{displaymath}
\left( \frac{1}{1+x} \right)
\end{displaymath}

2. Vertical bar delimiters:

\begin{displaymath}
\left| \frac{1}{1+x} \right|
\end{displaymath}

3. Delimiters don’t have to match:
We have now learnt enough to reproduce the equation shown in Chapter 1:

\begin{displaymath}
\left[ \frac{1}{1+x} \right]
\end{displaymath}

\begin{displaymath}
\newcommand{\pderiv}{\frac{\partial}{\partial}}
\newcommand{\e}{\mathrm{e}}
\begin{align*}
\pderiv{\mathcal{L}}{z_i^\rho}^2 &= -\pderiv{\rho_i}{z_i^\rho} \\
& \quad \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})}{(1-\e^{v_i})^2} + \e^{2v_i} \pderiv{v_i}{\rho_i} \right)
\end{align*}
\end{displaymath}

\textbf{Note:} 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:

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}
```

2. We now need a partial derivative:

```
\begin{displaymath}
\pderiv{}{}
\end{displaymath}
```

3. 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 caligraphic L:

```
\begin{displaymath}
\pderiv{^2 \mathcal{L}}{}
\end{displaymath}
```

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}
```

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)^2}
\end{displaymath}
```

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)^2}
\end{displaymath}
```
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( \pderiv{v_i}{\rho_i} \right)
\end{displaymath}

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}

9. This is followed by \( v_i \) times another 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}} + v_i \frac{}{} \frac{(1-e^{-v_i})}{(1-e^{-v_i})} \right)
\end{displaymath}

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( \pderiv{v_i}{\rho_i} \frac{e^{v_i}}{1-e^{v_i}} + v_i \frac{}{} (1-e^{-v_i}) \right)
\end{displaymath}
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:
\( +\e^{-2v_i} \)

(e) And finally we have the last term:
\( \pderiv{v_i}{\rho_i} \)

12. Putting it all together, we have:

\[
\begin{align*}
\pderiv{^2\mathcal{L}}{{z_i}^2} &= -\pderiv{\rho_i}{z_i} \left( \pderiv{v_i}{\rho_i} \frac{\e^{v_i}}{1-\e^{v_i}} + v_i \frac{\e^{v_i}\pderiv{v_i}{\rho_i}(1-\e^{v_i})}{1-\e^{v_i}} +\e^{-2v_i}\pderiv{v_i}{\rho_i}((1-\e^{-v_i})^2} \right)
\end{align*}
\]

13. And remember that if you haven’t already defined \texttt{\pderiv} and \texttt{\e}, you will need to do that in the preamble

\newcommand{\pderiv}[2]{\frac{\partial #1}{\partial #2}}
\newcommand{\e}{\mathrm{e}}

(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:

   \begin{displaymath}
   \begin{array}{rrr}
   0 & 1 & 19 \\
   -6 & 10 & 200 \\
   \end{array}
   \end{displaymath}

2. Let’s add some delimiters:

   \begin{displaymath}
   \left(\begin{array}{rrr}
   0 & 1 & 19 \\
   -6 & 10 & 200 \\
   \end{array}\right)
   \end{displaymath}
3. This example uses an invisible delimiter:

\begin{displaymath}
\begin{array}{rl}
-1 & x < 0 \\
0 & x = 0 \\
+1 & x > 0
\end{array}
\end{displaymath}

9.3.10 Vectors

Vectors can be produced using the command:

\texttt{\vec{variable}}

Example:

\begin{displaymath}
\vec{x}
\end{displaymath}

These days it is customary to typeset vectors in bold. This can be done by redefining the \texttt{\vec} command. You could use \texttt{\mathbf}, for example:
\renewcommand{\vec}[1]{\mathbf{#1}}
\begin{displaymath}
\vec{x}\cdot\vec{\xi} = z
\end{displaymath}

Exercise 23 (Maths: Vectors and Arrays)

Try to produce the following:

\[Ax = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \end{pmatrix} = y\]

As before, you can download or view the solution.

9.3.11 Mathematical Spacing

\LaTeX{} 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 \qquad. It also has a function name, diag, and it uses the \forall and \ldots symbols. It also redefines the \vec
command, as was done in the previous section, and it uses delimiters and the \texttt{array}

Try to reproduce the following output:

The set of linear equations:

\[ a_i x_i = b_i \quad \forall i = 1, \ldots, n \]

can be written as a matrix equation:

\[ \text{diag}(a)x = b \]

where \( x = (x_1, \ldots, x_n)^T \), \( b = (b_1, \ldots, b_n)^T \) and

\[
\text{diag}(a) = \begin{bmatrix}
ad_1 & 0 & \cdots & 0 \\
o & a_2 & \ddots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
o & \cdots & 0 & a_n \\
\end{bmatrix}
\]

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}

is used to define a new environment. As with new commands, you can use the optional argument \texttt{n-args} to define an environment with arguments, and \texttt{default} to define an environment with an optional argument.

The first argument \texttt{env-name} is the name of your new environment. Remember that the environment name must not have a backslash. The mandatory arguments \texttt{begin-code} and \texttt{end-code} indicate what \LaTeX{} 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, \texttt{exercise} that prints \texttt{Exercise} in bold and typesets the contents of the environment in italic. In other words, we want the following code:

\begin{exercise}
This is a sample.
\end{exercise}

Let’s first consider what we want this environment to do: we can get the word “Exercise” in bold by simply doing \texttt{\textbf{Exercise}}, and the italic font can be obtained by using the \texttt{itshape} environment. So, at the start of our new environment we need to do \texttt{\textbf{Exercise}} and we need to begin the \texttt{itshape} environment, and at the end of our new environment we need to end the \texttt{itshape} environment:
Let’s try it out:

\begin{exercise}
This is a sample.
\end{exercise}

\textbf{Exercise}  \textit{This is a sample.}

Not quite right. Let’s put a paragraph break after \textbf{Exercise}, and put one before it as well. The command \texttt{\par} can be used to make a paragraph break:

\begin{verbatim}
\newenvironment{exercise} % environment name
{\par\textbf{Exercise}\begin{itshape}\par} % begin code
{\end{itshape}} % end code
\end{verbatim}

Let’s have a look at the output now:

\textbf{Exercise}
\textit{This is a sample.}

One more thing, we need to remove the paragraph indentation. This can be done using the command \texttt{\noindent}:

\begin{verbatim}
\newenvironment{exercise}
{\par\noindent\textbf{Exercise}\begin{itshape}\par\noindent} % begin code
{\end{itshape}} % end code
\end{verbatim}

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{verbatim}
\begin{exercise}{An Example}
This is a sample.
\end{exercise}
\end{verbatim}

should produce the following result:

\textbf{Exercise (An Example)}
\textit{This is a sample.}
As with \texttt{newcommand}, \#1 is used to indicate the first argument. We can now modify the code as follows (modifications are indicated \textit{like this}):

\begin{verbatim}
\newenvironment{exercise}[1]{
  \par\noindent\textbf{Exercise (#1)}\begin{itshape}\par
  \noindent}
  {\end{itshape}}
\end{verbatim}

Exercise 25 (Defining a New Environment)

If you did any of the exercises from Exercise 8 to Exercise 18, go back to the document you created and define the \texttt{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 \texttt{\vspace{length}}. Again you can download or view an example.
Chapter 11
Counters

As we have seen, \LaTeX automatically generates numbers for chapters, sections, equations etc. These numbers are stored in counters. The names of these counters is usually the same as the name of the object with which it is associatated but without any backslash. For example, the \texttt{chapter command} has an associated counter called \texttt{chapter}, the \texttt{footnote command} has an associated counter called \texttt{footnote}, the \texttt{equation environment} has an associated counter called \texttt{equation}, the \texttt{figure environment} has an associated counter called \texttt{figure} and the \texttt{table environment} has an associated counter called \texttt{table}. There is also a counter called \texttt{page} that keeps track of the current page number.

The value of a counter can be displayed using the command

\begin{verbatim}
\texttt{\the counter}
\end{verbatim}

where \texttt{counter} is the name of the associated counter. Note that \texttt{counter} does not go in curly braces and adjoins \texttt{\the} (e.g. \texttt{\thepage}, \texttt{\thesection}, \texttt{\thechapter}).

Example:

\begin{verbatim}
This page is Page\thepage.  
The current chapter is Chapter\thechapter.
\end{verbatim}

This page is Page 115. The current chapter is Chapter 11.

New counters can be created using the command:

\begin{verbatim}
\texttt{\newcounter{counter-name} [outer-counter]}
\end{verbatim}

The \texttt{mandatory argument} \texttt{counter-name} is the name of your new counter (no backslash in the name). For example, let’s define a counter called \texttt{exercise} to keep track of each exercise.

\begin{verbatim}
\newcounter{exercise}
\end{verbatim}

we can now display the value of the counter using the command \texttt{\theexercise}. At the moment the counter has the value zero, the value can be changed using one of the following commands:
A couple of the commands above take a number \texttt{num} as one of the arguments. If you want to use another counter for this argument, you need to use \texttt{\value{counter}}

For example, if you want to set our new \texttt{exercise} counter to the same value as the \texttt{page} counter, you would do

\begin{verbatim}
\setcounter{exercise}{\value{page}}
\end{verbatim}

Let’s go back to the \texttt{exercise} environment you created in Exercise 25. The exercises really ought to have an associated number, and this number should be incremented each time we use the exercise environment. So let’s modify our code to do this. Modifications are illustrated like this:

\begin{verbatim}
\newcounter{exercise}
\newenvironment{exercise}{1}{}
{refstepcounter{exercise}\vspace{10pt}\par\noindent\textbf{Exercise \theexercise\ (#1)}}
{end{itshape}\vspace{10pt}\par}
\end{verbatim}

Note that the counter needs to be incremented before it is used. Since we’ve used \texttt{refstepcounter} instead of \texttt{stepcounter} we can cross-reference our \texttt{exercise} environment:

\begin{verbatim}
Exercise \ref{ex:simple} is a simple exercise.
\begin{exercise}{Simple Exercise}
\label{ex:simple}
This is a simple exercise.
\end{exercise}
\end{verbatim}

Exercise 1 is a simple exercise.

\textbf{Exercise 1 (Simple Exercise)}

\textit{This is a simple exercise.}
The counter representation can be changed by redefining \texttt{\theexercise} using the command \texttt{\renewcommand}. The following commands can be used to display the counter:

- \texttt{\arabic{counter}}: Arabic number (1, 2, 3, \ldots)
- \texttt{\Roman{counter}}: Uppercase Roman numeral (I, II, III, \ldots)
- \texttt{\roman{counter}}: Lowercase Roman numeral (i, ii, iii, \ldots)
- \texttt{\alph{counter}}: Lowercase letter (a, b, c, \ldots, z)
- \texttt{\Alph{counter}}: Uppercase letter (A, B, C, \ldots, Z)
- \texttt{\fnsymbol{counter}}: A footnote symbol (∗ † ‡ § ¶ \| ∗∗ †† ‡‡)

For example, to make the chapter numbers appear as uppercase Roman numerals you would do:

\begin{verbatim}
\renewcommand{\thechapter}{\Roman{chapter}}
\end{verbatim}

You may have noticed that \texttt{\newcounter} has an optional argument \texttt{outer-counter}. This is for use if you require the new counter to be reset every time \texttt{outer-counter} is incremented. For example, the section numbers in the \texttt{report} class are dependent on the chapter numbers. Each time a new chapter is started, the section numbers are reset. Suppose we want our \texttt{exercise} counter to be dependent on the \texttt{chapter} counter, we would do

\begin{verbatim}
\newcounter{exercise}[chapter]
\end{verbatim}

We now need to modify \texttt{\theexercise} so that it includes the chapter number:

\begin{verbatim}
\renewcommand{\theexercise}{\thechapter.\arabic{exercise}}
\end{verbatim}

Notice the use of \texttt{\thechapter} instead of, say, \texttt{\arabic{chapter}}. By using \texttt{\thechapter} we don’t need to keep track of the \texttt{chapter} counter format.

**Exercise 26 (Using Counters)**

Modify the document from Exercise 25 so that the \texttt{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 \textwidth, 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}  

where \texttt{cmd} is the particular length command (e.g. \parindent) and \texttt{length} is the new length. To display the value of a length do:

\the\texttt{cmd}  

Example:

\setlength{\parindent}{0pt}  

This is the first paragraph.

<table>
<thead>
<tr>
<th>Table 12.1: Units of Measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{pt}</td>
</tr>
<tr>
<td>\texttt{in}</td>
</tr>
<tr>
<td>\texttt{mm}</td>
</tr>
<tr>
<td>\texttt{cm}</td>
</tr>
<tr>
<td>\texttt{ex}</td>
</tr>
<tr>
<td>\texttt{em}</td>
</tr>
<tr>
<td>\texttt{sp}</td>
</tr>
<tr>
<td>\texttt{bp}</td>
</tr>
<tr>
<td>\texttt{dd}</td>
</tr>
<tr>
<td>\texttt{pc}</td>
</tr>
<tr>
<td>\texttt{cc}</td>
</tr>
<tr>
<td>\texttt{mu}</td>
</tr>
</tbody>
</table>
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 \TeX to stretch or contract the text to get the body of text as flushed with the margins as possible.

For example, the paragraph gap \parskip is usually set to 0pt plus 1pt. This means that the preferred gap is 0pt but \TeX 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 0.0pt.

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 MiKTeX, and the documentation should be in one of the subdirectories of \texmf\doc\latex. 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 screen is:

Bad command or file name

then you have either mistyped the command name, or you don’t have $\LaTeX$ 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 MiKTeX 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)
! 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 notepad and the MS-DOS Prompt, 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:

dir

at the MS-DOS prompt. 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 WinEdit or TeXnicCenter as they set the directory, and you only need to click a button, so you won’t get any typing errors.

• Error messages will usually look something like:

! Undefined control sequence.
1.1 \documentclass
\[a4paper,11pt]\{article\}

?
CHAPTER 13. COMMON ERRORS

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 line \LaTeX has processed so far, and staggered on the following 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: `\today, which is nice to know.`

The error in this case is the misspelling of the command `\today`. The error message will appear as follows:

! Undefined control sequence.  
1.8 The date today is: \today, which is nice to know.  
\LaTeX Prompt

At the \LaTeX prompt, you can either type `h` for a help message, or type `x` to exit \LaTeX and go back to your source code and fix the problem.

There follows below a list of common error messages.

13.1 * (No message, just an asterisk prompt!)

You’ve gone into \TeX! This is probably because you’ve forgotten the `\end{document}`. The asterisk is the \TeX prompt. At this point the best thing to do is type `\end{document}` at the prompt (followed by the return character `\`). and hope for the best.

13.2 Argument of \cune 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{document} (incorrect)`
instead of
\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:
\documentclass[a4paper]{article}
\begin{document}
usepackage{graphicx}
\end{document}

(incorrect)

\documentclass[a4paper]{article}
\usepackage{graphicx}
\begin{document}
\begin{document}
\end{document}

(correct)

\documentclass[a4paper]{article}
\usepackage{graphicx}
\begin{document}
\end{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 $.

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{document} (incorrect)

  instead of

  \begin{document} (correct)

• 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.

### 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.

\TeX 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 →.

  • Make sure that you have spelt the filename correctly.

    This error will be caused by, say,

  \documentclass[a4paper]{article} (incorrect)
instead of
\documentclass[a4paper]{article} \textit{(incorrect)}
If this is the case, simply type in the correct name at the prompt (followed by the return character)

- 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 \texttt{shapes.ps} in the subdirectory \texttt{pictures} then you would get a ‘file not found’ error message if you did
\begin{verbatim}
\includegraphics{shapes.ps} \textit{(incorrect)}
\end{verbatim}
instead of \begin{verbatim}\includegraphics{pictures/shapes.ps} \textit{(correct)}\end{verbatim}

- 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 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 \texttt{tabular} or \texttt{array} environment that is not allowed. The standard available characters are: \texttt{r} (right justified), \texttt{l} (left justified) and \texttt{c} (centred). This error will also occur if you have forgotten the argument to the \texttt{tabular} or \texttt{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 \texttt{#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 \texttt{\newcommand}, otherwise it will be assumed that the command has no arguments.

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 \texttt{item}
The command \texttt{item} may only appear in one of the list making environments (such as \texttt{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 \(\text{(wrong)}\)

instead of

\& our equipment \(\text{(correct)}\)

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 \(\text{\textit{wrong}}\)

instead of

expenditure came to \$2000.00 \(\text{\textit{correct}}\)

• 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 \LaTeX. For example:

\begin{equation}
% 
E = mc^2 
% \end{equation}
13.21 Missing \begin{document}

You have put some text outside of the document environment. Check the following:

- You have remembered \begin{document}
  This error would be caused by, say,
  \begin{verbatim}
  \documentclass[a4paper]{article}
  This is a simple document
  \end{document}
  \end{verbatim} (incorrect)
  instead of
  \begin{verbatim}
  \documentclass[a4paper]{article}
  \begin{document}
  This is a simple document
  \end{document}
  \end{verbatim} (correct)

- You haven’t placed any text before \begin{document}. For example:

  \begin{verbatim}
  \documentclass[a4paper]{article}
  This is a simple document
  \begin{document}
  \end{verbatim} (incorrect)
  instead of
  \begin{verbatim}
  \documentclass[a4paper]{article}
  \begin{document}
  This is a simple document
  \end{document}
  \end{verbatim} (correct)

- You haven’t missed out a backslash from either \documentclass or \begin{document}
  This error would be caused by, say,
  \begin{verbatim}
  documentclass[a4paper]{article}
  \end{verbatim} (incorrect)
  instead of
  \begin{verbatim}
  \documentclass[a4paper]{article}
  \end{verbatim} (incorrect)

13.22 Missing delimiter

You have forgotten to specify the type of delimiter you want (e.g. \( (\) \[[\]\{\}\}) \). (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:
\begin{verbatim}
  f(x) = \left\{
    \begin{array}{ll}
      0 & x \leq 0 \\
      1 & x > 1
    \end{array}
  \right.
\end{verbatim} (incorrect)

instead of
\[ f(x) = \begin{cases} 
0 & x \leq 0 \\
1 & x > 1 
\end{cases} \]
\[ \text{(correct)} \]

13.23 Missing $\textbackslash$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,

\begin{verbatim}
\begin{\texttt{sffamily} \texttt{(incorrect)}
\end{verbatim}

instead of

\begin{verbatim}
\begin{sffamily} \texttt{(correct)}
\end{verbatim}

13.24 Missing $\textbackslash$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 \texttt{eqnarray} environment inside a \texttt{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 \texttt{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 specifier should come first, followed by the height.

If you are using \texttt{\addtocounter} or \texttt{\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 \texttt{\value}. This error can be caused by, say,

\begin{verbatim}
\texttt{\setcounter{exercise}{\texttt{chapter}} \texttt{(incorrect)}}
\end{verbatim}

instead of

\begin{verbatim}
\texttt{\setcounter{exercise}{\value{chapter}} \texttt{(correct)}}
\end{verbatim}

13.26 Paragraph ended before $\textbackslash$begin was complete

You’ve probably missed a closing brace at the end of the argument to \texttt{\begin}. This error will be caused by, say,

\begin{verbatim}
\texttt{\begin{document} \texttt{(incorrect)}}
\end{verbatim}


CHAPTER 13. COMMON ERRORS

13.27 Runaway argument

There are a number of possible causes of this error:

- Paragraph breaks are not permitted in many command arguments. You should use the corresponding environment if possible. For example, this error message will be generated by doing, say,

\begin{document}
\textbf{This is a simple document.}
Here is the first paragraph.
Here is the second paragraph.
\end{document}

(incorrect)

instead of

\begin{bfseries}
This is a simple document.
Here is the first paragraph.
Here is the second paragraph.
\end{bfseries}

(correct)

- 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)
\end{thebibliography}

instead of

\begin{thebibliography}{1}
\bibitem{kopka95} A Guide to \LaTeX2e: document (correct)
\end{thebibliography}

13.28 Something’s wrong—perhaps a missing \texttt{item}.

You may have missed an \texttt{item} command. The first object in a list environment must either be an \texttt{item} command, or another list environment.

This error will be caused by, say,

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

(incorrect)
instead of
\begin{itemize}
\item Animal
\item Vegetable
\item Mineral
\end{itemize} (correct)

This error can also be caused by a missing \bibitem in the bibliography. For example:
\begin{thebibliography}{1}
\bibitem{KOPKA95} A Guide to \LaTeX\document (incorrect)
\end{thebibliography}

instead of
\begin{thebibliography}{1}
\bibitem{kopka95} A Guide to \LaTeX\document (correct)
\end{thebibliography}

13.29 There's no line here to end.

You have placed a line breaking command (\frac, \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.)

  You will get this error if you do, say,

  \texttt{This is a simple \LaTeX\ document} (incorrect)

  instead of

  \texttt{This is a simple \LaTeX\ document} (correct)

- Check that you have remembered the space when typing \ . For example:

  \texttt{This is a simple \LaTeX\document} (incorrect)

  instead of

  \texttt{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

  \texttt{man\oe uvre}

  or

  \texttt{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 \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, 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} (Correct)

instead of

\includegraphics{pictures\shapes.ps} (Incorrect)

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)
Bibliography


I would strongly recommend that you have a look at the \TeX{} Archive [6], particularly the Frequently Asked Questions and the On-Line Catalogue. It’s also a good idea to look at the documentation that was installed with your \TeX{}/\LaTeX{} distribution, which will usually be located in the directory c:\texmf\doc, and the on-line help via the Start Menu:

\begin{verbatim}
Start → Programs → MiKTeX → Help
\end{verbatim}
Index

Page numbers in italic indicate the principle definition.

| \! | 111 |
| \: | 35 |
| \: | 36 |
| \. | 35 |
| \( | 103 |
| \) | 103 |
| \| | 85 |
| \| | 111 |
| \. | 35 |
| \:= | 35 |
| \:= | 35 |
| \. | 102 |
| \. | 36 |
| \/ | 103 |
| \: | 111 |
| \: | 111 |
| \:= | 36 |
| \? | 35 |
| \[ | 7, 103 |
| \# | 34, 78, 114 |
| \# | 35 |
| \$ | 34, 86 |
| \$ | 35 |
| \% | 34 |
| \% | 35 |
| \& | 34, 59, 60 |
| \& | 35 |
| \^ | 34, 89, 92, 93 |
| \^ | 36 |
| \_ | 34, 89, 92, 93 |
| \_ | 35 |
| \{ | 5, 34 |
| \{ | 35, 103 |
| \\ | 103 |
| \\ | 103 |
| \\ | 5, 34 |
| \\ | 35, 103 |
| \\ | 7, 8, 59, 60 |
| \~ | 34 |
| \~ | 36 |
| \~ | 34, 51, 72 |

\] | 7, 103 |
| \\
| \\
| \\
| \\
| \\
| \\
| \AA | 36 |
| \aa | 36 |
| abstract | 44, 46, 59 |
| abstract environment | 46, 46, 58 |
| \abstractname | 84 |
| Acrobat | 11 |
| Acrobat | 11, 19 |
| Acrobat Reader | 2 |
| \addtocounter | 116, 127 |
| \AE | 36 |
| \ae | 36 |
| align environment | 87 |
| \Alph | 117 |
| \alph | 117 |
| \alpha | 90 |
| \alphabet | 100 |
| \appendix | 48 |
| \appendixname | 84 |
| \approx | 99 |
| \arabic | 117 |
| \arccos | 92 |
| \arcsin | 92 |
| \arctan | 92 |
| \arg | 92 |
| \argument | 6 |
| \array environment | 108, 108, 111, 124 |
| \ast | 100 |
| \asymp | 99 |
| \author | 45 |
| auxiliary file (.aux) | 53 |

B

\b | 36 |
\backslash | 103 |
\begin | 9, 104, 127 |
INDEX

em environment 44
em dash — 35
\emph 43, 44
en dash – 35
\end 9, 104
en dash – 35
\environnement 37, 39, 52
enumerate 9, 10, 43
\environnement 9
\environnement 86, 87, 115, 122, 125
\equiv 99
\eta 90
\exp 92
F
ffi 36
ffl 36
fi 36
\figure 71, 72, 115
\figurename 84
fl 36
\fnamesymbol 117
\footnote 33, 115
\footnotesize 44
\forall 101, 110
\frac 95
\framebox 8
\frown 99
G
\Gamma 90
\gamma 90
\langle 103
\LARGE 44
Large environment 43
\Large 44
\large 44
\LaTeX 5, 33, 34
\LaTeX{} 11, 19, 30, 68
\H 36
\hom 92
\hookleftarrow 100
\hookrightarrow 100
\hspace 73
\Huge 44
\huge 44
\hyphen- 35
I
\i 35, 35
\in 99
in-line maths 85
\includegraphics 65, 71
\index 79
\indexname 84
\inf 92, 92
\int 100
\iot 90
\item 9, 37, 41, 54, 124
\itemize 9, 37, 39, 41, 82, 124
\its 43, 44
\itsenvironment 43, 44, 112
J
\j 35, 35
K
\kappa 90
\ker 92
L
\Lambda 90
\lambda 90
\langle 103
\LARGE 44
\large 44
\LaTeX{} 11, 19, 30, 68
\LaTeX{} 5, 33, 34
\le 99
\lceil 103
\ldots 35, 101
\ldots 35, 101
\left 102, 102
\right 102, 102
\leftarrow 100
\Leftarrow 100
\leftharpoonup 100
\Leftrightarrow 100
\leftarrow 100
\leftharpoonup 100
\Leftarrow 100
\leftharpoonup 100
\Leftrightarrow 100
\leftharpoonup 100
\Leftrightarrow 100
\leftharpoonup 100
\Leftrightarrow 100
lengths 118
\leq 99
\lfloor 103
\lg 92
\lim 92, 92, 93
\liminf 92, 92
\limsup 92, 92
\listfigurename 84
\listoffigures 72
\listoftables 75
\listtablename 84
\ll 100
\ln 92
\log 92
\Longleftarrow 100
\longleftarrow 100
\Longleftrightarrow 100
\longleftrightarrow 100
\longmapsto 100
\Longrightarrow 100
\longrightarrow 100
M
\makeindex 79
makeindex 80, 81
\maketitle 45, 46, 49
mandatory argument see argument, mandatory
\mapsto 100
\markboth 58
\markright 59
math environment 85, 125
\mathbb 89
\mathbf 88
\mathcal 88
\mathfrak 89
\mathit 88
\mathrm 88
\mathsf 88
\mathtt 88
\max 92, 92
\mdseries 43
\medspace 111
\mid 99
MiKTeX 5, 11, 69
MiKTeX Options 69
\min 92, 92, 93
minipage environment 127
mktextlsr 69
\models 99
\mp 100
MS-DOS Prompt 12, 19, 28, 68, 69, 80, 120
\mu 90
\multicolumn 61, 61, 62, 125
\textbackslash N
\nearrow 100
\negmedspace 111
\negthickspace 111
\negthinspace 111
\neq 99
\newcommand 77, 78, 81, 82, 114
\newcounter 115, 117
\newenvironment 112
\ni 99
\noindent 113
\normalfont 43
\normalsize 44
\not 99
\notepad 12, 19, 28, 120
\notin 99
\nu 90
\nvarrow 100
O
\O 36
\o 36
\odot 100
\OE 36
\oe 35, 36
\oint 100
\Omega 90
\omega 90
\ominus 100
\operatorname 93
\operatornamewithlimits 93, 93
\oplus 100
optional argument see argument, optional
\oslash 100
\otimes 100
output file see DVI (.dvi)
P
\P 35
packages (.sty) 64
amsfonts............88, 89, 110
amsmath..............88, 89, 93
color..................79
colorspace.............63
datetimelabel........68–70
<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\small</td>
<td>44</td>
</tr>
<tr>
<td>\smile</td>
<td>99</td>
</tr>
<tr>
<td>source code</td>
<td>5, 11, 31–33</td>
</tr>
<tr>
<td>\sp</td>
<td>89, 89</td>
</tr>
<tr>
<td>spaces</td>
<td>32</td>
</tr>
<tr>
<td>\sqcap</td>
<td>100</td>
</tr>
<tr>
<td>\sqcup</td>
<td>100</td>
</tr>
<tr>
<td>\sqrt</td>
<td>97</td>
</tr>
<tr>
<td>\sqsubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\sqsupseteq</td>
<td>99</td>
</tr>
<tr>
<td>\SS</td>
<td>36</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\sup</td>
<td>92, 92</td>
</tr>
<tr>
<td>\supset</td>
<td>99</td>
</tr>
<tr>
<td>\supseteq</td>
<td>99</td>
</tr>
<tr>
<td>\swarrow</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>\textregistered</td>
<td>35</td>
</tr>
<tr>
<td>\textsc</td>
<td>43</td>
</tr>
<tr>
<td>\textsf</td>
<td>43</td>
</tr>
<tr>
<td>\textsl</td>
<td>43</td>
</tr>
<tr>
<td>\texttrademark</td>
<td>35</td>
</tr>
<tr>
<td>\thesetcounter</td>
<td>116, 116</td>
</tr>
<tr>
<td>\subsection</td>
<td>47</td>
</tr>
<tr>
<td>\subparagraph</td>
<td>47</td>
</tr>
<tr>
<td>\subset</td>
<td>99</td>
</tr>
<tr>
<td>\thesubseteq</td>
<td>99</td>
</tr>
<tr>
<td>\thesubsubsection</td>
<td>47</td>
</tr>
<tr>
<td>\succ</td>
<td>99</td>
</tr>
<tr>
<td>\succseq</td>
<td>99</td>
</tr>
<tr>
<td>\sum</td>
<td>100</td>
</tr>
<tr>
<td>\textup</td>
<td>43</td>
</tr>
<tr>
<td>\textwidth</td>
<td>119</td>
</tr>
<tr>
<td>\textshape</td>
<td>43</td>
</tr>
<tr>
<td>\textnormal</td>
<td>43</td>
</tr>
<tr>
<td>Command</td>
<td>Page(s)</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>\varsigma</td>
<td>90</td>
</tr>
<tr>
<td>\vartheta</td>
<td>89, 90</td>
</tr>
<tr>
<td>\vdash</td>
<td>99</td>
</tr>
<tr>
<td>\vdots</td>
<td>101</td>
</tr>
<tr>
<td>\vec</td>
<td>109, 109, 110</td>
</tr>
<tr>
<td>\vee</td>
<td>100</td>
</tr>
<tr>
<td>\vspace</td>
<td>75, 114</td>
</tr>
</tbody>
</table>

W

\wedge 100
\WinEdt 12, 19, 28, 33, 68, 81, 120
\wr 100

X

\xdvi 5
\Xi 90
\xi 90

Y

\YAP 5, 11–13
\yen 35

Z

\zeta 90