# Document Preparation with LaTeX

(Originally edited by David Budgen)
Edited by Sam Nelson
Department of Computing Science
University of Stirling

February 1990, Revised October 1990

# Contents

# Chapter 1

# Introducing LaTeX

## 1.1 What is LaTeX?

During the 1980's, one of the major changes that occurred in computing was the development of highly versatile output devices. Unlike earlier impact printers, the high-resolution matrix printers and laser printers are capable of printing a wide range of typefaces, fonts and sizes (these terms will be clarified later). LaTeX is one of the software tools that has been developed to allow the user to take full advantage of the power of such printing devices.

The popularity of *word processing* has led to the development of many software tools which are based upon *wysiwyg* techniques (what you see is what you get). While such systems can be excellent when used with fixed-pitch typewriter style printing forms, the results when used with mixed fonts and mixed typefaces are often far less impressive. This is largely because over the decades and centuries typesetters and compositors have developed a set of 'rules' which embody hard-won 'knowledge' about the effective use of such tools, and we have become used to seeing books and document laid out to these high standards.

Another reason for the lack of 'perfect' *wysiwyg* packages is that while printer technology involving resolutions in hundreds of pixels per inch is standard, display screen technology is still limited to less than 100 pixels per inch, except in very specialised (and very expensive) areas. Thus, it is technologically impossible to achieve true *wysiwyg* effects, unless the effects required are very limited in scope (boxes-and-lines diagrams or fixed-width fonts, say).

LaTeX is based upon an entirely different philosophy to that of *wysiwyg* tools. The idea in LaTeX is that the designer of a document should specify their layout requirement in an *abstract* manner, and that the program should then translate these into the necessary details of typeface, font and size, making use of a set of rules of 'style' that have been derived from type-setting experience. So the user of LaTeX is concerned only with specifying the *logical* design of their document in terms of chapters, sections, lists etc, rather than being concerned with physical layout.

The effect of this approach is that the document producer controls the appearance of the document indirectly, through a series of encodings which describe to the document processing package how the document should look. These descriptions take the form of ordinary text files produced with any ordinary text editor; indeed, the whole armoury of text-processing utilities may be used to 'attack' LaTeX source files, which can lead to useful short cuts, as will be seen later.

## 1.2 How it works

LaTeX is what is termed a 'mark up' language. The input to LaTeX consists of the raw text of a document, interspersed with *directives* that indicate how each part of a document is to be processed. LaTeX supplies a generous set of structures, as well as the means of adjusting some of their parameters where necessary. Overall, the effect is very like that of compiling a program.

The output from LaTeX is not immediately printable. While a number of files of information may be produced (the exact number depending upon the options selected), the main textual output is a *device-independent* file, usually given the extension `.dvi`. This file needs to be further processed so that it can be displayed on a screen or printed on a particular printer. A wide range of programs to performs the translation are available. For example, tools are available to transform `.dvi` files into line-printable output for cheap proof-reading purposes (highly recommended and environmentally sound), various different programs are available locally to preview output approximating the page display on a workstation screen (depending on the exact type of your workstation), and at least two programs are locally available to transform `.dvi` file into PostScript[1] for output to a laser printer for 'fair copy'.

LaTeX is itself built upon Donald Knuth's TeX typesetting language. TeX is enormously powerful, but writing in TeX is rather akin to writing programs in assembler—and is *not* recommended for the inexperienced. Because LaTeX is implemented as a set of style macros for TeX we occasionally become aware of its presence when errors occur, since some of the error messages may be generated from TeX rather than from LaTeX. Neither give particularly clear messages, but those from TeX can be particularly obscure!

## 1.3 This document

The purpose of this document is to support the 'no frills' use of LaTeX within the Department. The various chapters give basic guides to LaTeX document commands, running the available tools, and organisation of a large document, such as a dissertation. It provides hints and guidelines and has been assembled from a wide range of sources.

As a general point though, if you cannot find out how to obtain a particular effect with LaTeX fairly easily, don't waste your time, simply find a different form of expression. LaTeX discourages the production of output forms that are regarded as stylistically undesirable, and it is generally better to accept that in these matters 'LaTeX knows best'. It is probably only too true!

## 1.4 Other tools

UNIX does provide some useful tools that can be used with any form of document preparation. In particular, everyone should know about the following ones.

**wc** will count words and lines in a file.

**spell** will check the spelling of words in a file and provide a list of errors.

[1] PostScript is an example of a *Page Description Language*, a shorthand method for describing the layout of every pixel on a printed page.

We particularly recommend the use of the latter...

One other point ought to be mentioned here: laser printers are relatively slow and are expensive to run. Draft sections of a document need not be typeset simply for the purposes of proof-reading, and may simply be printed out on the standard line-printer. Please avoid laserprinting your output whenever possible.

## 1.5 Typography and other issues

### 1.5.1 Typographic terms

Most of us use these very casually (and erroneously). As an instant guide to correct use of someone else's technical terms, we offer the list below.

**typeface** is an 'abstract design idea for how letters are to be presented'. Examples of typefaces are Times New Roman, Helvetica and Baskerville. A typeface can be realised in various sizes and *fonts* (see below).

**font** describes a particular aspect of a typeface, often in a particular size. Examples of fonts are **bold**, roman, *italic*, *slanted* and condensed.

**point** is a printer's measure of size. A point can be taken as being approximately 1/72 of an inch. Font size is measured in points; for example, this document is set in 11-point type.

**serif** is a small lateral extension at the end of a stroke. These are found in various forms in many typefaces, and are considered to be an aid to faster reading of a document, since they help to create the imaginary line followed by the eye as a line of text is read. Most books (and this guide) are set in typefaces that have serifs.

For a fuller description of typographical terms, the book by Rubinstein described in the next section is particularly recommended.

### 1.5.2 Other reading

There is a short bibliography at the end of this document. The 'bible' of LaTeX was written by Leslie Lamport, who was its creator [Lamport86]. There is a copy in the library and various copies are owned by members of the Department (who are generally *very* reluctant to lend them out). It is not particularly well organised as a book, *but*, there is, at the time of writing, no alternative. For dissertation preparation these notes should suffice, although reference to Lamport's book may prove necessary on occasion.

Anyone wanting to know more about digital type-setting, and about typography in general, should consult the book *Digital Typography* by Richard Rubinstein [Rubinstein88]. This is a very interesting and well-written book that is saturated with useful references and gives lots of fascinating ideas.

TeX itself is thoroughly documented in the works by Donald Knuth, but these are NOT recommended as suitable reading for the novice.

# Chapter 2

# A Simple Example

The aim of this chapter is to give a brief example of the LaTeX structures and the way that LaTeX documents are organised. It should provide enough information for a user to be able to produce a relatively simple document and contains a small example of such a document in the form of the abstract for a larger paper. Chapter 3 in turn provides a much fuller description and one that is more geared to the production of dissertations. Chapter 9 describes a number of options, and gives some practical hints to help with document organisation and production.

## 2.1 General structure of a document

A LaTeX document has a structure which is somewhat like that of a program written in a block-structured language such as Pascal. The general form is as follows:

**Style Declarations.** These commands inform LaTeX about the way that the document is to be formatted and laid out on the page. The very first command must be a `\documentstyle` command which identifies which of the general document forms this is to be, namely `article`, `report`, `letter` or `book`.

(You should note that all LaTeX commands begin with a backslash '`\`' character to distinguish them. Some are placed on separate lines, others may be embedded in the text.)

Other style declarations are optional, and may be used to determine such forms as page numbering (arabic, roman, large roman etc), and to modify the values of particular parameters used by LaTeX to format the output, such as line spacing.

**Headers.** These are optional, and are generally used to format a document title block.

**Document Body.** This is the main body of the document, and is delimited by the commands `\begin{document}` and `\end{document}`. These effectively determine the *scope* of the LaTeX commands.

The main body of the document can have quite complex internal structures. In particular, it can be segmented into chapters (not for `article`), sections, subsections—all of which will be automatically numbered. (The numbering mechanism can be easily suppressed if required by appending an asterisk to the end of the section command.) Paragraphs are separated by an empty line in the source text. LaTeX provides a number of *list-making* structures for bulleted lists, numbered lists and highlighted lists, as well as the means of creating one's own forms! There are also font-changing commands that can be used to **embolden** or *italicise* words and phrases. Fuller descriptions of these features are given in Chapter 3.

## 2.2 Command formats

As already noted, LaTeX commands begin with a backslash character, '\'. Command arguments are enclosed in curly braces, and these are sometimes used to delimit the scope of more local options such as font changes.

The scope of an environment is normally delimited by using the form:

```
\begin{..environmentform..}
....
\end{..environmentform..}
```

and we have already seen an example of the use of this form with the `\begin{document}` and `\end{document}` commands. Other environments include `itemize` for bulleted lists, `enumerate` for numbered lists, `quote` for indented quotation blocks and so on. In general these forms can be nested without any problems.

More local options such as font changes can be denoted by a short form as in `{\bf ...}` to embolden the enclosed character string, and `{\em ...}` to italicise it.

## 2.3 A simple example

The example used below is based upon a simple technical article or paper. Figure 2.1 shows the resulting formatted output that will be generated from LaTeXwhen this is processed. For brevity, the bodies of some of the paragraphs have been abbreviated in the source text listing that follows:

```
% comment lines begin with the percent symbol and are ignored by
% LaTeX and TeX.  LaTeX commands begin with a back-slash.

\documentstyle[11pt,a4]{article}

\author{David Budgen \\ University of Stirling \\ Scotland}
\title{On A Use of Metrics for the Assessment of Software Designs}

\begin{document}     % this begins the actual document body
\maketitle           % this actually creates the title block

\begin{abstract}
The following description was an outline of a paper being submitted
for a conference.  It summarises an item of work, and gives the
supporting technical references.
\end{abstract}

\section{Introduction}
```

```
The work described here was performed as part of an Alvey-funded research
project (MDSE -- MASCOT Design Support Environment).  The project is
investigating ways of enabling a designer to produce and assess a design
using the MASCOT representation \cite{mascot}.

\section{Metrics for Design}

The lack of metrics that can be used to quantify the features of software
designs can be attributed partly to the lack of any standard design
representation.
   ......
This therefore makes MASCOT a widely-available ‘standard’ that
can be used to provide the basis for the work of this project.

The use of metrics for assessing the quality of code is quite
well-established \cite{conte}, and the value of many of the better-known
   ......
common design representation.

\section{Experimental Work}

The work described in this paper has taken a rather different approach,
similar to the ‘top-down’ approach used by Troy and Zweben \cite{troy};
   ......
that is more suited to a quantitative assessment.

We begin by selecting a set of {\em design principles}, which are global
descriptions of the properties of a design that are believed important.
We then derive a set of {\em general attributes} that reflect these
principles, and further decompose these into {\em specific attributes}.
Figure 1 shows the proposed hierarchy of principles and attributes.

The next step is MASCOT-specific, and involves identifying the
MASCOT design entities that will be expected to exhibit the
specific attributes, and from this, the measurable features that will
provide quantitative assessments of these attributes.

We are currently conducting a series of ‘design experiments’ within the
    ......
of the {\em design principles}.  The preliminary results from these will
be presented in this paper.

\begin{thebibliography}{6}

\bibitem{mascot} H R Simpson and K Jackson, {\em Process Synchronisation
in MASCOT}, The Computer Journal, {\bf 22}, 332, 1979
```

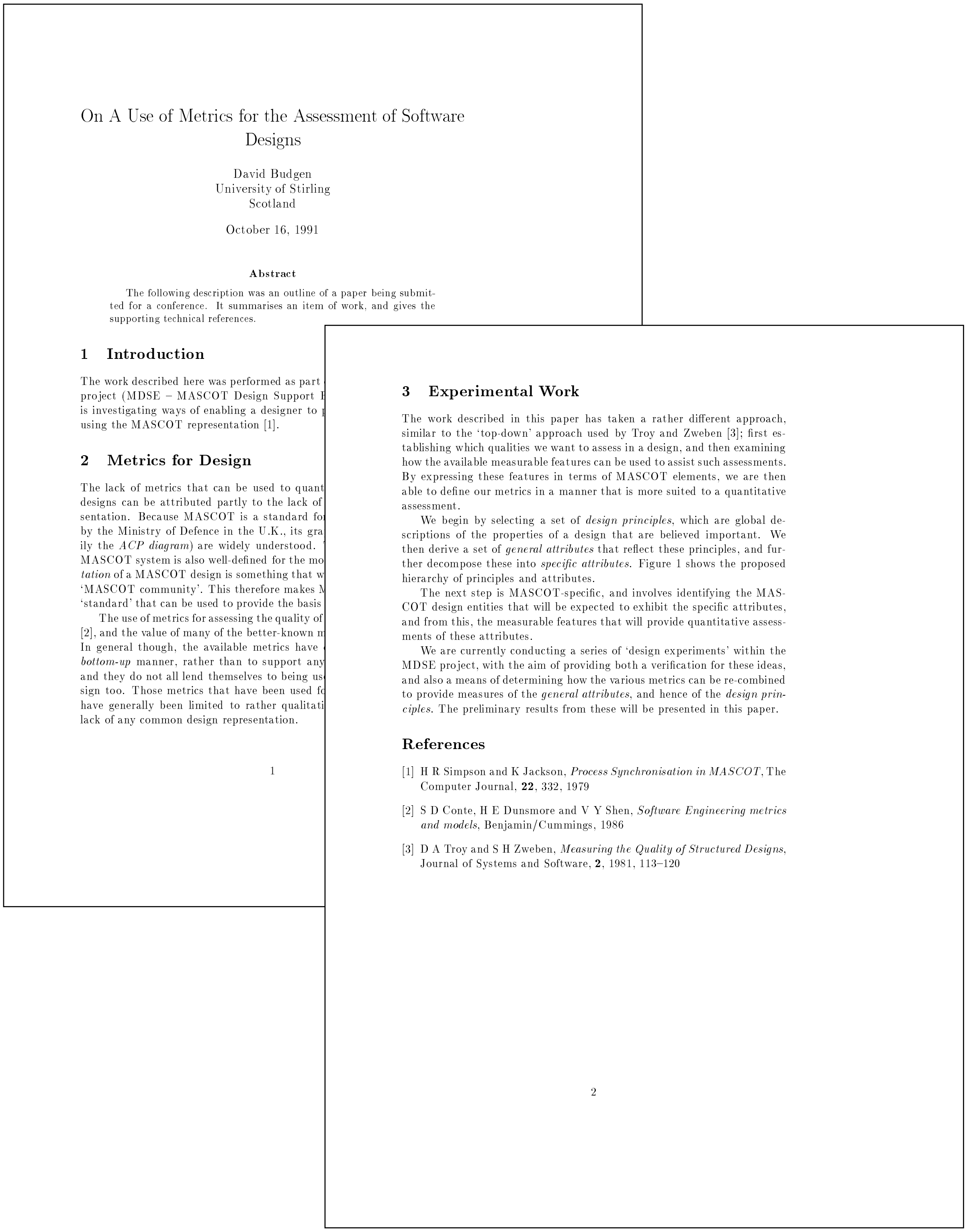
```
\bibitem{conte} S D Conte, H E Dunsmore and V Y Shen, {\em Software
Engineering metrics and models}, Benjamin/Cummings, 1986

\bibitem{troy}  D A Troy and S H Zweben, {\em Measuring the Quality of
Structured Designs}, Journal of Systems and Software, {\bf 2}, 1981, 113--120

\end{thebibliography}
\end{document}
```

This example contains a number of additional forms, most of which should be self-evident. In particular the combination `\\` forces a line break at that particular point in the text.

You might note that LaTeX cannot always produce perfect line breaks, although it does its very best using strict hyphenation rules. The first line of the second paragraph of §2 is slightly longer than the other lines, and the following extract from the run-time output produced by LaTeX shows how this is reported.

```
This is TeX, C Version 2.9 (preloaded format=lplain 89.3.20)  25 OCT 1990 09:48
**&/usr/lib/tex/fmt/lplain dbexample
(dbexample.tex
LaTeX Version 2.09 <25 Jan 1988>
(/usr/lib/tex/inputs/article.sty
Document Style 'article' <5 Feb 88>.
(/usr/lib/tex/inputs/art11.sty)
) (/usr/lib/tex/inputs/a4.sty) (dbexample.aux)
Overfull \hbox (4.05452pt too wide) in paragraph at lines 38--47
[]\elvrm The use of met-rics for as-sess-ing the qual-ity of code is quite well
-established
] [2] (dbexample.aux)
Output written on dbexample.dvi (2 pages, 5532 bytes).
```

This is a relatively infrequent problem, although it becomes a more severe one if you use narrow page widths (using two columns leads to a significant increase in such lines). It is rarely significant in terms of the usefulness of the output!



On A Use of Metrics for the Assessment of Software Designs

David Budgen
University of Stirling
Scotland

October 16, 1991

**Abstract**

The following description was an outline of a paper being submitted for a conference. It summarises an item of work, and gives the supporting technical references.

### 1 Introduction

The work described here was performed as part
project (MDSE – MASCOT Design Support
is investigating ways of enabling a designer to
using the MASCOT representation [1].

### 2 Metrics for Design

The lack of metrics that can be used to quant
designs can be attributed partly to the lack of
sentation. Because MASCOT is a standard fo
by the Ministry of Defence in the U.K., its gra
ily the *ACP diagram*) are widely understood.
MASCOT system is also well-defined for the mo
*tation* of a MASCOT design is something that w
'MASCOT community'. This therefore makes
'standard' that can be used to provide the basis

The use of metrics for assessing the quality of
[2], and the value of many of the better-known
In general though, the available metrics have
*bottom-up* manner, rather than to support any
and they do not all lend themselves to being us
sign too. Those metrics that have been used f
have generally been limited to rather qualitati
lack of any common design representation.

1

### 3 Experimental Work

The work described in this paper has taken a rather different approach, similar to the 'top-down' approach used by Troy and Zweben [3]; first establishing which qualities we want to assess in a design, and then examining how the available measurable features can be used to assist such assessments. By expressing these features in terms of MASCOT elements, we are then able to define our metrics in a manner that is more suited to a quantitative assessment.

We begin by selecting a set of *design principles*, which are global descriptions of the properties of a design that are believed important. We then derive a set of *general attributes* that reflect these principles, and further decompose these into *specific attributes*. Figure 1 shows the proposed hierarchy of principles and attributes.

The next step is MASCOT-specific, and involves identifying the MASCOT design entities that will be expected to exhibit the specific attributes, and from this, the measurable features that will provide quantitative assessments of these attributes.

We are currently conducting a series of 'design experiments' within the MDSE project, with the aim of providing both a verification for these ideas, and also a means of determining how the various metrics can be re-combined to provide measures of the *general attributes*, and hence of the *design principles*. The preliminary results from these will be presented in this paper.

### References

[1] H R Simpson and K Jackson, *Process Synchronisation in MASCOT*, The Computer Journal, **22**, 332, 1979

[2] S D Conte, H E Dunsmore and V Y Shen, *Software Engineering metrics and models*, Benjamin/Cummings, 1986

[3] D A Troy and S H Zweben, *Measuring the Quality of Structured Designs*, Journal of Systems and Software, **2**, 1981, 113–120

2

Figure 2.1: The formatted output

# Chapter 3

# LaTeX Structures

## 3.1 Introduction

This document was produced with the intention of assisting students in the preparation of their dissertations using LaTeX. The document works on two levels. Firstly it provides a short introduction to LaTeX. Secondly the text files from which it is produced form a set of templates you can use in creating your own LaTeX files. This chapter is also set using an increased interline spacing, as would be appropriate for a dissertation.

LaTeX is a collection of typesetting macros designed to make Donald Knuth's TeX [Knuth86] typesetting program more accessible. LaTeX is a powerful tool. This document only deals those aspects which seem to be essential in the preparation of a dissertation. For a comprehensive presentation the reader is referred to the LaTeX manual [Lamport86].

## 3.2 Organizing your text

Input to LaTeX is in the form of a text file. LaTeX input files have a `.tex` extension. The logical structure of a document is imposed by typesetting commands. It is good practice to split your text into logical units, each stored in a separate file, (see Chapter 4). Co-ordination is achieved through a *root* input file which specifies the document style and 'pulls in' the relevant text files. This document was produced with the following *root* file:

```
\documentstyle[11pt,a4wide,twoside]{report}

\pagestyle{headings}

\newcommand{\dspaceon}{\renewcommand{\baselinestretch}{1.3}\large\normalsize}
\newcommand{\dspaceoff}{\renewcommand{\baselinestretch}{1}\large\normalsize}

\title{Document Preparation with \LaTeX}
\author{Edited by David Budgen \\Department of Computing Science \\
University of Stirling}
\date{February 1990, Revised October 1990}

\begin{document}

\pagenumbering{roman}

\maketitle

\input{acks}
\tableofcontents

% The chapters as available

\include{intro}
\include{example}
\include{miniguide}
\include{documents}
\include{running}
\include{dvipsguide}
\include{previewer}
\include{graphics}
\include{hints}

\include{biblio}

\end{document}
```

A detailed description of this file follows. Firstly, the command

```
\documentstyle[11pt,a4wide,twoside]{report}
```

specifies the basic format of the document – this effects the layout of your text on the page, the size of headers, etc. In general LATEX commands are prefixed by a **** (backslash) character. LATEX provides a number of standard styles such as `report`, `article`, and `book`. Here we opt for the standard report style (`report`). Additional style options are selected within `[...]` brackets. Here the `11pt` option specifies the point size to be used throughout the document. A point is a measure of type size – `1pt` is approximately $\frac{1}{72}$ of an inch. Other sizes are 10pt (the default) and 12pt. The `a4wide` option enlarges the printing area on the page to fit better on to A4-sized paper, while the `twoside` option organises page margins for two-sided printing. This last option would not be appropriate for a thesis. (For many purposes the `a4wide` option produces a page that is really too full and with text that occupies too wide a line. The `a4` option may be much more suitable for articles, essays etc, although `a4wide` is probably more appropriate for the denser material of a dissertation!)

Comment lines begin with the `%` character, and are terminated by the end of a line. Comments are mostly useful in 'root' files such as this one, where they can be used to remind the author of items to be included later, or to exclude sections that have not yet been completed.

The commands used to specify these entities illustrate the notion of scope associated with certain commands known as declarations. Braces are used to delimit the scope of a declaration. For example, here is some *emphasised text* produced by the `\em` command. The associated LaTeX input takes the form:

```
For example, here is some {\em emphasised text} produced by ...
```

The scope of a declaration can also be delimited by the `\begin` and `\end` commands. For instance, consider *“some more emphasised text”* which was produced by the following fragment of LaTeX:

```
\begin{em}
``some more emphasised text''
\end{em}
```

Such a construct is called an environment, the name of the environment is specified within braces. Note that the `\em` declaration corresponds to the `em` environment. Environments are best used where a large section of text is involved. The largest text environment is the complete document and is delimited as follows:

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

The `\pagestyle` declaration controls the text printed in the header and footer regions of the page. The command `\pagestyle{plain}` generates an empty header and places the page number in the footer, whereas `\pagestyle{empty}` generates an empty header and footer. Roman and arabic numbering is achieved by the `\pagenumbering` declarations.

LATEX keeps track of various kinds of referencing information enabling the automatic generation of a table of contents by the `\tableofcontents` command. Similarly, lists of figures and tables can also be generated for free.

The `\include{intro}` command causes LATEX to include the file `intro.tex`. In this way the text of your document can be held in different files. In practice you will only work on one file (chapter) at a time. A mechanism to prevent the inclusion of particular files is therefore required. Two approaches are available. Firstly, as already mentioned, any line of the input file starting with a `%` symbol is ignored by LATEX. For instance, the following sequence of includes would result in only "chapter2" being included:

```
%\include{chapter1}
\include{chapter2}
%\include{chapter3}
     .
     .
     .
```

Alternatively, the `\includeonly` command enables you to specify within the preamble which of the include files should be processed. For instance, the inclusion of only "chapter2" can also be achieved by the following use of the `\includeonly` command:

```
\includeonly{chapter2}
     .
     .
     .
\include{chapter1}
\include{chapter2}
\include{chapter3}
     .
     .
     .
```

Finally, the `\appendix` command tells LATEX to treat the text which follows as appendices. This basically means that sectional units are labelled using letters.

## 3.3 Sectioning

LATEX provides a number of sectioning commands. Chapters are constructed by the `\chapter` command. The heading of this chapter was generated by the command

```
\chapter{\LaTeX\ Structures}
```

LaTeX takes care of chapter numbering automatically. A chapter is terminated either when another `\chapter` request is processed or when the `\end{document}` command is reached. Sectioning within a chapter is produced by the commands:

`\section`

`\subsection`

`\subsubsection`

For example this section was produced by the command:

```
\section{Sectioning}
```

All sectioning commands result in a new paragraph being started. One or more blank lines denote the end of a paragraph.

## 3.4 Displayed material

LaTeX provides a number of standard environments for displaying text, the most common ones are presented here.

### 3.4.1 Quotations

The **quote** environment is used to produce quotations, for example:

> "An opening quotation is a useful device, especially if it shows the author to be a broadly cultured Man (or Woman) of Science rather than a narrowly ignorant philistine. Newton, again, is good, particularly if he includes a few quaint words with initial capitals. Quotations from Ancient Greeks, if written in Greek characters without a translation, are highly sophisticated. Biblical quotes are considered rather tasteless nowadays, and *The Hitch-Hiker's Guide to the Galaxy* is passé. The *Rubaiyyat* of Omar Khayyam is forbidden."

*(Jon Stoney, New Scientist 20/27 December 1984)*

### 3.4.2 Verbatim

To prevent a block of lines from being re-arranged by LaTeX the `verbatim` environment should be used. You might use this to show a piece of code, for example

```
member(X,[X|_]).
member(X,[_|Y]):-
      member(X,Y).
```

The `dspaceoff\dspaceon` switch is used to obtain single spacing in such instances.

### 3.4.3 Lists

LaTeX provides three environments for creating lists: `itemize`, `enumerate` and `description`, for example:

- bullets are produced by the `itemize` environment
- and are useful when you wish to present a list of items
- which has no order.

The proceeding list was produced by the following LaTeX input:

```
\begin{itemize}
\item bullets are produced by the \verb+itemize+ environment
\item and are useful when you wish to present a list of items
\item which has no order.
\end{itemize}
```

Numbered items are produced by the `enumerate` environment

1. Tweedledum does not exist.
2. Tweedledee does not exist.
3. At least one of these sentences is false.

The proceeding list was produced by the following LaTeX input:

```
\begin{enumerate}
\item Tweedledum does not exist.
\item Tweedledee does not exist.
\item At least one of these sentences is false.
\end{enumerate}
```

while a list of labelled items is generated by the `description` environment

**grumble** to complain in a bad-tempered way.

**grumpy** bad-tempered.

**grunt** to make the gruff snorting sound characteristic of a pig.

The proceeding list was produced by the following LaTeX input:

```
begin{description}
\item[grumble]  to complain in a bad-tempered way.
\item[grumpy]  bad-tempered.
\item[grunt]  to make the gruff snorting sound characteristic of a
pig.
\end{description}
```

List-making environments may be nested (up to four levels of any type — which is too many). When nesting `itemize` environments, different symbols are used to begin each item, and when nesting `enumerate` environments, different forms of counter are used.

### 3.4.4 Formulae

LaTeX is particularly good for producing mathematical formulas, using the `math` environment which is delimited by `$` characters. Space limits all but a few basic examples.

This example of an inline formula $\sqrt{x+y}$ was produced by the following fragment of LaTeX:

```
This example of an inline formula: $\sqrt{x+y}$ was produced by
the following ...
```

While a formula displayed on a separate line, such as

$$\sqrt{x+y}$$

is achieved by the following fragment of LaTeX

```
While a formula displayed on a separate line, such as
\[ \sqrt{x+y} \]
is achieved by the following ...
```

The `displaymath` environment has the same effect as the `\[...\]` delimiters. Numbered equations are produced by the `equation` environment

$$(\forall x : N)(\forall y : N)(x = y) \vee (x \neq y) \tag{3.1}$$

Formulas which are too long to fit on a single line are handled by the `eqnarray` environment. See the LaTeX manual for more details.

## 3.5 Changing fonts

A font is a particular size and style of type, some examples follow:

**bf** **This bold face type** is produced by: `{\bf This bold face type}`

**em** *This emphasised type* is produced by: `{\em This emphasised type}`

**tt** `This typewriter type` is produced by: `{\tt This typewriter type}`

**sl** *This slanted type* is produced by: `{\sl This slanted type}`

**sc** THIS SMALL CAPS TYPE is produced by: `{\sc This small caps type}`

## 3.6 Footnotes

Footnotes[1] are generated by the `\footnote` command. The preceding footnote reference was produced by the following fragment of LaTeX:

```
Footnotes\footnote{Defn: A note printed at the bottom of the page.}
are generated by the ...
```

Note that there is no space between "Footnotes" and the `\footnote` command.

## 3.7 Figures and tables

The `figure` environment is usually used for the presentation of diagrams and pictures. Tabular information should be packaged up using the table environment. Both figures and tables require titles. These are produced by the `\caption` command. Note that figures and tables are never split across a page, they are said to "float". Figure 3.1 was produced using the `picture` environment while Table 3.1 was produced using the `tabular` environment which allows you to align text within boxed columns. The LaTeX source used to produce Table 3.1 is as follows:

```
\begin{table}
\centering
\begin{tabular}{|l|l|} \hline\hline
 {\em Formatted }  &   {\em Unformatted} \\ \hline\hline
$A \rightarrow B$  &   \verb+$A \rightarrow B$+ \\ \hline
$ \lambda ((x)b)$  &   \verb+$ \lambda((x)b)$+ \\ \hline
$ \neg A $         &   \verb+$\neg A$+ \\ \hline
$ A \vee B$        &   \verb+$A \vee B$+ \\ \hline
```

[1] Defn: A note printed at the bottom of the page.

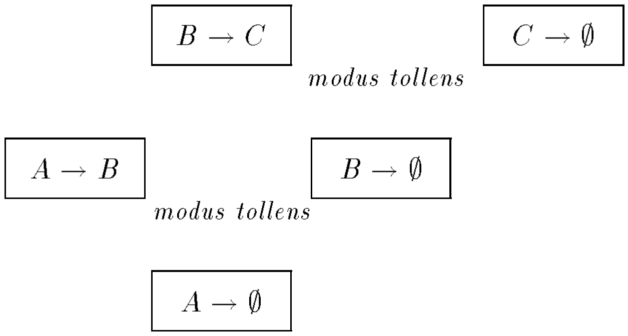



Figure 3.1: Example figure

| *Formatted* | *Unformatted* |
|---|---|
| $A \rightarrow B$ | `$A \rightarrow B$` |
| $\lambda((x)b)$ | `$ \lambda((x)b)$` |
| $\neg A$ | `$\neg A$` |
| $A \vee B$ | `$A \vee B$` |
| $A \wedge B$ | `$A \wedge B$` |

Table 3.1: Example table

```
$ A \wedge B$      &   \verb+$A \wedge B$+ \\ \hline
\end{tabular}
\caption{\label{formulas} Example table}
\end{table}
```

The references to Table 3.1 were produced by the `\ref` command, for example:

```
The references to Table~\ref{formulas} were produced by the
\verb+\ref+ command, for example:
```

The `\ref` command enables relative, as compared to absolute, referencing. A `\ref` command takes a *key* as its argument. The point of reference, the table in this case, is indicated by a corresponding `\label` command which appears within the table environment. For more details on the `tabular` and `picture` environments see the LaTeX manual.

## 3.8 Producing a bibliography

The bibliography list is produced with the `thebibliography` environment. This is a rather special form of list making environment, which will also produce its own heading. A simple example is:

```
\begin{thebibliography}{Rubinsteinetal}

\bibitem[Knuth86]{knuth}
Donald E Knuth, {\em The \TeX book}, Addison-Wesley, 1986,
ISBN 0-201-13447-0

\bibitem[Lamport86]{lamport}
Leslie Lamport, {\em \LaTeX\ A Document Preparation System},
Addison-Wesley, 1986, ISBN  0-201-15790-X

\bibitem[Rubinstein88]{rubinstein}
Richard Rubinstein, {\em Digital Typography: An Introduction to
Type and Composition for Computer System Design}, Addison-Wesley,
1988, ISBN 0-201-17633-5

\end{thebibliography}
```

The argument of the `thebibliography` environment should be a piece of text which is slightly wider than the widest item in the list.

Each `\bibitem` entry must have an associated label that can be referenced from the `\cite` command (see below). It may also have an optional argument preceding this (in square brackets) that is used to define the user's own reference labels that will appear in the document. If this is not used, then as a default LaTeX will produce numerical reference labels, so that in this case the argument for the `thebibliography` environment need only be a short string such as `{100}`.

The bibliography is most easily maintained by storing it in a separate file such as `biblio.tex`. This form is used in the root file example given at the beginning of this chapter.

Citations[2] are produced by the `\cite` command. Associated with the `\cite` command is a label, which is used to provide a reference to an entry in the bibliographic database. As an example, using the above items, the phrase

```
... in the \TeX book \cite{knuth} ...
```

would appear as

...in the TeXbook [Knuth86] ...

[2] Cross-references to publications.

Running LaTeX produces a number of files, one of which, the auxiliary (`.aux`) file, includes cross referencing information. The `.aux` file is also used in the creation of the table of contents (`.toc`), and the lists of figures (`.lof`) and tables (`.lot`). Modifications to your document which effect the table of contents, etc are incorporated the second time around. For example, the `\tableofcontents` command, which appears in the preamble, tells LaTeX to create a new `.toc` file and use the previous one (if any exists) to generate the contents page for inclusion in the current document. The `.toc` file, therefore, is one step out of phase with the document and consequently a second run is required. LaTeX in effect is a two-pass compiler.

## 3.9 Errors

LaTeX throws up errors as it finds them. Typical errors are missing closing brackets and mis-spelt commands. For example, the following error message was produced because of a mis-spelling of the `itemize` environment:

```
[5]
LaTeX error.  See LaTeX manual for explanation.
              Type  H <return>  for immediate help.
! Environment itemise undefined.
\@latexerr ...for immediate help.}\errmessage {#1}
                                                  \endgroup
l.226 \begin{itemise}

?
```

The `?` at the end of the error message indicates that LaTeX has suspended processing. It is possible to introduce a patch dynamically, however, this does not effect the source file which must also be modified. The interested reader is referred to chapter 6 of the LaTeX manual for more details. By simply pressing the *return* key LaTeX is instructed to skip over the error and continue processing the input. Note that a single error may cause many spurious errors. It is sensible to press on and identify as many errors as possible before going back to modify the source file. If you do want to terminate LaTeX before it is finished then type the character `e` followed by the 'return' character in response to the `?` prompt. A log (`.log`) file is generated which records the details of a text processing run and provides a permanent record of error messages.

# Chapter 4

# Structuring a Dissertation

## 4.1 Document Design

LaTeX is designed to be a tool that assists with the *logical* design of documents. Dissertations should also have a logical design, and so this chapter briefly examines the form of this, and discusses ways of mapping this design on to *physical* file structures.

Most dissertations have a structure that broadly approximates to the following:

1. **Preamble:** containing such items as dedication (if desired), acknowledgements etc. Usually less than a page in length.

2. **Introduction:** explaining *what* the project was about; describing the chief objectives; outlining any constraints on the solution; introducing **keywords** and explaining them. (Loosely conforms to ideas of *problem specification*.)

3. **Background:** describing techniques or tools appropriate to the problem and its solution. May be more than one chapter if there are several of these. It may be difficult to determine just how much to include under this on occasion, there is a need to provide enough information for an educated reader to understand the rest of the dissertation, without swamping them with unnecessary details.

4. **Solution:** describing how the problem was tackled by you (*design* plus a bit of the *implementation* information if appropriate). This is your chance to explain *what* you did, and *why* you made particular decisions.

5. **Results:** summarising the experiences of implementing the solution; modifications needed as a result of this; whether the effects were expected, whether the project results met the objectives etc, and if not – why.

6. **Conclusions:** providing a concise summary of what was achieved; how well it met the objectives; any more general observations about these; scope for any further development/extension of the ideas or your solution.

7. **Bibliography:** is an important component. It doesn't need to be large, *but* it should be enough to show that you read around the topic and looked for ideas other than those suggested by your supervisor!

8. **Appendices:** should be included as necessary. These supplement the material of the main chapters by providing the details that would obscure the structure of a chapter, but which would be needed by anyone who wanted to use your work.

The above is a very general framework, and will usually be implemented in some variant form. However, it does provide the start point for dissertation design. A suggested implementation plan is as follows:

1. List an outline of topics along the lines above.
2. Convert this to chapter headings.
3. For each chapter, list the topics to be covered.
4. Convert these to section headings and section content outlines (topic lists).
5. Check through for consistency and omissions.
6. Begin writing!

The order in which sections and chapters are written is a matter of personal preference. Provided that you have a good content plan for each chapter, the ordering of development can be relatively flexible. The rest of this chapter explains how you can *physically* structure your document so as to support this flexibility.

## 4.2 Sectioning into files

### 4.2.1 File hierarchy

LaTeX makes it easy to structure a document as a set of separate files. There are good arguments for making use of this. Among others these include:

- such files are easily maintained if the file structure reflects the document structure;
- sections can be moved within the document, or replaced with new versions, simply by changing links;
- diagrams etc can be developed and tested separately;

Experience with the development of large reports suggests that the following structure is useful during development as well as being easily maintained:

1. A *root* file which contains
   (a) option-setting commands that will have global effect, such as `\pagestyle` and any locally defined commands such as the `\dspaceon` and `\dspaceoff` commands shown in Chapter 3. Also title-setting commands.
   (b) Commands to create the `tableofcontents` and to set the page numbering forms (see later).
   (c) Commands to 'pull in' the text of the chapters.
   (d) Comments to remind the author of tasks to be performed in the future.

Essentially, this file reflects the top-level design of a document.

2. A file for each chapter, that in turn pulls in the files containing the text of each section. Use of this added level of indirection makes it easy to re-organise parts of a document if the original design proves incorrect.

3. A file of text for each *section*.

4. A file for each diagram/table. This allows these to be developed separately, using a temporary root file, so reducing processing time considerably, as these items often need a disproportionate number of re-tries in order to get them right.

The rest of this section describes how this structure can be organised using LaTeX.

### 4.2.2 The `include` and `input` options

LaTeX provides two mechanisms for 'pulling in' text from another file in the middle of reading a source file. Both of them read the file in directly, and differ only in their 'side effects'.

#### The `include` command

This has the effect of starting output on a new page. It also defines a 'logical entity' of the document, which will have its own `.aux` file. We normally use this command in the *root* file in order to pull in the chapter files and any other major units.

A (reduced) example of such a file is:

```
\documentstyle[11pt,a4wide]{report}

\pagestyle{headings}

\newcommand{\dspaceon}{\renewcommand{\baselinestretch}{1.3}\large\normalsize}
\newcommand{\dspaceoff}{\renewcommand{\baselinestretch}{1}\large\normalsize}

\title{My Dissertation}
\author{A Student \\Department of Computing Science \\
University of Stirling}
\date{January 1990}

\begin{document}

\pagenumbering{roman}

\maketitle

% \input{acks}
% \tableofcontents

% The chapters as available
```

```
\include{intro}
\include{chapter1}
% \include{chapter2}
% \include{chapter3}
% \include{chapter4}
% \include{chapter5}
% \include{appendixa}

\include{biblio}

\end{document}
```

#### The input command

This has no side effects in terms of output, and the text file read in is simply treated as though it were part of the outer file. It is normally used for reading in sections of a chapter, diagrams, complex examples etc.

An example of a `chapter` file might be:

```
% Chapter 2 of dissertation, stored in chapter2.tex

\chapter{Software tools}
\label{tools}

\input{sect2_1}
\input{sect2_2}
% \input{sect2_3}
```

The chapter heading is labelled to allow symbolic cross references to the chapter from later sections of the document. In this example, the third section of this chapter has yet to be prepared, and so has been commented out (or has been dropped from the plan!).

### 4.2.3 Page numbering

The default form is to use arabic numbering. If we include a `tableofcontents` in our document, then as the document grows so will the table, and when this spreads to further pages this will alter the page numbering for subsequent pages throughout the document.

A simple solution is to use roman numbers for the pages preceding the first chapter, as was shown in the *root* file used as an example in the previous section. The switch back to arabic numbering must then be made after the start of the first page. So the command

```
\pagenumbering{arabic}
```

must follow either a `\newpage` command at the end of the preamble, or appear after the first `\chapter` command.

## 4.3 Development Strategy

The *root* file is a realisation of the *logical* design of a document. It should therefore be complete when first entered, although the `%` character might need to be used to comment out most of the `\include` commands. By doing this, you ensure that none of the items in your plan get omitted by accident. (An alternative, and maybe better scheme, is to create the chapter and section files too, and to provide short summaries of *intentions* in the section files. This way, all of the document files are created at the start, although most will only have interim contents.)

LaTeX is a large and relatively slow program. Dissertations are typically quite large documents, and so we need a means of reducing the time spent processing text which is essentially fixed and completed. This is particularly important as the document increases in size. The mechanism for doing this is provided by the `\includeonly` command. This directs LaTeX to only process those `\include` entities that are listed in its argument. In doing this, LaTeX will still read the `.aux` files for the omitted entities, in order to get page numbering and cross-reference information, but it will not process the textual parts. This generally speeds up processing very significantly.

(**Note:** If you use the `\includeonly` facility, and have organised the page numbering so that you switch back to arabic numbers at the beginning of the first chapter, you need to keep this chapter in the `\includeonly` list to avoid odd effects with page numbers!)

# Chapter 5

# Generating Bibliographies with BibTEX

## 5.1 Introduction

BibTEX is a program used to generate a list of references (bibliography) from one or more bibliographic database files. BibTEX is quite involved, and although it may not seem worthwhile using it if you are citing only one or two references - it is useful for keeping sources centrally. Once you have started using BibTEX it is simple to maintain your list of references and to keep a note of books or articles that you have read in order to cite them in future documents.

If you would prefer to create your own bibliography, LaTEX provides an environment called **thebibliography** which is similar to the **itemize** environment. This is described in some detail in section 5.1.4.

### 5.1.1 The Bibliographic Database

The first step in using BibTEX is to create or use an existing bibliographic database (a `.bib` file). This file consists of entries of the form:

```
@ENTRYTYPE{key,
  REQUIREDFIELD = "value",
  REQUIREDFIELD = {value},
  OPTIONALFIELD = value,
  IGNOREDFIELD = abbrev }
```

BibTEX supports several entry types, such as BOOK, ARTICLE, PHDTHESIS, and each entry type has an associated list of fields. There are three different classes of fields: **required**, **optional** and **ignored**. **Required** fields must appear in the entry as a minimum, otherwise BibTEX will produce an error message. **Optional** fields will be used in the entry if present but can be omitted. They are used to provide the reader with additional information such as relevant chapters, the address of the publisher etc. **Ignored** fields are regarded as comments by BibTEX and as such are not printed out as part of the bibliography; they are typically used to annotate the entry for your own use.

As an example consider the following entry:

```
@STRING{addwes = "Addison-Wesley Publishing Company"}

@BOOK{latexbook,
  title = "\LaTeX~User's Guide \& Reference Manual",
  author = {Leslie Lamport\},
  publisher = addwes,
  year = 1986,
  note = "A comprehensive guide to preparing documents",
  relevant = "Appendix B covers \BibTeX~in some detail" }
```

In this example, the entry type is **BOOK**; the key is **latexbook**; the required fields are **title, author, publisher** and **year**; **note** is an optional field and **relevant** is an ignored field.

Strings in an entry can either be surrounded by braces or quotes as in the title and author fields above. If a string is used more than once, it is worth defining an abbreviation and using that as the value for a field. Abbreviations are defined by putting a @STRING command in the `.bib` file.

Further entry types and their required and optional fields are given in section 5.2.

### 5.1.2 Using BIBTEX with LATEX

The following commands should be used in your LATEX document to successfully create and refer to your bibliography:

- **bibliographystyle**{*style*} specifies the style of the source list. This must be placed after the **begin**{**document**} command. There are four standard bibliography styles:
  - **unsrt** Entries are sorted according to the order they appear in the document and numbered.
  - **plain** Entries are sorted alphabetically and numbered.
  - **alpha** Entries are sorted alphabetically and entry labels are formed from the author's name, e.g. 'Lam' would be the entry label for the example above.
  - **abbrv** Entries are sorted alphabetically but the first names, month names and journal names are abbreviated.
- The **bibliography** command is used to specify one or more files containing the bibliographic database. For example, the command
  **bibliography**{**myrefs**, **deptrefs**}
  specifies that the source list is to be obtained from the files **myrefs.bib** and **deptrefs.bib**. This command is used where you want to place the bibliography.
- The **cite**[**note**]{**key**, ...} command is used within the document text to refer to one or more entries in the bibliography. The note parameter is optional and if used gives additional information in the body of the text. For example to refer to the LATEX manual above, you could use the command **cite**[**AppendixB**]{**latexbook**}. The entry label [1, Appendix B] will then appear in the text of the document. You can also make use of the **nocite**{**key**, ...} command which puts references specified by the keys in the bibliography, but does not cite them in the document text. This should be used after the **begin**{**document**} command.

When you run `latex` on a document containing citations and bibliography commands, the auxiliary file (`.aux`) will contain cross-referencing information. Note: this will produce warnings stating that citations have been undefined. Running `bibtex` on your LaTeX document will read information from the auxiliary file and create a file with the extension `.bbl`. Any errors in the `.bib` file will be found at this stage and should be corrected before continuing. When `latex` is next run, the `\bibliography` in your document reads the `.bbl` file and generates the bibliography. Running `latex` will probably produce the warning :

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

You should then rerun `latex`.

If you have any problems with the `.bib` file or your LaTeX document, it may be worth deleting the `.aux and .bbl` files (i.e. the files generated by `latex` and `bibtex`) and starting again. Also if you add or remove a citation from your document you must rerun `latex` followed by `bibtex` to generate your bibliography. If you are not satisfied in any way with the output of BibTeX then you can edit the `.bbl` file.

### 5.1.3 Output from BibTeX

Given the example in section 5.1.1, the following output is generated :

[1] Leslie Lamport. LaTeX *User's Guide & Reference Manual.*Addison-Wesley Publishing Company, 1986. A comprehensive guide to preparing documents.

### 5.1.4 The `thebibliography` Environment

The `.bbl` file uses a type of list environment (such as the itemize enviroment), which has the form:

```
\begin{thebibliography}{xx}
\bibitem[label]{key}
...
\bibitem[label]{key}
\end{thebibliography}
```

The **thebibliography** environment has an argument which should be a string at least the length of the widest entry label in the source list. Instead of using the numbers generated by the environment as entry labels, you can specify your own by using an optional argument to bibitem, but remember to increase the length of the argument to the **thebibliography** command. Consider the following examples:

```
\begin{thebibliography}{1}
\bibitem{latexbook}
Leslie Lamport. {\em \LaTeX~User's Guide \& Reference Manual}.
Addison-Wesley Publishing Company, 1986.
\end{thebibliography}
```

```
\begin{thebibliography}{Name YY}
\bibitem[Lam 66]{latexbook}
Leslie Lamport. {\em \LaTeX~User's Guide \& Reference Manual}.
Addison-Wesley Publishing Company, 1986.
\end{thebibliography}
```

In the first example, the entry label is generated by the environment. Note that this will fail if you have more than nine entries, because the parameter to **thebibliography** command is not wide enough. The second example shows the use of your own entry labels.

## 5.2 Format of Bibliographic Entries

### 5.2.1 Text Fields

The bibliography style you choose determines how names, such as titles and authors will be formatted. You should therefore type the complete name and leave the formatting to BIBTEX.

If an entry has more than one author, then an "and" will separate the names; names in braces are considered as one name and will not be stylised by BIBTEX.

According to the bibliography style you choose, a title may be capitalised by BIBTEX (even if you had typed it in in lower case). In general titles of books are capitalised, while titles of articles are not. If you want a specific word or letter to remain capitalised, then enclose it in braces.

### 5.2.2 Entry Types

The following are the entry types allowed in the `.bib` file together with their required and optional fields.

- **article**. Required: **author**, **title**, **journal**, **year**. Optional: **volume**, **number**, **pages**, **month**, **note**.
- **book**. Required: **author** or **editor**, **title**, **publisher**, **year**. Optional: **volume**, **series**, **address**, **edition**, **month**, **note**.
- **booklet**. Required: **title**. Optional: **author**, **howpublished**, **address**, **month**, **year**, **note**. A booklet is a work that is printed and bound, but has no named publisher or sponsor.
- **inbook**. Required: **author** or **editor**, **title**, **chapter** and/or **pages**, **publisher**, **year**. Optional: **volume**, **series**, **address**, **edition**, **month**, **note**.
- **incollection**. Required: **author**, **title**, **booktitle**, **publisher**, **year**. Optional: **editor**, **chapter**, **pages**, **address**, **month**, **note**.
- **inproceedings**. Required: **author**, **title**, **booktitle**, **year**. Optional: **editor**, **pages**, **organization**, **publisher**, **address**, **month**, **note**.
- **manual**. Required: **title**. Optional: **author**, **organization**, **address**, **edition**, **month**, **year**, **note**. The manual entry is used for technical documentation.
- **mastersthesis** Required: **author**, **title**, **school**, **year**. Optional: **address**, **month**, **note**.
- **misc** Required: none; Optional: **author**, **title**, **howpublished**, **month**, **year**, **note**. Use this entry type if nothing else fits.

- **phdthesis**. Required: **author**, **title**, **school**, **year**. Optional: **address**, **month**, **note**.
- **proceedings**. Required: **title**, **year**. Optional: **editor**, **publisher**, **organization**, **address**, **month**, **note**.
- **techreport**. Required: **author**, **title**, **institution**, **year**. Optional: **type**, **number**, **address**, **month**, **note**.
- **unpublished**. Required: **author**, **title**, **note**. Optional: **month**, **year**.

#### Fields

Below is an explanation of some of the fields used in the entries:

- **address**. This is the address of the publisher. If it is a major publisher, then just give the city.
- **chapter**. This should be a number, rather than the name of the chapter.
- **pages**. One or more page numbers, or a range, such as 19–32.
- **note**. Any extra information that will help the reader locate the reference.

## 5.3 Additional Information

This guide is intended to be an introduction to BibTeX , additional information can be found in section 4.3.2 and Appendix B in the LaTeX manual. Also, an example bibliographic database (**xampl.bib**) can be found in /usr/fs/latex-eg. This gives both minimal and full entries for each of the entry types described in section 5.2.2.

# Chapter 6

# Running LaTeX and dvips

Input to LaTeX consists of a collection of source files, conventionally suffixed '`.tex`', arranged into a tree where the links in the tree are implemented by LaTeX `\input{}` and `\include{}` directives. In the case of a simple document, it is perfectly acceptable for this 'tree' to consist of a single file, of course. In examples that follow, it is assumed that the root of the tree is the file `root.tex`.

## 6.1 LaTeX

The tree of files can be given to LaTeX for processing with the command

```
latex root
```

LaTeX takes no command line options at all—all controls associated with the running of the program are presented to the program via the input text file. LaTeX is a very 'chatty' program (it is designed with inter-operating system portability in mind, and this occasionally shows!), but since it generally takes several seconds to complete a run even for a short document, this is no bad thing. The on-screen output for a 25-page document, consisting of about ten source files whose root is `root.tex` will look something like this:

```
This is TeX, C Version 2.9 (no format preloaded)
(root.tex
LaTeX Version 2.09 <25 Jan 1988>
(/usr/lib/tex/inputs/report.sty
Document Style ‘report’ <5 Feb 88>.
(/usr/lib/tex/inputs/rep11.sty) (/usr/lib/tex/inputs/titlepage.sty))
(/usr/lib/tex/inputs/a4wide.sty (/usr/lib/tex/inputs/a4.sty))
No file root.aux.
[0] (abstract.tex [0]) (intro.tex [1] [2] [3] [4]) [5] [6] (method.tex [7]
[8] [9] [10] [11]) [12] (results.tex [13] [14] [15] [16]) [17] (disc.tex
[18] [19]) [20] (conc.tex) [21] (acks.tex) [22] (refs.tex) [23] (root.aux
(abstract.aux) (intro.aux) (method.aux) (results.aux) (disc.aux) (conc.aux)
(acks.aux) (refs.aux))
Output written on root.dvi (25 pages, 80896 bytes).
Transcript written on root.log.
```

Most of this is irrelevant to the casual user of LaTeX, and can safely be treated merely as comforting output to convince the user that LaTeX is actually doing something. In case of problems with documents, however, it is generally useful to have an idea what is going on.

First, TeX and LaTeX announce themselves (remember that LaTeX is just TeX with a somewhat more usable 'front-end' bolted on) and then LaTeX uses the `\documentstyle{}` directive from the top of the root file to choose which style files to read, and reports on the list it chose. Next, it works its way through the tree of files which make up the document, laying out pages as it goes. Each file is announced as it is encountered, and output of each page is reported as it occurs by the appearance of the page number in square brackets. The output from LaTeX is sent to `root.dvi`. As LaTeX works its way through the document, it makes itself 'notes' about the positions of various elements such as figures, section headings, page counts, etc., in order to ease reprocessing and resolution of forward label references later. These are saved in the various '`.aux`' files mentioned in the commentary. The report '`No file root.aux`' is not an error message—it merely means that this is the first time LaTeX has seen the file `root.tex`, and thus `root.aux` has not been created as yet.

## 6.2 The `.dvi` (DeVice-Independent) file and post-processing

The intended result of a LaTeX run is the creation of a `.dvi` file containing a *device-independent* description of each page of the document. Sometimes, in order to create this, it will be necessary to run LaTeX on the document twice—similarly to a programming language compiler, LaTeX often needs two passes at a document to resolve forward references. An obvious case of this is the contents page, which appears at the front of a document, but cannot be created until LaTeX has seen the whole of the document.

The `.dvi` file contains details of the size, style and placement of each character to be found on each page of the final document. If LaTeX can be considered to be a document compiler, then a DVI post-processor, to continue the analogy, combines some of the functionality of an assembler and a linker. The same `.dvi` file may be used to generate *device-dependent* output for any number of output devices, the only limitation being the availability of the desired output device. The DVI post-processor takes the *device-independent* form of the document page descriptions, and produces output in a format suitable for delivery to a printer or similar device.

In general, on this site, there are two DVI post-processing routes to consider: one for on-screen presentation (if desired) and one for generation of the Postscript page description language understood by local laser-printers. The former route is handled mostly by the `xdvi` package described in Chapter 7, and the latter by a program called `dvips`. There is at least one other driver available, but only for local compatibility with old documents that require it; new users should not even consider attempting to use it.

## 6.3 The `dvips` post-processor

By the `.dvi` file stage, the *relative* sizes and positions of the characters on the final printed page have been fixed, but various options are available within to modify the *absolute* size and position of the entire page image. DVI post-processor options usually allow the user to control precisely which pages of the document should be printed out (users who insist on printing all 78 pages of a dissertation after a one-sentence change are carrying on an entirely unnecessary, wasteful practice which can be spotted and will be dealt with...) and various other internal workings of the program.

The `dvips` program provides a large number of command-line options for controlling aspects of the Postscript production process. The more significant of these are as follows:

**-r** Output is produced by default in 'first page first, last page last' order. Use `-r` to get 'last page first, first page last'.

**-c<number>** Generate '<number>' copies of the output pages.

**-p<number>** Make page '<number>' the first output page instead of the (default) first page of the document.

**-l<number>** Make page '<number>' the last output page instead of the (default) last page of the document. [1]

**-h header-file** Include the (assumed to be Postscript) file 'header-file' in the generated Postscript output in such a way as to be processed by the printer before the Postscript that describes the document. This is a 'hook' for access to a huge multitude of Postscript facilities, most of which are way beyond the scope of a beginner's introduction to LaTeX. For example, the generally-available header files 'multi.pro' and '4xA6.pro' may be used to print an entire document as 4 A6 quarter-pages on each A4 page. This is sometimes useful during drafting, and is often used locally to generate handouts consisting of the slides from a presentation written with SliTeX.

The above, plus default environment setups available to local users, should be adequate to allow straightforward use of `dvips`. The online manual page gives further information, and for the really committed user there is an extensive document on the package available from your local support office.

The `dvips` program is fairly 'chatty', although a command-line option (`-q`) is available to suppress all output other than real error messages. Terminal output from a sample run should look something like this:

```
This is dvips 5.493 Copyright 1986, 1992 Radical Eye Software
' TeX output 1993.03.07:1431' -> guide.ps
<tex.pro><special.pro>. [0] [1] [2] [1] [1] [2] [3] [4] [5] [6] [7] [8
<dbexample.ps>] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20]
[21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35]
[36] [37] [38] [39] [40<../../pix/trivial.ps>] [41<wdump.ps>] [42] [43] [44]
[45] [46]
```

The user is informed as each page is generated, and files other than the original DVI file are identified as they are read. The sample output above was generated from the DVI file associated with a version of this document.

[1] A single page of output may be obtained simply by giving the same number to '-p' and '-l'; e.g., to get just page 6, give the command-line options '-p6 -l6'.

## 6.4 Administrative hints

The result of a Postscript post-processor run is, as described above, a Postscript file (suffixed `.ps`) which can be sent to a laserprinter to produce the desired output pages. The Postscript files produced by DVI post-processors are generated relatively quickly, but tend to be quite large (about 40K for an average single page, rising non-linearly to about 250K for about 25-30 pages). Since disk space on local systems is almost always in short supply, it will be appreciated if users could remove Postscript files whenever possible, and try not to leave them around in between processing sessions. During the annual dissertation crisis experienced every spring semester, the amount of disk space which can be reclaimed simply by trawling user file systems for Postscript files created by `dvips` is phenomenal, but it is a task much better performed by the individual conscientious user than the desperate system administrator.

Despite the usual provision of several megabytes of memory in each laser printer, it is perfectly possible to generate Postscript files which exceed the capability of any particular printer to print them. It is effectively impossible to determine in advance whether any particular Postscript file will cause trouble, and in any case users who create large print jobs during prime time make themselves rapidly very unpopular anyway. Users are advised to use page-selection arguments to slice large documents up into several parts of perhaps 20-25 pages or about 250K and to check the size of any particular Postscript file *before* aiming it at a printer. Since diagrams included via `\special` can occupy sizeable amounts of disk space (200K or more for a full X screen dump) documents containing figures incorporated in this manner deserve special attention. It may sometimes be useful, having generated the diagram itself, simply to leave the space for it blank in the draft document, include the diagram proper at the last minute and print off the page bearing the diagram separately.

# Chapter 7

# Previewing LaTeX Documents

The usual cycle that an average LaTeX user goes through to produce their (eventually!) stunning final document involves viewing the output one or more times, where 'more' is often a much larger number than initially anticipated. If the method of viewing the document through its various steps towards completion is restricted to simply printing it out on the laserprinter, a large amount of paper is wasted on printing material to be read approximately once. Also, the time taken between sending the document to the laserprinter and the document appearing can sometimes be longer than is desirable for efficient document production. The time delay may be due to a large queue of multi-page documents being ahead of your one-page 'first try', or even the fact that someone has forgotten to refill the laserprinter paper tray[1].

Therefore, in an attempt to save the LaTeX user valuable time and effort (and also to avoid the needless destruction of acres of scarce forestry), various means are provided by which the LaTeX user can view a document without the aid of a laserprinter. The main machines within the department offer at least one form of previewing LaTeX documents.

## 7.1 Using `xdvi`

`xdvi` is a DVI file post-processor which generates its output in an X window. All that is necessary to start it up is the command

```
xdvi <filename>.dvi
```

In this case the `.dvi` filename extension may be omitted, as `xdvi` assumes that the extension is present. In keeping with other **X** applications, after a few seconds a flickering outline of the previewer window will appear. Positioning of this window is done in the usual manner. The first page of the document appears in this window. By positioning the mouse pointer at a required point within the window and holding down either the left mouse button, the right mouse button, or both mouse buttons simultaneously, gives varying degrees of magnification of that section of the document. This allows the close inspection of parts of a document as well as being able to see its full-page appearance. To exit from the `xdvi` previewer, simply type `q` to the window.

More extensive information is available from the `xdvi` manual pages and the next section provides a summary of the main options available. In particular, and in common with the

[1] Please take note of this intentional hint!

majority of X applications, `xdvi` accepts a large range of X default settings, described in the manual page, with which the user may tailor the program's behaviour. The behaviour seen by the casual user is just the 'default default' configuration, so to speak.

## 7.2 `xdvi` options

### 7.2.1 Options selected when xdvi is invoked

The most useful options here are:

**+page** Preceding a numeric value with a `+` symbol will cause the previewer to begin with the selected page (remember this is the DVI page number in the file).

**- size** This chooses a scaling factor. The default factor is a value of three. Using a larger value produces a smaller image, which may be useful if you just want to look at the layout on the page. As an example, using

```
xdvi - 5 test.dvi
```

will produce a scaled down image of the document *test.dvi* that will fit a complete page on to the screen, but which will not be particularly readable. (Note that a space is necessary between the '`-`' character and the digit.)

### 7.2.2 Commands used within `xdvi`

The following are particularly useful commands to remember:

**q** quits the program. Synonyms are *control-D* and *control-C*.

**n** moves on to display the next page, or the nth next page if you prefix it with a numeric value.

**p** moves back a page (or again, n pages if you provide a prefixed value.)

**g** moves to the given page, eg `4g` moves to the start of page 4.

**ˆ** moves to the 'home' position on a page (normally the top left corner).

**d** places the bottom of the page at the bottom of the display window.

## 7.3 Ghostview

When Postscript graphics are included in a document, `xdvi` is powerless to render them, because it knows nothing at all about Postscript. `xdvi` operates on the user's DVI file and TeX font files to generate a screen image, and at the point where the graphic is included, the DVI file simply has a note to the `dvips` post-processor about where to find the graphics. When `xdvi` encounters such a note, it simply leaves a blank box where a diagram will appear in the final printed version. Most of the time, this is fine, but there are circumstances where whatever is missing has a close enough relationship to the document that to view the document without it is insufficient for proof-reading purposes, and the user really needs a method of proof-reading the actual Postscript shortly to be sent to a printer.

Ghostscript is a public-domain software package that interprets Postscript page descriptions in an X window. It is adequate but suffers from a highly unsophisticated user interface rather akin to entering text direct from keyboard to laser printer. Thankfully for the majority of the user community, Ghostview is an X wrapper for Ghostscript which transforms the basic Ghostscript user interface into a full-scale X application, complete with menus, buttons, scrollbars and the like. The general previewing user is *strongly* recommended to stick to Ghostview.

Ghostview, as might be expected, operates on the Postscript version of the document, the `.ps` file. Probably the most common mistake made in the use of TeX and LaTeX previewers is forgetting to regenerate the file to be previewed and becoming increasingly annoyed at the computer's failure to take any notice of the last two hours' edit session. Remember, the Ghostview stage, if used, happens between `dvips` and `lp`!

To start up Ghostview, simply enter

```
ghostview file.ps
```

where 'file.ps' is the (full) name of the Postscript file you wish to preview. Ghostview is one of those programs whose behaviour is only really learnable by experiment, and there is little to be gained from listing the five most important options here. The manual page itself states

> Don't be alarmed by the number of options. Generally, one invokes `ghostview` with just one parameter, the name of the file to be previewed... The options provide a way to set X resources from the command line for a single invocation of `ghostview`.

It should be stated here that given that Ghostscript is not necessarily a complete, semantically-equivalent-to-a-laserprinter implementation of a Postscript interpreter. Indeed, the documentation that accompanies the software explicitly states that it is not, for copyright/licensing reasons. Users should remember that it is only *close* to Postscript, and should expect to see occasional 'blips' in its rendering of Postscript objects (not to mention the occasional straightforward core dump!).

# Chapter 8

# Generating graphics

LaTeX itself is a document preparation package overlaid on the TeX typesetting package. While it is possible to produce diagrams and pictures of acceptable appearance using LaTeX alone, the basic `picture` environment is difficult to learn, limited in scope, long-winded and, in general, extremely counter-intuitive. Its use is not recommended, except perhaps for very simple 'boxes-and-lines' drawings.

In principle, however, the PostScript page description language used to drive output devices is capable of individually controlling every single possible pixel which might appear on a printed page (a page of 300dpi A4 LaserWriter output is made up of about 8000000 pixels). Hence, any sufficiently well-behaved PostScript 'program' can be added to LaTeX output in such a way as to make the added text part of the document. All that is required is to tell LaTeX where on the page the extra PostScript text will draw whatever it is to draw and how large a box should be left to draw it, and LaTeX can be persuaded to fit the rest of the document around it. Graphics production methods used alongside LaTeX are then directed to generate drawings starting at the PostScript 'origin' page position, and when incorporated in the document appear at whatever 'current point' the LaTeX source specifies.

Several methods of generating graphics are available locally for LaTeX users:

**LaTeX `picture`** As mentioned above. Extensions, such as 'epic' and 'eepic' are available, which can ease some repetitive tasks, and clearly, this method is the most portable available;

**`xfig`** A `MacDraw`-like program available on X workstations which, along with conversion programs such as `fig2dev` and the 'graphics management' TransFig package, provides the cleanest LaTeX–Postscript combination available;

**`idraw`** Another drawing program available in some parts of the Department (at the whim of whatever C++ compiler happens to be available!) that generates its own Postscript. Experience suggests that this can be persuaded into LaTeX incorporation with a little tweaking;

**`xgraph`** A graph production program, capable of generating PostScript or HPGL. PostScript from `xgraph` can be incorporated into LaTeX with very little modification;

**`xwd,xpr`** Window dump files generated by the `xwd` program running on Sun or Hewlett-Packard workstations under the X Window system, converted to PostScript by `xpr`. Again, a little 'hacking' of the Postcript will be required.

LaTeX handles `picture` output for itself entirely internally, and its use will be discussed no further here; consult [Lamport86] for more details. The bulk of the remainder of this section concerns itself with 'hints and tips' for getting around some of the more common problems associated with the other methods mentioned above.

## 8.1 PostScript graphics files

In theory, any PostScript file can be incorporated into any other, so long as each obeys certain rules concerning what they are allowed to do to the 'PostScript engine' producing the output. The full collection of these rules is called the 'Adobe Document Structuring Conventions' (DSC) and while the phrase is widely-used, very little of this document is practically useful for day-to-day document preparation. The main points to note are:

`%%BoundingBox:` Does the file to be incorporated have a `%%BoundingBox:` comment somewhere near the top? If not, then one must be calculated. If the file is printable by itself, then the simplest way to do this is to print it out, draw around the diagram the smallest rectangle which contains it, and convert the dimensions of this rectangle to 'points'. For all practical purposes, there are 72 points to the inch, and 2.835 to the millimetre. Bear in mind that the printer's accuracy in grabbing a piece of paper from the feed tray is only around 1–2mm in any direction, and try not to become obsessed with accuracy.

All the utilities mentioned above can generate `%%BoundingBox:` comments in the right circumstances. However, PostScript appears on this site from all sorts of places, and it is wisest not to leave this to chance—*check*!

**Diagram origin** Occasionally, utilities which generate PostScript do so in 'egotistical' fashion, assuming that the user cannot possibly want anything else on the page with *its* graphics, and the output is rotated, scaled and centred on the page. This is all very well, but what is desirable for inclusion in LaTeX is a diagram whose lower-left `%%BoundingBox` corner is at the PostSript origin. With most printers, this means that on printout of the file on its own, the lower left corner of the diagram will be 'off the page' at the bottom left, because they are designed to believe that they are printing on an American 8.5"x11" page, rather than European A4 (210x297mm).

Usually, persuading the file to produce its output at the origin involves finding the PostScript `translate` operator responsible and commenting it out. Then, it may be necessary to 'translate' the `%%BoundingBox` comment to suit. `xfig`, `xgraph` and `MacDraw` can all be persuaded to produce 'origin graphics'; `xpr` will need a little strategic commenting.

**Graphics context** It is perfectly possible for a postScript graphic to leave the PostScript engine in a state in which it cannot continue, or to 'mess about' with the state so that it is pointless to continue. The obvious example of this is the `showpage` operator. If the file has been produced with a view to printing it out alone, then somewhere at the tail of the file a `showpage` will appear, which to the PostScript engine means 'this page description is complete; print it out'. In the case of a figure in a LaTeX document, we would prefer the rest of the page, after the figure, to be added before that page is printed. Spurious `showpage`s can be safely commented out.

Complications can also be caused by `save/restore` and `gsave/grestore` appearing in the included file. This is an area best avoided if possible, where, to use the time-honoured phrase, 'experience and informed courage count for much'. Access to various PostScript manuals [Adobe85a, Adobe85b] is also useful here.

`xfig`, `xgraph` and `idraw` will produce `showpage`less code; `xpr`, again, may need some help.

## 8.2 Incorporating the image into LaTeX

This section covers only the incorporation of graphics as LaTeX floating figures. Most of its suggestions can be generalised to cover other uses, but what appears here should suffice for most purposes.

Every object placed on a page by TeX and LaTeX is enclosed in a box. Usually there is a hierarchy of boxes involved in all but the simplest objects. For example, a letter in a word appears in a box which contains only that letter, which appears in a box containing all or part of the word (which might, after all be hyphenated), which appears in a box containing a line of text, which appears vertically between all the other line boxes which constitute a paragraph, etc...

The general principle of graphics incorporation is to persuade LaTeX to allocate a box of the right size and shape in the right place, in which the diagram will be drawn using the LaTeX `\special{}` directive. Continuing the analogy drawn several times in this document to LaTeX as a programming language, `\special{}` is similar in approach to the 'assembly language insert' features often found in lower-level and systems programming languages.

Text found between the braces in `\special{}` is not 'interpreted' at all by LaTeX, but passed on to the tools used to process the `.dvi` file. Thus, the use of `\special{}` is almost always 'non-portable', in the sense that changes may well have to be made to generate a document containing the directive at a different site or on a different computer system. This important point is often ignored by LaTeX users, but is only too well-understood by producers of camera-ready conference proceedings. Collecting a large number of separately produced articles together may sound like a simple business, given the relatively high level of compatibility of basic LaTeX systems, but the addition of graphics to each paper usually means coming to terms with a large number of different methods of graphics production and incorporation, and tempers can become frayed. If you are asked to send a document in LaTeX source form to another site, agree with the remote site *in advance* how graphics, if there are any, are to be presented.

The two `dvips` tools used on site take different approaches to diagram incorporation with `\special{}`. By far the best way to explain this is with reference to an example. Suppose, for example, that the diagram to be included is held in a PostScript file called `trivial.ps`, has a correct `%%BoundingBox`, draws from the origin, and is 162 by 54 points in size (2.25"x0.75" or 57x19mm). Assume also that the document uses `11pt.sty` and `a4wide.sty`.

The `dvips` program offers a fairly sophisticated set of 'hooks' on which to hang Postscript graphics. The style file `epsf.sty` contains a (locally-added) TeX macro `\epsffig{}` which does almost all of the work for itself, as well as others which take care of more complex examples. In this case, since the PostScript file contains a sensible `%%BoundingBox`, the figure can be produced simply with the LaTeX source given in Fig. 8.2. In this case, the various calculations are done partly by the TeX macros, and partly by `dvips` itself, and the

```
\begin{figure}[htb]
\epsffig{trivial.ps}
\label{examplefig}
\caption{An example of a figure.}
\end{figure}
```

Figure 8.1: Simple figure incorporation using `dvips` and `epsf.sty`.

`\special{}` is 'hidden' in the `\epsffig{}` macro. Note that, since `\epsffig` was added locally, it should not be relied upon when shipping documents off-site. It is intended purely as a 'simplest route' option to solve straightforward incorporation problems.

The general form of Postscript-graphic incorporation into LaTeX is via the `\special{}` directive. If the Postscript file you wish to incorporate is considered 'well-formed', then the locally-available Perl script `pssp` will generate a LaTeX fragment to incorporate that file, consisting of a `\vspace{}` directive to allocate some vertical space and a `\special{}` directive to pull the file in. Running the command `pssp` with no arguments gives a summary of usage: there may even be a real manual page available by the time you read this document.

Practically speaking, you should approach the incorporation of PostScript graphics into LaTeX in the following manner:

1. Do your best to generate the graphics in such a way as to follow the conventions mentioned above. If that fails, do your best to handcraft the conventions in the manner described above.

2. Try the methods described above to incorporate the graphics file. In each case, you should be able to get *some sort* of output, even if it looks odd. The best way to fix problems in this field is first to get to the stage where you have *some* output, then work towards making it *correct* output. Previewers and the ability to print out only one problematic page come in very useful here.

3. Investigate the 'periphery' of each of the techniques described above, to see if a more complex form of one of them solves your problem. For example, `dvips` gives a large collection of different options for `\special{}`, allowing extensive flexibility in scaling, rotating, clipping and translating PostScript pictures. One of these configurations is almost bound to work. It may be that you have not fully exploited the options to the Postscript generation program; go back and check the manual pages to see if a better way can be found.

Eventually, you may give up and come and ask. There is considerable expertise around the Department in this sort of field, and it is *very* rare now that a PostScript picture simply cannot be persuaded to behave as a LaTeX figure. Please make sure you have exhausted the available information first!

## 8.3 Example output

The figure 'floating' somewhere around this paragraph (Fig. 8.3) is the example used above as a demonstration of technique. It was generated using `xfig` and `fig2dev` and has a

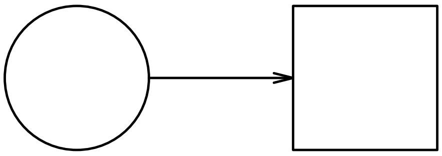


Figure 8.2: An example of a figure.

`%%BoundingBox` comment on line 6 of the PostScript file which reads

```
%%BoundingBox: 0 0 162 54
```

indicating first the $x$ and $y$ co-ordinates (in points) of the lower left corner of the diagram (0,0) and then the $x$ and $y$ co-ordinates of the upper right corner (162,54). Since this document is processed using the newer of the two `dvips` programs available, the LaTeX source quoted in Fig. 8.2 was used to incorporate it. The centred figure is 162 points wide, and has whitespace either side of it approximately 139 points wide, in order to centre it in a `\textwidth` of 441 points.

Figure 8.3 shows a screen dump taken from one of the Department's Hewlett-Packard workstations. The various screen windows show the following:

**upper left** Some of the source file of a version of this document

**lower left** Log output from a `dvips` run

**centre** `xdvi` previewer output, with the 'magnifying glass' active

**lower right** A screenful of 'chatty' LaTeX log output

This figure was produced by adding the '`xwd | xpr`' command line to the window manager menu as an option, so that

- no command line asking for a screen dump appears on the screen dump itself (considered terribly unprofessional);
- the mouse was free to be used to demonstrate the 'magnifying glass' facility of `xdvi`.

Close inspection of this figure reveals quite a lot about the actual operation of the LaTeX document production system.

## 8.4 Conclusion

In the final analysis, what matters is not that the first-chosen technique is the one that is used for figure production, but that the figure is successfully produced in reasonable time. There have been many cases where hours or even days have been spent trying to pummel one PostScript file into shape, where time might have been more efficiently spent in using another graphic generation package to copy the chosen picture by hand. The methods described above are all known to work reasonably well, but bugs can appear in the strangest of places, and it is best to be flexible so as not to waste time. One of the methods is bound to work successfully... ...we hope!

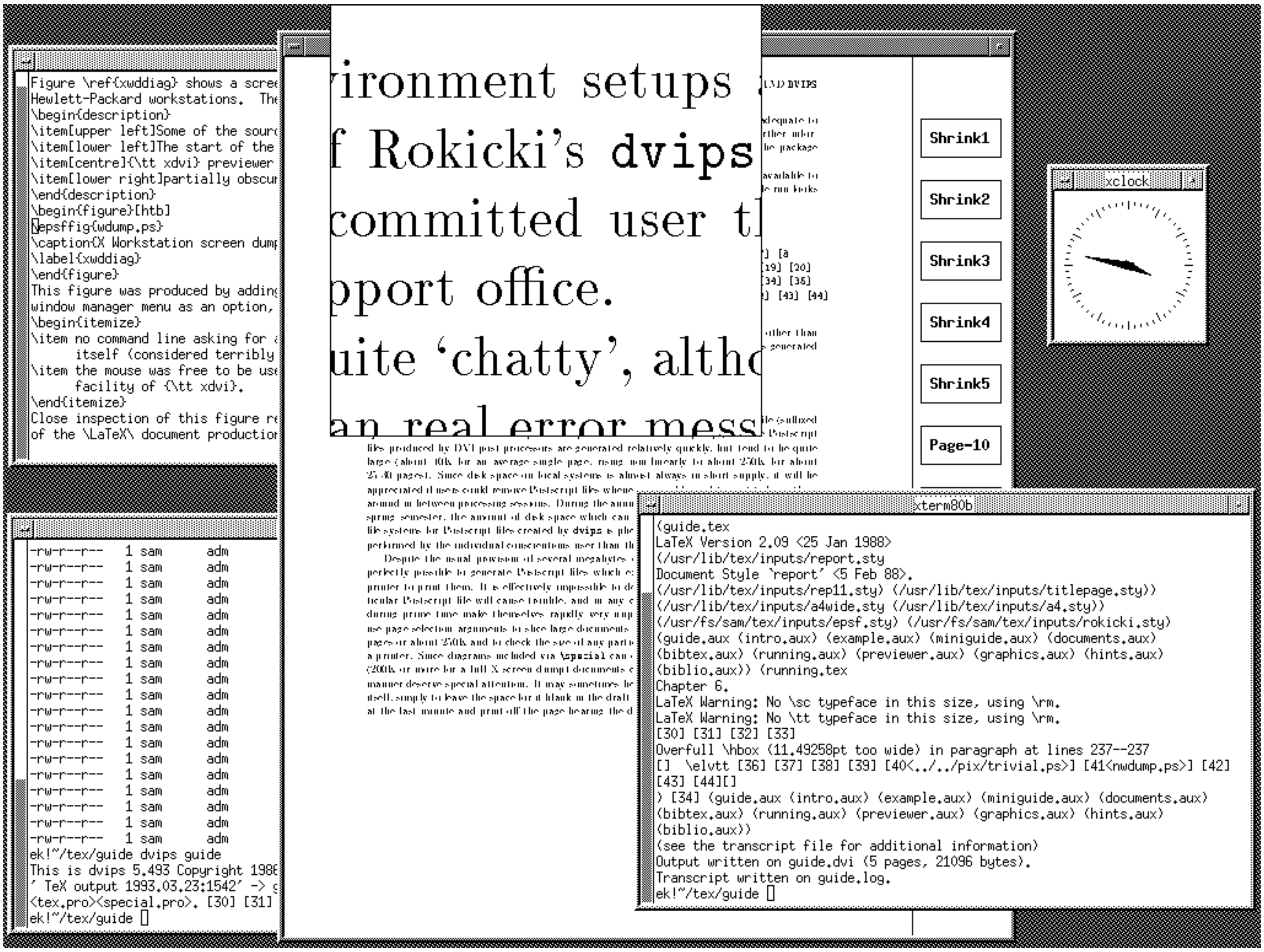



Figure 8.3: X Workstation screen dump.

# Chapter 9

# Some Hints & Useful References

These are grouped together broadly, but are not ordered in any particular sense.

## 9.1 LaTeX commands

**Quote Marks.** The single quote mark key produces a closing '9' quotation mark. For an opening '6' quotation mark, use the backquote character (upper left somewhere on most keyboards). For double quotes, type two single quote or backquote characters. (The double quote mark character is unsuitable and is not normally used in ordinary text.)

**Line Breaks.** Can be forced by the use of two backslash characters, as in `\\`. It is mainly used with the `tabular` environment, to separate fields in a `\title` or `\author` block, and for setting poetry! Otherwise, only use with great care—if at all.

**Page Breaks.** These can be forced by using the `\newpage` command, but as with line breaks, use sparingly if at all.

**Reserved Characters.** The following ten characters have special roles and cannot be included directly in normal text:

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

All of them, bar `~ ^ \` can be included in text by quoting them with a preceding backslash character `\`. As an example, we use `\&` to produce the & character.

**Accents & Special Characters.** These can be produced by using the large range of character commands provided in LaTeX. For details see the LaTeX manual [Lamport86], Chapter 3. There is a large range of mathematical symbols, accents, and special characters such as £ which is produced by `\pounds`.

**Extra spacing.** On occasion, we need a little extra space between words or characters (an example might be where the last letter of a word in italics would 'collide' with a closing parenthesis character). The use of the character pair `\/` directs LaTeX to leave a little extra space on such occasions. As a general rule though, leave spacing to LaTeX to organise.

**Dashes.** The 'minus' character `-` prints as a short hyphen. For a longer dash, as in 'pp21–29' use two dash characters, as in `21--29`. For a punctuation dash—you should use three as in `---`.

**Includeonly.** Using the `\includeonly` command not only ensures that the size of the `.dvi` file is reduced, it also speeds up the processing time used by LaTeX. As your file gets larger, this becomes a more significant benefit! Insert the `\includeonly` command after the initial line of the root document.

## 9.2 Style and Use

**Appearance.** Don't try to produce perfect copy from the start. Get the structure and content of the document right, and only then, at the last stage, sort out any odd formatting problems such as widows and orphans (see below), or extra spaces.

**Emphasis.** Typography uses *italics* rather than underlining for emphasis. Don't overdo the use of emphasis, as it soon loses its effect. A further useful convention is to set the first occurrence of a significant **technical term** in boldface.

**Widows & Orphans.** A **widow** is the last line of a paragraph appearing as the top line of a page. Similarly, an **orphan** is the *first* line of a paragraph appearing as the last line of a page. The page-breaking algorithm in LaTeX is quite good, but these still occur from time to time. Careful use of `\newpage` can remove orphans, but leave such tasks until the very end of document production. Widows are harder to remove.

**Style Options.** A dissertation should be produced using the following `documentstyle` options.

- `report` style, which provides for the use of `chapter` headings.
- `11pt` size. The default size for LaTeX is 10pt, which is really too small when used with an A4-sized page.
- `a4` or `a4wide` options. These enlarge the printed area to make better use of a sheet of A4 paper. The `a4` option leaves a larger margin and is probably the more suitable one to use. (The `a4wide` option makes use of the `a4` option and they are mutually exclusive.)
- `dspaceon`, which will need to be defined as in the first example of Chapter 3. Stretching the interline spacing by a factor of 1.3 is about right, but remember to turn it off for example and mathematical expressions as these may look odd in such spacing.

**Glossary.** Appending a glossary of technical terms seems to be a growing trend, but if you do so, take care to obtain some good definitions. A glossary is really only justified where the topic makes use of many specialised terms.

**Index.** You *can* produce these with LaTeX, but the provision of an index is NOT recommended for dissertations. Better to provide a `tableofcontents` at the beginning, since this guides the reader more effectively.

## 9.3 Ethical and Social Issues

Our laser printers are used by staff, honours students and postgraduate students in the Department. Anti-social behaviour easily leads to frayed tempers and worse! Please remember always that you are not the *sole* user of the printers and try to consider the needs of others.

**Print Runs.** These should be kept short using the `dvips` options available. Few of us enjoy standing by the printer watching someone's life history appear while waiting for one page of output. In particular, only print the pages that you need, and print them only when you need them.

**Paper Supply.** It is easy to reload the printer tray (the need is indicated on an Apple/Sun LaserWriter by the amber light staying on continuously, and by a displayed message on HP LaserJets). If you are printing lots of pages, go and make sure that there is paper available. Don't overfill the feed tray, as this may cause the printer to jam; about half full is sufficient.

**Drafting.** Do it elsewhere, not on the laser printers please. `xdvi` is quite competent at the job. We would prefer to avoid the need to impose paper quotas, but such options are open to us...

# Bibliography


[Knuth86] Donald E Knuth, *The TEXbook*, Addison-Wesley, 1986, ISBN 0-201-13447-0

[Lamport86] Leslie Lamport, LATEX *A Document Preparation System*, Addison-Wesley, 1986, ISBN 0-201-15790-X

[Rubinstein88] Richard Rubinstein, *Digital Typography: An Introduction to Type and Composition for Computer System Design*, Addison-Wesley, 1988, ISBN 0-201-17633-5

[Adobe85a] Adobe Systems Inc., PostScript Language Reference Manual, Addison-Welsey, 1985, ISBN 0-201-10174-2

[Adobe85b] Adobe Systems Inc., PostScript Language Tutorial and Cookbook, Addison-Wesley, 1985, ISBN 0-201-10179-3