A LaTeX2e Gallimaufry

Techniques, Tips, and Traps

Peter Grogono

Original version for LaTeX: January 1996.

This version for LaTeX2e: March 2001.

Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd West
Montréal, Québec H3G 1M8

Contents

1 Introduction **1**

2 The Program `latex` and Friends **1**

3 Preparing the Input **4**

3.1 Document Structure 4

3.2 Front Matter 5

3.3 The Body of the Document 6

3.4 Cross References 14

4 Making Lists **14**

5 Math Mode **15**

6 Vertical Alignment **18**

6.1 The `tabbing` Environment 18

6.2 The `tabular` Environment 19

6.3 The `array` Environment 21

6.4 The `eqnarray` Environment 22

7 Floats **22**

8 The Preamble **23**

8.1 Dimensions and Glue 23

8.2 Defining Commands 25

8.3 Defining Environments 27

9 Additional Features **28**

9.1 Packages 28

9.2 Pictures 30

9.3 Making a bibliography 31

9.4 Making an index 32

10 Problems **32**

11 Further Reading **33**

A LaTeX2e Gallimaufry

Techniques, Tips, and Traps

Peter Grogono

1 Introduction

TeX is a typesetting program designed and implemented by Donald Knuth, now an Emeritus Professor of Stanford University. The letters in the name are Greek capitals — tau, epsilon, chi — and TeX is pronounced "tecch", rhyming with "yecch". LaTeX2e is an extensive set of TeX commands written by Leslie Lamport with the intention of making TeX usable to ordinary mortals.

LaTeX2e is not a user-friendly, "what you see is all you've got" word-processor. It is a powerful program designed to produce documents that can be reproduced directly in journals and books. LaTeX2e is harder to use than programs like Word or WordPerfect, but it has many advantages for the production of technical text in general and software documents. There are many reasons to use LaTeX2e:

- ◇ Ability to prepare all kinds of documents, from an office memo to a multi-volume book.
- ◇ Very high-quality output, as good as the finest books.
- ◇ Automatic generation of table of contents, list of figures, section numbers, cross-references, bibliography, and index.
- ◇ Advanced facilities for setting mathematical and program text.
- ◇ Highly customizable.

There are also a few reasons for not using LaTeX2e:

- ◇ It's hard to learn.
- ◇ It's not WYSIWYG.
- ◇ People will keep asking you why you don't use W***.

2 The Program `latex` and Friends

This section describes the use of LaTeX2e in the kind of command-line environment provided by UNIX^{TM}, LINUX, and MS-DOS. In these environments, different vesions of LaTeX2e all behave in pretty much the same way. There are also implementations of LaTeX2e for window environments; these usually come with their own documentation and are not described here. Neverthless, the programs and files tend to be similar and so the principles described here are generally applicable.

Figure 1 shows various programs and files used by LaTeX2e and the relations between them. The LaTeX2e formatting program, `latex`, reads a file called *name*`.tex`, where *name* is a name of your choice and `.tex` is the required extension. Running `latex` generates several files with the same name but different extensions. The most important output file is *name*`.dvi`, the *device independent output* file. To obtain formatted text, you need a *driver program* that can read `.dvi` files. The diagram shows two such programs: `dvips`, which produces a PostScript (`.ps`) file; and `xdvi`, which displays the output on the screen.

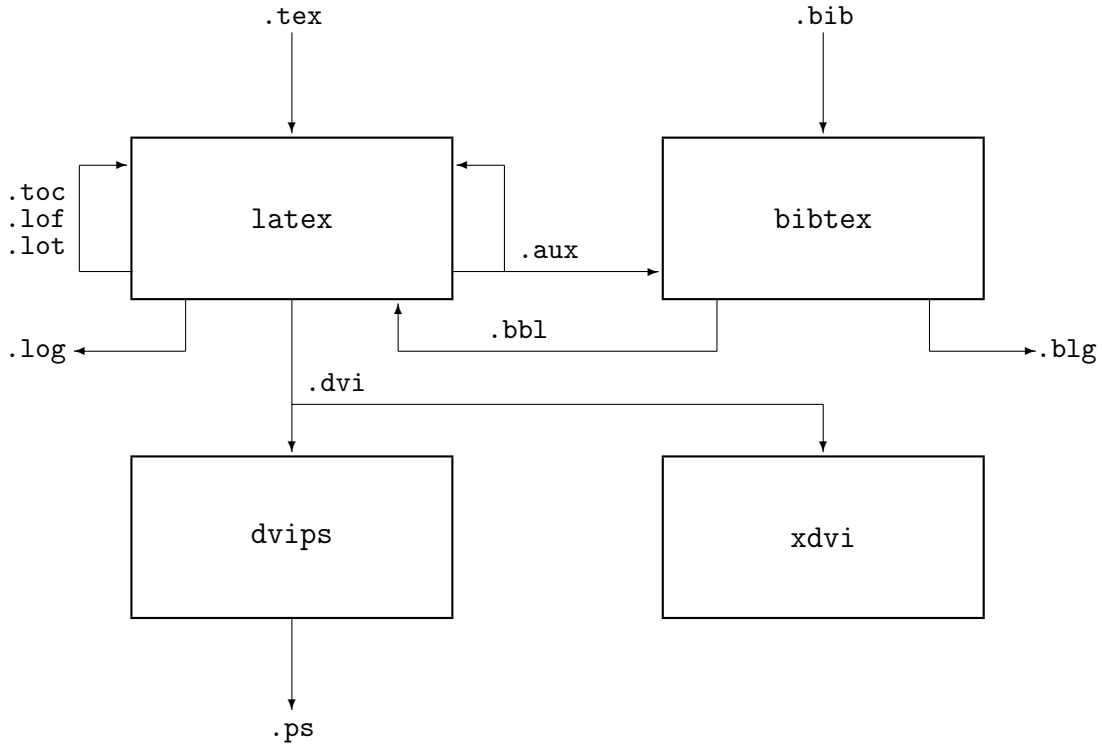

Figure 1: LaTeX2e programs and files

The `latex` program produces a number of other files, including: `.aux`, the auxiliary file; `.toc`, the table of contents; `.lof`, the list of figures; `.lot`, the list of tables; and `.log`, the log file. The first four of these contain information that is needed for the subsequent runs. For example, if your document includes cross-references, `latex` writes the numbers and page numbers to the `.aux` file. If you print your document after running `latex` once, forward references will appear as `??`. When you run `latex` again, it will read the `.aux` file and insert the correct numbers.

The `.log` file contains a report of its activities that you can usually ignore but may need if you have serious problems.

If you use the program `bibtex` to create a bibliography, you must first write a `.bib` file. The `bibtex` program reads this file, an `.aux` file created by `latex`, and creates a `.bbl` file that is read by `latex` during the next pass. It also creates a `.blg` (`bibtex` log) file containing a report of its progress.

There are two alternatives to `latex`: `pslatex` and `pdflatex`. The program `pslatex` uses PostScript fonts instead of Computer Modern fonts but is otherwise similar to `latex`. Using `pslatex` and then `dvips` yields a smaller PostScript file but has the drawback that the PostScript fonts are smaller than the corresponding Computer Modern fonts.

The program `pdflatex` creates a `.pdf` (portable document format) file instead of a `.dvi` file but otherwise behaves in the same way as `latex`. You can use Adobe's Acrobat reader, instead of `xdvi` and `dvips`, to view and print the `.pdf` file.

```
1       tony> ls
        tex.bib       texintro.tex
2       tony> l texintro
        This is TeX, Version 3.14159 (Web2C 7.3.1)
        LaTeX Warning: There were undefined references.
        LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
3       tony> bibtex texintro
        This is BibTeX, Version 0.99c (Web2C 7.3.1)
4       tony> makeindex texintro
        This is makeindex, version 2.13 [07-Mar-1997] (using kpathsea).
5       tony> latex texintro
        This is TeX, Version 3.14159 (Web2C 7.3.1)
        LaTeX Warning: There were undefined references.
        LaTeX Warning: Label(s) may have changed. Rerun to get cross-references right.
6       tony> latex texintro
        This is TeX, Version 3.14159 (Web2C 7.3.1)
        Output written on texintro.dvi (32 pages, 102328 bytes).
7       tony> dvips texintro > texintro.ps
        This is dvips(k) 5.86 Copyright 1999 Radical Eye Software (www.radicaleye.com)
8       tony> pdflatex texintro
        This is pdfTeX, Version 3.14159-13d (Web2C 7.3.1)
        Output written on texintro.pdf (32 pages, 351420 bytes).
9       tony> ls
        tex.bib       texintro.blg  texintro.ilg  texintro.pdf  texintro.toc
        texintro.aux  texintro.dvi  texintro.ind  texintro.ps
        texintro.bbl  texintro.idx  texintro.log  texintro.tex
```

Figure 2: Transcript of a LaTeX2e session using UNIX

Figure 2 is an abbreviated transcript of a UNIX session. These programs are quite noisy; most of the output that they generate has been omitted from the transcript. The following notes explain what is going on.

- Line 1. At the beginning of the session, the working directory contains only `texintro.tex` (the source file for this manual) and `tex.bib` (the bibliography for this manual).
- Line 2. `latex` produces warnings about undefined references and changed labels when it is run for the first time.
- Line 3. `bibtex` creates the bibliography.
- Line 4. `makeindex` creates the index.
- Line 5. Running `latex` again shows that there are still problems with references and labels.
- Line 6. The warnings disappear when `latex` is run for the third time.
- Line 7. `dvips` converts the `.dvi` data to postcript format. Unlike the other programs, `dvips` writes the PostScript to standard output.
- Line 8. `pdflatex` writes messages that are very similar to `latex`'s messages.
- Line 9. Listing the directory again shows that the programs have generated a number of intermediate files.

You can use any editor to prepare the `.tex` and `.bib` files, provided that it generates (or can be persuaded to generate) plain ASCII characters. Some advanced editors, such as EMACS, have built-in facilities to simplify the production of LaTeX2e source. TeX packages for PCs usually come with specialized editors with special features such as colour coding of text and one-click formatting.

3 Preparing the Input

The best way to learn how to prepare a LaTeX2e input file is to compare LaTeX2e input files with the output that they generate.

```
\documentclass[options]{format}
  preamble
\begin{document}
  body
\end{document}
```

Figure 3: The format of a LaTeX2e input file

3.1 Document Structure

Figure 3 shows the format of a LaTeX2e input file. The *options* field following `\documentclass` may be omitted. If present, it contains one or more of the following options, separated by commas.

- ◇ The size of the base font for the document. This must be `10pt`, `11pt`, or `12pt`. The default size is `10pt`.
- ◇ The paper size. The following sizes are available:

`letterpaper` (11 × 8.5 in)	`a4paper` (297 × 210 mm)
`legalpaper` (14 × 8.5 in)	`a5paper` (210 × 148 mm)
`executivepaper` (10.5 × 7.25 in)	`b5paper` (250 × 176 mm)

  These options define suitable dimensions for the margins, headers, and footers. You may find that there is not enough text on a page for your taste. For example, the text in `letterpaper` style is about 5 inches wide and $7\frac{1}{2}$ inches high. In this case, you can define the text dimensions yourself, as described in Section 8.1.
- ◇ The option `landscape` reverses the length and width dimensions of the paper. If you use landscape format, you may have to inform your printer as well.
- ◇ The options `onecolumn` and `twocolumn` determine the number of columns on the page. The default is `onecolumn`.
- ◇ The options `oneside` and `twoside` determine how page numbers are printed. For two-sided printing, odd page numbers are on the right and even page numbers are on the left. You may have to tell your printer that you are printing on both sides if you use `twoside`.

The *format* field must be one of `article`, `book`, `report`, or `letter`. The most common format, used for this document, is `article`.

```
\usepackage{latexsym}
\usepackage{makeidx}
\makeindex
\textwidth    6.5in
\textheight     9in
\topmargin   -0.5in
\oddsidemargin  0in
\evensidemargin 0in
\parskip        1.5ex plus 1ex minus 0.5ex
\parindent      0em
\pagestyle{headings}
\setcounter{secnumdepth}{5}
\setcounter{tocdepth}{2}
\newcommand{\tx}{\TeX}
\newcommand{\ltx}{\LaTeX2e}
```

Figure 4: Part of the preamble for this manual

The *preamble* contains definitions that apply to the entire document. Figure 4 shows an extract from the preamble for this manual. The manual requires two "packages": `latexsym`, for special symbols; and `makeidx`, for indexing commands. The command `\makeindex` tells LaTeX2e to construct an index. The next few commands set dimensions for the document; `\pagestyle` sets the page style, and so on. Don't be intimidated by the complexity of the preamble: you can start off with *nothing* in the preamble and add things to it as you need them. Section 8 explains some of the commands in Figure 4 and describes other things that you can put in the preamble.

There may be occasions on which you would like to use "double spacing", a concept that has little relevance for fine typesetting but remains as a hangover from the days of typewriters. LaTeX2e has a value called `\baselinestretch` which has a default value of 1. If you alter this value, the space between adjacent lines of text will change. For example, a value of 1.5 gives one-and-a-half spacing, and a value of 2 gives double spacing. You can also use odd values such as 1.25 and you can reduce the spacing by using a value less than 1. To change `\baselinestretch`, include this command in the preamble:

```
\renewcommand{\baselinestretch}{1.5}
```

The *body* of the document, which contains everything that you want to appear in the output, is described in the next section.

3.2 Front Matter

You can either construct your own title or use LaTeX2e commands to construct the title material for you. If you decide to let LaTeX2e do the work, include the following commands in the preamble:

- ◇ `\title{`*text*`}` defines the text of the title. You can use `\\` to indicate where you want line breaks.

- ◇ `\author{`*names*`}` lists the name(s) of the author(s). If there is more than one author, separate the names with `\and` commands, *not* commas or "and". You can include author's affiliations and addresses, using `\\` to separate lines.
- ◇ `\date{`*text*`}` defines the text of the date. If you omit this command, LaTeX2e will use the date on which the document is formatted. If you do not want any date, write `\date{}`.
- ◇ `\thanks{`*text*`}` produces a footnote to the title. You can include `\thanks` in any of the three commands above (although it would be unusual to footnote a date).

In the body of the text, after `\begin{document}`, write `\maketitle`, with no arguments, to generate the title.

After the title, you can use these commands:

- ◇ `\tableofcontents` writes a table of contents.
- ◇ `\listoffigures` writes a list of figures.
- ◇ `\listoftables` writes a list of tables.
- ◇ `\begin{abstract}` *text* `\end{abstract}` writes an abstract consisting of *text*.

3.3 The Body of the Document

The body of the document contains text interspersed with commands. The source file should contain only ASCII graphic characters. The graphic characters (codes 33 through 127) are processed as you would expect (with a few exceptions explained below); all other characters are treated as "white space", as described in Section 3.3.1.

There are ten characters that TeX processes in a special way. They are

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

It is easy to include the first seven of these characters in your document. You just write `\` in front of them: `\#` translates to #, and so on. You can obtain the last three by using the command `\verb` (see Section 3.3.2). For example, `\verb"~\~"` translates to `~\~`. The sequence `\\` does not print `\`: it is a LaTeX2e command which inserts a line break in the output text.

The role of the special characters is described briefly here. Detailed explanation of their use appears in later sections.

- ◇ `#` introduces a formal command parameter. You will not need it unless you are defining your own commands (Section 8.2).
- ◇ `$` switches between text mode and math mode (Section 5).
- ◇ `%` introduces a comment. The comment consists of all text between `%` and the end of the current line.
- ◇ `&` is used to align columns in tables and arrays (Section 6.2).
- ◇ `{` and `}` are used to enclose arguments of commands and for grouping in general.
- ◇ `_` (underscore) and `^` provide subscripts and superscripts in math mode (Section 5).
- ◇ `\` introduces LaTeX2e and TeX commands.
- ◇ `~` prints a blank character but inhibits line breaking.

Aïda translated *déchaîné* as "unbridled" and *deçà* as "this side of". Professors Srīnivāsa Aiyaṅgār and Stanisław Świerczkowski agreed that these were correct translations.

```
A\"\i da       translated \textit{d\'echa\^\i n\'e}    as ``unbridled'' and
\textit{de\c c\`a} as ``this side of''. Professors Sr\=\i niv\=asa
Aiya\.ng\=ar and Stanis\l aw \'Swierczkowski     agreed    that these were
correct                          translations.
```

Figure 5: A short story with accents and the TEX input that generated it

TEX is quite good at hyphenating words to improve line breaks. If you encounter a word that it cannot hyphenate correctly, you can help it by putting `\-` (a "discretionary" hyphen) in the word, as in `gal\-axy`.

TEX provides accents for most languages that are based on or can be transcribed in Roman letters. Figure 5 illustrates more than you are likely to need. Before you can put an accent on an "i" or a "j", you should remove the dot by writing `\i` or `\j` rather than just `i` or `j`, as in Srīnivāsa.

3.3.1 Spaces and Punctuation

Unlike most word processors, LATEX2e does not count blanks. If you want a particular amount of white space, either horizontally or vertically, you have to ask for it. Here is a summary of the way in which LATEX2e handles white space in the input file.

- ◇ A single blank creates an inter-word space. The precise width of this space will vary in justified text.
- ◇ Many blanks are equivalent to a single blank. A line break is also equivalent to a single blank. You can see these conventions at work in Figure 5.
- ◇ Blanks and line breaks that immediately follow a command are discarded.

  For example, the command `\S` prints the section symbol, §. If you write `\S␣␣␣␣4.5` (in which each ␣ denotes a blank), LATEX2e will print §4.5.

  Sometimes you will want a blank to follow a command such as `\S`. There are three ways to do this:

  - enclose the command in braces: `{\S}`;
  - append a null argument: `\S{}`; or
  - append a normal space: `\S\␣`.
- ◇ White space following the argument of a command is *not* discarded.

  This **sentence** was **entered** as: `This \textbf{sentence} was \textbf{entered} as:`.
- ◇ Two or more consecutive line breaks (in other words, one or more blank lines) in the input file signal the end of a paragraph. The command `\par` has the same effect as a blank line.
- ◇ The tilde (`~`) acts as a "hard blank". It produces an inter-word space and is never replaced by a line break.

TEX usually manages to get the right spacing between punctuation symbols. Occasionally you have to help it. If you write `Mr.␣Vo` (where ␣ indicates a blank) in your input file, TEX will think that the period signifies the end of a sentence and will print "Mr. Vo". To obtain the correct spacing, enter `Mr.\␣Vo`, which prints as "Mr. Vo". It would be even better to enter `Mr.~Vo`, because this would prevent a line break between `Mr.` and `Vo`.

Use the left and right single quote characters rather than the double quote character (`"`). To obtain opening and closing double quotes, write ` `` ` and `''`, respectively. For example, ` ``quoted text'' ` yields "quoted text". On most keyboards, left quote (`` ` ``) is the top left key, below ESC, and right quote (`'`) is at the right of the keyboard, just left of the ENTER key.

Use "`-`" (hyphen) to hyphenate words, as in "object-oriented". Use "`--`" (two hyphens) to separate numbers; `555--1212` yields 555–1212. Use "`---`" (three hyphens) to make an "em dash" in running text — like the one you just read.

If you want a minus sign, use math mode (see Section 5). The input "`5-2=3`" yields 5-2=3, which looks awful. Inform LATEX2e that this is mathematics by enclosing the equation between $ signs. "`$5-3=2$`" yields $5 - 3 = 2$.

3.3.2 Sizes and Shapes

Compared to commercial word processors, LATEX2e has a rather limited range of font styles and sizes. Although it is possible to incorporate many fonts into LATEX2e files, doing so is quite difficult and has the undesirable effect of making the LATEX2e source files less portable.

The font sizes provided by LATEX2e are: tiny, scriptsize, footnotesize, small, normalsize, large, Large, LARGE, huge, and Huge. To obtain Elephant, write "`{\Large Elephant}`". The braces `{...}` are used here as scope delimiters. If you wrote simply "`\Large Elephant`" or "`\Large{Elephant}`", all of the text until the next size command would be Large.

You can change the text style by using either a *command* or a *declaration*. The commands and declarations are shown in Figure 6. Use commands to change the style of a single word or phrase: to obtain "this is an *italic* word", write "`this is an \textit{italic} word`". Use declarations to change the style of an extended body of text. The declaration stays in effect until the end of the current scope, indicated by "`}`" or "`\end{...}`". For example, to obtain "lots of nonsense, *lots more nonsense,* and even more nonsense", write "`{ lots of nonsense, \slshape lots more nonsense, \upshape and even more nonsense}`". Note how `{` and `}` limit the scopes of the declarations.

The shapes, series, and families can be used in combination. Since there are four shapes, two series, and three families, you might think that there are 24 different styles. Unfortunately, this is not the case because not all of the combinations give distinct results. Figure 7 shows the combinations that work on my system.

All of the styles except typewriter are proportionately spaced. There is a command that enables you to print any characters, even `\`, in typewriter style. The command `\verb`*csc* prints all the characters in the string *s* in typewriter style without translation. The character *c* can be any character that does not appear in *s*. For example, "`\verb#/\/\#`" yields `/\/\`. The command `\verb*` has the same effect, but translates each blank character to "␣".

Shapes

Description	Command	Declaration	Example
Upright	`\textup`	`\upshape`	Upright letters.
Italic	`\textit`	`\itshape`	*Italic letters.*
Slanted	`\textsl`	`\slshape`	*Slanted letters.*
Small capitals	`\textsc`	`\scshape`	SMALL CAPITAL LETTERS.

Series

Description	Command	Declaration	Example
Medium	`\textmd`	`\mdseries`	Medium upright letters.
Bold	`\textbf`	`\bfseries`	**Bold upright letters.**

Families

Description	Command	Declaration	Example
Roman	`\textrm`	`\rmfamily`	Roman letters.
Sans serif	`\textsf`	`\sffamily`	Sans serif letters.
Typewriter	`\texttt`	`\ttfamily`	`Typewriter letters.`

Figure 6: Fonts: shapes, series, and families

roman medium upright
roman medium italic
roman medium slanted
ROMAN MEDIUM SMALL CAPS
roman bold face upright
roman bold face italic
roman bold face slanted

`typewriter medium upright`
`typewriter medium italic`
`typewriter medium slanted`
`TYPEWRITER MEDIUM SMALL CAPS`

sans serif medium upright
sans serif medium slanted
sans serif bold face upright

Figure 7: The styles that work

3.3.3 Fonts

Changing fonts is a cloudy area of LaTeX2e. The cloud does have a silver lining: it is difficult to use LaTeX2e to produce the kind of multi-font mish-mash that one so often sees from inferior word processors (and amateur typographers). One of the reasons for the complexity is that LaTeX2e has quite sophisticated facilities for managing fonts: this manual uses 64 different fonts.

Unless you tell it otherwise, LaTeX2e uses the Computer Modern series of fonts designed by Donald Knuth with assistance by Hermann Zapf. Most people use these fonts and, as a result, LaTeX2e documents are instantly recognizable. The alternatives below are available on the department's computers.

- ◇ The American Mathematical Society commissioned Hermann Zapf to design a series of fonts specifically for mathematics; these are called the *Euler fonts* after Leonhard Euler (1707–83). The Euler fonts are often used with the *Concrete fonts*, a variation of Computer Modern. The

book *Concrete Mathematics* [GKP89] is set in Euler and Concrete fonts. You can obtain the Euler and Concrete fonts by writing the following comand in the preamble:

```
\usepackage{concrete}
```

- ◇ You can also write

  ```
  \usepackage{pandora}
  ```

  in the preamble. The Pandora fonts are also a unified set of fonts for both prose and mathematical type setting.
- ◇ As mentioned in Section 2, the program `pslatex` makes LaTeX2e use PostScript fonts rather than Computer Modern fonts.

3.3.4 Footnotes

Footnotes are easy to obtain in LaTeX2e: write `\footnote{`*note*`}` in the main text at the point where the footnote should be referenced. The text *note* is the text of your footnote. By default, footnotes are numbered consecutively in arabic $(1, 2, 3, \ldots)$ for the whole document (that is, the numbering does not restart on each page). It is possible to use symbols for footnotes: see [Lam86] or [Lam95].

3.3.5 Environments

An "environment" in LaTeX2e is a construction that begins with `\begin{`*name*`}` and ends with `\end{`*name*`}`. There are a number of built-in environments and Section 8.3 explains how you can define your own environments. Figure 8 shows some of the environments provided by LaTeX2e.

The environment `document` contains everything that you want printed; this is why you follow the preamble with `\begin{document}` and end your file with `\end{document}`. The environment `center`, as you would expect, centers text. If you write in your input file

```
\begin{center}
This is the first line of centered text, \\
and this is the second.
\end{center}
```

then LaTeX2e will print

This is the first line of centered text,
and this is the second.

The `verbatim` environment is similar to the `\verb` command, but allows you to display several lines of typewriter text exactly as you input it. Start a new line and write just `\begin{verbatim}` on it. Then write as much text as you want, using any characters you need. Finally, on a line by itself, write `\end{verbatim}`.

The `verbatim` environment does not work inside other environments. For example, you cannot centre a `verbatim` environment. It is tempting to use the `\verbatim` environment whenever you cannot figure out how to make LaTeX2e do something. Don't: there is no point in using a powerful type-setting program to mimic a typewriter.

Name	Description	Section
`array`	Math arrays	6.3
`center`	Centered text	3.3.5
`description`	List with named items	4
`document`	The entire document	3.3.5
`enumerate`	List with numbered items	4
`eqnarray`	Equations	6.4
`figure`	Floating figures	7
`itemize`	List with bullet items	4
`list`	Basic list	8.3
`minipage`	Small "page"	3.3.5
`picture`	Primitive diagram	9.2
`quotation`	Long quotation	3.3.5
`quote`	Short quotation	3.3.5
`tabbing`	Tab definition and use	6.1
`table`	Floating table	7
`tabular`	Table with text	6.2
`verbatim`	Text as entered	3.3.5

Figure 8: Some of the environments provided by LaTeX2e

The `alltt` environment is similar to the `verbatim` environment except that the characters `\`, `{`, and `}` are processed in the usual way; consequently, commands in an `alltt` environment are executed. If you use `alltt`, you must include the command `\usepackage{alltt}` in the preamble.

> Use the `quote` environment or the `quotation` environment for quotations. This text was obtained by writing `\begin{quote} Use the . . . paragraph.\end{quote}`. Use `quote` for short quotations and `quotation` for quotations that contain more than one paragraph.

The `minipage` environment produces, as its name suggests, a "mini page" within the document. It is useful for creating areas that may contain text with paragraph breaks, figures, tables, and so on. A minipage environment has an optional parameter for alignment (`t` for top, `b` for bottom) and a required parameter for its width. The text on the left below produces the minipage on the right. The `\fbox` command, which encloses its argument in a box, is used here to show the size of the minipage.

```
\fbox{\begin{minipage}[t]{2.5in}
This minipage is 2.5 inches
wide and aligned at the top.

In a minipage, paragraphs
are set without indentation
or extra vertical space.
\end{minipage}}
```

This minipage is 2.5 inches wide and aligned at the top.
In a minipage, paragraphs are set without indentation or extra vertical space.

3.3.6 Sections

You can divide your document into parts, chapters, sections, subsections, subsubsections, paragraphs, and subparagraphs. For each level of the hierarchy, there is a command with the corresponding name. For example, you are now reading a subsubsection introduced by the command `\subsubsection{Sections}`.

The commands `\part` and `\chapter` cannot be used with the `article` style but are available in the `book` and `report` styles.

3.3.6.1 Controlling the Numbering.

This section was produced by a `\paragraph` command. The preamble of this document contains these commands:

```
\setcounter{secnumdepth}{5}
\setcounter{tocdepth}{2}
```

The following subparagraphs explain the effect of these commands.

3.3.6.1.1 Section Numbering.

The counter `\secnumdepth` controls the level to which parts are numbered. The default value is 3, which means that sections, subsections, and subsubsections are numbered but paragraphs and subparagraphs are not numbered. In this document, `\secnumdepth` is set to 5, which means that paragraphs and subparagraphs (such as this one) are numbered as well.

If a section command name is followed by *, LaTeX2e does not generate a number or an entry in the table of contents. For example, the command `\subsection*{The Joy of \TeX}` writes a subsection title "The Joy of TeX" without a section number.

3.3.6.1.2 Entries in the Table of Contents.

The counter `\tocdepth` determines what will appear in the table of contents. The default value is 3, which means that sections, subsections, and subsubsections are listed. In this document, it is set to 2, so that only sections and subsections entries appear in the table of contents.

3.3.7 Pages

TeX is quite good at deciding where to start a new page. If you have to force a page break, use one of the following commands.

- ◇ `\pagebreak` is a strong hint that this is a good place to start a new page. In fact, `\pagebreak` on a line by itself will always start a new page. `\pagebreak` in the middle of a paragraph, however, will not start a new page until the end of the paragraph (assuming that the paragraph fits onto the page). If `\pagebreak` appears close to a natural page break, LaTeX2e will stretch the text a little so that bottom margins match.
- ◇ `\newpage` will create a short page without stretching text. If there are waiting floats (see Section 7) that fit onto the current page, they will be included.
- ◇ `\clearpage` is like `\newpage` but does not put waiting floats on the current page.

ˆ	\textasciicircum
˜	\textasciitilde
\	\textbackslash
|	\textbar
•	\textbullet
©	\textcopyright
$	\textdollar
...	\textellipsis
>	\textgreater
<	\textless
®	\textregistered
£	\textsterling
TM	\texttrademark
␣	\textvisiblespace

Table 1: LaTeX2e text-mode commands

Front material, such as the table of contents, usually has roman page numbers ($i, ii, iii, \ldots$) and the body of the document has arabic page numbers ($1, 2, 3, \ldots$). To set the page number style and also initialize the page counter to 1, use either `\pagenumbering{roman}` or `\pagenumbering{arabic}`.

There are two commands for setting the page style. The command `\pagestyle` is used in the preamble to set the page style for the entire document. The command `\thispagestyle` determines the style for the current page. Both commands expect an argument.

- ◇ `\pagestyle{plain}` gives you page numbers centered at the bottom of the page and no headings. It is the default page style.
- ◇ `\thispagestyle{empty}` says that the current page should have no headings or page number. You can use this for title pages.
- ◇ `\pagestyle{headings}` tells LaTeX2e to write section headings and page numbers at the top of each page. It is the page style used by this document.
- ◇ `\pagestyle{myheadings}` tells LaTeX2e that you will define your own headings. To do so, write `\markright{`*heading text*`}`. The *heading text* will be written at the top of every page, with the page number on the right. You can use `\markright` commands within the text to change the heading, but it is sometimes hard to get them in the right place.
- ◇ If you are printing on both sides of the page, you may want different headers on left and right pages. To obtain this, write `\markboth{`*left heading text*`}{`*right heading text*`}`.

If you want more elaborate headers or footers, consider using the package `fancyheadings`.

3.3.8 Common Characters

Table 1 shows some LaTeX2e text-mode commands. Although there are short versions for many of them, the long versions are sometimes preferable. For example, the symbol `<` is font-dependent, but `\textless` is not.

3.4 Cross References

The command `\label{`*name*`}` creates a cross-reference entry consisting of *name*, which is a name that you choose, a position number, and a page number. Elsewhere in the document, you can refer to the name by writing `\ref{`*name*`}` and LaTeX2e will print the position number. Alternatively, you can write `\pageref{`*name*`}` and LaTeX2e will write the page number.

For example, the heading for this section is:

```
\subsection{Cross References}\label{sec:xref}\index{cross-referencing}
```

The `\subsection` command writes the heading; the `\label` command sets the label "`sec:xref`" to "3.4", and the `\index` command creates an entry in the index. You can use the label value by writing `Section~\ref{sec:xref}`, which yields "Section 3.4", or by writing `page~\pageref{sec:xref}`, which yields "page 14".

The value of the "position number" depends on the location of the `\label` command. The smallest possible scope is used. Within a section, the label will be the section number. Within a subsection, the label will be the subsection number. If the label appears after `\item` in an enumerated list, its value will be the number of the list item.

4 Making Lists

LaTeX2e provides several kinds of list-making environments, and you can also define your own. A list has the following structure.

- A command of the form `\begin{`*kind*`}`, where *kind* is one of: `itemize`, for bullet lists; `enumerate`, for numbered lists; or `description`, for definition lists.
- One or more list items, each beginning with `\item`.
- A command of the form `\end{`*kind*`}`, where *kind* is the same as at the beginning of the list.

Figure 9 shows the text used to generate the bullet list that you have just read. The spacing and layout in the input file have no effect on the appearance of the printed list. In particular, you can use indentation to clarify the structure of the input file, as if it were a program.

The `\item` command takes an optional argument: if you write `\item[$\circ$]` in an `itemize` list, you will get a circle instead of a bullet. In a `description` list, every item should have an argument.

Section 8.3 gives the LaTeX2e definition for the diamond lists in this document and explains how you can define your own list-making environments.

```
\begin{itemize}
  \item A command of the form \verb"\begin{"\textit{kind}\verb"}",
        where \textit{kind} is one of \verb"itemize",
        \verb"enumerate", or \verb"description".
  \item One or more list items, each beginning with \verb"\item".
  \item A command of the form \verb"\end{"\textit{kind}\verb"}",
        where \textit{kind} is the same as at the beginning of the
        list.
\end{itemize}
```

Figure 9: Input text for an itemized list

5 Math Mode

TEX is intended for "the creation of beautiful books — and especially for books that contain a lot of mathematics" [Knu86b, page v]. This section introduces a few of the basic principles of mathematical typesetting.

A small piece of mathematical text is enclosed between $ signs. For example, `$x^2+y^2=r^2$` produces $x^2 + y^2 = r^2$. If you want "display math", centered on a line by itself, use two $ signs, as in Figure 10. TEX provides a large selection of math symbols. Figure 11 shows the symbols that you are most likely to need.

$$\mathcal{L}\left\{\frac{\partial w}{\partial x}\right\} = \int_0^\infty e^{-st}\,\frac{\partial w}{\partial x}\,dt = \frac{\partial}{\partial x}\int_0^\infty e^{-st} w(x,t)\,dt = \frac{\partial}{\partial x}\mathcal{L}\{w(x,t)\}.$$

```
$$ {\cal L} \left\{ \frac{\partial w}{\partial x} \right\}
=  \int_0^{\infty} e^{-st} \, \frac{\partial w}{\partial x} \, dt
= \frac{\partial}{\partial x} \int_0^{\infty} e^{-st} w(x,t) \, dt
= \frac{\partial}{\partial x} {\cal L} \{ w(x,t) \} . $$
```

Figure 10: An equation [Kre79, page 559] and the TEX input used to generate it

In math mode, TEX ignores all spaces in the input. Spaces are not required except when their omission would make TEX misunderstand a command. The space in `$\sin x$` (which yields $\sin x$) is required; if you omitted it, TEX would complain that `\sinx` was not defined.

TEX associates a category with each character and uses the category to determine spacing. For example, there is more space in $x = y$ than in (z) because "=" is a relational symbol but "(" and ")" are delimiters. You can make fine adjustments to the spacing using the commands `\;` (thick space), `\:` (medium space), `\,` (thin space), and `\!`(negative thin space). The space between the following bars is thick, medium, and thin, respectively: ||||.

The input `$\{x\mid P(x)\}$` yields $\{x \mid P(x)\}$. You might feel that the braces are too close to the text they enclose; if so, you could add thin space, as in `$\{\,x\mid P(x)\,\}$`, which yields $\{\, x \mid P(x) \,\}$. You can use negative space to create new characters: `$\mu\,[\![E]\!]$` yields $\mu\,[\![E]\!]$.

$\pm$	\pm	$\bigcirc$	\bigcirc	$\leftrightarrow$	\leftrightarrow
$\times$	\times	$\dagger$	\dagger	$\Leftrightarrow$	\Leftrightarrow
$\div$	\div	$\ddagger$	\ddagger	$\uparrow$	\uparrow
$*$	\ast	$<$	<	$\downarrow$	\downarrow
$\star$	\star	$\leq$	\le	$\Updownarrow$	\Updownarrow
$\circ$	\circ	$>$	>	$\nearrow$	\nearrow
$\bullet$	\bullet	$\geq$	\ge	$\swarrow$	\swarrow
$\cdot$	\cdot	$=$	=	$\rightsquigarrow$	\leadsto
$\cap$	\cap	$\neq$	\ne	$\top$	\top
$\cup$	\cup	$\subset$	\subset	$\bot$	\bot
$\vee$	\vee, \lor	$\subseteq$	\subseteq	$\forall$	\forall
$\wedge$	\wedge, \land	$\in$	\in	$\exists$	\exists
$\neg$	\neg, \lnot	$\supset$	\supset	$\backslash$	\backslash
$\Longleftrightarrow$	\iff	$\supseteq$	\supseteq	∞	\infty
$\setminus$	\setminus	$\equiv$	\equiv	$\Box$	\Box
$\diamond$	\diamond	$\sim$	\sim	$\Diamond$	\Diamond
$\triangleleft$	\triangleleft	$\simeq$	\simeq	$\triangle$	\triangle
$\triangleright$	\triangleright	$\approx$	\approx	$\clubsuit$	\clubsuit
$\oplus$	\oplus	$\cong$	\cong	$\diamondsuit$	\diamondsuit
$\ominus$	\ominus	$\leftarrow$	\leftarrow	$\heartsuit$	\heartsuit
$\otimes$	\otimes	$\rightarrow$	\rightarrow	$\spadesuit$	\spadesuit
$\oslash$	\oslash	$\longrightarrow$	\longrightarrow	$\S$	\S
$\odot$	\odot	$\Leftarrow$	\Leftarrow	$\copyright$	\copyright
$\lfloor$	\lfloor	$\lceil$	\lceil	$\langle$	\langle
$\rfloor$	\rfloor	$\rceil$	\rceil	$\rangle$	\rangle

Most of these symbols can be used in math mode only. The space around a symbol depends on whether it is an operator, a relation, or a delimiter. The table shows only some of the arrows; the others have analogous names.

Figure 11: Assorted symbols

TEX believes that the symbols < and > are relations, not delimiters. If you use them as delimiters, the spacing will be wrong, as in $< x, y >$. The angle brackets, `\langle` and `\rangle`, are classified as delimiters: `$\langle x,y\rangle$` yields $\langle x, y\rangle$. If you want to use < and > as brackets, include the following definitions in the preamble (see Section 8):

```
\newcommand{\la}{\mathopen{<}}
\newcommand{\ra}{\mathclose{>}}
```

With these definitions in place, `$\la x,y\ra$` expands to $<x, y>$. The commands `\mathopen` and `\mathclose` define the symbols to be opening and closing math delimiters and determine the spacing that TEX will use in printing them.

In math mode, TEX spaces letters on the assumption that they are products of single-letter variable names. For example, `$affix$` yields $affix$ rather than *affix*. Words in math mode should be enclosed in an "mbox", with appropriate font changes if necessary. To obtain *affix* in math mode, write `$...\mbox{\itshape affix}...$`.

$\alpha, \ldots, \omega$	`\alpha, \ldots, \omega`
$\Gamma, \Delta, \ldots$	`\Gamma, \Delta, \ldots`
$\mathcal{A}, \ldots, \mathcal{Z}$	`{\cal A,\ldots,Z}`
$1+2+3+\cdots+N$	`1+2+3+\cdots+N`
a_k	`a_k`
$\Gamma^a_{b,c}$	`\Gamma_{b,c}^a`
e^{-kx^2}	`e^{-kx^2}`
e^{-kx^2}	`e^{\textstyle-kx^2}`
$\lvert\sin x\rvert \le 1$	`|\sin x| \le 1`
$\sum_{i=0}^{i=\infty} \frac{1}{2^n}$	`\sum_{i=0}^{i=\infty} \frac{1}{2^n}`
$\displaystyle\sum_{i=0}^{i=\infty} \frac{1}{2^n}$	`\displaystyle\sum_{i=0}^{i=\infty} \frac{1}{2^n}`
$\frac{x-y}{x+y}$	`\frac{x-y}{x+y}`
$\displaystyle\frac{x-y}{x+y}$	`\displaystyle\frac{x-y}{x+y}`
$\sqrt{s(s-a)(s-b)(s-c)}$	`\sqrt{s(s-a)(s-b)(s-c)}`
$E \stackrel{*}{\rightarrow} K$	`E \stackrel{*}{\rightarrow} K`
$\overline{x^2+y^2}$	`\overline{x^2+y^2}`
$\underline{A+B}$	`\underline{A+B}`
$\vec{x} - \hat{y}$	`\vec{x} - \hat{y}`
$\widehat{A+B}$	`\widehat{A+B}`
$x_1+\underbrace{x_2+\cdots+x_{n-1}}_{n-2}+x_n$	`x_1+\underbrace{x_2+\cdots+x_{n-1}}_{n-2}+x_n`
$\sum_{\substack{0\le i<M\\0\le j<N}} \binom{i}{j}$	`\sum_{0\le i<M \atop 0\le j<N} {i \choose j}`

Figure 12: Examples of mathematical text

Figure 11 shows an assortment of mathematical and other symbols. A few of the more obscure symbols are not built into LaTeX2e; if you want to use them, include `\usepackage{latexsym}` in the preamble. The obscure symbols are: `\lhd`, ($\lhd$), `\rhd`, ($\rhd$), `\unlhd`, ($\unlhd$), `\unrhd`, ($\unrhd$), `\sqsubset`, ($\sqsubset$), `\sqsupset`, ($\sqsupset$), `\Join`, ($\bowtie$), `\leadsto`, ($\leadsto$), `\mho`, ($\mho$), `\Box`, ($\Box$), and `\Diamond`, ($\Diamond$).

Figure 12 provides some examples of mathematical typesetting. Most of the commands work in combinations: note, for example, the use of nested superscripts. TeX formats material between single $ signs in `\textstyle` and material between double $ signs in `\displaystyle`. You can write these commands explicitly to change the style: Figure 12 includes examples of display style.

The brackets $ . . . $ and $$ . . . $$ write equations without numbers. If you want numbered equations, use the `equation` environment. Equations can be labelled for cross-referencing (see Section 3.4). Equation (1) below is obtained by writing

```
\begin{equation} \label{imagine} e^{-i\pi} = -1 \end{equation}
```

and is referenced by writing `(\ref{imagine})`.

$$e^{-i\pi} = -1 \tag{1}$$

Bracket symbols (`()`, `[]`, and `{}`) and certain others will stretch to the size required to enclose a complex expression. To achieve this effect, write `\left` before the opening bracket and `\right` before the closing bracket. For example, the input `$$ \left(1 + \frac{x}{y} \right)^{2k+1} $$` creates

$$\left(1 + \frac{x}{y}\right)^{2k+1}$$

The commands `\left` and `\right` must be used in pairs; if you don't want one of the brackets, write a period instead, as in `\left.` or `\right.`. For example, the input

```
$$ f(x) = \left\{ \begin{array}{ll}
   \sin x/x, & \mbox{if $x \ne 0$;} \\
   0,        & \mbox{otherwise.}
\end{array} \right. $$
```

creates

$$f(x) = \begin{cases} \sin x/x, & \text{if } x \ne 0; \\ 0, & \text{otherwise.} \end{cases}$$

You should not use the style commands of Section 3.3.2 in math mode. If you want to use *single letters* in a particular font in math mode, use one of these commands:

```
\mathrm     \mathsf     \mathtt     \mathit     \mathbf     \mathcal
```

For example, `$\mathbf{x}\cdot\mathbf{y}=\mathsf{T}^i_{jk}$` produces $\mathbf{x} \cdot \mathbf{y} = \mathsf{T}^i_{jk}$.

6 Vertical Alignment

LaTeX2e provides several ways of aligning text vertically. The simplest, but least useful, is the `tabbing` environment, described next. For most purposes, the `tabular` environment, described in Section 6.2, is better. The `array` environment, described in Section 6.3, is similar to the `tabular` environment, but it is intended for mathematical text.

6.1 The `tabbing` Environment

Within a tabbing environment, `\=` defines a tab stop, `\>` moves to the next tab stop, and `\\` indicates a line break. Figure 13 shows a simple example of a `tabbing` environment.

This scheme doesn't work very well if the columns in the first row are shorter than the columns in subsequent rows. If you write `\kill` at the end of a line instead of `\\`, the line will not be printed but any tab definitions that it contains remain active. Figure 14 shows a tabbing environment with a `\kill`ed line. LaTeX2e moves to the defined tab position even if doing so requires moving backwards; this causes some of the letters to be superimposed in Figure 14.

```
\begin{tabbing}
Telephone: \= 848 3012 \\
Fax: \> 848 2830 \\
\end{tabbing}
```

Telephone: 848 3012
Fax: 848 2830

Figure 13: A simple `tabbing` environment and the formatted output

```
\begin{tabbing}
xxxxx \= xxxxx \= xxxxx \= xxxxx \= \kill
Fix \> the \> fax \> before \> the \\
fox \> discombobulates \> it. \\
\end{tabbing}
```

Fix the fax before the
fox discombobulates it.

Figure 14: Another `tabbing` environment and the formatted output

6.2 The `tabular` Environment

The `tabular` and `array` environments provide powerful facilities for formatting tables, arrays, and other kinds of aligned text. Figure 17 shows several objects created by `tabular` and `array` environments.

The basic form of a `tabular` environment is:

```
\begin{tabular}[pos]{cols}
  rows
\end{tabular}
```

A `tabular` environment produces a TEX "box" and behaves like a single, very large character. This means that you can put tables side by side simply by writing one after the other. The *pos* argument specifies the vertical positioning of the table and is useful only when there is something else on the same line as the table — such as another table. If it is omitted, the centres of the tables are aligned. If it is present, it must be `t` (align the tops) or `b` (align the bottoms).

The *cols* argument defines the format of each column and the text between columns. For simple tables, the column specifier is `l` (left of column), `c` (centered in column), or `r` (right of column). The intercolumn specifier is `|` (vertical rule) or omitted (no rules between columns). For example, `|cc|` specifies a vertical rule, two centered columns with intercolumn space between them, and another vertical rule.

A *row* consists of the text for each column of the row, with `&` between columns and `\\` to indicate the end of the row. You can also use `\hline` to draw a horizontal rule between rows. Figure 15 shows a simple table constructed using these rules. Note that "thin space" (`\,`: see Section 5) is preferable to a comma for numbers greater than 999.

The column specifier `p{`*width*`}` creates a column with the given width set in paragraph mode. For example, a column specified by `p{2in}` contains text in a column 2 inches wide.

The intercolumn specifier `@{`*text*`}` creates a column separator containing the given text. If you don't want any space between columns, use `@{}` as the intercolumn specifier. If you want a vertical line as part of the intercolumn text, use `@{...\vline...}`, not `@{...|...}`. In Figure 16, one of the intercolumn specifiers is `@{.}`, which has the effect of aligning the decimal points of the numbers.

```
\begin{tabular}{| l | r |} \hline
Description & Quantity    \\ \hline
Hook        & $354$       \\
Line        & $1\,476$    \\
Sinker      & $23\,749$   \\ \hline
\end{tabular}
```

Description	Quantity
Hook	354
Line	1 476
Sinker	23 749

Figure 15: A simple `tabular` environment and the formatted output

You can create a row item that spans several columns by writing

`\multicolumn{`*num*`}{`*col*`}{`*item*`}`

The field *num* specifies the number of columns to span. The field *col* is a column specifier for the item. The field *item* contains the item itself. Figure 16 uses spanning items for the headings of the table.

```
\begin{tabular}{| p{10pc} | r @{.} l | l |} \hline
\multicolumn{1}{|c|}{Description} &
\multicolumn{2}{|c|}{Value} &
\multicolumn{1}{|c|}{Unit} \\ \hline\hline
Gravitational constant & 0 & 00000006673
                       & $\mbox{cm}^3/\mbox{g sec}^2$ \\
Velocity of light      & $299\,792$ & 4562
                       & $\mbox{km}/\mbox{sec}$ \\
Reciprocal fine-structure constant, $hc/e^2$
                       & 137 & 0360 & \\ \hline
\end{tabular}
```

Description	Value	Unit
Gravitational constant	0.00000006673	$\mathrm{cm}^3/\mathrm{g\ sec}^2$
Velocity of light	299 792.4562	km/sec
Reciprocal fine-structure constant, hc/e^2	137.0360	

Figure 16: Another `tabular` environment and the formatted output

You will have noticed that in Figure 16 the superscripts in the first row are too close to the top of the bounding box. You can correct this by writing `\rule{0pt}{12pt}` anywhere in the same row. The command `\rule{`w`}{`h`}` creates a rectangular blob of ink with width w and height h. Setting h to 12 points pushes the top line of the box up a bit; setting w to 0 points ensures that you cannot see the blob: an invisible blob is called "strut". The value "12" is obtained by trial and error. Section 8.1 explains "points".

Concordia
UNIVERSITY

$$\begin{array}{rcl} f(x) & = & x^3 \text{ where } -\pi < x < \pi \\ & & \quad \text{and } f(x+2\pi) = f(x) \\ g(x) & = & e^{\lambda(1-x^2)} \text{ where } -1 < x < 1 \\ & & \quad \text{and } g(x+2) = g(x) \end{array}$$

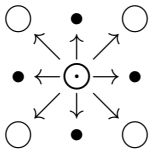

Figure 17: Examples of `tabular` and `array` environments

```
\begin{figure}
\hrule\vskip 4pt
\begin{tabular}{@{}c@{}}
\Huge Concordia \\ \hline
\rule{0pt}{11pt} \large U\hfill N\hfill I\hfill V\hfill E\hfill
                        R\hfill S\hfill I\hfill T\hfill Y
\end{tabular}
\hfill
$ \begin{array}{rcl}
f(x) & = & x^3  \begin{array}[t]{ll}
                   \mbox{where} & -\pi < x < \pi \\
                   \mbox{and}   & f(x + 2\pi) = f(x)
                 \end{array} \\
g(x) & = & e^{\lambda(1-x^2)}  \begin{array}[t]{ll}
                                   \mbox{where} & -1 < x < 1 \\
                                   \mbox{and}   & g(x + 2) = g(x)
                                 \end{array} \\
\end{array} $
\hfill
$ \renewcommand{\arraystretch}{0.8}
\begin{array}{c@{}c@{}c@{}c@{}c}
\bigcirc &            & \bullet    &             & \bigcirc \\
         & \nwarrow   & \uparrow   & \nearrow    &          \\
\bullet  & \leftarrow & \bigodot   & \rightarrow & \bullet  \\
         & \swarrow   & \downarrow & \searrow    &          \\
\bigcirc &            & \bullet    &             & \bigcirc \\
\end{array} $
\caption{Examples of \texttt{tabular} and \texttt{array} environments}
\label{fig:tab}
\vskip 6pt\hrule
\end{figure}
```

Figure 18: The source code for Figure 17

6.3 The `array` Environment

The `array` environment is essentially the same as the `tabular` environment. The main difference is that an `array` environment can only be used in math mode (between `$...$` or `$$...$$`). All of the features of column specifiers provided by the `tabular` environment are also available in `array` environments. The objects at the centre and right of Figure 17 were produced by the `tabular` and `array` environments shown in Figure 18.

6.4 The eqnarray Environment

The `eqnarray` and `eqnarray*` environments provide a simple way of constructing array environments in the common case of equations centered around a single operator. They are identical except that `eqnarray` numbers the equations and `eqnarray*` does not. Each line should contain three expressions separated by `&` and each line but the last should end with `\\`. These environments are used in text mode, *not* in math mode. Figure 19 shows the input for an `eqnarray*` environment and the output produced.

```
\begin{eqnarray*}
\xi & = & \theta - \theta_0 \\
    & = & \xi_0\cos\left(
            \sqrt{\frac{gb}{l}+\epsilon}
          \right)
\end{eqnarray*}
```

$$\begin{aligned}\xi &= \theta - \theta_0 \\ &= \xi_0 \cos\left(\sqrt{\frac{gb}{l} + \epsilon}\right)\end{aligned}$$

Figure 19: An `eqnarray*` environment and the formatted output

7 Floats

A "float" is a piece of text, usually a figure or a table, that must not be split between two pages. The text in a float is not necessarily printed where it occurs in the input file; instead, LaTeX2e prints it on the next occasion where there is enough space for it. The `figure` and `table` environments produce floating text. The floats in this document have the structure shown below.

```
\begin{figure}[pos]
\hrule                  % optional rule
\vskip 6pt              % space below rule
\begin{center}          % optional centering
  text of figure
\end{center}            % optional centering
\caption{title}
\label{name}
\vskip 6pt              % space above rule
\hrule                  % optional rule
\end{figure}
```

The position field, *pos*, consists of any combination of the letters `h` (here), `t` (top of page), `b` (bottom of page), and `p` (on a page containing figures only). If you omit the position field and the square brackets, LaTeX2e assumes `[tbp]`. Sometimes, in spite of all your efforts, LaTeX2e refuses to put the float where you want it. If this happens, including `!` in the *pos* field will make LaTeX2e "try harder" [Lam95, page 197]. The *text of figure* may contain anything acceptable to LaTeX2e.

The command `\caption{`*title*`}` prints "Figure *n*:" followed by *title*. Short captions are centered and long captions are printed as running text. The value of *n* will be 1 for the first figure, 2 for the second, and so on. The command `\label{`*name*`}` assigns the value of *n* to the string *name*. The `\label` command should either come after the `\caption` command or be included in the argument of the `\caption` command.

If you change both occurrences of `figure` to `table`, the effect will be exactly the same except that the caption will say "Table n:". Figures and tables are numbered separately.

If you do not want the figure centered, omit the lines labelled "optional centering". If you do not want rules above and below your figures, omit the lines labelled "optional rule" and "spacing".

Write `\listoffigures` if you want a list of figures and `\listoftables` if you want a list of tables.

8 The Preamble

The text between `\documentclass` and `\begin{document}` is called the *preamble*. The preamble contains definitions that apply to the entire document. Sections 8.2 and 8.3 describe these definitions. Writing good preambles requires some knowledge of dimensions and glue, which are described next.

8.1 Dimensions and Glue

Figure 20 shows the units of measurement provided by TEX. You won't need most of them: it is simplest to use inches for large dimensions and points for small dimensions, although there are a few occasions when other units are useful. TEX does all of its length calculations in integer "scaled points". Since a scaled point is 5.36×10^{-7} cm (about 1% of the wavelength of light), you will not notice rounding errors. The only problem with such a small unit is that it limits the size of documents that you can produce. If your pages are more than 16384 pt (about 18 feet) long, you should consider using another text processor.

Abbreviation	Name	Relations
`pt`	point	$1\,\text{pt} \approx 0.013837\,\text{in}$
`pc`	pica	$1\,\text{pc} = 12\,\text{pt}$
`in`	inch	$1\,\text{in} = 72.27\,\text{pt}$
`bp`	big point	$1\,\text{in} = 72\,\text{bp}$
`cm`	centimetre	$1\,\text{in} = 2.54\,\text{cm}$
`mm`	millimetre	$1\,\text{cm} = 10\,\text{mm}$
`dd`	didot point	$1\,\text{dd} = \frac{1238}{1157}\,\text{pt}$
`cc`	cicero	$1\,\text{cc} = 12\,\text{dd}$
`sp`	scaled point	$1\,\text{pt} = 65536\,\text{sp}$

Figure 20: Units of Measurement

Whenever TEX expects a dimension, you should provide a number followed by one of the unit abbreviations in the left column of Figure 20. The unit is required even if the length is zero: write `0pt` or `0in`, not just `0`.

There are two dimensions that are not mentioned in Figure 20 because they do not have fixed sizes. They are the `em` and the `ex`. Their size depends on the current font: `em` is a horizontal dimension and `ex` is a vertical dimension. The line ⎯ is 1 em wide and the line ı is 1 ex high in the current

font. If you use multiples of these units for horizontal and vertical dimensions of your document, the actual size will change if you change the base font size.

The space in TEX documents is filled with *glue*. Glue may be rigid or elastic. The glue between paragraphs in this document, for example, is elastic. The variation helps TEX to choose page breaks at sensible places. Paragraph conventions for this document are specified by the following definitions in the preamble:

```
\parskip        6pt plus 3pt minus 2pt
\parindent      0pt
```

The first declaration says that `\parskip`, the glue between paragraphs, defaults to 6 points, but it can be stretched to as much as $6 + 3$ points or shrunk to as little as $6 - 2$ points. The second declaration says that the horizontal glue at the beginning of a paragraph is zero; in other words, there is no paragraph indent.

You can introduce horizontal glue into a document by writing `\hspace{`*length*`}` and vertical space by writing `\vspace{`*length*`}`. The parameter *length* can be in any units, but the units must be specified. These comands are "smart" in the sense that `\vspace` has no effect at a pagebreak and `\hspace` has no effect at a line break. If you want space at the top or bottom of a page, use `\vspace*{`*length*`}` and if you want space at the ends of a line, use `\hspace*{`*length*`}`.

There is a special kind of glue that is infinitely stretchable; it is called `\fill` and it is used in the commands `\vfill` and `\hfill`. The title page of this document uses `\vspace*{1in}` to provide space at the top and bottom of the page, and `\vfill` three times to provide space between the four centered blocks of text. There are also variants of `\hfill` that fill space with either a solid or a dotted line. The following line was produced by writing "`left\dotfill centre\hrulefill right`":

left ..... centre_____right

If you want to find out the value of a length, such as a LATEX2e default length, write `\the` in front of it. For example, `\the\parskip` yields "7.07185pt plus 4.71457pt minus 2.35728pt" in the formatted text.

Figure 21 shows the dimensions of a LATEX2e page. There are three regions of text: the header, the main body, and the footer. The dimensions are given by the names in the diagram except that

$$\begin{aligned} \textit{margin} &= \texttt{\textbackslash oddsidemargin} + 1\,\text{inch} && \text{(odd numbered pages)} \\ \textit{margin} &= \texttt{\textbackslash evensidemargin} + 1\,\text{inch} && \text{(even numbered pages)} \\ \textit{topmargin} &= \texttt{\textbackslash topmargin} + 1\,\text{inch} \end{aligned}$$

The lengths that appear after the names in Figure 21 apply to this document.

The only lengths that you will normally need to change from the defaults are the width and height of the text. For $8\frac{1}{2} \times 11$ inch paper, include the following definitions in the preamble. The actual margins will be 1 inch wide, as the formula given above for *margin* shows. You can specify the dimensions in TEX style, as on the left below, or in LATEX2e style, as on the right.

```
\textwidth    6.5in
\textheight     9in
\oddsidemargin  0in
\evensidemargin 0in
```

```
\setlength{\textwidth}{6.5in}
\setlength{\textheight}{9in}
\setlength{\oddsidemargin}{0in}
\setlength{\evensidemargin}{0in}
```

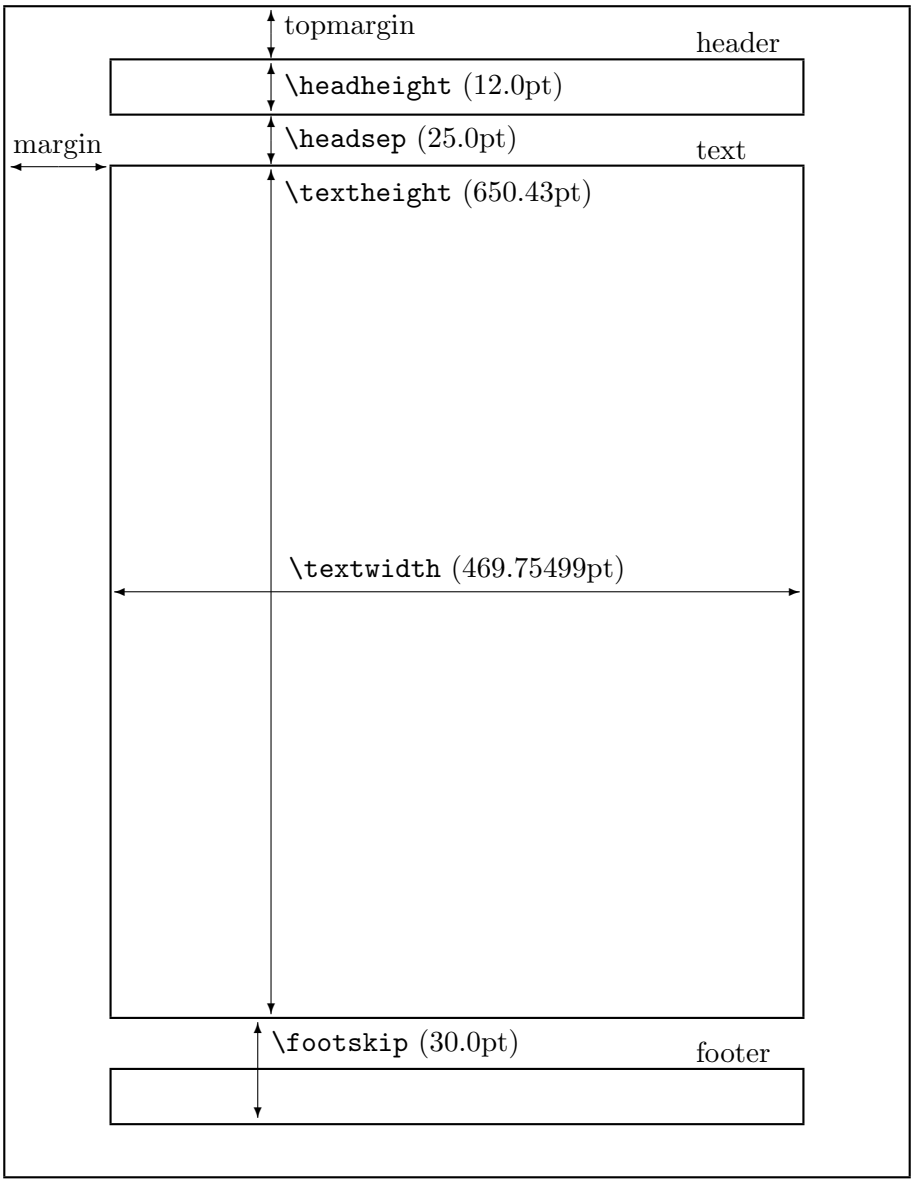

Figure 21: Page Dimensions

8.2 Defining Commands

You can define your own commands (or "macros") for a document. A command definition appears in the preamble and has the form

`\newcommand{\`*name*`}[`*n*`]{`*body*`}`

The *name* field is the name of the new command. The name must consist entirely of letters. The case of letters is significant. The field `[`*n*`]` is optional; if it is present, $n \geq 1$ gives the number of parameters of the command. The body of the command consists of arbitrary text. Within the text, `#`*n* refers to the *n*th argument. To invoke the command, write `\`, the command name, and

Def: `\newcommand{\xs}[1]{Section~\ref{#1}}`
Use: `\xs{sec:cmd}`
Out: Section 8.2

Def: `\newcommand{\kw}[1]{\ensuremath{\mbox{\texttt{#1}}}}`
Use: `Testing with \kw{if}: $\kw{if}\;(x>y)\ldots$`
Out: Testing with `if`: if $(x > y)\ldots$

Def:
```
\newcommand{\cpp}{\textsf{C}%
   \raise.22ex\hbox{\footnotesize +}%
   \raise.22ex\hbox{\footnotesize +}}
```
Use: `You either love {\cpp} or you hate it.`
Out: You either love C++ or you hate it.

Def: `\newcommand\setabstr}[2]{\ensuremath{\left\{\,#1\mid#2\,\right\}}}`
Use: `\setabstr{n}{K\le\sum_{i=1}^n\frac{1}{i}<K+1}`
Out: $\left\{ n \mid K \le \sum_{i=1}^{n} \frac{1}{i} < K + 1 \right\}$

Def: `\newcommand{\pd}[3]{\frac{\partial^{#3}#1}{\partial#2^{#3}}}`
Use: `\nabla^2\phi=\pd{\phi}{x}{2}+\pd{\phi}{y}{2}+\pd{\phi}{z}{2}`
Out: $\nabla^2\phi = \dfrac{\partial^2\phi}{\partial x^2} + \dfrac{\partial^2\phi}{\partial y^2} + \dfrac{\partial^2\phi}{\partial z^2}$

Def: `\newcommand{\all}[2]{\forall\,#1\mbox{\texttt{\LARGE .}}#2}`
Use: `$\all{x}{\all{y}{P(x)\Rightarrow P(y)\iff\lnot P(x)\lor P(y)}}$`
Out: $\forall\, x \,.\, \forall\, y \,.\, P(x) \Rightarrow P(y) \iff \neg P(x) \lor P(y)$

Figure 22: Examples of macros

the arguments, with each argument between braces:

`\`*`name`*`{...}{...}...`

The rather ugly table in Figure 22 shows some macros that I have found useful. Each entry in the table has three lines: "Def" shows the definition of the new command; "Use" shows an example of its use in a source file; and "Out" shows the formatted output.

The first macro, `\xs`, cross-references a section of the document. The word "Section" avoids inconsistent usage (such as "Section", "Sec.", and "section") and the tilde (`~`) prevents a line break before the number. You can define similar macros for referencing figures, tables, equations, etc.

The second macro, `\kw`, might be useful in an article about programming ("kw" is short for "keyword"). The `\ensuremath` ensures that `\kw` can be used in either text or math mode, the `\mbox` ensures that a keyword will not be split across two lines, and the `\texttt` says that keywords will be set in typewriter font. If you later decided that sans serif was more appropriate, you would simply change the definition.

The definition of `\cpp` uses the TEX commands `\raise` and `hbox` to produce a reasonable approximation to the "official" name as it appears in publications by AT&T.

The last three examples show how macro definitions can simplify math input. Note the use of `\left` and `\right` to obtain "expanding" brackets in `\setabstr`.

8.3 Defining Environments

You can define a new environment in the preamble by writing

`\newenvironment{`*name*`}[`*n*`]{`*opening*`}{`*closing*`}`

The field *name* is the name of the new environment. It does not begin with \, but *name* must not be a valid command. The field `[`*n*`]` specifies the number of arguments; it is omitted if there are no arguments. When the environment is used, LATEX2e will process the text *opening*, then the text between `\begin{`*name*`}` and `\end{`*name*`}`, and finally the text *closing*.

Environment arguments are referenced in the same way as command arguments, using `#`*n*, but in the *opening* text only.

Example 1: Defining an Environment

The text from the line above to the box below was generated by an environment defined as follows:

```
\newcounter{exnum}
\newenvironment{example}[1]{\penalty-2000\vskip1.5\parskip
\refstepcounter{exnum}{\bfseries Example \arabic{exnum}: #1}%
\penalty 10000}{$\Box$}
```

The purpose of this environment is to provide numbered examples in a document. The first line defines the new counter `exnum`. The next three lines define an environment called `example` with one argument. The TEX command `\penalty-2000` says that, if we are close to the bottom of the page, this would be a good place to start a new page. The `\vskip` command introduces vertical glue equivalent to 1.5 times the glue between paragraphs. `\refstepcounter{exnum}` increments the example counter and makes the counter a value that can be cross-referenced (see Section 3.4). Next comes the bold text that will introduce the example: "Example", the example number in arabic, and the example title. The TEX command `\penalty 10000` imposes the maximum penalty for a page break at this point; this ensures that there will almost never be a page break between the title and the body of the example. (LATEX2e uses the same trick to prevent a section title appearing at the bottom of a page.) Finally, the closing text in the last line prints a box immediately after the text of the example.

When a definition occupies several lines, as this one does, it is sometimes necessary to put `%` at the ends of the lines. If the `%` is omitted, TEX inserts blanks which at best spoil the appearance of the output and at worst cause obscure error messages. The example ends here → □

Figure 23 shows the definition of the diamond list used in this document. To define a new list, include `\begin{list}` in the *opening* text and `\end{list}` in the *closing* text. The command

```
\newenvironment{diamonds}{\begin{list}{$\diamond$}{%
\labelsep 0.7em   % Glue between label and text
\leftmargin 2em   % Relative position of left margin within list
\topsep 0pt       % Vertical glue above the list
\parsep 2pt plus 1pt minus 1pt     % Paragraph glue within list
\itemsep 2pt plus 1pt minus 1pt}}  % Glue between items
{\end{list}}
```

Figure 23: The environment definition for diamond lists

`\begin{list}{`*default-label*`}{`*declarations*`}`

creates a list environment with the given *default-label*. In Figure 23, the default label is a `\diamond`. The format of the list is determined by setting various spacing parameters in the *declarations* part.

9 Additional Features

There is not space in a short introduction to describe all of the features of LaTeX2e (let alone those of TeX!) completely. This section provides brief sketches of a few advanced features. You will need one of the books cited in Section 11 to use them seriously.

9.1 Packages

A "package" or "style file" is a file with suffix `.sty` containing LaTeX2e definitions. The command `usepackage` takes the name of a style file as its argument; `\usepackage` commands are usually placed directly after the `\documentclass` commmand. For example, if you have a bunch of neat definitions, put them into a file called `tricks.sty` and begin your documents with

```
\documentclass[11pt]{article}
\usepackage{tricks}
```

If you write your own style files, you should include them in the same directory as your LaTeX2e source file.

There are a number of useful LaTeX2e packages. Unfortunately, there is not space to describe them here, but you can read about them in *The LaTeX2e] Companion* [GMS94]. They include: `fancyheadings` for headers and footers; `floatfig` for narrow floating figures; `multicol` for multiple columns; `psfig` for including PostScript pictures (see Section 9.2); `supertab` for better tables; `varioref` for flexible cross-referencing; `verbatim` and `moreverb` for more sophisticated verbatim text; and many others. Most of the commonly-used packages have been installed on the department's servers. If you need a package that is not installed, look for it at `http://www.tug.org`, the web page of TUG, the TeX Users Group.

If you submit a technical paper to a conference or journal, you may find that there is a LaTeX2e style file already designed for you. Examples at the time of writing (February 2001) include:

- IEEE styles: `http://www-isl.stanford.edu/ieee`.

◇ Springer styles: http://www.springer.de/author/index.html#Books.

◇ ACM TOPLAS style: http://www.acm.org/toplas/style/index.html.

9.1.1 Sideways Printing

- To format the entire document in landscape mode, use

```
\usepackage[landscape]{geometry}
```

- To print a single page in landscape mode, use:

```
\usepackage{lscape}
....
\begin{landscape}
....
\end{landscape}
```

  To print figures and tables in landscape mode, use package `rotating`. The environments `sidewaystable` and `sidewaysfigure` act like `table` and `figure`, respectively, but print the float sideways.

See http://texblog.wordpress.com/2007/11/10/landscape-in-latex/.

9.1.2 Graphics

Use package `graphicx`. For each image, use the command `\includegraphics`. For example:

```
\includegraphics[width=5in]{pend-small.png}
\includegraphics[scale=3]{pend-small.png}
```

To make text flow around figures:

```
\usepackage{wrapfigure}
....
\begin{wrapfigure}{r}{0.5\textwidth}
  \begin{center}
    \includegraphics[width=0.48\textwidth]{gull}
  \end{center}
  \caption{A gull}
\end{wrapfigure}
```

Alignment can normally be either `l` for left, or `r` for right. Lowercase `l` or `r` forces the figure to start precisely where specified (and may cause it to run over page breaks), while capital `L` or `R` allows the figure to float. If you defined your document as `twosided`, the alignment can also be `i` for inside or `o` for outside, as well as `I` or `O`. The width is obviously the width of the figure.

See http://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions.

9.1.3 Euler Style

Use packages `beton` (for concrete fonts) and `euler` (for mathematical symbols).

9.1.4 Strikeout

Use package `ulem` and the command `\sout{...}`.

Commands in `ulem`:

`\uline`	Single underline
`\uuline`	Double underline
`\uwave`	Wavy underline
`\sout`	Line through text
`\xout`	Text marked over with slashes

See `http://www.mackichan.com/index.html?techtalk/531.htm~mainFrame`.

NOTE: Package `ulem` changes `\emph` so that it underlines!

9.2 Pictures

LaTeX2e can draw simple pictures: Figure 1 on page 2 and Figure 21 on page 25 of this document are examples. The commands for constructing pictures are quite extensive and will not be described here. You should consult [Lam86] or [Lam95] if you want to draw pictures using LaTeX2e.

LaTeX2e allows you to import pictures drawn using other programs. You can use `xfig` to draw diagrams in a X windows environment. The following notes outline the procedure for incorporating a picture drawn with `xfig` into a LaTeX2e file.

1. Create the diagram using `xfig`. Make sure that you create an encapsulated PostScript file as well as the normal `xfig` output.

   Assume that you have used `xfig` to create a diagram and to create the files `diag.fig` and `diag.eps`. You need `diag.fig` if you want to modify the diagram later. You need `diag.eps` to incorporate into the LaTeX2e input.

2. Invoke the package `psfig` by including this line in the preamble:

```
\documentclass{article}
\usepackage{psfig}
\begin{document}
\raisebox{-18pt}%
   {\psfig{figure=/pkg/misc/inputs/ConUlogo.eps,height=36pt}}
\quad
\begin{tabular}[t]{@{}c@{}}
\Huge Concordia\\ \hline
\rule{0pt}{11pt} \large U\hfill N\hfill I\hfill V\hfill E\hfill
                        R\hfill S\hfill I\hfill T\hfill Y\\
\end{tabular}
\vskip 36pt\hrule
\end{document}
```

Figure 24: Using the Concordia logo

```
\usepackage{psfig}
```

3. At the point where you want your picture to appear, use the command `\psfig` to create a LaTeX2e box that contains the picture. You can use either `height=` or `width=` to scale the picture. (Actually you can use both, but this will usually distort the picture.)

   For example, to insert a picture 3cm high, write:

   ```
   \psfig{figure=diag.eps,height=3cm}
   ```

Figure 24 shows a complete LaTeX2e input file that gives a heading with the university logo and the words "Concordia University" as they appear in Figure 17. You must be running LaTeX2e on the Computer Science network to obtain the logo with the given path.

9.3 Making a bibliography

The program BibTeX is used in conjunction with LaTeX2e to provide bibliographies. You can store a number of citations in a file with suffix `.bib`, cite them in your documents using the command `\cite`, and have your bibliography generated automatically. The bibliography on page 33 was created in this way. Briefly, the steps are as follows; for details, see [Lam95].

- ◇ Construct a bibliography containing all the documents you want to cite.
- ◇ Wherever you need a citation, write `\cite{`*key*`}`, where *key* is the key for the referenced document in the bibliography.
- ◇ Write `\bibiographystyle{`*style*`}` somewhere near the beginning of the your document. There are many styles, including: `alpha`, used in this manual; `plain`; `unsrt`, unsorted; and `ieeetr` for IEEE transactions. Some styles require a package: for example, to use the `chicago` style, you must include `\usepackage{chicago}` in the preamble.
- ◇ Write `\bibliography{`*file,file,...*`}` where you want the bibliography to appear. Include the names of all files containing bibliographic references that are cited in your document.

- ◇ Run `latex` on the input file.
- ◇ Run `bibtex` on the input file (`bibtex` should be part of the LaTeX2e installation).
- ◇ Run `latex` on the input file again.

9.4 Making an index

LaTeX2e will not generate an index automatically — which is a good thing because indexes created in this way are useless. However, you can index documents if you are prepared to do the work. Briefly, the steps are as follows; for details, see [Lam95].

- ◇ Write `\usepackage{makeidx}` in the preamble.
- ◇ Write `\makeindex` in the preamble.
- ◇ Write `\printindex` at the place in the document where you want the index — usually at the end of your document.
- ◇ Write `\index{`*word*`}` beside each word that you want to appear in the index. Don't leave blanks: a typical entry looks like this: `callipygian\index{callipygian}`.
- ◇ Run `latex` on the input file.
- ◇ Run `makeindex` on the input file (`makeindex` should be part of the LaTeX2e installation).
- ◇ Run `latex` on the input file again.

There are a number of tricks you can use to make the index more useful. For example,

```
\index{medicine!holistic}
```

will create an entry for "medicine" with a subentry for "holistic" and

```
\index{backslash@\verb=\=}
```

will create an entry consisting of "\" but alphabetized as "backslash".

10 Problems

TeX operates like a compiler attempting to translate your "program" into its "machine language" — the `.dvi` file. When it cannot understand your instructions, it stops and complains. This section does not explain all the problems you may encounter — that would require an encyclopedia — but it provides brief descriptions of common problems and solutions.

- ◇ You will sometimes find errors in the table of contents or in cross-references. The cure is simple: run LaTeX2e again.
- ◇ When TeX cannot figure out how to format a paragraph without exceeding the current text width, it reports an "overfull hbox". There is not much you can do about this except rewrite the paragraph or add discretionary hyphens (Section 3.3).
- ◇ When TeX stops with an error, you can type RETURN to make it start again or `x` RETURN to make it give up. Some errors create cascades of diagnostics; in this case, it is best to give up. But you can sometimes save time by noting the error and restarting TeX to discover the next error.

- ◇ A common mistake is to leave out a $ sign. TEX will detect this because it does not allow paragraph breaks in math mode. A consequence of this convention is that, if TEX encounters a blank line in math mode, it will complain about a missing $!

  When TEX thinks it has discovered a "missing" $ sign, it usually inserts a $ into the text and says "`Missing $ inserted`". This often causes more problems than the original error.
- ◇ If you leave out `\begin{document}`, LATEX2e generates numerous error messages but it does *not* tell you about the missing command.
- ◇ Most TEX and LATEX2e commands and environments nest nicely. There are a few exceptions. The message "`Not in outer par mode`" means that you tried to do something in math mode (or in a `\parbox`) that is allowed only in text mode. In particular, the `verbatim` environment may cause problems if it is used inside another environment.

11 Further Reading

Leslie Lamport designed LATEX and wrote the original, and best, user guide [Lam86], now superseded by the second edition, [Lam95]. If you want one book on LATEX2e get this one. Much of the development of LATEX2e was undertaken by a German group, who have published their own "companions" [GMS94, GR99] the second of which discusses how LATEX2e can be combined with web software.

Kopka and Daly have prepared an excellent book [KD95] that has both good examples and comprehensive references. Norman Walsh's book [Wal94] contains a wealth of practical information, including instructions for obtaining *emtex*, a public domain implementation of TEX and LATEX2e available on the internet. Unfortunately, it does not describe *miktex*, a more recent version of LATEX2e.

Knuth, the designer of TEX, has written a whole series of books about TEX and Metafont, his font designing program. For detailed information about how TEX works and its commands, get *The TEXbook* [Knu86b]; if you still can't figure out what is going on, you can read the source code of TEX and guess what it is trying to do [Knu86a].

[Abr90] Paul W. Abrahams. *TEX for the Impatient*. Addison-Wesley, 1990.

[Dil93] Antoni Diller. *LATEX Line by Line*. Wiley, 1993.

[GKP89] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathenatics: a Foundation for Computer Science*. Addison-Wesley, 1989.

[GMS94] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LATEX Companion*. Addison-Wesley, 1994.

[GR99] Michel Goossens and Sebastian Rahtz. *The LATEX Web Companion*. Addison-Wesley, 1999.

[Hah91] Jane Hahn. *LATEX for Everyone*. Personal TEX Inc, 1991.

[KD95] Helmut Kopka and Patrick W. Daly. *A Guide to LATEX2e: Document Preparation for Beginners and Advanced Users*. Addison-Wesley, second edition, 1995.

[Knu86a] Donald Erwin Knuth. *TEX: The Program*, volume B of *Computers & Typesetting*. Addison-Wesley, 1986.

[Knu86b] Donald Erwin Knuth. *The TEXBook*, volume A of *Computers & Typesetting*. Addison-Wesley, 1986.

[Kre79] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, fourth edition, 1979.

[Lam86] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, 1986. With illustrations by Duane Bibby.

[Lam95] Leslie Lamport. *LATEX: A Document Preparation System*. Addison-Wesley, second edition, 1994, 1995. With illustrations by Duane Bibby.

[Wal94] Norman Walsh. *Making TEX Work*. O'Reilly & Associates, Inc, 1994.

Index

\abstract environment, 6
accents, 7
Acrobat reader, 2
alignment, vertical, 18
\all (macro), 26
alltt environment, 11
alpha (bibliography style), 31
array environment, 21
article format, 4
\author, 6
.aux file, 2

\baselinestretch, 5
.bbl file, 2
.bib file, 2
bibliography, 31
bibtex program, 2, 32
blank
    ␣, 8
    "hard", 7, 8
.blg file, 2
body, of document, 6
book format, 4
brackets, 18
break
    line, 6–8, 18, 26
    page, 12, 27
bullet list, 14

callipygian, 32
\caption, 22
center environment, 10
chapter, 12
chicago (bibliography style), 31
\clearpage, 12
column spanning, 20
command, 8
comment, 6
Computer Modern fonts, 9
Concrete fonts, 9
contents, table of, 6
\cpp (macro), 26
cross-referencing, 14
    equations, 18
    sections, 26

\date, 6
declaration, 8
defining
    commands, 25
    environments, 27
definition list, 14
delimiter, 16
dimension, 23
    of page, 24
display math, 15
document
    body, 6
    structure, 4
document environment, 10
\documentclass, 4
\dotfill, 24
double spacing, 5
.dvi file, 1
dvips program, 1, 2

elephant, 8
em (dimension), 23
em dash, 8
EMACS, 4
empty page style, 13
environment, 10
    abstract, 6
    alltt, 11
    array, 21
    defining, 27
    document, 10
    eqnarray, 22
    eqnarray*, 22
    figure, 22
    for lists, 14
    minipage, 11
    quotation, 11
    quote, 11
    tabbing, 18
    table, 22
    tabular, 19
    verbatim, 10
.eps file, 30
eqnarray environment, 22
eqnarray* environment, 22

equation, 18
Euler fonts, 9
`ex` (dimension), 23

family, of font, 8
`fancyheadings` package, 13
`\fbox`, 11
`.fig` file, 30
`figure` environment, 22
figures, list of, 6
file
    input, 4, 14, 31
    style, 28
`\fill`, 24
floating figures and tables, 22
font, 4, 8, 9
    concrete, 9
    Euler, 9
    Pandora, 10
    PostScript, 2, 10
    size of, 8
footer, 24
footnote, 10
    for title, 6
`\footnotesize`, 8
front matter, 5

glue, 23, 24

header, 24
`headings` page style, 13
`\hfill`, 24
`\hrulefill`, 24
`\hspace`, 24
`\hspace*`, 24
`\Huge`, 8
`\huge`, 8
hyphenation, 7, 8

`ieeetr` (bibliography style), 31
index, 32
`\index`, 32
input file, 4, 14, 31
inter-word space, 7
intercolumn specifier, 19

`\kw` (macro), 26

`\label`, 14, 22
`landscape` option, 4
`\langle`, 16
`\LARGE`, 8
`\Large`, 8
`\large`, 8
`latex` program, 2, 32
`latexsym` package, 17
`\left`, 18
left quote, 8
letter format, 4
line
    break, 6–8, 18, 26
    spacing, 5
list, 14
    environments, 14
    of figures, 6
    of tables, 6
`\listoffigures`, 23
`\listoftables`, 23
`.lof` file, 2
`.log` file, 2
`.lot` file, 2

macro, 25
`makeidx` package, 32
`makeindex` program, 32
`\maketitle`, 6
`\markboth`, 13
`\markright`, 13
math
    examples, 17
    mode, 15
    symbols, 16
`\mathclose`, 16
`\mathopen`, 16
`\mbox`, 16
medium space (in math mode), 15
`\mid`, 15
`minipage` environment, 11
minus sign, 8
missing $, 33
`\multicolumn`, 20
`myheadings` page style, 13

negative thin space (in math mode), 15
`\newcommand`, 25
`\newpage`, 12
`\normalsize`, 8

not in outer par mode, 33
numbered list, 14
numbering
    equations, 18
    pages, 13
    sections, 12

one-and-half spacing, 5
`onecolumn` option, 4
`oneside` option, 4
optional argument for `\item`, 14
options for `\documentclass`, 4
overfull hbox, 32

package, 13, 28
    `concrete`, 9
    `latexsym`, 5
    `makeidx`, 5
    `pandora`, 10
    `psfig`, 30
page, 12
    break, 12, 27
    dimensions, 24
    numbers, 13
    style, 13
`\pagebreak`, 12
`\pagenumbering`, 13
`\pageref`, 14
`\pagestyle`, 13
Pandora fonts, 10
paper size, 4
paragraph, 12
`\parskip`, 24
part, 12
`\pd` (macro), 26
`.pdf` file, 2
`pdflatex` program, 2
`\penalty`, 27
picture, 30
`plain` (bibliography style), 31
`plain` page style, 13
positioning floats, 22
PostScript, 1, 3, 28, 30
    fonts, 2, 10
preamble, 5, 23
`\printindex`, 32
problems, 32
`.ps` file, 1

`psfig` package, 30
`pslatex` program, 2, 10
punctuation, 7, 8

`quotation` environment, 11
`quote` environment, 11
quotes, 8

`\range`, 16
`\ref`, 14
`\refstepcounter`, 27
relation, 16
report format, 4
`\right`, 18
right quote, 8

`\scriptsize`, 8
`secnumdepth`, 12
section, 12
    numbering, 12
series, of font, 8
`\setabstr` (macro), 26
shape, of font, 8
size
    of font, 4, 8
    of paper, 4
`\small`, 8
space, 7
    between columns, 19
    between lines, 5
    between paragaphs, 24
    between words, 7
    in math mode, 15
    white, 7
spanning columns, 20
structure, of document, 4
strut, 20
style
    file, 28
    in math mode, 18
    of font, 8
    of page, 13
subscript, 6
subsection, 12
superscript, 6

`tabbing` environment, 18
`table` environment, 22
table of contents, 6, 12

tables, list of, 6
`tabular` environment, 19
`\thanks`, 6
`\the`, 24
thick space (in math mode), 15
thin space (in math mode), 15
`\thispagestyle`, 13
`\tiny`, 8
`\title`, 5
`.toc` file, 2
TUG (TEX Users Group), 28
two-sided printing, 4
`twocolumn` option, 4
`twoside` option, 4
typewriter font, 8

units of measurement, 23
`unsrt` (bibliography style), 31
`\usepackage`, 28

variable size symbols, 18
`\verb`, 6
`\verb*`, 8
`verbatim` environment, 10
vertical alignment, 18
`\vfill`, 24
`\vspace`, 24
`\vspace*`, 24

web software, 33
white space, 7
    in math mode, 15

`xdvi` program, 1, 2
`xfig` program, 30
`\xs` (macro), 26