University of Oxford

Department of Statistics

Handbook for Further LaTeX

Trinity Term 2011

Susan Hutchinson

Computing support specialist

20 June 2011

Version 1.2. Minor changes for TT 2010

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Professor Brian Ripley for his permission to use material from exercises he devised for the Department of Statistics as part of an introductory LaTeX course.

I have also made use of ideas from Dr Nicola Talbot's excellent documents *LaTeX for Complete Novices* and *Using LaTeX to write a PhD Thesis*. These are exceptionally clear and relevant and can be found at **`http://theoval.cmp.uea.ac.uk/~nlct/latex/`**.

Both have probably forgotten more LaTeX than I will ever know.

1. Introduction

This handbook contains the following documents:

- A set of exercises exploring different LaTeX features.
- The answers to the exercises.
- The slides.

2. Before you start

2.1 Downloading files

Follow these commands to download some sample files which you will need for the exercises.

1. Create a new folder
2. Browse to **`http://www.stats.ox.ac.uk/pub/susan/oucs_latex/samples.zip`** to download a bundle of useful files that you will need today. I have also included a copy of this document as some of the features are only obvious when viewed online.
3. Right click on **`samples.zip`** and select **Extract All...** to extract all the files.

2.2 TEXnicCenter

We will be using TEXnicCenter to create, edit, compile and view LATEX files. To start TEXnicCenter go to Start ▶ All Programs ▶ TEXnicCenter. A window like Figure 2.1 should appear:

Take a little time to explore TEXnicCenter. There are three windows below the menus and formatting buttons. They are:

Navigator window The narrow window on the lefthand side is used to help you find your way around larger projects.

Edit window The large window is where files are edited.

Output window The long thin window at the bottom is where the output from compilation appears. This is where you should look when you have problems.

Find the **Output Profile** drop down menu box. The **Output Profile** shows **LaTeX** $=>$ **DVI** in Figure 2.1. See what other options there are.

Find the following buttons:

- **Build and view current file (Ctrl+Shift+F5)**
- **Build Output (F7)**
- **Build current file (Ctrl+F7)**
- **View Output (F5)**
- **Find (Ctrl+F)**
- **Replace (Ctrl+H)**
- **Save (Ctrl+S)**

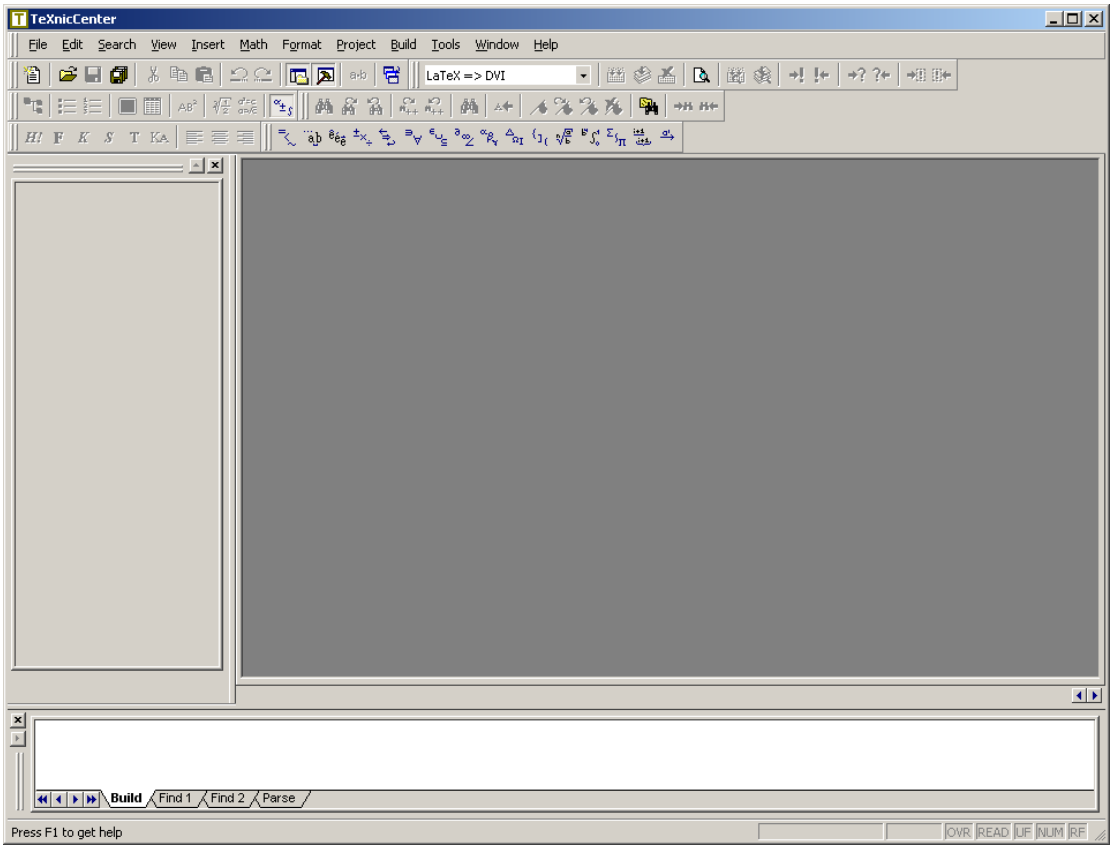

Figure 2.1: The TEXnicCenter: Start–up screen

The keyboard shortcuts are included after each command. The buttons are used for compiling, viewing, saving and editing LaTeX documents.

Most of the other buttons are used for typesetting. See if you can find some of these:

- **Bold**
- ***Slanted***
- **Align left**
- **Itemization**
- **Equation array**
- **Footnote**

Now you have explored the TEXnicCenter it is time to start using LaTeX.

2.3 LaTeX

We will be using **LaTeX** ⇒ **PDF** to compile `.tex` files today. This method of compilation offers a richer environment for graphics.

The exercises in this document are only suggestions. If there are other packages and techniques that you want to explore then please do so and we will do our best to assist you but priority will be given to those needing help with these exercises.

Most sets of exercises include a “If you have time” section. These are designed to develop a particular technique. Not all the information needed to complete the exercises is necessarily

included – you may need to search online or read the documentation.

2.4 Typesetting conventions

Distinct fonts and structures have been used to distinguish between various LaTeX features.

i. Text that needs to be typeset exactly as it appears in the exercises looks like this:

```
\begin{document}
\usepackage{parskip}
\begin{document}
Hello.
\end{document}
```

[For the curious this is achieved adding the `\usepackage{Verbatim}` to the preamble and enclosing the text in a `Verbatim` environment.]

ii. TeXnicCenter output profiles and other commands appear in a small sans-serif font like this: **LaTeX** $=>$ **PDF**.

iii. The output from a LaTeX compilation which appears in the display window I have enclosed between two lines as follows:

Here's a reference to Figure 2.1 in Chapter 2.

---

3. Getting Started

The first exercise is a refresher to get you started.

i. Start TEXnicCenter and create a new project using **File ► New Project...**. A window like Figure 3.1 should appear.
ii. Give the project a sensible name, perhaps `First`.
iii. Make sure that **Uses BibTeX** and **Uses MakeIndex** are checked.
iv. Click on **OK** when you're done.

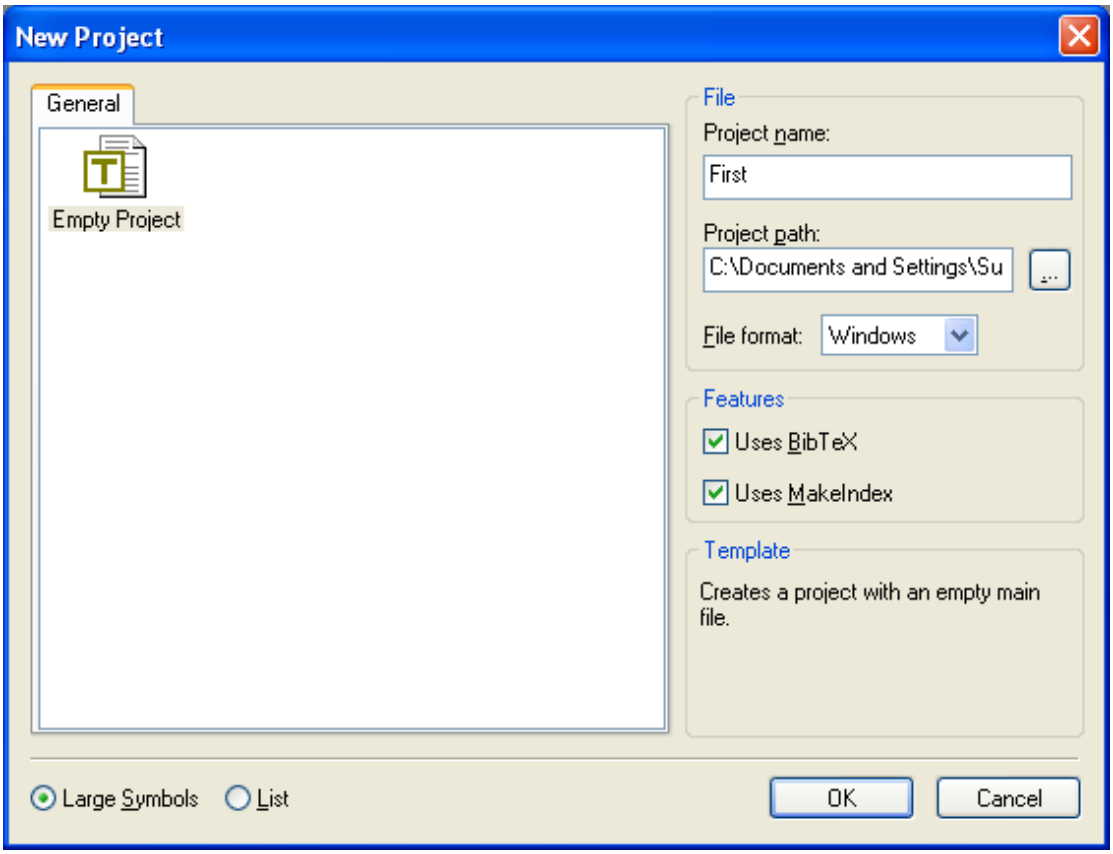

Figure 3.1: Creating a new project

Note that if you prefer to use Wordpad or Notepad and the Command Prompt window then do so.

▷ Exercise 1 **Creating a sample file**

Insert the following lines into your new project and then save it.

```
\documentclass[a4paper,12pt]{report}
\begin{document}
\title{My sample document}
\author{My Name}
\date{November 2009}
\maketitle
\pagenumbering{roman}
\tableofcontents
\chapter*{Acknowledgements}
\pagenumbering{arabic}
\chapter{Introduction}
\chapter{Background}
\chapter{Recent developments}
\chapter{Areas for study}
\chapter{Conclusion}
\end{document}
```

Most of the markup in the document should be familiar but please ask if any of the commands are not clear.

Making sure that **LaTeX** $=>$ **PDF** is displayed in the **Output Profile** box, click on the **Build and view current file** button to compile and view your file. There should be no errors, but there's not a lot of useful information here, either. We will develop this document as the course progresses. Remember, too, that you will need to compile the document twice in order to see the information in the table of contents.

From now on I won't ask you to click on the compile button and view button but just say recompile your LaTeX file.

▷ Exercise 2 **Changing the document**

1. Add your name and a title to the title page.
2. Add some text to the Introduction chapter. Perhaps you could write a couple of sentences about your studies or a hobby.

When you've saved the changes, recompile the file. The preview window – in this case Adobe Reader – will reload automatically with the new version after the compilation has completed successfully.

Before we go on to the next set of exercises make sure you are confident with the edit $\rightarrow$ compile $\rightarrow$ preview cycle. You will be doing this many times during the course.

4. Long documents

It is easier to manage longer documents like a thesis or a dissertation if they are organised in separate files. An advantage of this approach is that certain chapters only can be selected for compilation. For example, a chapter with lots of pictures can be omitted to speed up the compile → edit → preview cycle while writing other parts of the document. If this file containing the chapter has previously been compiled then references to figures in that chapter will still work.

▷ Exercise 3 **Splitting up a large document**

Make a backup copy of your project. Now we are going to remove all the text associated with the first chapter and put it in a separate file. Of course the words you added in the previous exercise may well be different from mine!

1. Copy the lines

   ```
   \chapter{Introduction}
   Here are some words for the first chapter.

   There's not a lot but it's a start.
   ```

   into a new file and save it as **intro.tex**. You should now have two edit windows as shown in Figure 4.1.

2. Delete these lines from your main file, **First.tex**.

3. Add the line \include{intro} to **First.tex**.

4. Recompile **First.tex** and preview it.

   *Make sure that you always compile your main file; compiling **intro.tex** will produce many errors and will fail.*

It should look exactly the same as the original version.

Now replace \chapter{Background} with \include{background} and create a new file called **background.tex** with the first line \chapter{Background} and recompile. Again, after compilation the document should look exactly the same.

▷ Exercise 4 **Conditional compilation**

We are now going to test that references will work even if we leave out some chapter files. Now make these changes:

1. Add these lines to **background.tex**

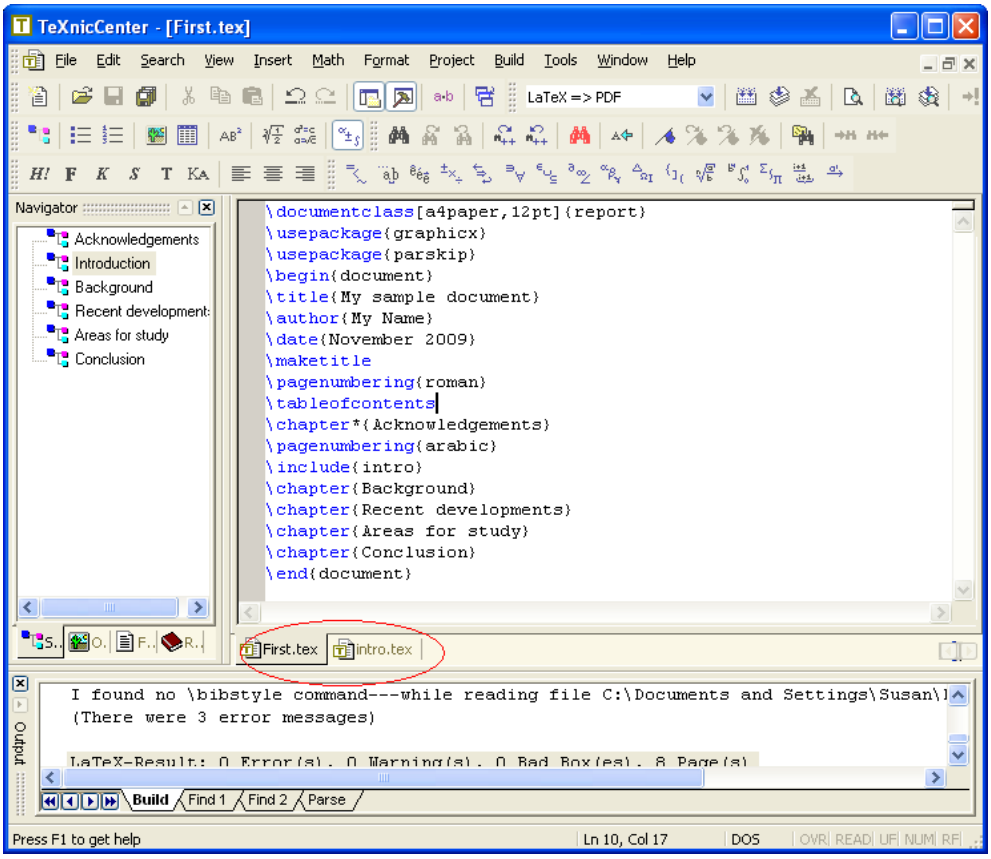

Figure 4.1: TEXnicCenter window showing two edit tabs

```
\label{ch:background}
I'm going to add some words and a picture in the second chapter.
\begin{figure}[ht]
\centering
\includegraphics{ox-crest.png}
\caption{The University of Oxford crest}
\label{fig:OxCrest}
\end{figure}
```

2. Make sure that the file **`ox-crest.png`** that you downloaded earlier is in the same folder as your **`.tex`** files.
3. Add **`usepackage`**{**`graphicx`**} to the preamble in the main file, **`First.tex`**.
4. Add these lines to **`intro.tex`**

   ```
   Here's a reference to Figure~\ref{fig:OxCrest} in
   Chapter~\ref{ch:background}.
   ```

   and recompile *twice* and preview the output.

You should see something like this in your preview window.

Here's a reference to Figure 2.1 in Chapter 2.

Now we are going to run a conditional compilation, including only **`intro.tex`** and excluding **`background.tex`**.

1. Add the line **includeonly**{**intro**} to the preamble of **second.tex**.
2. Save the file and recompile.

The new document will be missing Chapter 2 but the references will still be correct and there will still be an entry labelled 'Background' in the table of contents.

If you have time...

If you have time put all the information in the preamble in a separate file. See if you can explain how the **input**{**file.tex**} command differs from **include**{**file.tex**}. You may find google useful. Would you use **input**{**file.tex**} or **include**{**file.tex**} for the information in the preamble?

Chapter 10 contains the answers.

5. Customising LaTeX

Often you will find that you are repeating the same LaTeX markup over and over. The next exercise demonstrates how to create your own commands which can then be reused throughout your document.

For example if we have many occurrences of the phrase 'Oxford University' in our document it is much quicker to add the line `\newcommand{\ou}{Oxford University}` to the preamble. Subsequently, whenever we want to write 'Oxford University' all that is needed is to include the markup `\ou` in the text.

▷ Exercise 5 **Creating a new command**

Make a backup copy of your project. We are now going to add the new command to the preamble in the main file and make use it in both this document and in **intro.tex**.

You may want to remove or comment out (using a %) the `\includeonly...` line.

1. Add `\newcommand{\ou}{Oxford University}` to the preamble.
2. Add a suitable sentence such as 'I am grateful to \ou for the support I have received' to the Acknowledgements chapter. You may need to add {} after \ou to make sure the spacing is correct.
3. Add another sentence including \ou to your **intro.tex**.
4. Save all the files and recompile. Open a preview window and check that you can see 'Oxford University' in the text.
5. At this point you realise that it should be 'University of Oxford' not 'Oxford University'. Change the `\newcommand{\ou}...` line in your main file and recompile.

This demonstrates the advantage of using `\newcommand` for frequently used phrases or commands. If you need to make a change, it is only necessary to do it in one place – that is in the preamble.

▷ Exercise 6 **Using arguments with new commands**

Let's assume our document is going to make reference to several different Universities. Whenever we do that we want to change the font to bold, and make the colour magenta.

1. We need to add `\usepackage{xcolor}` to the preamble.
2. Add a line

   ```
   \newcommand{\Uni}[1]{\textbf {\color{magenta} University of #1}}
   ```

   to the preamble.

3. Now add two or more entries to your text using the **Uni** command. Perhaps you could have **Uni**{**Manchester**} and **Uni**{**Cambridge**}.
4. See if you can make the font bold and sans serif and change the colour to blue. You will need to add the **textsf** command.

Chapter 10 on page 22 contains answers where they don't appear in the text.

▷ Exercise 7 **Customising existing commands**

It is also possible to modify existing commands so that they behave as you want. The classic example of this is the **today** command which by default produces:

June 20, 2011

---

which is an American format. Many people prefer a date of the form 20 June 2011.

The standard definition of **today** is

```
\newcommand{\today}{\ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
  \space\number\day, \number\year}
```

Using **renewcommand** see if you can change the format of the **today** command. You will need to add this to the preamble. See chapter 10 for the answer.

If you have time...

See if you can make the colour of your new **Uni** command an argument as well so that the command is **Uni**{**green**}{**Leeds**} for example. Again, see Chapter 10 for the answer.

6. Creating a slide show

If you are writing papers or other technical documents in LaTeX you will probably need to give a presentation at some point. The decision to use a LaTeX slide–making environment is often based on the same considerations: typesetting maths is straightforward, creating precisely your own customisations is possible.

There are several slide-making document classes available, including the original class **slides** and the more powerful **Prosper**. One of the most popular and powerful classes is **beamer**. Early distributions of LaTeX did not always include Beamer, but up-to-date disbributions do.

Using Beamer it is possible to create complex and sophisticated slideshows which include equations, pictures and graphics, bibliographies, sound and animations. Effects such as pauses and transitions are simple to create; familiar LaTeX commands work.

▷ Exercise 8 **Creating a slide show**

i. Creating a minimal slide show based on the example below or use the example file provided. A very simple beamer **.tex** file could look like this:

```
\documentclass[pdf]{beamer}
% Put packages here
\usetheme{Warsaw} % Choose a theme
\usecolortheme{rose}  % Choose a colour scheme
\title{My first slideshow}
\subtitle{I hope you like it}
\author{Susan Hutchinson}
\institute{University of Oxford}
\date{March 2010}
% End of preamble
\begin{document}
\maketitle
\begin{frame}
\frametitle{Outline}
\tableofcontents
\end{frame}
\section{Introduction}
\begin{frame}
\frametitle{My first slide}
Contains very little information
\end{frame}
\end{document}
```

- Save this in a new project called **slideshow** and compile it with **LaTeX** $=>$ **PDF**.

- Open the PDF file - and Adobe Acrobat should start - then select **Window ▶ Full Screen Mode**. You should now be able to use the arrow or **Page Up** and **Page Down** keys to display the slides.

ii. Develop your slide show by changing themes and colours. The lines

```
\usetheme{Warsaw}
\usecolortheme{rose}
```

are used to set up the theme and colour scheme. Change and recompile your slide show a couple of times with different themes and colours. Do not spend too long on this. Most of the installed themes and colours are listed in the Tables 6.1 and 6.2.

AnnArbor	Antibes	Bergen	Berkeley	Berlin	Boadilla
CambridgeUS	Copenhagen	Darmstadt	Dresden	Frankfurt	Goettingen
Hannover	Ilmenau	JuanLesPins	Luebeck	Madrid	Malmoe
PaloAlto	Pittsburgh	Rochester	Singapore	Szeged	Warsaw

Table 6.1: Some Beamer themes

albatross	beaver	beetle	crane	dolphin	dove	fly
lily	orchid	rose	seagull	seahorse	whale	wolverine

Table 6.2: Some Beamer colours

iii. Make the following changes to your slide show

- Add some more content so that there are three or four slides
- Add some maths.
- Add pauses and overlays; the slides contain some examples of the markup for pauses and overlays.
- Add at least one more `\section` command before a frame so that the contents of the second slide is changed. Remember that you will need to compile twice in order for new sections to be displayed.

iv. Explore the examples provided and the package documentation in the **beamer** package folder. This can be found at `C:\Program Files\MiKTeX 2.7\tex\latex\beamer`.

v. Using the **beamer** documentation and Google see if you can work out how to create a handout rather than a presentation. In the handout version slides with pauses and overlays appear as one slide with all the information, rather than a series of slides.

If you have time...

Using the **beamer** documentation and Google if you get stuck, see if you can add a movie to your slide show. You can download a movie file from **http://www.stats.ox.ac.uk/pub/susan/oucs_latex/movie.avi** . Apologies for the poor quality and to those who don't enjoy cycling!

See Chapter 10 for a suggested solution.

7. Exploring packages

Many packages are available in LaTeX. These add features or change the behaviour of existing packages. We will be looking at just a few commonly used packages in this set of exercises.

Make a backup copy of your project.

▷ Exercise 9 **The `fancyhdr` package**

The basic headers provided by LaTeX are rather limited. The **`fancyhdr`** packages adds more information in the header and footer and extends the customistations you can make, allowing you, for example, to add your name to the footer of each page.

1. Add **`usepackage`**{**`fancyhdr`**} to the preamble.
2. Add **`pagestyle`**{**`fancy`**} just after the **`maketitle`**.
3. Compile your main file and preview it. Do you notice any difference?

It could be that you can't see any difference to your document. If each chapter is only a page long then nothing will have changed. This is because header information is only added to the second and subsequent pages of any chapter. Add something the following to a chapter:

```
\newpage
With a second page. It may be that headings
only appear on the second
page of a chapter.
\section{My big idea}
Here's the second page.

With a second paragraph.
\newpage
Again we need a new page to see what the
default headings look like.
\subsection{More detail}
Why this is an excellent idea.
\newpage
More information about my very good idea.
```

and recompile (possibly a couple of times if you want references to be correct). Now look at the document. You should notice

at the top of the second page. If you have included sections as in the example above you should see something like this.

Notice how the titles and numbers of the chapter and section now appear in the header. We are now going to customise the `fancyhdr` package by including our name in the footer.

▷ Exercise 10 Using the `texdoc` command

LaTeX packages mostly come with documentation. Some LaTeX editors provide this documentation as part of their help pages, others don't. We are now going to read the documentation for `fancyhdr` outside TeXnicCenter. We will use a command prompt and the `texdoc` command.

i. Open a command prompt window from **start ► All Programs ► Accessories ► Command Prompt**. You should see a window like Figure 7.1.

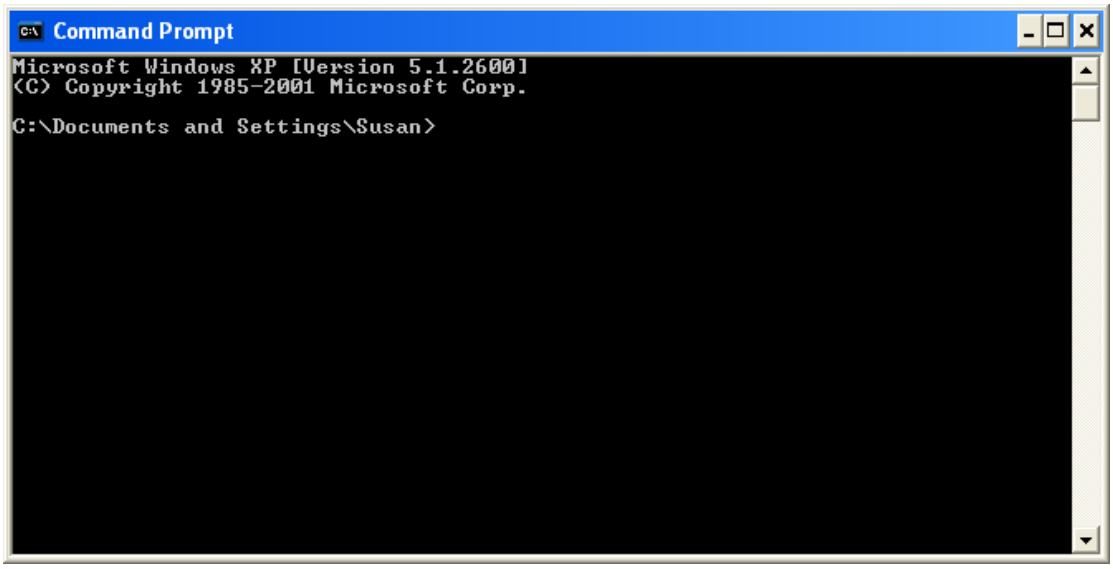

Figure 7.1: The command prompt window

ii. Type in the command `texdoc fancyhdr` and press enter. You may need to click on a link in a browser window but you should finally see a page like that in Figure 7.2:

From this document see if you can find out how to make the following changes to the footer:

- Add your name to the lefthand side of the footer.
- Move the page number from the centre to the righthand side.

▷ Exercise 11 Creating a glossary

I suggest you look at `finalglos.pdf` to see how your final version should look.

A glossary allows you to create a list of technical terms and their definitions. In LaTeX you can automate this quite a lot. Definitions can be reused, entries are sorted, and, if you are producing

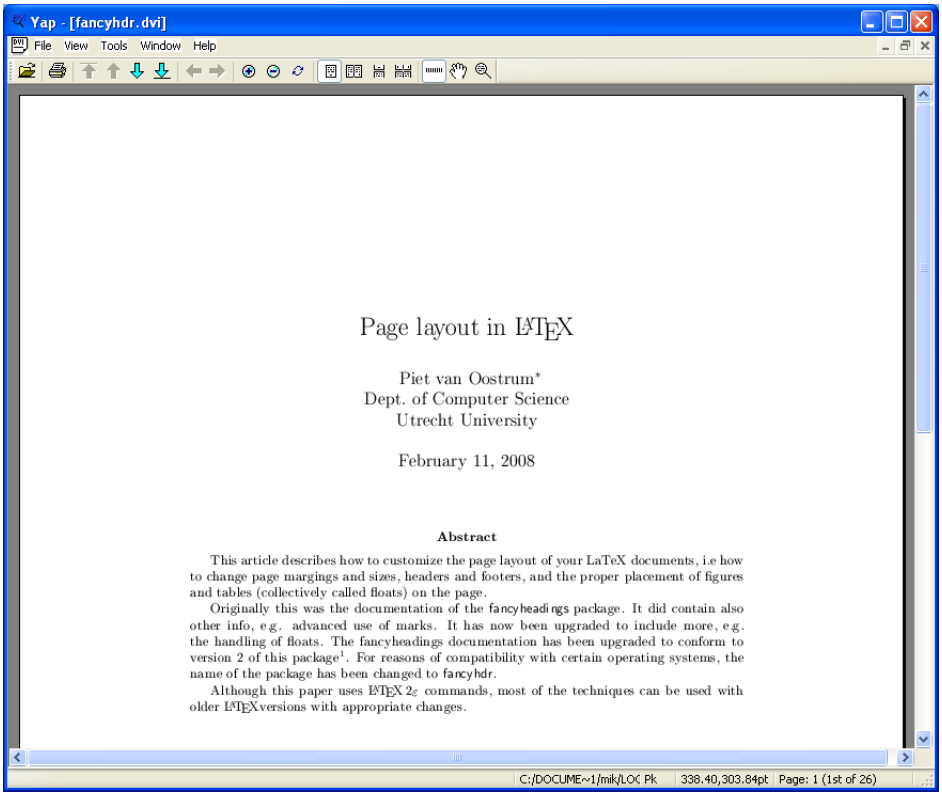

Figure 7.2: fancyhdr documentation

a PDF document, then these can be linked from the words in the text to the relevant entry in the glossary.

Learning a new package is sometimes a challenge. The following suggestions techniques that I have used successfully.

- Do a proof of concept by creating a small working example.
- Make only small changes at a time and that what you have done works as you would expect.

How I made it work

i. First I read the documentation using **`texdoc glossary`**. I found it quite difficult to understand as there was a *lot* of information. Try it!
ii. Googled. When I searched for 'glossary package latex' the one of the first links was to **`http://theoval.sys.uea.ac.uk/~nlct/latex/thesis/node25.html`** which contained a useful summary. Note that this page was written by the author of the glossary package and so is likely to be authoritative.

Now to get started with your own glossary.

Cut and paste the following into a separate file called **`glos.tex`** starting here:

```
\documentclass[12pt,a4paper]{article}
\usepackage{glossary}
\makeglossary
\begin{document}
I'm going to write a paragraph
```

```
\glossary{name=paragraph,description=sentences
on a linked theme}
to demonstrate how glossaries \glossary{name=glossary,description=an
explanation of terms}
work. Glossaries usually appear at the end of documents, but can be
referred to at any time. In particular, I will describe how glossaries
work in  \LaTeX \glossary{name=latex,description=a mathematical typesetting  language}.
\printglossary
\end{document}
```

Compiling a **.tex** file that contains a glossary is a slightly more complicated procedure.

i. Build **glos.tex** with **LaTeX** => **PDF**.

ii. Now in the command prompt window make sure you are in the same folder as your **glos.tex** file.

iii. Enter the command

makeindex -t glos.glg -o glos.gls -s glos.ist glos.glo

iv. Build **glos.tex** twice more and then look at the contents. You should see something like Figure 7.3.

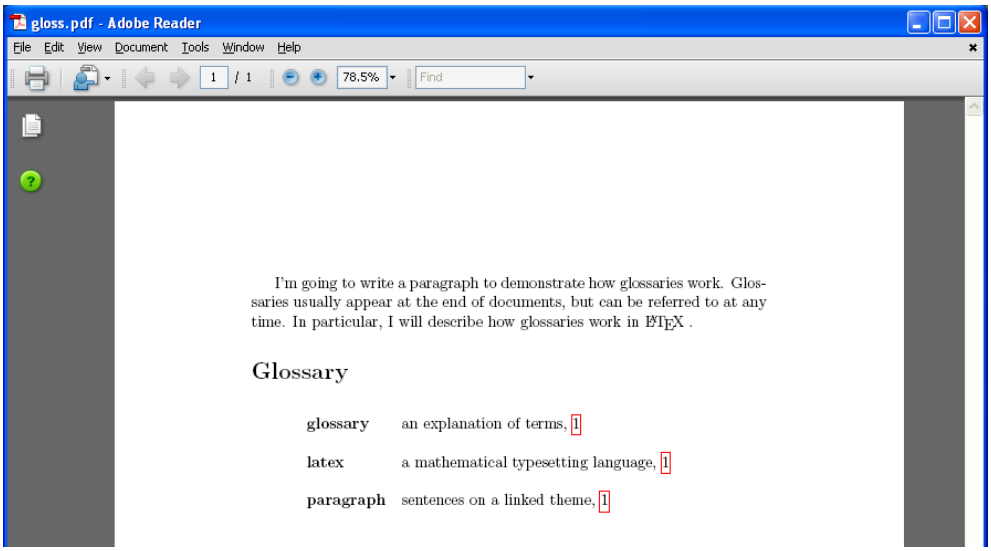

Figure 7.3: The first attempt at a glossary

It is possible to configure TEXnicCenter to build glossaries but this method will work whatever editor you are using.

Let's see how we can make this better. I would like the words in the text to be linked to the entry in the glossary. Reading the documentation it appears that **xglossary** would work as long as the **\hyperref** package is used. So we now have

```
\documentclass[12pt,a4paper]{article}
\usepackage{hyperref}
\usepackage{glossary}
\makeglossary
\begin{document}
I'm going to write a paragraph \xglossary{name=paragraph,
description=sentences on a linked theme}
to demonstrate how glossaries \xglossary{name=glossary,
```

```
description=an explanation of terms}
work. Glossaries usually appear at the end of
documents, but can be referred to at any time.
In particular, I will describe how glossaries work
in \LaTeX\ \xglossary{name=latex,description=a mathematical
typesetting language}. I will also describe amendments
\xglossary{name=amendments,description=a set of
changes or improvements} that can be made.
\end{document}
```

And my version:

I'm going to write a paragraph to demonstrate how glossaries work. Glossaries usually appear at the end of documents, but can be referred to at any time. In particular, I will describe how glossaries work in LaTeX for example. I will also describe amendments that can be made.

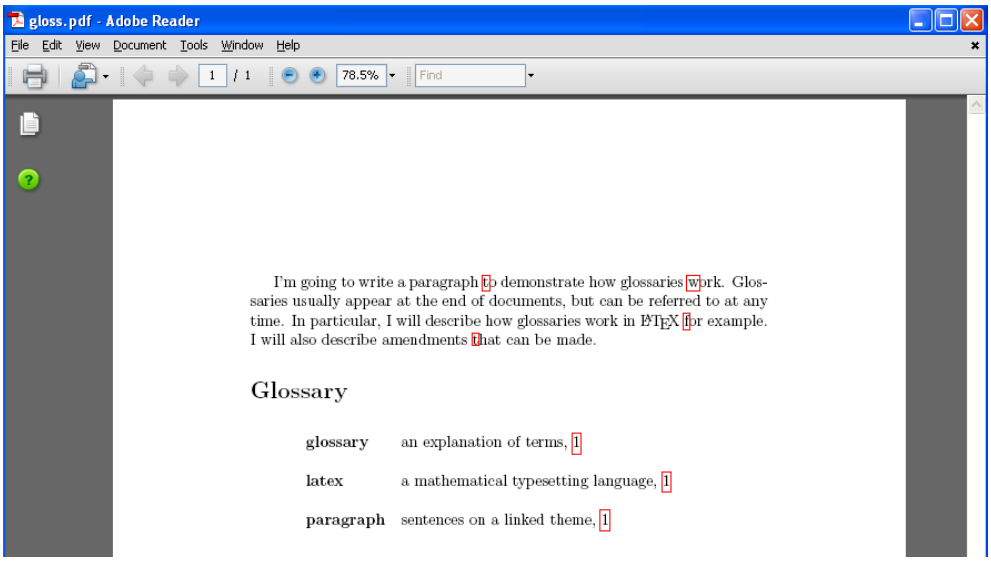

Figure 7.4: The second attempt at a glossary

I noticed that when I viewed the file displayed in Figure 7.4 that the links were to the first letter of the word *after* the glossary not to the word I intended. So I moved the words to after the glossary entry and surrounded them by {}. The paragraph now looks like this:

```
I'm going to write a \xglossary{name=paragraph,
description=sentences on a linked theme}{paragraph}
to demonstrate how \xglossary{name=glossary,
description=an explanation of terms}{glossaries}
work. Glossaries usually appear at the end of
documents, but can be referred to at any time.
In particular, I will describe how glossaries work
in \xglossary{name=latex,description=a mathematical
typesetting language}{\LaTeX}. I will also describe
\xglossary{name=amendments,description=a set of
changes or improvements}{amendments} that can be made.
```

And again:

I'm going to write a paragraph to demonstrate how glossaries work. Glossaries usually appear at the end of documents, but can be referred to at any time. In particular, I will describe how glossaries work in LaTeX. I will also describe amendments that can be made.

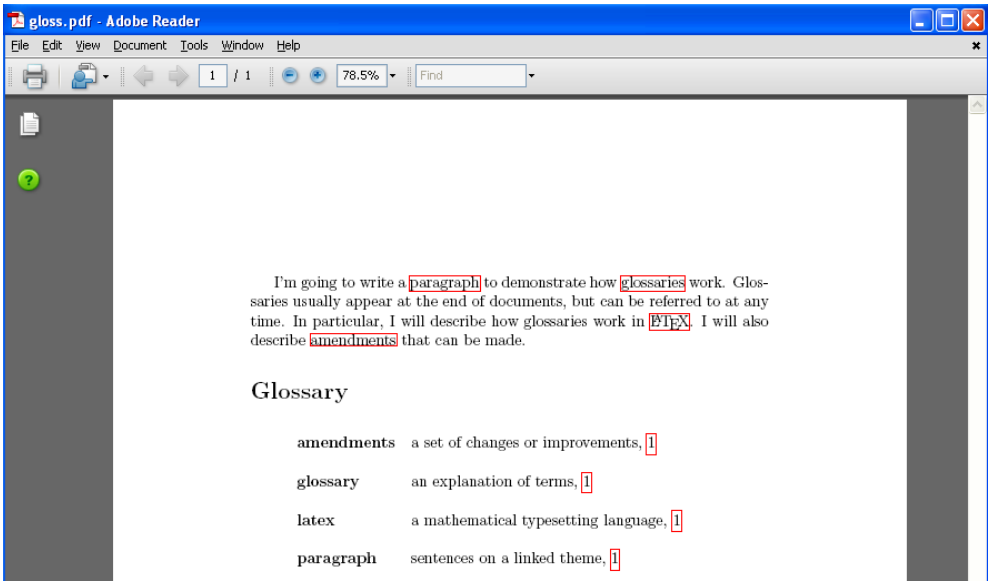

Figure 7.5: The third attempt at a glossary

As you can see from Figure 7.5 the whole word links to its glossary entry.

What happens when a new entry is added? Run the command `makeindex -t glos.glg ...` again.

Finally what happens if I want to reuse entries. It makes sense to create definitions separately and then use a shortcut when linking text to the glossary. So my final version is:

```
\storeglosentry{glos:P}{name=paragraph,
description=sentences on a linked theme}
\storeglosentry{glos:G}{name=glossary,
description=an explanation of terms}
\storeglosentry{glos:L}{name=latex,
description=a mathematical typesetting
language}
\storeglosentry{glos:A}{name=amendments,
description=a set of changes or improvements}

I'm going to write a \useGlosentry{glos:P}
{paragraph} to demonstrate how \useGlosentry{glos:G}
{glossaries} work. \useGlosentry{glos:G}{Glossaries}
usually appear at the end of documents, but can be
referred to at any time. In particular, I will
describe how glossaries work in \useGlosentry{glos:L}
{\LaTeX}. I will also describe \useGlosentry{glos:A}
{amendments} that can be made.
```

And finally...

I'm going to write a paragraph to demonstrate how glossaries work. Glossaries usually appear at the end of documents, but can be referred to at any time. In particular, I will describe how glossaries work in LaTeX. I will also describe amendments that can be made.

8. Glossary

amendments a set of changes or improvements, 16, 17

glossary an explanation of terms, 16, 17

latex a mathematical typesetting language, 16, 17

paragraph sentences on a linked theme, 16, 17

9. Conclusion

This course has only touched on more advanced features of LaTeX.

Here are some links which

10. Answers

\include or \input?

See page 9. I think it makes more sense to use `\input{file.tex}` if you store the information from the preamble in a separate file. This information is always needed. It is also useful to do this so that it can be reused whenever you start a new LaTeX document.

Using arguments with new commands

See page 10.

```
\newcommand{\Uni}[1]{\textsf {\textbf {\color{blue} University of #1}}}
```

Customising existing commands

See page 11.

```
\renewcommand{\today}{\number\day\space \ifcase\month\or
  January\or February\or March\or April\or May\or June\or
  July\or August\or September\or October\or November\or December\fi
   \space\number\year}
```

If you have time. . .

```
\newcommand{\Uni}[2]{\textsf {\textbf {\color{#1} University of #2}}}
```

Creating a slide show

See page 13 **If you have time. . .**

There may well be other solutions but I found the following was needed to make the movie display

i. Add `\usepackage{multimedia}` to the preamble.
ii. Add `\movie[width=5cm, height=4cm]{Play me}{movie.avi}` to the frame.

Exploring packages

Reading this sort of documentation is not always easy, I find. There is often a lot of detail, particularly at the beginning and there is rarely a simple 'Getting started' section or instructions

for frequently-used customisations. I must admit that I struggled to find the answer here and had to resort to Wikipedia! I added the following the preamble beneath `\usepackage{fancyhdr}`.

```
\fancyfoot{}
\lfoot{Susan Hutchinson}
\cfoot{}
\rfoot{\thepage}
```

This was quite easy to find out; the problem I had was that the page number was still centered on the first page of each chapter and no name appeared. This is perfectly acceptable, but I wanted my footer on *all* the pages. The solution was to change `\pagestyle{fancy}}` to `\pagestyle{fancyplain}}`