Introductory LaTeX

Dr Nicola Talbot

nlct@cmp.uea.ac.uk

Centre for Staff and Educational Development

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary

- Basic text, punctuation, accents and symbols.
- Simple font changing commands.
- Document classes, sectioning commands, and title pages.
- Centering and one-sided justification.
- Defining new commands.
- Converting to PostScript or Portable Document Format.
- Lists
- Tabulated material.
- Basic mathematics.
- Cross-referencing

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Summary Continued

- Packages
- Citations
- Lengths
- Boxes and minipages
- Incorporating images
- Figures and tables
- Creating slides
- Defining new environments
- Counters
- Writing packages and class files

Course Materials

- Each participant should have been given a handout.
- In addition, the following material is available on the web:
  - On-line version of the slides (HTML and PDF)
  - Advice and solutions to the exercises
  - Explanations to common errors
  - Terminology so that you can check the definition of a *keyword*.
  - PostScript version of the slides

  These can be found at:

  http://theoval.cmp.uea.ac.uk/~nlct/latex/csed/csed.html

Some Notes

- At the end of each topic there will be an exercise for you to do to give you some practical experience with the topic.
- Be sure to read the instructions given in the handout, and pay particular attention to any **Notes**.
- If you find yourself struggling, just do the parts of the exercises marked ⓔ. If you're speeding ahead, try doing the additional bits, marked ⚠.
- If you skip ahead, please save your questions until everyone else reaches that topic.

Some Notes

- At the end of each topic there will be an exercise for you to do to give you some practical experience with the topic.
- Be sure to read the instructions given in the handout, and pay particular attention to any **Notes**.
- If you find yourself struggling, just do the parts of the exercises marked ⓔ. If you're speeding ahead, try doing the additional bits, marked ⚠.
- If you skip ahead, please save your questions until everyone else reaches that topic.

Some Notes

- At the end of each topic there will be an exercise for you to do to give you some practical experience with the topic.
- Be sure to read the instructions given in the handout, and pay particular attention to any **Notes**.
- If you find yourself struggling, just do the parts of the exercises marked ⓔ. If you're speeding ahead, try doing the additional bits, marked ⚠.
- If you skip ahead, please save your questions until everyone else reaches that topic.

Some Notes

- At the end of each topic there will be an exercise for you to do to give you some practical experience with the topic.
- Be sure to read the instructions given in the handout, and pay particular attention to any **Notes**.
- If you find yourself struggling, just do the parts of the exercises marked ⓔ. If you're speeding ahead, try doing the additional bits, marked ⚠.
- If you skip ahead, please save your questions until everyone else reaches that topic.

Some More Notes

- Pay particular attention to Important information.
- Anything displayed *like this* indicates the type of thing you should insert at that point.
- LaTeX and UNIX are case-sensitive, so be sure to type commands exactly as they appear in the handout.
- A triangle ▶ indicates something to be typed in at the command prompt. For example:

  ```
  ▶latex filename
  ```

  (Remember to press the return key at the end of the line.)

Some More Notes

- Pay particular attention to Important information.
- Anything displayed *like this* indicates the type of thing you should insert at that point.
- LaTeX and UNIX are case-sensitive, so be sure to type commands exactly as they appear in the handout.
- A triangle ▶ indicates something to be typed in at the command prompt. For example:

  ```
  ▶latex filename
  ```

  (Remember to press the return key at the end of the line.)

Some More Notes

- Pay particular attention to Important information.
- Anything displayed *like this* indicates the type of thing you should insert at that point.
- LaTeX and UNIX are case-sensitive, so be sure to type commands exactly as they appear in the handout.
- A triangle ▶ indicates something to be typed in at the command prompt. For example:

  ```
  ▶latex filename
  ```

  (Remember to press the return key at the end of the line.)

Some More Notes

- Pay particular attention to Important information.
- Anything displayed *like this* indicates the type of thing you should insert at that point.
- LaTeX and UNIX are case-sensitive, so be sure to type commands exactly as they appear in the handout.
- A triangle ▶ indicates something to be typed in at the command prompt. For example:

  ```
  ▶latex filename
  ```

  (Remember to press the return key at the end of the line.)

What is TEX?

- TEX is a typesetting *language* written by Donald Knuth.
- Original format of TEX called: “plain TEX”.
- Plain TEX easy for simple documents (without equations, chapters etc).
- Otherwise very tricky.

What is LaTeX?

- Leslie Lamport wrote a format of TeX called LaTeX.
- Simple documents slightly harder to produce in LaTeX than plain TeX.
- Otherwise much easier to use.
- Since LaTeX is a format of TeX, you may get TeX as well as LaTeX error messages.
- We will be using LaTeX 2_ε version.
- Old LaTeX2.09 version *very* out of date.

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ►`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ►`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ►`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter (e.g. ►`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ▶`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ►`gcc HelloWorld.c`)
   - If there are errors, return to Step 1.
   - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ►`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ►`javac HelloWorld.java`)
   - If there are errors, return to Step 1.
   - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter
   (e.g. ►`java HelloWorld`).

Programming Languages

C

1. Write/edit source code in text file (e.g. `HelloWorld.c`)
2. Compile source code. (e.g. ▶`gcc HelloWorld.c`)
    - If there are errors, return to Step 1.
    - If successful, executable file created (e.g. `HelloWorld.exe`)
3. Run executable (e.g. ▶`HelloWorld`).

Java

1. Write/edit source code in text file (e.g. `HelloWorld.java`)
2. Compile source code. (e.g. ▶`javac HelloWorld.java`)
    - If there are errors, return to Step 1.
    - If successful, Java bytecode file created (e.g. `HelloWorld`)
3. Load Java bytecode into Java interpreter (e.g. ▶`java HelloWorld`).

LaTeX

1. Write/edit *source code* in text file (e.g. `HelloWorld.tex`)
2. LaTeX source code. (e.g. ▶`latex HelloWorld.tex`)
   - If there are errors, return to Step 1.
   - If successful, device independent file (DVI) created (e.g. `HelloWorld.dvi`)
3. Load DVI file into DVI viewer (e.g. ▶xdvi `HelloWorld.dvi`).

LaTeX is **NOT** a word processor!

LATEX

1. Write/edit *source code* in text file (e.g. `HelloWorld.tex`)
2. LATEX source code. (e.g. ▶`latex HelloWorld.tex`)
   - If there are errors, return to Step 1.
   - If successful, device independent file (DVI) created (e.g. `HelloWorld.dvi`)
3. Load DVI file into DVI viewer (e.g. ▶xdvi `HelloWorld.dvi`).

LATEX is **NOT** a word processor!

Pros and Cons of LaTeX

☞ Can only view your document once you have LaTeXed your *source code*.

  ☹ Can't see how things appear while you type.

  ☺ Tend to spend more time writing the actual text.

☹ Need to remember *command* names (or have book by you.)

☺ Unless you are using a front-end.

☹ Tricky to start with

☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.

☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

- ☞ Can only view your document once you have LaTeXed your *source code*.
  - ☹ Can't see how things appear while you type.
  - ☺ Tend to spend more time writing the actual text.
- ☹ Need to remember *command* names (or have book by you.)
- ☺ Unless you are using a front-end.
- ☹ Tricky to start with
- ☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.
- ☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

☞ Can only view your document once you have LaTeXed your *source code*.

- ☹ Can't see how things appear while you type.
- ☺ Tend to spend more time writing the actual text.

☹ Need to remember *command* names (or have book by you.)

☺ Unless you are using a front-end.

☹ Tricky to start with

☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.

☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

- ☞ Can only view your document once you have LaTeXed your *source code*.
  - ☹ Can't see how things appear while you type.
  - ☺ Tend to spend more time writing the actual text.
- ☹ Need to remember *command* names (or have book by you.)
- ☺ Unless you are using a front-end.
- ☹ Tricky to start with
- ☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.
- ☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

☞ Can only view your document once you have LaTeXed your *source code*.

- ☹ Can't see how things appear while you type.
- ☺ Tend to spend more time writing the actual text.

☹ Need to remember *command* names (or have book by you.)

☺ Unless you are using a front-end.

☹ Tricky to start with

☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.

☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

☞ Can only view your document once you have LaTeXed your *source code*.

- ☹ Can't see how things appear while you type.
- ☺ Tend to spend more time writing the actual text.

☹ Need to remember *command* names (or have book by you.)

☺ Unless you are using a front-end.

☹ Tricky to start with

☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.

☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

☞ Can only view your document once you have LaTeXed your *source code*.

  - ☹ Can't see how things appear while you type.
  - ☺ Tend to spend more time writing the actual text.

☹ Need to remember *command* names (or have book by you.)

☺ Unless you are using a front-end.

☹ Tricky to start with

☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.

☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX

☞ Can only view your document once you have LaTeXed your *source code*.

  ☹ Can't see how things appear while you type.

  ☺ Tend to spend more time writing the actual text.

☹ Need to remember *command* names (or have book by you.)

☺ Unless you are using a front-end.

☹ Tricky to start with

☺ But once you get the hang of it, it becomes a lot easier to do more complicated things.

☺ *Source code* for large documents comparatively small compared with word processors even if document contains many pictures. Can easily transfer file onto disk — useful if co-authoring a document.

Pros and Cons of LaTeX cont.

- ☺ Large documents don’t usually affect your typing speed (as long as you have a decent text editor). With word processors the whole document is constantly being reformatted as you type.
- ☺ Automatically follows most laws of typography, particularly when typesetting mathematics. Many word processors don’t do a very good job typesetting equations.
- ☺ Documents created using LaTeX tend to have a more professional look than those created using a word processor.
- ☺ Free! (Although some front-ends, such as WinEdt, are shareware.)

Pros and Cons of LaTeX cont.

- ☺ Large documents don't usually affect your typing speed (as long as you have a decent text editor). With word processors the whole document is constantly being reformatted as you type.
- ☺ Automatically follows most laws of typography, particularly when typesetting mathematics. Many word processors don't do a very good job typesetting equations.
- ☺ Documents created using LaTeX tend to have a more professional look than those created using a word processor.
- ☺ Free! (Although some front-ends, such as WinEdt, are shareware.)

Pros and Cons of LaTeX cont.

☺ Large documents don't usually affect your typing speed (as long as you have a decent text editor). With word processors the whole document is constantly being reformatted as you type.

☺ Automatically follows most laws of typography, particularly when typesetting mathematics. Many word processors don't do a very good job typesetting equations.

☺ Documents created using LaTeX tend to have a more professional look than those created using a word processor.

☺ Free! (Although some front-ends, such as WinEdt, are shareware.)

Pros and Cons of LaTeX cont.

- ☺ Large documents don’t usually affect your typing speed (as long as you have a decent text editor). With word processors the whole document is constantly being reformatted as you type.
- ☺ Automatically follows most laws of typography, particularly when typesetting mathematics. Many word processors don’t do a very good job typesetting equations.
- ☺ Documents created using LaTeX tend to have a more professional look than those created using a word processor.
- ☺ Free! (Although some front-ends, such as WinEdt, are shareware.)

Logging On

- We will be using Exceed.
- Exceed allows us to access UNIX computers from Windows NT.
- You should all have an ITCS username and password.
- To run Exceed, double-click on the icon `Staff1`.

Using UNIX

- UNIX is command line driven.
- Commands are typed in at the command prompt.
- Some simple commands:

ls lists the contents of the current directory.

mkdir makes a new directory. e.g.

```
▶mkdir latex
```

will create a directory called latex.

cd change directory. e.g.

```
▶cd latex
```

will change to the directory called latex.

cp copies a file to a new location. We will be using this command in some of the exercises.

xedit runs the text editor for the X Window System.

Initialising Your Account

- To initialise your account so that you can use LaTeX, you must type the following commands at the command prompt:

  ```
  ▶cd ~/.uea-options
  ▶touch tex
  ▶logout
  ```

  This will exit Exceed, so you will now have to run Exceed again.

- Once you have done this, you don't need to do it again.

Special Characters

- The following 10 symbols have special meaning and should be used with care:

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

- The percent symbol % is a comment character. Everything from the % symbol onwards until the end of the line will be ignored by LaTeX.

```
Some text % a comment.
```

INPUT

```
Some text
```

OUTPUT

- The other special characters will be discussed later.

Commands (Macros)

- *Commands* are available to specify how to format parts of the document.
- Commands are either made up of a single special character (e.g. $) or a backslash followed by a single symbol (e.g. `\=`) or a backslash followed by one or more alphabetical characters (e.g. `\today`).
- Any spaces following a command name made up of alphabetical characters are ignored. Notice the difference between

```
\oe
```

INPUT

```
œ
```

OUTPUT

and

```
\o e
```

INPUT

```
øe
```

OUTPUT

Commands (Macros)

- *Commands* are available to specify how to format parts of the document.
- Commands are either made up of a single special character (e.g. $) or a backslash followed by a single symbol (e.g. \=) or a backslash followed by one or more alphabetical characters (e.g. `\today`).
- Any spaces following a command name made up of alphabetical characters are ignored. Notice the difference between

```
\oe
```

INPUT

```
œ
```

OUTPUT

and

```
\o  e
```

INPUT

```
øe
```

OUTPUT

Commands (Macros)

- *Commands* are available to specify how to format parts of the document.

- Commands are either made up of a single special character (e.g. $) or a backslash followed by a single symbol (e.g. \=) or a backslash followed by one or more alphabetical characters (e.g. `\today`).

- Any spaces following a command name made up of alphabetical characters are ignored. Notice the difference between

```
\oe
```

INPUT

œ

OUTPUT

and

```
\o e
```

INPUT

øe

OUTPUT

Commands (Macros)

- *Commands* are available to specify how to format parts of the document.
- Commands are either made up of a single special character (e.g. `$`) or a backslash followed by a single symbol (e.g. `\=`) or a backslash followed by one or more alphabetical characters (e.g. `\today`).
- Any spaces following a command name made up of alphabetical characters are ignored. Notice the difference between

```
\oe
```

INPUT

œ

OUTPUT

and

```
\o e
```

INPUT

øe

OUTPUT

Commands (Macros)

- *Commands* are available to specify how to format parts of the document.

- Commands are either made up of a single special character (e.g. `$`) or a backslash followed by a single symbol (e.g. `\=`) or a backslash followed by one or more alphabetical characters (e.g. `\today`).

- Any spaces following a command name made up of alphabetical characters are ignored. Notice the difference between

```
\oe
```

Input

œ

Output

and

```
\o e
```

Input

øe

Output

Commands (Macros)

- *Commands* are available to specify how to format parts of the document.
- Commands are either made up of a single special character (e.g. `$`) or a backslash followed by a single symbol (e.g. `\=`) or a backslash followed by one or more alphabetical characters (e.g. `\today`).
- Any spaces following a command name made up of alphabetical characters are ignored. Notice the difference between

```
\oe
```

INPUT

```
œ
```

OUTPUT

and

```
\o e
```

INPUT

```
øe
```

OUTPUT

Grouping

- Segments of code may be *grouped* by placing it within { and }
- Most commands occurring within a group will be local to that group.

```
Some text. {This
text is \em within
a group.} Some more
text.
```

INPUT

Some text. This text is *within a group.* Some more text.

OUTPUT

- A command may be grouped to avoid placing a space after it:

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

INPUT

manœuvre

OUTPUT

Grouping

- Segments of code may be *grouped* by placing it within { and }
- Most commands occurring within a group will be local to that group.

```
Some text.  {This
text is \em within
a group.} Some more
text.
```

INPUT

Some text. This text is *within a group.* Some more text.

OUTPUT

- A command may be grouped to avoid placing a space after it:

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

INPUT

manœuvre

OUTPUT

Grouping

- Segments of code may be *grouped* by placing it within { and }
- Most commands occurring within a group will be local to that group.

```
Some text.  {This
text is \em within
a group.} Some more
text.
```

INPUT

Some text. This text is *within a group.* Some more text.

OUTPUT

- A command may be grouped to avoid placing a space after it:

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

INPUT

manœuvre

OUTPUT

Command Arguments

- Some commands take one or more *arguments*. For example, the command `\textbf` takes one argument, and will typeset that argument in a bold font.

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

INPUT

Some bold text

OUTPUT

- Note that if the argument consists of more that one character, it must be grouped using braces { }, if not, only the first object will be taken as the argument:

```
\textbf Some bold
text
```

INPUT

Some bold text

OUTPUT

Command Arguments

- Some commands take one or more *arguments*. For example, the command `\textbf` takes one argument, and will typeset that argument in a bold font.

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

INPUT

Some bold text

OUTPUT

- Note that if the argument consists of more that one character, it must be grouped using braces { }, if not, only the first object will be taken as the argument:

```
\textbf Some bold
text
```

INPUT

Some bold text

OUTPUT

Optional Arguments

- Some commands have *optional arguments*.
- Optional arguments are always enclosed in square brackets [ ]:

```
A new\\ line.
```

INPUT

A new
line.

OUTPUT

```
A new\\[5mm]
line.
```

INPUT

A new

line.

OUTPUT

- Optional arguments *almost* always come before mandatory arguments (although there are a few exceptions.)

Optional Arguments

- Some commands have *optional arguments*.
- Optional arguments are always enclosed in square brackets `[ ]`:

```
A new\\ line.
```

INPUT

A new
line.

OUTPUT

```
A new\\[5mm]
line.
```

INPUT

A new

line.

OUTPUT

- Optional arguments *almost* always come before mandatory arguments (although there are a few exceptions.)

Optional Arguments

- Some commands have *optional arguments*.
- Optional arguments are always enclosed in square brackets `[ ]`:

```
A new\\ line.
```

INPUT

A new
line.

OUTPUT

```
A new\\[5mm]
line.
```

INPUT

A new

line.

OUTPUT

- Optional arguments *almost* always come before mandatory arguments (although there are a few exceptions.)

Environments

- An *environment* is different to a command.
- `\begin{`*name*`}` indicates the beginning of an environment.
- `\end{`*name*`}` indicates the end of an environment.

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

  INPUT

  Some sans-serif text.

  OUTPUT
- Environments form *implicit* grouping. Changes made within an environment are usually local.
- Environments may also have optional or mandatory *arguments*.

Environments

- An *environment* is different to a command.
- \begin{*name*} indicates the beginning of an environment.
- \end{*name*} indicates the end of an environment.

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

  INPUT

  Some sans-serif text.

  OUTPUT
- Environments form *implicit* grouping. Changes made within an environment are usually local.
- Environments may also have optional or mandatory *arguments*.

Environments

- An *environment* is different to a command.
- `\begin{`*name*`}` indicates the beginning of an environment.
- `\end{`*name*`}` indicates the end of an environment.
-
  ```
  \begin{sffamily}
  Some sans-serif text.
  \end{sffamily}
  ```
  INPUT

  Some sans-serif text.

  OUTPUT
- Environments form *implicit* grouping. Changes made within an environment are usually local.
- Environments may also have optional or mandatory *arguments*.

Environments

- An *environment* is different to a command.
- `\begin{name}` indicates the beginning of an environment.
- `\end{name}` indicates the end of an environment.

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

  INPUT

  Some sans-serif text.

  OUTPUT
- Environments form *implicit* grouping. Changes made within an environment are usually local.
- Environments may also have optional or mandatory *arguments*.

Environments

- An *environment* is different to a command.
- \begin{*name*} indicates the beginning of an environment.
- \end{*name*} indicates the end of an environment.
-
```
\begin{sffamily}
Some sans-serif text.
\end{sffamily}
```

INPUT

Some sans-serif text.

OUTPUT

- Environments form *implicit* grouping. Changes made within an environment are usually local.
- Environments may also have optional or mandatory *arguments*.

Environments

- An *environment* is different to a command.
- \begin{*name*} indicates the beginning of an environment.
- \end{*name*} indicates the end of an environment.
-
  ```
  \begin{sffamily}
  Some sans-serif text.
  \end{sffamily}
  ```

  INPUT

  Some sans-serif text.

  OUTPUT
- Environments form *implicit* grouping. Changes made within an environment are usually local.
- Environments may also have optional or mandatory *arguments*.

Spaces

- Consecutive spaces are treated as one single space.

```
Some text.
```

INPUT

Some text.

OUTPUT

- CR-LF or tab characters are treated as a space.
- To force a space after a command, use \␣:

```
\LaTeX\ is great!
```

INPUT

LaTeX is great!

OUTPUT

- Also use \␣ after lowercase abbreviations:

```
e.g.\ like this.
```

INPUT

e.g. like this.

OUTPUT

Spaces

- Consecutive spaces are treated as one single space.

```
Some text.
```

Input

Some text.

Output

- CR-LF or tab characters are treated as a space.
- To force a space after a command, use \␣:

```
\LaTeX\ is great!
```

Input

LaTeX is great!

Output

- Also use \␣ after lowercase abbreviations:

```
e.g.\ like this.
```

Input

e.g. like this.

Output

Spaces

- Consecutive spaces are treated as one single space.

```
Some text.
```

INPUT

Some text.

OUTPUT

- CR-LF or tab characters are treated as a space.
- To force a space after a command, use \␣:

```
\LaTeX\ is great!
```

INPUT

LaTeX is great!

OUTPUT

- Also use \␣ after lowercase abbreviations:

```
e.g.\ like this.
```

INPUT

e.g. like this.

OUTPUT

Spaces

- Consecutive spaces are treated as one single space.

```
Some text.
```

INPUT

Some text.

OUTPUT

- CR-LF or tab characters are treated as a space.
- To force a space after a command, use \␣:

```
\LaTeX\ is great!
```

INPUT

LaTeX is great!

OUTPUT

- Also use \␣ after lowercase abbreviations:

```
e.g.\ like this.
```

INPUT

e.g. like this.

OUTPUT

Paragraphs

- Completely blank lines indicate the end of a paragraph.

```
Here is the first
paragraph.

This is the start
of the second
paragraph.
```

INPUT

Here is the first paragraph.

This is the start of the second paragraph.

OUTPUT

- A paragraph break can also be specified by the command `\par`

Creating a Document

- At the start of any LaTeX 2$_{\varepsilon}$ file, you must have the *command*

  ```
  \documentclass[options]{class}
  ```

  which declares what *class file* to use. For example:

  ```
  \documentclass[a4paper,12pt]{article}
  ```

- All the text that is actually contained in the document must be enclosed in a document *environment*:

  ```
  \begin{document}
  ```

  specifies the start of the document, and

  ```
  \end{document}
  ```

  specifies the end of the document.

Creating a Document

- At the start of any LaTeX 2_ε file, you must have the *command*

  ```
  \documentclass[options]{class}
  ```

  which declares what *class file* to use. For example:

  ```
  \documentclass[a4paper,12pt]{article}
  ```

- All the text that is actually contained in the document must be enclosed in a document *environment*:

  ```
  \begin{document}
  ```

  specifies the start of the document, and

  ```
  \end{document}
  ```

  specifies the end of the document.

Creating a Document

All LaTeX documents must have the following three lines:

```
\documentclass[a4paper,12pt]{article}

\begin{document}

\end{document}
```

Note that the text in blue may change, but the rest must occur exactly as above.

Exercise 1 : Creating a Simple Document (See Page 1)

- Open xedit, and type in the contents of Figure 1 on page 2 of the handouts.

- Save your file as `exercise1.tex`.

- Go to the command window, and type:

  ```
  ▶latex exercise1.tex
  ```

- If there were errors, a question mark (`?`) will appear. Type `h` for help, *Return* to continue or `x` to quit. Go back to xedit to fix the problem.

- To view the typeset document, type the following at the command prompt:

  ```
  ▶xdvi exercise1.dvi
  ```

Font Changing Commands
verses
Font Changing Declarations

A Font Changing Command is something that does not affect the rest of the document. It effectively says: *do this to the following object*. For example, `\textbf{A}` says: "make the following object bold", where the following object is the letter 'A'.

A Font Changing Declaration is something that affects the document from that point onwards. For example, `\bfseries` will switch to a bold font from the point where it is declared, onwards.

Font Changing Commands
verses
Font Changing Declarations

A Font Changing Command is something that does not affect the rest of the document. It effectively says: *do this to the following object*. For example, `\textbf{A}` says: "make the following object bold", where the following object is the letter 'A'.

A Font Changing Declaration is something that affects the document from that point onwards. For example, `\bfseries` will switch to a bold font from the point where it is declared, onwards.

Font Changing Commands
verses
Font Changing Declarations

A Font Changing Command is something that does not affect the rest of the document. It effectively says: *do this to the following object*. For example, `\textbf{A}` says: “make the following object bold”, where the following object is the letter ‘A’.

A Font Changing Declaration is something that affects the document from that point onwards. For example, `\bfseries` will switch to a bold font from the point where it is declared, onwards.

Font Changing Commands
verses
Font Changing Declarations

A Font Changing Command is something that does not affect the rest of the document. It effectively says: *do this to the following object*. For example, `\textbf{A}` says: "make the following object bold", where the following object is the letter 'A'.

A Font Changing Declaration is something that affects the document from that point onwards. For example, `\bfseries` will switch to a bold font from the point where it is declared, onwards.

Font Changing Commands

Command	Sample Input	Sample Output	
\textrm{*text*}	\textrm{Roman}	Roman	Family
\textsf{*text*}	\textsf{Sans serif}	Sans serif	
\texttt{*text*}	\texttt{typewriter}	`typewriter`	
\textmd{*text*}	\textmd{medium}	medium	Weight
\textbf{*text*}	\textbf{bold}	**bold**	
\textup{*text*}	\textup{upright}	upright	Shape
\textit{*text*}	\textit{italic}	*italic*	
\textsl{*text*}	\textsl{slanted}	*slanted*	
\textsc{*text*}	\textsc{Small Caps}	SMALL CAPS	
\emph{*text*}	\emph{emphasized}	*emphasized*	
\textnormal{*text*}	\textnormal{default}	default	

Font Changing Declarations

Declaration	Sample Input	Sample Output	
`\rmfamily`	`\rmfamily Roman`	Roman	Family
`\sffamily`	`\sffamily Sans serif`	Sans serif	
`\ttfamily`	`\ttfamily typewriter`	`typewriter`	
`\mdseries`	`\mdseries medium`	medium	Weight
`\bfseries`	`\bfseries bold`	**bold**	
`\upshape`	`\upshape upright`	upright	Shape
`\itshape`	`\itshape italic`	*italic*	
`\slshape`	`\slshape slanted`	*slanted*	
`\scshape`	`\scshape Small Caps`	Small Caps	
`\em`	`\em emphasized`	*emphasized*	
`\normalfont`	`\normalfont default`	default	

Font Changing Examples

1.

```
\em Some emphasized text.
```

INPUT

Some emphasized text.

OUTPUT

2.

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

INPUT

Some *emphasized* text.

OUTPUT

3.

```
\sffamily Some
\textsl{slanted} text.
```

INPUT

Some *slanted* text.

OUTPUT

Font Changing Examples

4.

```
\scshape Some more
\upshape text.
```

INPUT

SOME MORE text.

OUTPUT

5.

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

INPUT

Some emphasized *text.*

OUTPUT

6.

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

INPUT

Some bold text.

OUTPUT

Font Changing Environments

- An *environment* can also be used to change the font locally.
- The name of the environment is the same as the *declaration*, *without* the preceding \.
- Example:

```
Some normal text.
\begin{bfseries}
Some bold text.
\end{bfseries}
Back to normal text.
```

INPUT

Some normal text. **Some bold text.** Back to normal text.

OUTPUT

Changing the Font Size

Declaration	Environment	Sample
`\tiny`	tiny	tiny text
`\scriptsize`	scriptsize	script sized text
`\footnotesize`	footnotesize	footnote sized text
`\small`	small	small text
`\normalsize`	normalsize	normal sized text
`\large`	large	large text
`\Large`	Large	even larger
`\LARGE`	LARGE	larger still
`\huge`	huge	huge
`\Huge`	Huge	really huge

Font Size Changing Examples

1.

```
Some normal sized text.
{\small Some small
text.} Normal again.
```

INPUT

Some normal sized text. Some small text. Normal again.

OUTPUT

2.

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

INPUT

Some **large bold** text.

OUTPUT

Font Size Changing Examples

4.

```
\begin{footnotesize}
Some text.
\end{footnotesize}
```

Input

Some text.

Output

5.

```
Some normal text.
\begin{tiny}
\itshape
Some tiny italic text.
\end{tiny}
Back to normal.
```

Input

Some normal text. *Some tiny italic text.* Back to normal.

Output

Exercise 2 : Fonts (Page 2)

- Go back to the file you created in Exercise 1.
- Typeset the first paragraph in a large sans serif font, keeping the second paragraph in normal size Roman font.
- Emphasize the word “Footnotes” in the second paragraph, and then change the entire paragraph to italic.

Symbols

Remember the special characters? What if you actually want those characters to appear? And what about other symbols not on your keyboard?

\%	%	\textasciicircum	^	\P	¶	\ldots	. . .
\$	$	\textasciitilde	~	\S	§	\textbar	|
\#	#	\textbackslash	\	\yen	¥	\textgreater	>
\&	&	\copyright	©	\i	ı	\textless	<
\{	{	\textregistered	®	\j	ȷ	\textbullet	•
\}	}	\texttrademark	™	\ddag	‡	\pounds	£
_	_	\textvisiblespace	␣	\dag	†		

Examples of Symbols

1.

```
\pounds 43.50
```

Input

£43.50

Output

2.

```
A, B \& C
```

Input

A, B & C

Output

3.

```
As she opened the
door, she saw \ldots
```

Input

As she opened the door, she saw . . .

Output

Punctuation Symbols

- Standard punctuation characters can be entered as normal: , . ; : ? !
- Some symbols are obtained through a particular combination of characters.

Quotes		Dashes			Foreign Punctation	
`	‘	-	-	hyphen	?`	¿
'	’	--	–	en dash	!`	¡
``	“	---	—	em dash		
''	”	$-$	−	minus		

- A thin space \, can be used to separate a single quote from a double quote. For example, '\,'' produces ’”

Examples

1.

```
See pages 23--30
```

INPUT

See pages 23–30

OUTPUT

2.

```
She opened the box ---
the twenty-second
one --- and
fainted in surprise.
```

INPUT

She opened the box — the twenty-second one — and fainted in surprise.

OUTPUT

3.

```
``She said to me:
`is that it?'\,''
```

INPUT

“She said to me: ‘is that it?’ ”

OUTPUT

Accents

The accent commands place the required accent over or under the specified character.

	Example			Example	
Definition	Input	Output	Definition	Input	Output
\'{*object*}	\'{c}	ć	\={*object*}	\={c}	c̄
\`{*object*}	\`{c}	c̀	\.{*object*}	\.{c}	ċ
\^{*object*}	\^{c}	ĉ	\~{*object*}	\~{c}	c̃
\"{*object*}	\"{c}	c̈	\v{*object*}	\v{c}	č
\u{*object*}	\u{c}	c̆	\H{*object*}	\H{c}	c̋
\t{*object*}	\t{cc}	c͡c	\c{*object*}	\c{c}	ç
\d{*object*}	\d{c}	c̣	\b{*object*}	\b{c}	c̱

Example of Words with Accents

1.	`Caf\’e` INPUT	Café OUTPUT
2.	`R\^ole` INPUT	Rôle OUTPUT
3.	`P\^at\’e` INPUT	Pâté OUTPUT
4.	`Na\"{\i}ve` INPUT	Naïve OUTPUT

Note the use of the dotless i.

Ligatures

`\AE`	Æ	`\ae`	æ	`\OE`	Œ	`\oe`	œ
`fi`	ﬁ	`ffi`	ﬃ	`fl`	ﬂ	`ffl`	ﬄ

Foreign Symbols

`\AA`	Å	`\aa`	å	`\L`	Ł	`\l`	ł
`\O`	Ø	`\o`	ø	`\SS`	SS	`\ss`	ß

Examples of Words Containing Ligatures

1.

```
Man{\oe}uvre
```

INPUT

Manœuvre

OUTPUT

2.

```
{\AE}olian
```

INPUT

Æolian

OUTPUT

3.

```
{\OE}sophagus
```

INPUT

Œsophagus

OUTPUT

4.

```
fluffier
```

INPUT

fluffier

OUTPUT

Exercise 3 : Punctuation, Accents and Symbols (Page 4)

Create a LaTeX document that will produce the output shown in Figure 2 on page 5 of the handouts.

Note the following:

- Accent commands take one *argument* which must be the character you want the accent over.
- If you want an accent over an i use a dotless i.
- Remember to either *group* a command that produces a ligature, or place a space after it.

Document Classes

- As we have already seen, the document class is specified using the command:
  `\documentclass[`*option-list*`]{`*class*`}`
- There are many class files available. The standard ones are:

  article report book slides letter

- Which class file should I use?
  - ☞ What kind of document do you want to write?
  - ☞ Do you need an abstract? (article or report class)
  - ☞ Do you want only sections (article class), or do you need chapters as well (report or book class)?

Document Classes

- As we have already seen, the document class is specified using the command:
  \documentclass[*option-list*]{*class*}
- There are many class files available. The standard ones are:

  article report book slides letter

- Which class file should I use?
  ☞ What kind of document do you want to write?
  ☞ Do you need an abstract? (article or report class)
  ☞ Do you want only sections (article class), or do you need chapters as well (report or book class)?

Document Classes

- As we have already seen, the document class is specified using the command:
  `\documentclass[`*option-list*`]{`*class*`}`
- There are many class files available. The standard ones are:

  article report book slides letter

- Which class file should I use?
  - ☞ What kind of document do you want to write?
  - ☞ Do you need an abstract? (article or report class)
  - ☞ Do you want only sections (article class), or do you need chapters as well (report or book class)?

Document Classes

- As we have already seen, the document class is specified using the command:
  `\documentclass[`*option-list*`]{`*class*`}`
- There are many class files available. The standard ones are:

  article report book slides letter

- Which class file should I use?
  - ☞ What kind of document do you want to write?
  - ☞ Do you need an abstract? (article or report class)
  - ☞ Do you want only sections (article class), or do you need chapters as well (report or book class)?

Document Classes

- As we have already seen, the document class is specified using the command:
  `\documentclass[`*option-list*`]{`*class*`}`
- There are many class files available. The standard ones are:

  article report book slides letter

- Which class file should I use?
  - ☞ What kind of document do you want to write?
  - ☞ Do you need an abstract? (article or report class)
  - ☞ Do you want only sections (article class), or do you need chapters as well (report or book class)?

Document Classes

- As we have already seen, the document class is specified using the command:
  `\documentclass[`*option-list*`]{`*class*`}`
- There are many class files available. The standard ones are:

  article report book slides letter

- Which class file should I use?
  - ☞ What kind of document do you want to write?
  - ☞ Do you need an abstract? (article or report class)
  - ☞ Do you want only sections (article class), or do you need chapters as well (report or book class)?

Standard Class File Options

Some of the more common options:

`onecolumn`	Format document in one column format
`twocolumn`	Format document in two column format
`titlepage`	Make the title page appear on a separate page
`notitlepage`	Make the title appear at the top of the first page of the document
`oneside`	Format the document for one-sided printing
`twoside`	Format the document for two-sided printing
`portrait`	Format the document in portrait orientation
`landscape`	Format the document in landscape orientation
`10pt`	Make the normal font be 10pt
`11pt`	Make the normal font be 11pt
`12pt`	Make the normal font be 12pt

Standard Sectioning Commands

`\part`[*short title*]{*Title*}

`\chapter`[*short title*]{*Title*}

`\section`[*short title*]{*Title*}

`\subsection`[*short title*]{*Title*}

`\subsubsection`[*short title*]{*Title*}

`\paragraph`[*short title*]{*Title*}

`\subparagraph`[*short title*]{*Title*}

- Some of these commands may not be available in certain class files.
- All parts, chapters and sections, sub-sections and sub-sub-sections will be numbered automatically.
- `\paragraph` and `\subparagraph` are usually defined to have running unnumbered headings, but this can be changed.

Examples of Sectioning Commands

1.

```
\section{Introduction}
\LaTeX\ documents
are\ldots
```

Input

1 Introduction

LaTeX documents are. . .

Output

2.

```
\subsection{Macros}
Macros\ldots
```

Input

1.1 Macros

Macros. . .

Output

3.

```
\section*{Unnumbered
Sections}
```

Input

Unnumbered Sections

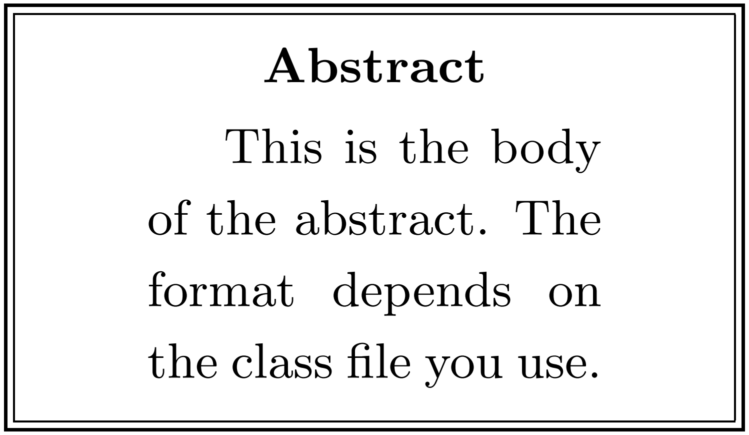
Output

Abstract

- Some *class files* such as article and report define an abstract *environment*.
- Example:

```
\begin{abstract}
This is the body of
the abstract.
The format depends
on the class
file you use.
\end{abstract}
```

INPUT



Abstract

This is the body of the abstract. The format depends on the class file you use.

OUTPUT

Title Page

- To create a title page, you first need to store information using the commands:

  ```
  \author{Author Names}
  \title{Document Title}
  \date{Date}
  ```

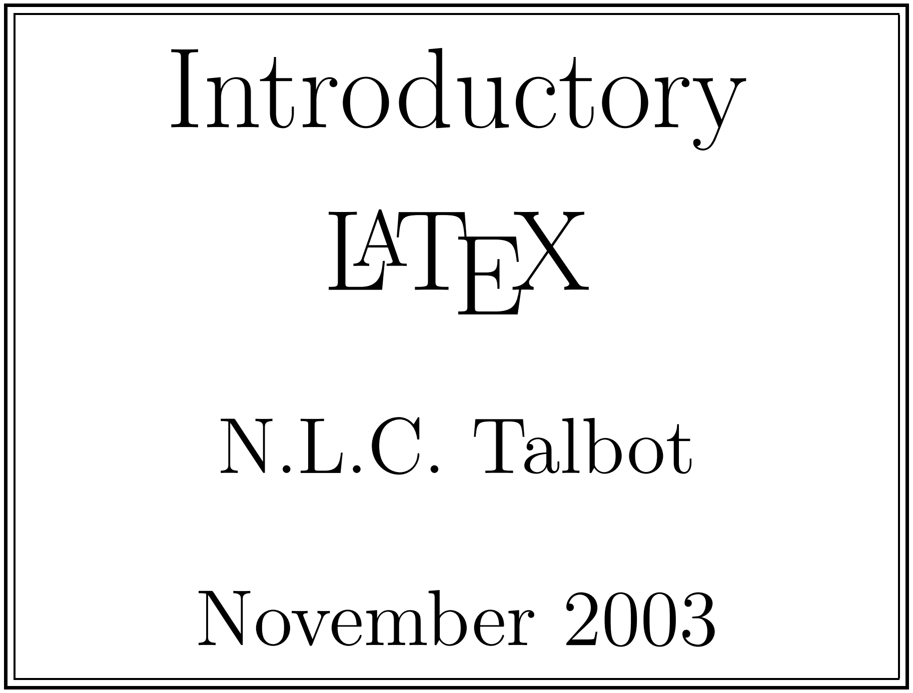
- The information is then displayed using the command:

  ```
  \maketitle
  ```

Example Title Page

```
\author{N.L.C. Talbot}
\title{Introductory
 \LaTeX}
\date{November 2003}

\maketitle
```

Input



Introductory

LaTeX

N.L.C. Talbot

November 2003

Output

Appendices

- To switch to appendices, use the command:

  ```
  \appendix
  ```

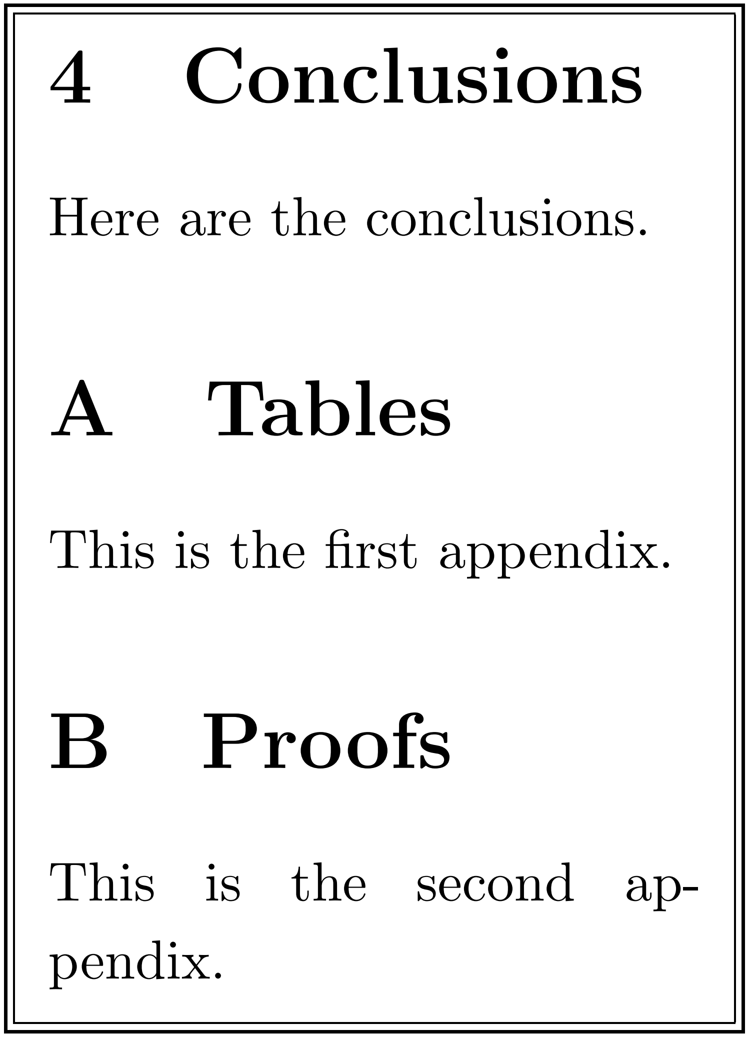
  at the start of the appendices.

- Continue to use `\chapter` or `\section` commands, depending on the *class file*.

Example

```
% This is the 4th section
\section{Conclusions}
Here are the conclusions.

\appendix
\section{Tables}
This is the first appendix.

\section{Proofs}
This is the second
appendix.
```

INPUT



4 Conclusions

Here are the conclusions.

A Tables

This is the first appendix.

B Proofs

This is the second appendix.

OUTPUT

Table of Contents

- The table of contents can be produced using the command:

  ```
  \tableofcontents
  ```

- The sectioning commands write information to the table of contents file (`.toc`).
- Next time you LaTeX your document, the table of contents file will be read, and the contents page will be generated.
- If you have a particularly long chapter or section title, you can use the optional argument of the sectioning command to specify a shorter title for the contents.
- The `\tableofcontents` command is usually placed after the `\maketitle` command.

Table of Contents

- The table of contents can be produced using the command:

  ```
  \tableofcontents
  ```

- The sectioning commands write information to the table of contents file (`.toc`).

- Next time you LaTeX your document, the table of contents file will be read, and the contents page will be generated.

- If you have a particularly long chapter or section title, you can use the optional argument of the sectioning command to specify a shorter title for the contents.

- The `\tableofcontents` command is usually placed after the `\maketitle` command.

Table of Contents

- The table of contents can be produced using the command:

  `\tableofcontents`

- The sectioning commands write information to the table of contents file (`.toc`).
- Next time you LaTeX your document, the table of contents file will be read, and the contents page will be generated.
- If you have a particularly long chapter or section title, you can use the optional argument of the sectioning command to specify a shorter title for the contents.
- The `\tableofcontents` command is usually placed after the `\maketitle` command.

Table of Contents

- The table of contents can be produced using the command:

  ```
  \tableofcontents
  ```

- The sectioning commands write information to the table of contents file (`.toc`).
- Next time you LaTeX your document, the table of contents file will be read, and the contents page will be generated.
- If you have a particularly long chapter or section title, you can use the optional argument of the sectioning command to specify a shorter title for the contents.
- The `\tableofcontents` command is usually placed after the `\maketitle` command.

Table of Contents

- The table of contents can be produced using the command:

  `\tableofcontents`

- The sectioning commands write information to the table of contents file (`.toc`).
- Next time you LaTeX your document, the table of contents file will be read, and the contents page will be generated.
- If you have a particularly long chapter or section title, you can use the optional argument of the sectioning command to specify a shorter title for the contents.
- The `\tableofcontents` command is usually placed after the `\maketitle` command.

Page Styles

- Page numbers appear automatically
- By default, in the article class file the page numbers appear centred in the footer.
- The page style (how headers and footers appear) can be changed using the command:

  `\pagestyle{`*style*`}`
- The most common styles are: `plain`, `empty` and `headings`.
- The style for the current page only can be set using:

  `\thispagestyle{`*style*`}`

Page Styles

- Page numbers appear automatically
- By default, in the article class file the page numbers appear centred in the footer.
- The page style (how headers and footers appear) can be changed using the command:

  ```
  \pagestyle{style}
  ```

- The most common styles are: `plain`, `empty` and `headings`.
- The style for the current page only can be set using:

  ```
  \thispagestyle{style}
  ```

Page Styles

- Page numbers appear automatically
- By default, in the article class file the page numbers appear centred in the footer.
- The page style (how headers and footers appear) can be changed using the command:

  `\pagestyle{`*style*`}`

- The most common styles are: `plain`, `empty` and `headings`.
- The style for the current page only can be set using:

  `\thispagestyle{`*style*`}`

Page Styles

- Page numbers appear automatically
- By default, in the article class file the page numbers appear centred in the footer.
- The page style (how headers and footers appear) can be changed using the command:

  `\pagestyle{`*style*`}`
- The most common styles are: `plain`, `empty` and `headings`.
- The style for the current page only can be set using:

  `\thispagestyle{`*style*`}`

Page Styles

- Page numbers appear automatically
- By default, in the article class file the page numbers appear centred in the footer.
- The page style (how headers and footers appear) can be changed using the command:

  ```
  \pagestyle{style}
  ```
- The most common styles are: `plain`, `empty` and `headings`.
- The style for the current page only can be set using:

  ```
  \thispagestyle{style}
  ```

Page Numbering Styles

- Page numbers are displayed in Arabic by default.
- The format can be changed using:

  `\pagenumbering{`*style*`}`

- Standard styles are: `arabic`, `roman`, `Roman`, `alph` and `Alph`.
- `\pagenumbering` will also reset the page number back to one.
- Standard practice is to use lowercase Roman numbering for the front matter, and arabic numbering for the main matter. Example:

```
\maketitle
\pagenumbering{roman}
\tableofcontents                  % Front matter
\begin{abstract} ... \end{abstract}
\chapter{Introduction}            % Start of main matter
\pagenumbering{arabic}
```

Page Numbering Styles

- Page numbers are displayed in Arabic by default.
- The format can be changed using:

  `\pagenumbering{`*style*`}`
- Standard styles are: `arabic`, `roman`, `Roman`, `alph` and `Alph`.
- `\pagenumbering` will also reset the page number back to one.
- Standard practice is to use lowercase Roman numbering for the front matter, and arabic numbering for the main matter. Example:

```
\maketitle
\pagenumbering{roman}
\tableofcontents              % Front matter
\begin{abstract} ... \end{abstract}
\chapter{Introduction}        % Start of main matter
\pagenumbering{arabic}
```

Page Numbering Styles

- Page numbers are displayed in Arabic by default.
- The format can be changed using:

  `\pagenumbering{`*style*`}`
- Standard styles are: `arabic`, `roman`, `Roman`, `alph` and `Alph`.
- `\pagenumbering` will also reset the page number back to one.
- Standard practice is to use lowercase Roman numbering for the front matter, and arabic numbering for the main matter. Example:

```
\maketitle
\pagenumbering{roman}
\tableofcontents                % Front matter
\begin{abstract} ... \end{abstract}
\chapter{Introduction}          % Start of main matter
\pagenumbering{arabic}
```

Page Numbering Styles

- Page numbers are displayed in Arabic by default.
- The format can be changed using:

  `\pagenumbering{`*style*`}`
- Standard styles are: `arabic`, `roman`, `Roman`, `alph` and `Alph`.
- `\pagenumbering` will also reset the page number back to one.
- Standard practice is to use lowercase Roman numbering for the front matter, and arabic numbering for the main matter. Example:

```
\maketitle
\pagenumbering{roman}
\tableofcontents                  % Front matter
\begin{abstract} ... \end{abstract}
\chapter{Introduction}            % Start of main matter
\pagenumbering{arabic}
```

Page Numbering Styles

- Page numbers are displayed in Arabic by default.
- The format can be changed using:

  \pagenumbering{*style*}

- Standard styles are: `arabic`, `roman`, `Roman`, `alph` and `Alph`.
- `\pagenumbering` will also reset the page number back to one.
- Standard practice is to use lowercase Roman numbering for the front matter, and arabic numbering for the main matter. Example:

```
\maketitle
\pagenumbering{roman}
\tableofcontents                    % Front matter
\begin{abstract} ... \end{abstract}
\chapter{Introduction}              % Start of main matter
\pagenumbering{arabic}
```

Exercise 4 : Sectioning Commands etc (Page 6)

- Copy over the file sectioning.tex:

  ```
  ▶cp /home/sys/gcc/insecure/sectioning.tex .
  ```

  Load the file sectioning.tex into xedit, and find the line that says:

  ```
  % CHAPTER : Introduction
  ```

  On the following line, insert the line

  ```
  \chapter{Introduction}
  ```

- Go through the rest of the file, and insert the appropriate sectioning commands.
- Use \maketitle to make the title information appear.
- Make the table of contents appear, by placing the command \tableofcontents at the place where you want it to appear.

Paragraph Formatting

By default, paragraphs are fully justified, however the justification can be changed, either by a *declaration*, or an *environment*.

Declaration	`\raggedright`	`\raggedleft`	`\centering`
Environment	flushleft	flushright	center

Examples (environments)

1.

```
\begin{flushright}
Some right
justified text.
\end{flushright}
```

Input

Some right justified
text.

Output

2.

```
\begin{center}
Some more text,
this time it is centred.
\end{center}
```

Input

Some more text,
this time it is
centred.

Output

Examples (declarations)

The justification declarations must be applied to *whole* paragraphs.

1. Justification applied to entire paragraph:

```
{\raggedleft Some right
justified text.\par}
```

INPUT

```
Some right justified
               text.
```

OUTPUT

2. Paragraph break has been omitted, causing the text to be formatted according to the justification in effect at the start of the paragraph:

```
{\raggedleft Some right
justified text.}
```

INPUT

```
Some right justified
text.
```

OUTPUT

Centering a Single Line of Text

There is also a *command* to centre a single line of text:

\centerline{*text*}

Example:

```
\centerline{Some centred text}
```

INPUT

Some centred text

INPUT

New Lines

- To force a new line: `\\[`*length*`]` or `\newline`

-
  ```
  Line one\\
  Line two\\[20pt]
  Line three
  ```

  Input

  ```
  Line one
  Line two


  Line three
  ```

  Output

-
  ```
  \begin{flushright}
  Line one\\
  Line two\\[20pt]
  Line three
  \end{flushright}
  ```

  Input

  ```
                Line one
                Line two


              Line three
  ```

  Output

Line breaks

- To break a line but keeping the text fully justified use: `\linebreak[n]`

-
  ```
  A short fully
  justified paragraph.
  ```
  Input

  A short fully justi-
  fied paragraph.

  Output

-
  ```
  A short \linebreak fully
  justified paragraph.
  ```
  Input

  A   short
  fully   justified
  paragraph.

  Output

Preventing Line breaks

- To prevent a line break use: `\nolinebreak[n]`

-
  ```
  A short fully
  justified paragraph.
  ```
  Input

  A short fully justified
  paragraph.

  Output

-
  ```
  A short fully
  justified\nolinebreak\
  paragraph.
  ```
  Input

  A short fully justi-
  fied paragraph.

  Output

Unbreakable Spaces

Alternatively, use a tilde ˜ to produce a space that can not be broken by a new line. For example:

```
Numbers such as the 3
in Example 3, should
never occur at the
start of a new line.
```

INPUT

```
Numbers such as the 3 in Example
3, should never occur at the start
of a new line.
```

OUTPUT

```
Numbers such as the 3
in Example~3, should
never occur at the
start of a new line.
```

INPUT

```
Numbers such as the 3 in Exam-
ple 3, should never occur at the
start of a new line.
```

OUTPUT

Page Breaks

- To force a ragged page break, use:

  `\newpage`

- To force a vertically justified page break, use:

  `\pagebreak[`*n*`]`

- To prevent a page break, use:

  `\nopagebreak[`*n*`]`

- To force a page break, and process all unprocessed floats, use:

  `\clearpage`

Exercise 5 : Paragraph Formatting (Page 7)

- Reproduce the output shown in Figure 3 on page 7 of the handouts.
- Consider whether to use *declarations* or *environments* or the `\centerline` command.

Defining New Commands

- To define a new command use:

  \newcommand{*cmd-name*}[*nargs*][*default*]{*text*}

- *cmd-name* is the name of the new command (remember the backslash)

- *nargs* is the numbers of *arguments* the new command takes (default 0)

- *default* is the default value for the first argument should an *optional argument* be required

- *text* is what LaTeX should do every time it encounters this command.

- Existing commands can be redefined using `\renewcommand` instead of `\newcommand`, but **never** redefine a command whose existing meaning is unknown to you.

Defining New Commands

- To define a new command use:

  `\newcommand`{*cmd-name*}[*nargs*][*default*]{*text*}

- *cmd-name* is the name of the new command (remember the backslash)
- *nargs* is the numbers of *arguments* the new command takes (default 0)
- *default* is the default value for the first argument should an *optional argument* be required
- *text* is what LaTeX should do every time it encounters this command.
- Existing commands can be redefined using `\renewcommand` instead of `\newcommand`, but **never** redefine a command whose existing meaning is unknown to you.

Defining New Commands

- To define a new command use:

  `\newcommand`{*cmd-name*}[*nargs*][*default*]{*text*}

- *cmd-name* is the name of the new command (remember the backslash)
- *nargs* is the numbers of *arguments* the new command takes (default 0)
- *default* is the default value for the first argument should an *optional argument* be required
- *text* is what LaTeX should do every time it encounters this command.
- Existing commands can be redefined using `\renewcommand` instead of `\newcommand`, but **never** redefine a command whose existing meaning is unknown to you.

Defining New Commands

- To define a new command use:

  `\newcommand{`*cmd-name*`}[`*nargs*`][`*default*`]{`*text*`}`

- *cmd-name* is the name of the new command (remember the backslash)
- *nargs* is the numbers of *arguments* the new command takes (default 0)
- *default* is the default value for the first argument should an *optional argument* be required
- *text* is what LaTeX should do every time it encounters this command.
- Existing commands can be redefined using `\renewcommand` instead of `\newcommand`, but **never** redefine a command whose existing meaning is unknown to you.

Defining New Commands

- To define a new command use:

  `\newcommand`{*cmd-name*} [*nargs*] [*default*] {*text*}

- *cmd-name* is the name of the new command (remember the backslash)
- *nargs* is the numbers of *arguments* the new command takes (default 0)
- *default* is the default value for the first argument should an *optional argument* be required
- *text* is what LaTeX should do every time it encounters this command.
- Existing commands can be redefined using `\renewcommand` instead of `\newcommand`, but **never** redefine a command whose existing meaning is unknown to you.

Defining New Commands

- To define a new command use:

  `\newcommand`{*cmd-name*} [*nargs*] [*default*] {*text*}

- *cmd-name* is the name of the new command (remember the backslash)
- *nargs* is the numbers of *arguments* the new command takes (default 0)
- *default* is the default value for the first argument should an *optional argument* be required
- *text* is what LaTeX should do every time it encounters this command.
- Existing commands can be redefined using `\renewcommand` instead of `\newcommand`, but **never** redefine a command whose existing meaning is unknown to you.

Why Define New Commands?

- To reduce lengthy typing:

```
\newcommand{\introLaTeX}{%
\emph{Introductory \LaTeX}}

The \introLaTeX\ course
is run by CSED\ldots
```

INPUT

The *Introductory* LaTeX course is run by CSED…

OUTPUT

Why Define New Commands?

- To ensure consistency:

```
\newcommand{\envname}[1]{%
\textsf{#1}}

The \envname{abstract}
environment\ldots
```

INPUT

The abstract environment…

OUTPUT

Examples

```
% First define the new command
\newcommand{\price}[2]{\pounds #1.#2}
% Now you can use it
The price is \price{2}{50}.
```

INPUT

The price is £2.50.

OUTPUT

```
\newcommand{\cost}[2][17.5]{%
The cost is \pounds #2 excl.\ VAT
@ #1\%}
%
\cost{100}.\\
\cost[0.0]{50}
```

INPUT

The cost is £100 excl. VAT @ 17.5%.
The cost is £50 excl. VAT @ 0.0%

OUTPUT

Exercise 6 : Defining New Commands (Page 8)

- Create a new document called `exercise6.tex`.
- Define the command `\timeofday` (in the *preamble*). This command should take two parameters, the first is the hour and the second is the number of minutes passed the hour. For example, the command `\timeofday{10}{25}` should produce the output: 10:25.
- Create the output shown in Figure 4 on page 8 where the time is produced using the `\timeofday` command.
- Once you have done this, change the definition of the command so that the time is displayed in bold and the hours and minutes are separated with a dash instead of a colon (e.g. **10-25**).

PostScript

- DVI files can be converted to PostScript using dvips:

```
▶dvips -o filename.ps filename.dvi
```

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

    – Rearrange pages (psbook)

    – Print multiple logical pages on a single sheet (psnup)

    – Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ```
  ▶dvips -o filename.ps filename.dvi
  ```

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

    – Rearrange pages (psbook)

    – Print multiple logical pages on a single sheet (psnup)

    – Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ▶`dvips -o filename.ps filename.dvi`

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

  - Rearrange pages (psbook)
  - Print multiple logical pages on a single sheet (psnup)
  - Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ```
  ▶dvips -o filename.ps filename.dvi
  ```

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

  - Rearrange pages (psbook)
  - Print multiple logical pages on a single sheet (psnup)
  - Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ▶`dvips -o filename.ps filename.dvi`

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

  – Rearrange pages (psbook)

  – Print multiple logical pages on a single sheet (psnup)

  – Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ▶`dvips -o filename.ps filename.dvi`

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:
  - ☞ Use the pstricks package and create some very fancy graphics.
  - ☞ Incorporate PostScript commands into your document.
  - ☞ Use the psutils suite:
    - Rearrange pages (psbook)
    - Print multiple logical pages on a single sheet (psnup)
    - Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ```
  ▶dvips -o filename.ps filename.dvi
  ```

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

  - Rearrange pages (psbook)
  - Print multiple logical pages on a single sheet (psnup)
  - Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ```
  ▶dvips -o filename.ps filename.dvi
  ```

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:
  - ☞ Use the pstricks package and create some very fancy graphics.
  - ☞ Incorporate PostScript commands into your document.
  - ☞ Use the psutils suite:
    - Rearrange pages (psbook)
    - Print multiple logical pages on a single sheet (psnup)
    - Convert, e.g., US letter documents to A4 paper (psresize)

PostScript

- DVI files can be converted to PostScript using dvips:

  ▶`dvips -o filename.ps filename.dvi`

- The PostScript file can either be sent directly to a PostScript printer, or can be loaded into ghostview where you can preview it and send it to a non-PostScript printer.

- If you use dvips you can:

  ☞ Use the pstricks package and create some very fancy graphics.

  ☞ Incorporate PostScript commands into your document.

  ☞ Use the psutils suite:

  – Rearrange pages (psbook)
  – Print multiple logical pages on a single sheet (psnup)
  – Convert, e.g., US letter documents to A4 paper (psresize)

Portable Document Format (PDF)

- The proliferation of the world wide web has encouraged many authors to create PDF documents which can be viewed in Adobe's Acrobat Reader.

- PostScript files can be converted to PDF using ps2pdf.

- Alternatively, PDF output can be obtained from LaTeX documents using PDFLaTeX instead of LaTeX.

  ▶`pdflatex filename`

- You can have hyperlinks, both internal and external.

- You can access Adobe Acrobat menu functions. For example: Summary

  See "The LaTeX Web Companion" [4, chapter 2] for more details

PostScript and PDF Output

If you use either dvips or PDFLaTeX, you can:

- Include PostScript, PDF or PNG images in your document.
- Have colour
- Use PostScript fonts
- rotate reflect or resize text.

Exercise 7 (Page 8)

- Try converting the document that you modified in Exercise 4 (sectioning.tex) into a PostScript file.

  ```
  ▶dvips -o sectioning.ps sectioning
  ```

- Now view it using ghostview:

  ```
  ▶ghostview sectioning.ps
  ```

- Try using PDFLaTeX to generate a PDF version of this document:

  ```
  ▶pdflatex sectioning.tex
  ```

  You can view it using xpdf:

  ```
  ▶xpdf sectioning.pdf
  ```

List Making Environments

The itemize *environment* produces an *unordered* list.

```
\begin{itemize}
\item The first item
\item The second item
\item The third item
\end{itemize}
```

Input

- The first item
- The second item
- The third item

Output

Nested itemize environments

Up to four itemize environments may be nested:

```
\begin{itemize}
\item The first item
  \begin{itemize}
   \item First item
   of nested list
   \item Second item
   of nested list
  \end{itemize}
\item The second item
\end{itemize}
```

INPUT

- The first item
  - First item of nested list
  - Second item of nested list
- The second item

OUTPUT

Numbered Lists

The enumerate *environment* produces an *ordered* list.

```
\begin{enumerate}
\item The first item
\item The second item
\item The third item
\end{enumerate}
```

INPUT

1. The first item
2. The second item
3. The third item

OUTPUT

Nested enumerate environments

Up to four enumerate environments may be nested:

```
\begin{enumerate}
\item The first item
  \begin{enumerate}
   \item First item
   of nested list
   \item Second item
   of nested list
  \end{enumerate}
\item The second item
\end{enumerate}
```

INPUT

1. The first item
   (a) First item of nested list
   (b) Second item of nested list
2. The second item

OUTPUT

Nested itemize and enumerate environments

itemize and enumerate environments may be nested:

```
\begin{enumerate}
\item The first item
  \begin{itemize}
   \item First item
   of nested list
   \item Second item
   of nested list
  \end{itemize}
\item The second item
\end{enumerate}
```

INPUT

1. The first item
   - First item of nested list
   - Second item of nested list
2. The second item

OUTPUT

Description

```
\begin{description}
\item[Cabbage] A large
round green vegetable
\item[Brussel sprout] A
small round
green vegetable
\end{description}
```

INPUT

Cabbage A large round green vegetable

Brussel sprout A small round green vegetable

OUTPUT

Exercise 8 : Lists (Page 9)

- Create a document that produces the output shown in Figure 5 on page 9 of the handouts.
- You will need to use nested enumerate *environments*

ⓔ Start with a simple list:

1. Animal
2. Vegetable
3. Mineral

and add to it.

⚠ Convert the enumerate environments to itemize environments, and redefine the item labels.

Tabulated Material

- Material can be aligned in rows and columns using the tabular *environment*:

  `\begin{tabular}`{*column specifiers*}

  Each column is specified by one of the following letters: `l` (left aligned) `c` (centred) or `r` (right aligned).

  Examples:

  1. `\begin{tabular}{ccc}`

     Three columns, all centred.

  2. `\begin{tabular}{lr}`

     Two columns, the first left justified, the second right justified.

- Within the tabular environment:
  - The special character & moves to the next column.
  - The new line command \\ is used to move on to the next row.

Tabulated Material

- Material can be aligned in rows and columns using the tabular *environment*:

  `\begin{tabular}{`*column specifiers*`}`

  Each column is specified by one of the following letters: `l` (left aligned) `c` (centred) or `r` (right aligned).

  Examples:

  1. `\begin{tabular}{ccc}`

     Three columns, all centred.

  2. `\begin{tabular}{lr}`

     Two columns, the first left justified, the second right justified.

- Within the tabular environment:

  – The special character & moves to the next column.

  – The new line command \\ is used to move on to the next row.

Tabulated Material

- Material can be aligned in rows and columns using the tabular *environment*:

  `\begin{tabular}{`*column specifiers*`}`

  Each column is specified by one of the following letters: `l` (left aligned) `c` (centred) or `r` (right aligned).

  Examples:

  1. `\begin{tabular}{ccc}`

     Three columns, all centred.

  2. `\begin{tabular}{lr}`

     Two columns, the first left justified, the second right justified.

- Within the tabular environment:

  – The special character & moves to the next column.

  – The new line command \\ is used to move on to the next row.

Tabulated Material

- Material can be aligned in rows and columns using the tabular *environment*:

  `\begin{tabular}{`*column specifiers*`}`

  Each column is specified by one of the following letters: `l` (left aligned) `c` (centred) or `r` (right aligned).

  Examples:

  1. `\begin{tabular}{ccc}`

     Three columns, all centred.

  2. `\begin{tabular}{lr}`

     Two columns, the first left justified, the second right justified.

- Within the tabular environment:

  – The special character & moves to the next column.

  – The new line command `\\` is used to move on to the next row.

Tabulated Material

- Material can be aligned in rows and columns using the tabular *environment*:

  `\begin{tabular}{`*column specifiers*`}`

  Each column is specified by one of the following letters: `l` (left aligned) `c` (centred) or `r` (right aligned).

  Examples:

  1. `\begin{tabular}{ccc}`

     Three columns, all centred.

  2. `\begin{tabular}{lr}`

     Two columns, the first left justified, the second right justified.

- Within the tabular environment:
  - The special character `&` moves to the next column.
  - The new line command `\\` is used to move on to the next row.

Simple Example

- This example has only 2 columns
- So each row can have no more than 1 &
- Column 1 is left justified, column 2 is centred.

```
\begin{tabular}{lc}
Item & Cost\\
CD & \pounds 11.75\\
Video & \pounds 14.10\\
Total & \pounds 25.85
\end{tabular}
```

Input

Item	Cost
CD	£11.75
Video	£14.10
Total	£25.85

Output

Be careful not to confuse l (ell) with 1 (one)!

Adding Horizontal and Vertical Lines

- Vertical lines are specified in the tabular environment argument using the vertical bar character |

  `\begin{tabular}{|l|c|}`

- Horizontal lines:
  - Spanning all columns:

    `\hline`
  - Spanning from column *n* to *m*:

    `\cline{`*n*`-`*m*`}`

- `\hline` and `\cline` can only be used at the start of a row.

Adding Horizontal and Vertical Lines

- Vertical lines are specified in the tabular environment argument using the vertical bar character |

  ```
  \begin{tabular}{|l|c|}
  ```

- Horizontal lines:
  - Spanning all columns:

    ```
    \hline
    ```

  - Spanning from column *n* to *m*:

    ```
    \cline{n-m}
    ```

- `\hline` and `\cline` can only be used at the start of a row.

Adding Horizontal and Vertical Lines

- Vertical lines are specified in the tabular environment argument using the vertical bar character |

  `\begin{tabular}{|l|c|}`

- Horizontal lines:
  - Spanning all columns:

    `\hline`

  - Spanning from column *n* to *m*:

    `\cline{`*n*`-`*m*`}`

- `\hline` and `\cline` can only be used at the start of a row.

Adding Horizontal and Vertical Lines

- Vertical lines are specified in the tabular environment argument using the vertical bar character |

  ```
  \begin{tabular}{|l|c|}
  ```

- Horizontal lines:
  - Spanning all columns:

    ```
    \hline
    ```

  - Spanning from column *n* to *m*:

    ```
    \cline{n-m}
    ```

- `\hline` and `\cline` can only be used at the start of a row.

Example

Our simple example can be modified:

```
\begin{tabular}{|l|c|}
\hline
Item & Cost\\
\hline\hline
CD & \pounds 11.75\\
Video & \pounds 14.10\\
\hline
Total & \pounds 25.85\\
\hline
\end{tabular}
```

INPUT

Item	Cost
CD	£11.75
Video	£14.10
Total	£25.85

OUTPUT

Spanning Columns

- An entry can span several columns:

  `\multicolumn`{*cols*}{*align*}{*text*}

- *cols* is the number of columns to be spanned.
- *align* is the alignment for this entry. This can only contain one alignment letter (e.g. `c`), but can also contain vertical bar specifiers.
- *text* The text for this entry.
- Can also be used to change the specification for a single column.

```
\begin{tabular}{|l|cc|}\hline
 & \multicolumn{2}{c|}{Cost}\\
Item & ex VAT & inc VAT (@17.5\%)\\\hline\hline
CD & \pounds 10.00 & \pounds 11.75\\
Video & \pounds 12.00 & \pounds 14.10\\\hline
\multicolumn{1}{l|}{Total} & \pounds 22.00 &
\pounds 25.85\\\cline{2-3}
\end{tabular}
```

INPUT

	Cost	
Item	ex VAT	inc VAT (@17.5%)
CD	£10.00	£11.75
Video	£12.00	£14.10
Total	£22.00	£25.85

OUTPUT

Exercise 9 : Tabulated Material (Page 10)

- When creating tables, it's best to start with a simple table, and then add to it.

  ⓔ Try creating the output shown in Figure 6 on page 10

  ⚠ Try creating the output shown in Figure 7 on page 10.

- The table is created using just one tabular *environment*. The lines `Equipment Expenditure` and `Travel Expenditure` span all 5 columns.

- Once you've finished it, centre the table, using the `\centerline` command.

Basic Mathematics

- TEX has two modes:
    1. Text mode
    2. Maths mode
- So far we have only been using text mode.
- All maths must be placed in maths mode, as the fonts and spacing are different.
    - Text mode: y-c=mx
    - Maths mode: $y - c = mx$
- There are two types of maths: *in-line* maths and *displayed* maths

In-Line Mathematics

- In-line maths occurs within a line of text.

- Symbols such as $\sum$ are small so that it doesn't take up to much space.

- Can be broken across a line.

- For in-line maths, use the math environment:

  ```
  \begin{math}
  ...
  \end{math}
  ```

- Shorthand notation: `\( ... \)`

- Even shorter notation: `$ ...$`

  Another special character!

In-Line Mathematics

- In-line maths occurs within a line of text.
- Symbols such as $\sum$ are small so that it doesn't take up to much space.
- Can be broken across a line.
- For in-line maths, use the math environment:

  ```
  \begin{math}
  ...
  \end{math}
  ```

- Shorthand notation: `\( ... \)`
- Even shorter notation: `$ ...$`

  Another special character!

In-Line Mathematics

- In-line maths occurs within a line of text.
- Symbols such as $\sum$ are small so that it doesn't take up to much space.
- Can be broken across a line.
- For in-line maths, use the math environment:

  ```
  \begin{math}
  ...
  \end{math}
  ```

- Shorthand notation: `\( ... \)`
- Even shorter notation: `$ ...$`

  Another special character!

In-Line Mathematics

- In-line maths occurs within a line of text.
- Symbols such as $\sum$ are small so that it doesn't take up to much space.
- Can be broken across a line.
- For in-line maths, use the math environment:

  ```
  \begin{math}
  ...
  \end{math}
  ```

- Shorthand notation: `\( ... \)`
- Even shorter notation: `$ ...$`

  Another special character!

In-Line Mathematics

- In-line maths occurs within a line of text.
- Symbols such as $\sum$ are small so that it doesn't take up to much space.
- Can be broken across a line.
- For in-line maths, use the math environment:

  ```
  \begin{math}
  ...
  \end{math}
  ```

- Shorthand notation: `\( ... \)`
- Even shorter notation: `$ ...$`

  Another special character!

In-Line Mathematics

- In-line maths occurs within a line of text.
- Symbols such as $\sum$ are small so that it doesn't take up to much space.
- Can be broken across a line.
- For in-line maths, use the math environment:

  ```
  \begin{math}
  ...
  \end{math}
  ```

- Shorthand notation: `\( ... \)`
- Even shorter notation: `$ ...$`

  Another special character!

Examples

```
I can refer to the variable $x$, or the
formula \(y = m x + c\).
```

INPUT

I can refer to the variable $x$, or the formula $y = mx + c$.

OUTPUT

```
The $i$th element of the vector $\vec{a}$ has the value
$2i$ for $i = 1 \ldots m$.
```

INPUT

The $i$th element of the vector $\vec{a}$ has the value $2i$ for $i = 1 \ldots m$.

OUTPUT

Displayed Mathematics

- Displayed maths is placed on a line of its own so that it stands out better.
- Symbols such as $\displaystyle\sum$ are larger.
- Lots of different environments go into displayed math mode.
- Basic ones:
  1. The displaymath environment displays a *single* unnumbered line of maths.

     `\begin{displaymath} ... \end{displaymath}`

     Shorthand: `\[ ... \]`
  2. The equation environment displays a *single* numbered line of maths.

     `\begin{equation} ... \end{equation}`

Displayed Mathematics

- Displayed maths is placed on a line of its own so that it stands out better.
- Symbols such as $\sum$ are larger.
- Lots of different environments go into displayed math mode.
- Basic ones:
  1. The displaymath environment displays a *single* unnumbered line of maths.

     ```
     \begin{displaymath} ... \end{displaymath}
     ```

     Shorthand: `\[ ... \]`
  2. The equation environment displays a *single* numbered line of maths.

     ```
     \begin{equation} ... \end{equation}
     ```

Displayed Mathematics

- Displayed maths is placed on a line of its own so that it stands out better.
- Symbols such as $\sum$ are larger.
- Lots of different environments go into displayed math mode.
- Basic ones:

  1. The displaymath environment displays a *single* unnumbered line of maths.

     ```
     \begin{displaymath} ... \end{displaymath}
     ```

     Shorthand: `\[ ... \]`

  2. The equation environment displays a *single* numbered line of maths.

     ```
     \begin{equation} ... \end{equation}
     ```

Displayed Mathematics

- Displayed maths is placed on a line of its own so that it stands out better.
- Symbols such as $\sum$ are larger.
- Lots of different environments go into displayed math mode.
- Basic ones:

  1. The displaymath environment displays a *single* unnumbered line of maths.

     ```
     \begin{displaymath} ... \end{displaymath}
     ```

     Shorthand: `\[ ... \]`

  2. The equation environment displays a *single* numbered line of maths.

     ```
     \begin{equation} ... \end{equation}
     ```

Displayed Mathematics

- Displayed maths is placed on a line of its own so that it stands out better.
- Symbols such as $\displaystyle\sum$ are larger.
- Lots of different environments go into displayed math mode.
- Basic ones:

  1. The displaymath environment displays a *single* unnumbered line of maths.

     ```
     \begin{displaymath} ... \end{displaymath}
     ```

     Shorthand: `\[ ... \]`

  2. The equation environment displays a *single* numbered line of maths.

     ```
     \begin{equation} ... \end{equation}
     ```

Displayed Mathematics

- Displayed maths is placed on a line of its own so that it stands out better.
- Symbols such as $\displaystyle\sum$ are larger.
- Lots of different environments go into displayed math mode.
- Basic ones:
  1. The displaymath environment displays a *single* unnumbered line of maths.

     `\begin{displaymath} ... \end{displaymath}`

     Shorthand: `\[ ... \]`
  2. The equation environment displays a *single* numbered line of maths.

     `\begin{equation} ... \end{equation}`

Examples

```
The function
\begin{displaymath}
f(x) = 4x + 1
\end{displaymath}
is linear
```

INPUT

The function

$$f(x) = 4x + 1$$

is linear

OUTPUT

```
The function
\begin{equation}
f(x) = 4x + 1
\end{equation}
is linear
```

INPUT

The function

$$f(x) = 4x + 1 \qquad (1)$$

is linear

OUTPUT

Subscripts and Superscripts

- Subscripts are created using the command: `\sb{`*subscript*`}`
  - Shorthand: `_{`*subscript*`}`
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp{`*superscript*`}`
  - Shorthand: `^{`*superscript*`}`
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Subscripts and Superscripts

- Subscripts are created using the command: `\sb`{*subscript*}
  - Shorthand: `_`{*subscript*}
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp`{*superscript*}
  - Shorthand: `^`{*superscript*}
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Subscripts and Superscripts

- Subscripts are created using the command: `\sb`{*subscript*}
  - Shorthand: `_`{*subscript*}
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp`{*superscript*}
  - Shorthand: `^`{*superscript*}
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Subscripts and Superscripts

- Subscripts are created using the command: `\sb{`*subscript*`}`
  - Shorthand: `_{`*subscript*`}`
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp{`*superscript*`}`
  - Shorthand: `^{`*superscript*`}`
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Subscripts and Superscripts

- Subscripts are created using the command: `\sb`{*subscript*}
  - Shorthand: `_`{*subscript*}
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp`{*superscript*}
  - Shorthand: `^`{*superscript*}
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Subscripts and Superscripts

- Subscripts are created using the command: `\sb`{*subscript*}
  - Shorthand: `_`{*subscript*}
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp`{*superscript*}
  - Shorthand: `^`{*superscript*}
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Subscripts and Superscripts

- Subscripts are created using the command: `\sb{`*subscript*`}`
  - Shorthand: `_{`*subscript*`}`
  - Example: `$a\sb{0}$` or `$a_{0}$` or `$a_0$` all produce: a_0
- Superscripts are created using the command `\sp{`*superscript*`}`
  - Shorthand: `^{`*superscript*`}`
  - Example: `$x\sp{2}$` or `$x^{2}$` or `$x^2$` all produce: x^2
- Subscripts and superscripts can be combined.

  Example: `$a_0^2$` produces a_0^2

Examples

```
A quadratic function:
\[f(x) = a_0 + a_1 x + a_2 x^2\]
```

INPUT

A quadratic function:

$$f(x) = a_0 + a_1 x + a_2 x^2$$

OUTPUT

```
Compare $a_b^c$ with $a_{b^c}$.
```

INPUT

Compare a_b^c with a_{b^c}.

OUTPUT

Fractions and Square Roots

- Fractions are produced using:
  `\frac`{*numerator*}{*denominator*}

- Roots are produced using:
  `\sqrt[`*n*`]`{*maths*}

- Example:

```
\begin{displaymath}
f(x_1, x_2) = x_1^2
+ e^{x_2} +
\frac{\sqrt[3]{a}}%
{1+\sqrt{x_2}}
\end{displaymath}
```

INPUT

$$f(x_1, x_2) = x_1^2 + e^{x_2} + \frac{\sqrt[3]{a}}{1+\sqrt{x_2}}$$

OUTPUT

Function Names

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

Wrong: `$log x$` $logx$

Correct: `$\log x$` $\log x$

Commands in blue indicate commands that can take a limit which can be specified using the _ command.

Examples

```
\begin{displaymath}
\exp(ix) = \sin(x) + i\cos(x)
\end{displaymath}
```

INPUT

$$\exp(ix) = \sin(x) + i\cos(x)$$

OUTPUT

```
\begin{displaymath}
\lim_{x \rightarrow 0} f(x)
\end{displaymath}
```

INPUT

$$\lim_{x \rightarrow 0} f(x)$$

OUTPUT

Lower Case Greek Letters

\alpha	α	\beta	β	\gamma	γ
\delta	δ	\epsilon	ϵ	\varepsilon	ε
\zeta	ζ	\eta	η	\theta	θ
\vartheta	ϑ	\iota	ι	\kappa	κ
\lambda	λ	\mu	μ	\nu	ν
\xi	ξ	\pi	π	\varpi	ϖ
\rho	ρ	\varrho	ϱ	\sigma	σ
\varsigma	ς	\tau	τ	\upsilon	υ
\phi	ϕ	\varphi	φ	\chi	χ
\psi	ψ	\omega	ω		

Upper Case Greek Letters

`\Gamma`	Γ	`\Delta`	Δ	`\Theta`	Θ
`\Lambda`	Λ	`\Xi`	Ξ	`\Pi`	Π
`\Sigma`	Σ	`\Upsilon`	Υ	`\Phi`	Φ
`\Psi`	Ψ	`\Omega`	Ω		

Example:

```
\begin{displaymath}
e^{i\theta} = \cos\theta
+ i\sin\theta
\end{displaymath}
```

INPUT

$$e^{i\theta} = \cos\theta + i\sin\theta$$

OUTPUT

Symbols with Two Sizes

The following symbols have different sizes depending on whether they are in displayed maths or in-line maths:

`\bigcap`	$\bigcap$	$\displaystyle\bigcap$	`\bigcup`	$\bigcup$	$\displaystyle\bigcup$	`\bigodot`	$\bigodot$	$\displaystyle\bigodot$
`\bigotimes`	$\bigotimes$	$\displaystyle\bigotimes$	`\bigoplus`	$\bigoplus$	$\displaystyle\bigoplus$	`\bigsqcup`	$\bigsqcup$	$\displaystyle\bigsqcup$
`\biguplus`	$\biguplus$	$\displaystyle\biguplus$	`\bigvee`	$\bigvee$	$\displaystyle\bigvee$	`\bigwedge`	$\bigwedge$	$\displaystyle\bigwedge$
`\coprod`	$\coprod$	$\displaystyle\coprod$	`\int`	$\int$	$\displaystyle\int$	`\oint`	$\oint$	$\displaystyle\oint$
`\prod`	$\prod$	$\displaystyle\prod$	`\sum`	$\sum$	$\displaystyle\sum$			

Examples

```
\begin{displaymath}
f(x) = \sum_{i=0}^{n}
\alpha_i x^i
\end{displaymath}
```

INPUT

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

OUTPUT

```
In text :
\begin{math}
f(x) = \sum_{i=0}^n
\alpha_i x^i
\end{math}
```

INPUT

In text : $f(x) = \sum_{i=0}^{n} \alpha_i x^i$

OUTPUT

Exercise 10 : Basic Mathematics (Page 11)

ⓔ Produce the output shown in Figure 8 on page 11 of the handouts.

⚠
- Produce the output shown in Figure 9 on page 11 of the handouts.
- Remember to typeset the $f(x)$, f and x in the text in maths mode.

Try making the equation a numbered equation.

Delimiters

- Placing brackets around a tall object in maths mode, such as fractions, does not look right if you use normal sized brackets. For example:

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

INPUT

$$(\frac{1}{1+x})$$

OUTPUT

- Under such circumstances, it is better to use the commands: `\left`*delimiter* and `\right`*delimiter*

- Note that you must always have matching `\left` and `\right` commands, although the delimiters used may be different.

Delimiters

- Placing brackets around a tall object in maths mode, such as fractions, does not look right if you use normal sized brackets. For example:

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

INPUT

$$(\frac{1}{1+x})$$

OUTPUT

- Under such circumstances, it is better to use the commands: `\left`*delimiter* and `\right`*delimiter*

- Note that you must always have matching `\left` and `\right` commands, although the delimiters used may be different.

Delimiters

- Placing brackets around a tall object in maths mode, such as fractions, does not look right if you use normal sized brackets. For example:

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

Input

$$(\frac{1}{1+x})$$

Output

- Under such circumstances, it is better to use the commands: `\left`*delimiter* and `\right`*delimiter*

- Note that you must always have matching `\left` and `\right` commands, although the delimiters used may be different.

Delimiters

(	$($	)	$)$	[	$[$	]	$]$
\{	$\{$	\}	$\}$	|	$\vert$	\\\|	$\Vert$
/	$/$	\backslash	$\backslash$	\langle	$\langle$	\rangle	$\rangle$
\lfloor	$\lfloor$	\rfloor	$\rfloor$	\lceil	$\lceil$	\rceil	$\rceil$
\uparrow	$\uparrow$	\downarrow	$\downarrow$	\Uparrow	$\Uparrow$	\Downarrow	$\Downarrow$
\updownarrow	$\updownarrow$	\Updownarrow	$\Updownarrow$				

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

Examples

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

INPUT

$$\left(\frac{1}{1+x}\right)$$

OUTPUT

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

INPUT

$$\left|\frac{1}{1+x}\right|$$

OUTPUT

Arrays

- Arrays can be created using the array *environment*.
- Similar to the tabular environment, but must be in maths mode.
- Elements are arranged in rows and columns to form mathematical structures such as vectors and matrices.

```
\begin{displaymath}
\begin{array}{cc}
0 & 1 \\
2 & 3
\end{array}
\end{displaymath}
```

INPUT

$$\begin{array}{cc} 0 & 1 \\ 2 & 3 \end{array}$$

OUTPUT

```
\begin{displaymath}
\left (
\begin{array}{cc}
0 & 1 \\
2 & 3
\end{array}
\right )
\end{displaymath}
```

INPUT

$$\left(\begin{array}{cc} 0 & 1 \\ 2 & 3 \end{array} \right)$$

OUTPUT

Delimiters Don’t Need to Match

```
\begin{displaymath}
\left[
\begin{array}{cc}
0 & 1 \\
2 & 3
\end{array}
\right\}
\end{displaymath}
```

INPUT

$$\left[\begin{array}{cc} 0 & 1 \\ 2 & 3 \end{array} \right\}$$

OUTPUT

Example Using Invisible Delimiter

```
\begin{displaymath}
f(x) =
\left \{
\begin{array}{cl}
0 & x \leq 0 \\
1 & x > 0
\end{array}
\right .
\end{displaymath}
```

INPUT

$$f(x) = \left\{ \begin{array}{cl} 0 & x \leq 0 \\ 1 & x > 0 \end{array} \right.$$

OUTPUT

Exercise 11 : Arrays (Page 12)

ⓔ Create the output shown in Figure 10 on page 12 of the handouts.

⚠
- Create the output shown in Figure 11 on page 12 of the handouts.
- You will need the following commands:

`\cdots`	$\cdots$
`\vdots`	$\vdots$
`\ddots`	$\ddots$
`\neq`	$\neq$

Multiline Formulæ

- The displaymath and equation environments only allow one line of mathematics.
- The eqnarray environment allows multiple equations to be aligned.
- The eqnarray environment has three columns: the first is right aligned, the second is centrally aligned and the third is left aligned.
- Each line is numbered in the eqnarray environment.
- The eqnarray* environment is unnumbered.
- To suppress line numbering in the eqnarray, use the command `\nonumber` on the appropriate line.

Multiline Formulæ

- The displaymath and equation environments only allow one line of mathematics.
- The eqnarray environment allows multiple equations to be aligned.
- The eqnarray environment has three columns: the first is right aligned, the second is centrally aligned and the third is left aligned.
- Each line is numbered in the eqnarray environment.
- The eqnarray* environment is unnumbered.
- To suppress line numbering in the eqnarray, use the command `\nonumber` on the appropriate line.

Multiline Formulæ

- The displaymath and equation environments only allow one line of mathematics.
- The eqnarray environment allows multiple equations to be aligned.
- The eqnarray environment has three columns: the first is right aligned, the second is centrally aligned and the third is left aligned.
- Each line is numbered in the eqnarray environment.
- The eqnarray* environment is unnumbered.
- To suppress line numbering in the eqnarray, use the command `\nonumber` on the appropriate line.

Multiline Formulæ

- The displaymath and equation environments only allow one line of mathematics.
- The eqnarray environment allows multiple equations to be aligned.
- The eqnarray environment has three columns: the first is right aligned, the second is centrally aligned and the third is left aligned.
- Each line is numbered in the eqnarray environment.
- The eqnarray* environment is unnumbered.
- To suppress line numbering in the eqnarray, use the command `\nonumber` on the appropriate line.

Multiline Formulæ

- The displaymath and equation environments only allow one line of mathematics.
- The eqnarray environment allows multiple equations to be aligned.
- The eqnarray environment has three columns: the first is right aligned, the second is centrally aligned and the third is left aligned.
- Each line is numbered in the eqnarray environment.
- The eqnarray* environment is unnumbered.
- To suppress line numbering in the eqnarray, use the command `\nonumber` on the appropriate line.

Multiline Formulæ

- The displaymath and equation environments only allow one line of mathematics.
- The eqnarray environment allows multiple equations to be aligned.
- The eqnarray environment has three columns: the first is right aligned, the second is centrally aligned and the third is left aligned.
- Each line is numbered in the eqnarray environment.
- The eqnarray* environment is unnumbered.
- To suppress line numbering in the eqnarray, use the command `\nonumber` on the appropriate line.

```
\begin{eqnarray}
\ln(f(x)) & = & x^2 + \frac{1}{x+3}\\
f(x) & = & \exp \left ( x^2
+ \frac{1}{x+3} \right )
\end{eqnarray}
```

INPUT

$$\ln(f(x)) = x^2 + \frac{1}{x+3} \tag{2}$$

$$f(x) = \exp\left(x^2 + \frac{1}{x+3}\right) \tag{3}$$

OUTPUT

```
\begin{eqnarray}
\ln(f(x)) & = & x^2 + \frac{1}{x+3} \nonumber\\
f(x) & = & \exp \left ( x^2
+ \frac{1}{x+3} \right )
\end{eqnarray}
```

INPUT

$$\begin{array}{rcl}
\ln(f(x)) & = & x^2 + \dfrac{1}{x+3} \\
f(x) & = & \exp\left(x^2 + \dfrac{1}{x+3}\right) \qquad (4)
\end{array}$$

OUTPUT

Exercise 12 : Multiline Formulæ (Page 13)

ⓔ Produce the output shown in Figure 12 on page 13 of the handouts.

⚠ Produce the output shown in Figure 13 on page 13 of the handouts.

- You will need the following commands:

`\approx`	$\approx$	`\pm`	$\pm$
`\partial`	∂	`\leq`	$\leq$
`\varepsilon`	ε		

Cross-Referencing

- Assign a textual label using \label{*string*}

  Example:

  ```
  \section{Introduction}
  \label{sec:intro}
  ```

  Example:

  ```
  \begin{equation}
  E = mc^2
  \label{eqn:einstein}
  \end{equation}
  ```

- Refer to the object using \ref{*string*}.

- Refer to the page that the object is on using \pageref{*string*}.

Cross-Referencing

- Assign a textual label using `\label{`*string*`}`

  Example:

  ```
  \section{Introduction}
  \label{sec:intro}
  ```

  Example:

  ```
  \begin{equation}
  E = mc^2
  \label{eqn:einstein}
  \end{equation}
  ```

- Refer to the object using `\ref{`*string*`}`.

- Refer to the page that the object is on using `\pageref{`*string*`}`.

Cross-Referencing

- Assign a textual label using \label{*string*}

  Example:

  ```
  \section{Introduction}
  \label{sec:intro}
  ```

  Example:

  ```
  \begin{equation}
  E = mc^2
  \label{eqn:einstein}
  \end{equation}
  ```

- Refer to the object using \ref{*string*}.

- Refer to the page that the object is on using \pageref{*string*}.

Cross-Referencing

- Assign a textual label using `\label{`*string*`}`

  Example:

  ```
  \section{Introduction}
  \label{sec:intro}
  ```

  Example:

  ```
  \begin{equation}
  E = mc^2
  \label{eqn:einstein}
  \end{equation}
  ```

- Refer to the object using `\ref{`*string*`}`.

- Refer to the page that the object is on using `\pageref{`*string*`}`.

Cross-Referencing

- Assign a textual label using `\label{`*string*`}`

  Example:

  ```
  \section{Introduction}
  \label{sec:intro}
  ```

  Example:

  ```
  \begin{equation}
  E = mc^2
  \label{eqn:einstein}
  \end{equation}
  ```

- Refer to the object using `\ref{`*string*`}`.

- Refer to the page that the object is on using `\pageref{`*string*`}`.

Examples

```
\section{Introduction}
\label{sec:intro}
\ldots
See Section~\ref{sec:intro}
for a brief introduction.
```

INPUT

1 Introduction

…See Section 1 for a brief introduction.

OUTPUT

```
See subsection~\ref{sec:ex}
for examples.

\subsection{Examples}
\label{sec:ex}
```

INPUT

See subsection 2.3 for examples.

2.3 Examples

OUTPUT

Examples

```
See Appendix~\ref{apd:tables}
for tables\ldots
\appendix
\section{Tables}\label{apd:tables}
```

INPUT

See Appendix A for tables…

A Tables

OUTPUT

```
\begin{equation}
\label{eqn:Emc}
E = mc^2
\end{equation}
\ldots
See Equation~\ref{eqn:Emc}
on page~\pageref{eqn:Emc}.
```

INPUT

$$E = mc^2 \qquad (5)$$

…See Equation 5 on page 253.

OUTPUT

Exercise 13 (Page 14)

- Reproduce the document shown in Figure 14 on page 14 of the handouts using `\label` and `\ref`. You will need to remember how to:
  - create sections
  - emphasize text
  - create numbered equations
  - have in-line mathematics

⚠ Also try inserting an extra section between the introductory section and the section on Bayes' Theorem, and try inserting another equation, to see how LaTeX automatically updates the cross-references.

- Try adding a title and table of contents

Packages

- *Packages* are files with the extension `.sty`
- Packages can redefine existing commands, or provide new commands.
- To include a package in your document:

  `\usepackage`[*options*]{*package-name*}

- This command can only be used in the *preamble*.

Packages

- *Packages* are files with the extension `.sty`
- Packages can redefine existing commands, or provide new commands.
- To include a package in your document:

  `\usepackage` [*options*] {*package-name*}

- This command can only be used in the *preamble*.

Packages

- *Packages* are files with the extension `.sty`
- Packages can redefine existing commands, or provide new commands.
- To include a package in your document:

  \usepackage[*options*]{*package-name*}

- This command can only be used in the *preamble*.

Packages

- *Packages* are files with the extension `.sty`
- Packages can redefine existing commands, or provide new commands.
- To include a package in your document:

  `\usepackage`[*options*]{*package-name*}

- This command can only be used in the *preamble*.

Examples

- The babel package redefines names such as “Chapter” to another language.
- ukdate redefines `\today`, as does datetime.
- xr defines a new command

  `\externaldocument{`*filename*`}`

  allowing you to refer to labels defined in another document.
- varioref defines the command `\vref` which works much like `\ref` but also adds the page number, e.g. `Fig~\vref{fig:ex}` can produce: “Fig 2 on page 42”, or “Fig 2 on the following page” or simply “Fig 2”.

Examples

- The babel package redefines names such as “Chapter” to another language.
- ukdate redefines `\today`, as does datetime.
- xr defines a new command

  `\externaldocument{`*filename*`}`

  allowing you to refer to labels defined in another document.
- varioref defines the command `\vref` which works much like `\ref` but also adds the page number, e.g. `Fig~\vref{fig:ex}` can produce: “Fig 2 on page 42”, or “Fig 2 on the following page” or simply “Fig 2”.

Examples

- The babel package redefines names such as “Chapter” to another language.
- ukdate redefines `\today`, as does datetime.
- xr defines a new command

  ```
  \externaldocument{filename}
  ```

  allowing you to refer to labels defined in another document.
- varioref defines the command `\vref` which works much like `\ref` but also adds the page number, e.g. `Fig~\vref{fig:ex}` can produce: “Fig 2 on page 42”, or “Fig 2 on the following page” or simply “Fig 2”.

Examples

- The babel package redefines names such as “Chapter” to another language.
- ukdate redefines `\today`, as does datetime.
- xr defines a new command

  `\externaldocument{`*filename*`}`

  allowing you to refer to labels defined in another document.
- varioref defines the command `\vref` which works much like `\ref` but also adds the page number, e.g. `Fig~\vref{fig:ex}` can produce: “Fig 2 on page 42”, or “Fig 2 on the following page” or simply “Fig 2”.

Examples cont.

- hyperref defines commands that allows you to have active links in your document if used in combination with PDFLATEX. E.g.

```
\href{http://www.tex.ac.uk/}{%
the \TeX\ Archive}
```

INPUT

the TEX Archive

OUTPUT

- ifpdf defines the conditional `\ifpdf` which can be used to determine whether LATEX or PDFLATEX is being used. E.g.

```
\ifpdf
A PDF\LaTeX\ document
\else
A \LaTeX\ document
\fi
```

INPUT

A PDFLATEX document

OUTPUT

Examples cont.

- hyperref defines commands that allows you to have active links in your document if used in combination with PDFLATEX. E.g.

```
\href{http://www.tex.ac.uk/}{%
the \TeX\ Archive}
```

INPUT

the TEX Archive

OUTPUT

- ifpdf defines the conditional `\ifpdf` which can be used to determine whether LATEX or PDFLATEX is being used. E.g.

```
\ifpdf
A PDF\LaTeX\ document
\else
A \LaTeX\ document
\fi
```

INPUT

A PDFLATEX document

OUTPUT

Example

```
\documentclass[a4paper]{article}
\begin{document}
\today
\end{document}
```

Input

```
October 4, 2004
```

Output

```
\documentclass[a4paper]{article}
\usepackage[short]{datetime}
\begin{document}
\today
\end{document}
```

Input

```
Mon 4th Oct, 2004
```

Output

Example

```
\documentclass[a4paper]{article}
\begin{document}
\today
\end{document}
```

INPUT

October 4, 2004

OUTPUT

```
\documentclass[a4paper]{article}
\usepackage[short]{datetime}
\begin{document}
\today
\end{document}
```

INPUT

Mon 4th Oct, 2004

OUTPUT

Self-Extracting Documentation

- *Packages* not currently on your TeX installation can be downloaded from the TeX archive.
- Increasingly packages are bundled up with their documentation in a file with the extension `.dtx`
- The package should also come with a driver or installation script (`.ins`)
- The documentation can usually be obtained by LaTeXing the `.dtx` file. For example:

```
▶latex datetime.dtx
```

- The package can be extracted by LaTeXing the installation script. For example:

```
▶latex datetime.ins
```

Self-Extracting Documentation

- *Packages* not currently on your TEX installation can be downloaded from the TEX archive.

- Increasingly packages are bundled up with their documentation in a file with the extension `.dtx`

- The package should also come with a driver or installation script (`.ins`)

- The documentation can usually be obtained by LATEXing the `.dtx` file. For example:

  ▶`latex datetime.dtx`

- The package can be extracted by LATEXing the installation script. For example:

  ▶`latex datetime.ins`

Self-Extracting Documentation

- *Packages* not currently on your TEX installation can be downloaded from the TEX archive.
- Increasingly packages are bundled up with their documentation in a file with the extension .dtx
- The package should also come with a driver or installation script (.ins)
- The documentation can usually be obtained by LATEXing the .dtx file. For example:

  ```
  ▶latex datetime.dtx
  ```
- The package can be extracted by LATEXing the installation script. For example:

  ```
  ▶latex datetime.ins
  ```

Self-Extracting Documentation

- *Packages* not currently on your TEX installation can be downloaded from the TEX archive.
- Increasingly packages are bundled up with their documentation in a file with the extension `.dtx`
- The package should also come with a driver or installation script (`.ins`)
- The documentation can usually be obtained by LATEXing the `.dtx` file. For example:

  ```
  ▶latex datetime.dtx
  ```
- The package can be extracted by LATEXing the installation script. For example:

  ```
  ▶latex datetime.ins
  ```

Self-Extracting Documentation

- *Packages* not currently on your TEX installation can be downloaded from the TEX archive.
- Increasingly packages are bundled up with their documentation in a file with the extension `.dtx`
- The package should also come with a driver or installation script (`.ins`)
- The documentation can usually be obtained by LATEXing the `.dtx` file. For example:

  ▶`latex datetime.dtx`
- The package can be extracted by LATEXing the installation script. For example:

  ▶`latex datetime.ins`

Exercise 14 (Page 15)

- Go back to the file `sectioning.tex` you edited in Exercise 4
  - Use the babel package. There are many language options available just list the contents of the babel directory:

    ▶`ls /sw4/teTeX/texmf/tex/generic/babel/`

    The options have the same name as the `.sty` files.

- Or go back to the document you created in Exercise 13.
  - Use the hyperref package to make the cross-references active, and use PDFLaTeX instead of LaTeX to produce a PDF file.

⚠ If you want to try extracting documentation and code from a `.dtx` file, you can copy the datetime package over:

▶`cp /home/sys/gcc/insecure/datetime.* .`

Citations

- thebibliography environment

```
\begin{thebibliography}{2}

\bibitem{clarke83} G. M. Clarke and D. Cooke.
\emph{A basic course in statistics}.
Chapman and Hall, 2nd edition, 1983.

\bibitem{goossens93} M. Goossens and F. Mittelbach.
\emph{The \LaTeX\ companion}.
Addison-Wesley, 1993.

\end{thebibliography}
```

- Use \cite[*text*]{*key-list*} to cite a reference in the bibliography

Example

```
See Goossens \emph{et
al.}~\cite{goossens93}
\ldots

\begin{thebibliography}{1}
\bibitem{goossens93}
M. Goossens and
F. Mittelbach.
\emph{The \LaTeX\
companion}.
Addison-Wesley, 1993.
\end{thebibliography}
```

INPUT

See Goossens *et al.* [1] . . .

References

[1] M. Goossens and F. Mittelbach. *The LaTeX companion*. Addison-Wesley, 1993.

OUTPUT

BibTEX

Use BibTEX to automatically generate thebibliography environment.

- Large database (`.bib`) containing many references.
- BibTEX will only include those that are cited in the document.
- Entries sorted.
- Entries consistently formatted.

Bibliography Database (.bib)

```
@entry type{keyword,
    field = "text",
    ⋮
    field = "text"
}
```

```
@book{kreyszig88,
   author    = "Kreyszig, Erwin",
   title     = "Advanced Engineering Mathematics",
   publisher = "Wiley",
   edition   = "6th",
   year      = 1988
}
```

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“abbrv” style)	
`"Alex Thomas von Neumann"`	A.T. von Neumann	
`"John Chris {Smith Jones}"`	J.C. Smith Jones	
`"van de Klee, Mary-Jane"`	M.-J. van de Klee	
`"Smith, Jr, Fred John"`	F.J. Smith, Jr	
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz	
Compare last example with:		
`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“`abbrv`” style)
`"Alex Thomas von Neumann"`	A.T. von Neumann
`"John Chris {Smith Jones}"`	J.C. Smith Jones
`"van de Klee, Mary-Jane"`	M.-J. van de Klee
`"Smith, Jr, Fred John"`	F.J. Smith, Jr
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz

Compare last example with:

`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“`abbrv`” style)	
`"Alex Thomas von Neumann"`	A.T. von Neumann	
`"John Chris {Smith Jones}"`	J.C. Smith Jones	
`"van de Klee, Mary-Jane"`	M.-J. van de Klee	
`"Smith, Jr, Fred John"`	F.J. Smith, Jr	
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz	
Compare last example with:		
`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“`abbrv`” style)	
`"Alex Thomas von Neumann"`	A.T. von Neumann	
`"John Chris {Smith Jones}"`	J.C. Smith Jones	
`"van de Klee, Mary-Jane"`	M.-J. van de Klee	
`"Smith, Jr, Fred John"`	F.J. Smith, Jr	
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz	

Compare last example with:

`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“`abbrv`” style)	
`"Alex Thomas von Neumann"`	A.T. von Neumann	
`"John Chris {Smith Jones}"`	J.C. Smith Jones	
`"van de Klee, Mary-Jane"`	M.-J. van de Klee	
`"Smith, Jr, Fred John"`	F.J. Smith, Jr	
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz	
Compare last example with:		
`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“`abbrv`” style)	
`"Alex Thomas von Neumann"`	A.T. von Neumann	
`"John Chris {Smith Jones}"`	J.C. Smith Jones	
`"van de Klee, Mary-Jane"`	M.-J. van de Klee	
`"Smith, Jr, Fred John"`	F.J. Smith, Jr	
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz	
Compare last example with:		
`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Author Format

Authors should be entered in one of the following formats:

- *forenames von surname*
- *von surname*, *forenames*
- *von surname*, *jr*, *forenames*

Examples:

`Entry`	Output (“`abbrv`” style)	
`"Alex Thomas von Neumann"`	A.T. von Neumann	
`"John Chris {Smith Jones}"`	J.C. Smith Jones	
`"van de Klee, Mary-Jane"`	M.-J. van de Klee	
`"Smith, Jr, Fred John"`	F.J. Smith, Jr	
`"Maria {\uppercase{d}e La} Cruz"`	M. De La Cruz	
Compare last example with:		
`"Maria De La Cruz"`	M. D. L. Cruz	(Incorrect!)

Multiple Authors

Multiple authors should be separated by the keyword and

```
@book{goossens97,
   author = "Goossens, Michel and Rahtz, Sebastian and
             Mittelbach, Frank",
   title = "The \LaTeX\ graphics companion: illustrating
            documents with \TeX\ and {PostScript}",
   publisher = "Addison Wesley Longman, Inc",
   year = 1997
}
```

Month Entries

- Bibliography styles always have three-letter abbreviations for months: `jan`, `feb`, `mar`, . . .
- Always use these abbreviations for consistency.

```
@inproceedings{talbot97,
   author    = "Talbot, Nicola and Cawley, Gavin",
   title     = "A fast index assignment algorithm for
                robust vector quantisation of image data",
   booktitle = "Proceedings of the I.E.E.E. International
                Conference on Image Processing",
   address   = "Santa Barbara, California, USA",
   month     = oct,
   year      = 1997
}
```

Example (`incollection`)

```
@incollection{wainwright,
   author    = "Wainwright, Robert B.",
   title     = "Hazards from {Northern} Native Foods",
   booktitle = "\emph{Clostridium botulinum}: Ecology and
                Control in Foods",
   chapter   = 12,
   pages     = "305--322",
   editor    = "Hauschild, Andreas H. W. and Dodds,
                Karen L.",
   publisher = "Marcel Dekker, Inc",
   year      = 1993
}
```

Declaring Databases and Bibliography Style

In your LaTeX *source code* (`.tex`):

- Declare the bibliography style:
  `\bibliographystyle{`*style-name*`}`

  Common Styles:

plain	Entries sorted alphabetically with numeric labels.
unsrt	Entries printed in order of citation with numeric labels.
alpha	Entries sorted alphabetically with labels formed from author's name and year of publication.
abbrv	Entries sorted alphabetically with first name, month and journal names abbreviated.

- Declare the bibliography database:
  `\bibliography{`*name*`}`

Declaring Databases and Bibliography Style

In your LaTeX *source code* (`.tex`):

- Declare the bibliography style:
  `\bibliographystyle{`*style-name*`}`
  Common Styles:

  plain — Entries sorted alphabetically with numeric labels.

  unsrt — Entries printed in order of citation with numeric labels.

  alpha — Entries sorted alphabetically with labels formed from author's name and year of publication.

  abbrv — Entries sorted alphabetically with first name, month and journal names abbreviated.

- Declare the bibliography database:
  `\bibliography{`*name*`}`

Declaring Databases and Bibliography Style

In your LaTeX *source code* (`.tex`):

- Declare the bibliography style:
  `\bibliographystyle{`*style-name*`}`
  Common Styles:

plain	Entries sorted alphabetically with numeric labels.
unsrt	Entries printed in order of citation with numeric labels.
alpha	Entries sorted alphabetically with labels formed from author's name and year of publication.
abbrv	Entries sorted alphabetically with first name, month and journal names abbreviated.

- Declare the bibliography database:
  `\bibliography{`*name*`}`

Example

In `filename.tex` (where `database.bib` contains the bibliography database):

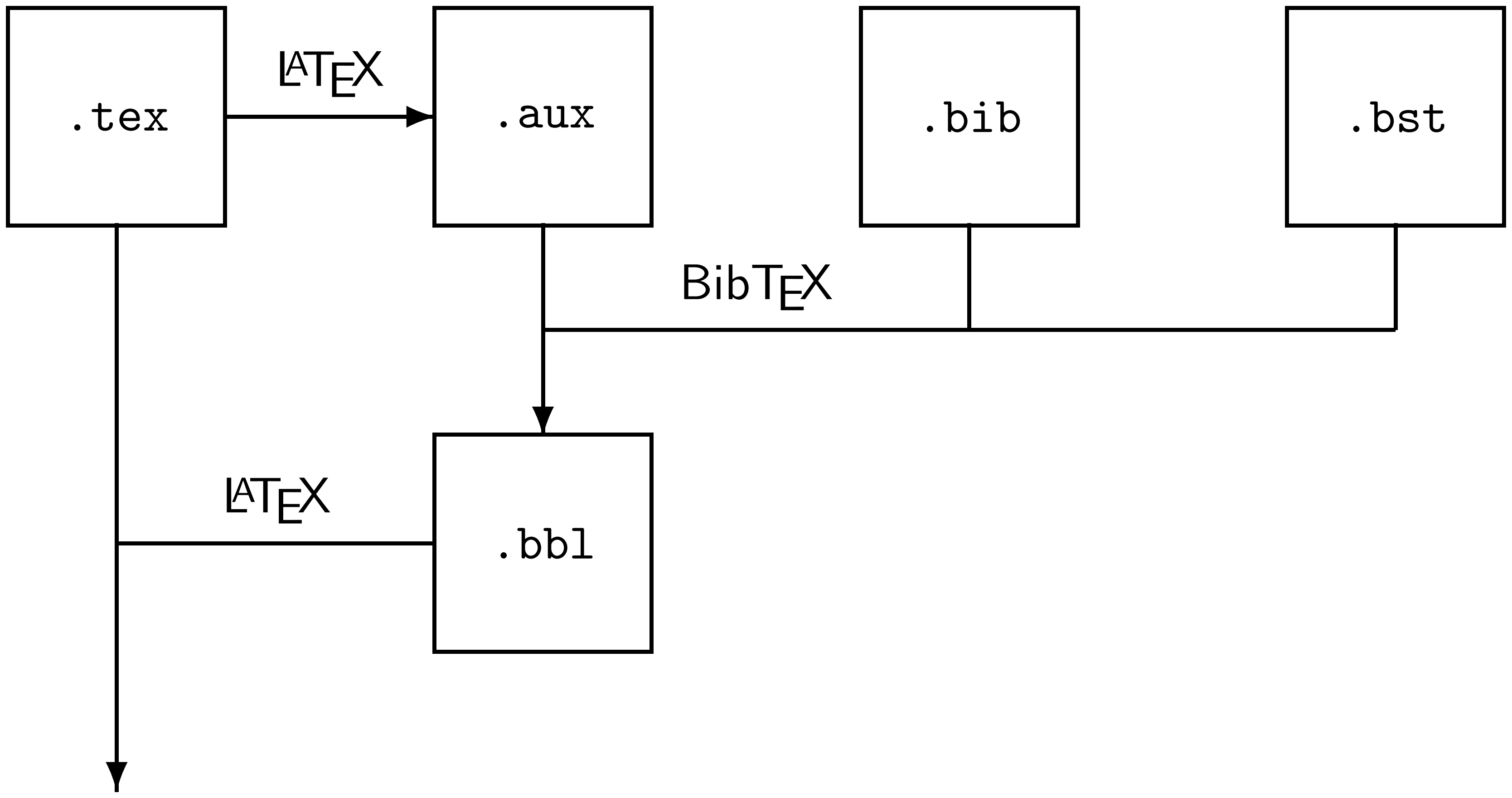
```
This is the document \ldots

\bibliographystyle{plain}
\bibliography{database}
```

INPUT

At the command prompt:

```
▶latex filename
▶bibtex filename
▶latex filename
▶latex filename
```

LaTeX/BibTeX Process



\bibliographystyle{**plain**}

[1] Gavin C. Cawley and Nicola L. C. Talbot. A fast index assignment algorithm for vector quantization over noisy transmission channels. *I.E.E. Electronic Letters*, 32(15):1343–1344, July 1996.

[2] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, 6th edition, 1988.

[3] Nicola L. C. Talbot and Gavin C. Cawley. A quadratic index assignment algorithm for vector quantisation over noisy transmission channels. In *Proceedings of the Institute of Acoustics Autumn Conference on Speech and Hearing*, volume 18, pages 195–199, November 1996.

[4] Nicola L. C. Talbot and Gavin C. Cawley. A fast index assignment algorithm for robust vector quantisation of image data. In *Proceedings of the I.E.E.E. International Conference on Image Processing*, Santa Barbara, California, USA, October 1997.

\bibliographystyle{alpha}

[CT96] Gavin C. Cawley and Nicola L. C. Talbot. A fast index assignment algorithm for vector quantization over noisy transmission channels. *I.E.E. Electronic Letters*, 32(15):1343–1344, July 1996.

[Kre88] Erwin Kreyszig. *Advanced Engineering Mathematics*. Wiley, 6th edition, 1988.

[TC96] Nicola L. C. Talbot and Gavin C. Cawley. A quadratic index assignment algorithm for vector quantisation over noisy transmission channels. In *Proceedings of the Institute of Acoustics Autumn Conference on Speech and Hearing*, volume 18, pages 195–199, November 1996.

[TC97] Nicola L. C. Talbot and Gavin C. Cawley. A fast index assignment algorithm for robust vector quantisation of image data. In *Proceedings of the I.E.E.E. International Conference on Image Processing*, Santa Barbara, California, USA, October 1997.

\bibliographystyle{**ieeetr**}

[1] E. Kreyszig, *Advanced Engineering Mathematics*. Wiley, 6th ed., 1988.

[2] N. L. C. Talbot and G. C. Cawley, “A quadratic index assignment algorithm for vector quantisation over noisy transmission channels,” in *Proceedings of the Institute of Acoustics Autumn Conference on Speech and Hearing*, vol. 18, pp. 195–199, Nov. 1996.

[3] G. C. Cawley and N. L. C. Talbot, “A fast index assignment algorithm for vector quantization over noisy transmission channels,” *I.E.E. Electronic Letters*, vol. 32, pp. 1343–1344, July 1996.

[4] N. L. C. Talbot and G. C. Cawley, “A fast index assignment algorithm for robust vector quantisation of image data,” in *Proceedings of the I.E.E.E. International Conference on Image Processing*, (Santa Barbara, California, USA), Oct. 1997.

\bibliographystyle{acm}

[1] Cawley, G. C., and Talbot, N. L. C. A fast index assignment algorithm for vector quantization over noisy transmission channels. *I.E.E. Electronic Letters 32*, 15 (July 1996), 1343–1344.

[2] Kreyszig, E. *Advanced Engineering Mathematics*, 6th ed. Wiley, 1988.

[3] Talbot, N. L. C., and Cawley, G. C. A quadratic index assignment algorithm for vector quantisation over noisy transmission channels. In *Proceedings of the Institute of Acoustics Autumn Conference on Speech and Hearing* (Nov. 1996), vol. 18, pp. 195–199.

[4] Talbot, N. L. C., and Cawley, G. C. A fast index assignment algorithm for robust vector quantisation of image data. In *Proceedings of the I.E.E.E. International Conference on Image Processing* (Santa Barbara, California, USA, Oct. 1997).

\bibliographystyle{**apalike**}

This style file requires the apalike package.

Cawley, G. C. and Talbot, N. L. C. (1996) A fast index assignment algorithm for vector quantization over noisy transmission channels. *I.E.E. Electronic Letters*, 32(15):1343–1344.

Kreyszig, E. (1988) *Advanced Engineering Mathematics*. Wiley, 6th edition.

Talbot, N. L. C. and Cawley, G. C. (1996). A quadratic index assignment algorithm for vector quantisation over noisy transmission channels. In *Proceedings of the Institute of Acoustics Autumn Conference on Speech and Hearing*, volume 18, pages 195–199.

Talbot, N. L. C. and Cawley, G. C. (1997). A fast index assignment algorithm for robust vector quantisation of image data. In *Proceedings of the I.E.E.E. International Conference on Image Processing*, Santa Barbara, California, USA.

Customising a BiBTeX Style

- If you want a BibTEX style file (`.bst`) that is slightly different from one that already exists, you can try copying the existing file to a new name and make minor modifications.
- Drawback: BibTEX is a low level language and is not for the faint-hearted!
- Consider using `makebst` instead. The file `makebst.tex` is an interactive TEX/LATEX script that can be used to create a customised `.bst` file. Simply type:

  ```
  ▶latex makebst
  ```

  at the command prompt, and follow the instructions.

Customising a BiBTeX Style

- If you want a BibTEX style file (`.bst`) that is slightly different from one that already exists, you can try copying the existing file to a new name and make minor modifications.

- Drawback: BibTEX is a low level language and is not for the faint-hearted!

- Consider using `makebst` instead. The file `makebst.tex` is an interactive TEX/LATEX script that can be used to create a customised `.bst` file. Simply type:

  ```
  ▶latex makebst
  ```

  at the command prompt, and follow the instructions.

Customising a BiBTeX Style

- If you want a BibTEX style file (`.bst`) that is slightly different from one that already exists, you can try copying the existing file to a new name and make minor modifications.
- Drawback: BibTEX is a low level language and is not for the faint-hearted!
- Consider using `makebst` instead. The file `makebst.tex` is an interactive TEX/LATEX script that can be used to create a customised `.bst` file. Simply type:

  ```
  ▶latex makebst
  ```

  at the command prompt, and follow the instructions.

Exercise 15 (Page 16)

- Produce a BibTeX database that contains the references shown in Figure 15 on page 17 of the handouts, and create the document shown in that figure.

- Try changing the bibliography style so that the entries are printed in order of citation. (You need the unsrt style for this). Try other styles, such as alpha, abbrv and acm, to see the differences between styles.

- If you have a number of citations, such as [3,2,4], you might prefer to have it printed as a range, such as [2–4], instead. There is a *package* called citesort that redefines the `\cite` command that will do this. Try using this package with the unsrt bibliography style.

⚠ If you are feeling adventurous, try creating your own customised bibliography style using makebst.

Lengths

- LaTeX has commands that represent lengths, such as `\textwidth`.
- There are two types of lengths: *rigid* and *rubber*.
- A rigid length is a fixed length, such as `4in`.
- A rubber length is a length with a certain amount of elasticity, for example: `2in plus 0.1in minus 0.1in`.

  A rubber length is a way of telling LaTeX your preferred length, and the amount of deviation from that length which you are prepared to put up with.
- Most LaTeX lengths are rigid lengths.

Lengths

- LaTeX has commands that represent lengths, such as `\textwidth`.
- There are two types of lengths: *rigid* and *rubber*.
- A rigid length is a fixed length, such as `4in`.
- A rubber length is a length with a certain amount of elasticity, for example: `2in plus 0.1in minus 0.1in`.

  A rubber length is a way of telling LaTeX your preferred length, and the amount of deviation from that length which you are prepared to put up with.
- Most LaTeX lengths are rigid lengths.

Lengths

- LaTeX has commands that represent lengths, such as `\textwidth`.
- There are two types of lengths: *rigid* and *rubber*.
- A rigid length is a fixed length, such as `4in`.
- A rubber length is a length with a certain amount of elasticity, for example: `2in plus 0.1in minus 0.1in`.

  A rubber length is a way of telling LaTeX your preferred length, and the amount of deviation from that length which you are prepared to put up with.
- Most LaTeX lengths are rigid lengths.

Lengths

- LaTeX has commands that represent lengths, such as `\textwidth`.
- There are two types of lengths: *rigid* and *rubber*.
- A rigid length is a fixed length, such as `4in`.
- A rubber length is a length with a certain amount of elasticity, for example: `2in plus 0.1in minus 0.1in`.

  A rubber length is a way of telling LaTeX your preferred length, and the amount of deviation from that length which you are prepared to put up with.
- Most LaTeX lengths are rigid lengths.

Lengths

- LaTeX has commands that represent lengths, such as `\textwidth`.
- There are two types of lengths: *rigid* and *rubber*.
- A rigid length is a fixed length, such as `4in`.
- A rubber length is a length with a certain amount of elasticity, for example: `2in plus 0.1in minus 0.1in`.

  A rubber length is a way of telling LaTeX your preferred length, and the amount of deviation from that length which you are prepared to put up with.
- Most LaTeX lengths are rigid lengths.

Lengths

- LaTeX has commands that represent lengths, such as `\textwidth`.
- There are two types of lengths: *rigid* and *rubber*.
- A rigid length is a fixed length, such as `4in`.
- A rubber length is a length with a certain amount of elasticity, for example: `2in plus 0.1in minus 0.1in`.

  A rubber length is a way of telling LaTeX your preferred length, and the amount of deviation from that length which you are prepared to put up with.
- Most LaTeX lengths are rigid lengths.

Common Units

pt	Point ($\frac{1}{72.27}$in)
bp	Big point, or PostScript point ($\frac{1}{72}$in)
mm	Millimetre (2.845pt)
cm	Centimetre (28.45pt)
in	Inch (25.4mm)
ex	Height of lowercase x in current font
em	Width of capital M in current font

Changing Lengths

- A length can be assigned a new value using the command:

  \setlength{*cmd*}{*length*}

  For example:

  ```
  \setlength{\textwidth}{6in}
  ```

- A length can be incremented using the command:

  \addtolength{*cmd*}{*length*}

  so to make the text width 1in wider than it was previously, do:

  ```
  \addtolength{\textwidth}{1in}
  ```

Changing Lengths

- A length can be assigned a new value using the command:

  ```
  \setlength{cmd}{length}
  ```

  For example:

  ```
  \setlength{\textwidth}{6in}
  ```

- A length can be incremented using the command:

  ```
  \addtolength{cmd}{length}
  ```

  so to make the text width 1in wider than it was previously, do:

  ```
  \addtolength{\textwidth}{1in}
  ```

Changing Lengths

- A length can be assigned a new value using the command:

  ```
  \setlength{cmd}{length}
  ```

  For example:

  ```
  \setlength{\textwidth}{6in}
  ```

- A length can be incremented using the command:

  ```
  \addtolength{cmd}{length}
  ```

  so to make the text width 1in wider than it was previously, do:

  ```
  \addtolength{\textwidth}{1in}
  ```

Changing Lengths

- A length can be assigned a new value using the command:

  \setlength{*cmd*}{*length*}

  For example:

  ```
  \setlength{\textwidth}{6in}
  ```

- A length can be incremented using the command:

  \addtolength{*cmd*}{*length*}

  so to make the text width 1in wider than it was previously, do:

  ```
  \addtolength{\textwidth}{1in}
  ```

Lengths

- There are three more commands that can change a length, and they are:

  ```
  \settowidth{cmd}{text}
  \settoheight{cmd}{text}
  \settodepth{cmd}{text}
  ```

  These set the length *cmd* to the width, height or depth of the *text*. Note that the actual text itself is not displayed.

- To create a new length:

  ```
  \newlengthcmd
  ```

- To display the value of a length:

  ```
  \thecmd
  ```

Lengths

- There are three more commands that can change a length, and they are:

  ```
  \settowidth{cmd}{text}
  \settoheight{cmd}{text}
  \settodepth{cmd}{text}
  ```

  These set the length *cmd* to the width, height or depth of the *text*. Note that the actual text itself is not displayed.

- To create a new length:

  ```
  \newlengthcmd
  ```

- To display the value of a length:

  ```
  \thecmd
  ```

Lengths

- There are three more commands that can change a length, and they are:

  ```
  \settowidth{cmd}{text}
  \settoheight{cmd}{text}
  \settodepth{cmd}{text}
  ```

  These set the length *cmd* to the width, height or depth of the *text*. Note that the actual text itself is not displayed.

- To create a new length:

  ```
  \newlengthcmd
  ```

- To display the value of a length:

  ```
  \thecmd
  ```

Example

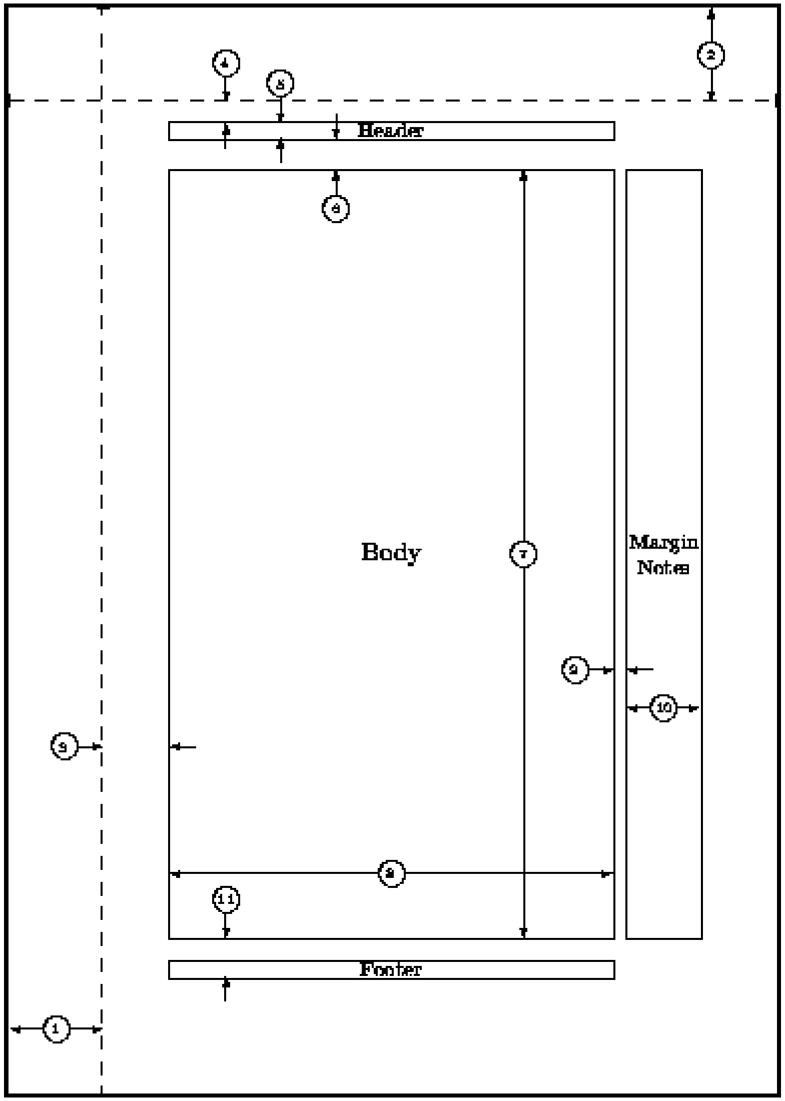
```
% define new length
\newlength\mylen
% set it to the width of the text
\settowidth{\mylen}{Hello}
% Display the value
Width=\the\mylen.
```

INPUT

Width=22.50005pt.

OUTPUT

Layout Lengths



1	one inch + `\hoffset`
2	one inch + `\voffset`
3	`\oddsidemargin`
4	`\topmargin`
5	`\headheight`
6	`\headsep`
7	`\textheight`
8	`\textwidth`
9	`\marginparsep`
10	`\marginparwidth`
11	`\footskip`

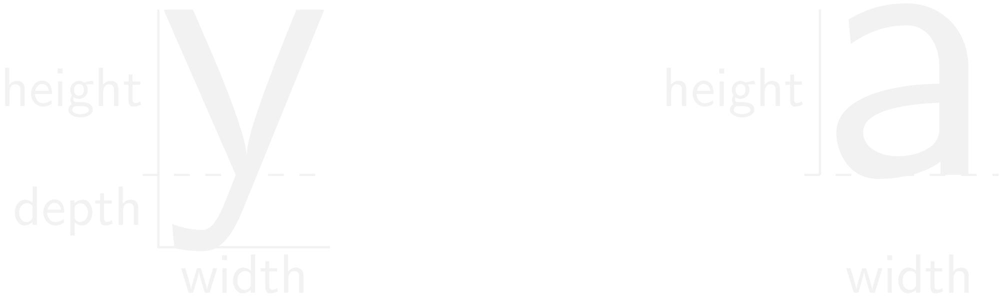
(Diagram generated using the layout package)

Exercise 16 (Page 17)

- Go back to the document you created in Exercise 1
- Change the paragraph indentation (`\parindent`) to 0pt
- Change the gap between paragraphs (`\parskip`) to 3ex.

Boxes

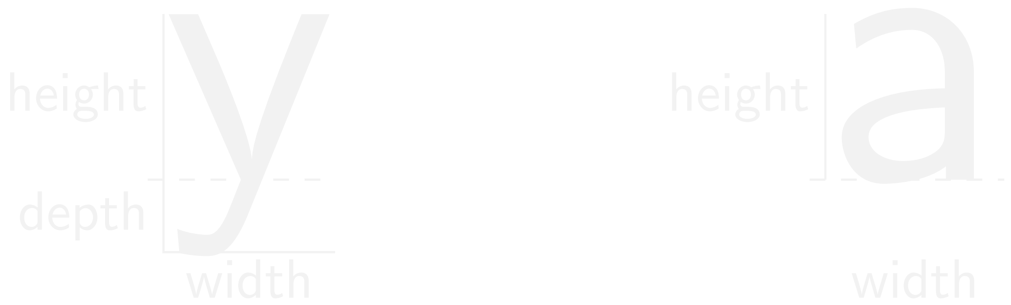
- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter



- More complicated boxes are made up of smaller boxes.
- Example: The phrase “cabbages and peas” is made up of 15 boxes: cabbages and peas
- Between the boxes is “glue”.
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

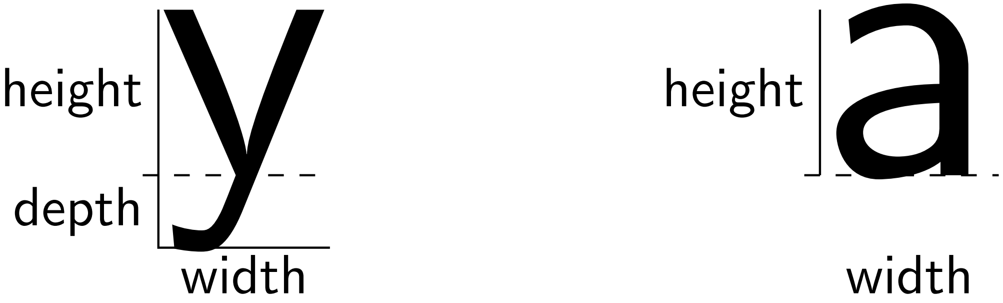
- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter



- More complicated boxes are made up of smaller boxes.
- Example: The phrase “cabbages and peas” is made up of 15 boxes: cabbages and peas
- Between the boxes is “glue”.
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

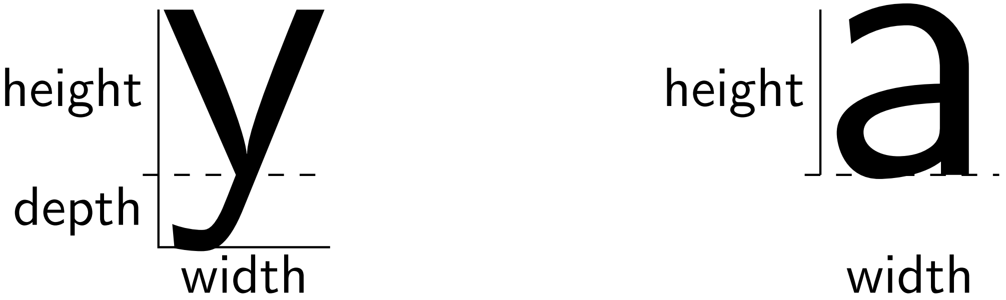
- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter



- More complicated boxes are made up of smaller boxes.
- Example: The phrase "cabbages and peas" is made up of 15 boxes: cabbages and peas
- Between the boxes is "glue".
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

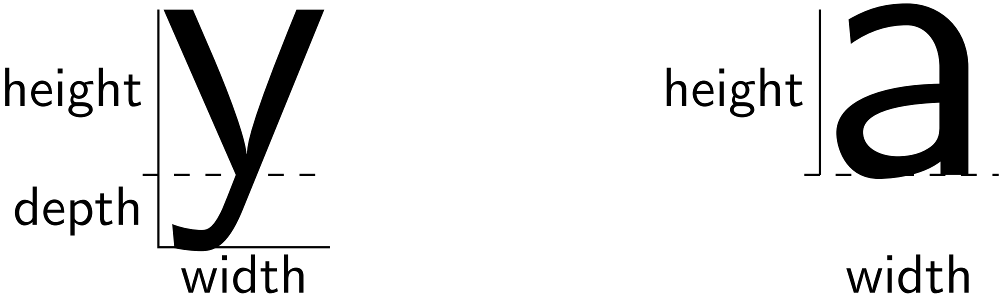
- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter



- More complicated boxes are made up of smaller boxes.
- Example: The phrase “cabbages and peas” is made up of 15 boxes: cabbages and peas
- Between the boxes is “glue”.
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

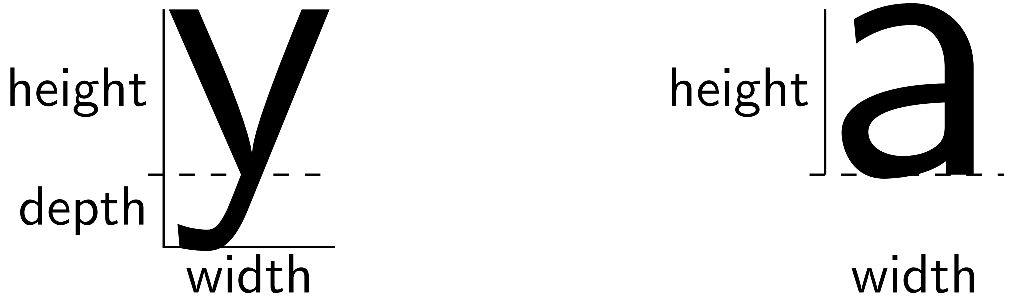
- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter



- More complicated boxes are made up of smaller boxes.
- Example: The phrase “cabbages and peas” is made up of 15 boxes: $\boxed{\text{c}}\boxed{\text{a}}\boxed{\text{b}}\boxed{\text{b}}\boxed{\text{a}}\boxed{\text{g}}\boxed{\text{e}}\boxed{\text{s}}$ $\boxed{\text{a}}\boxed{\text{n}}\boxed{\text{d}}$ $\boxed{\text{p}}\boxed{\text{e}}\boxed{\text{a}}\boxed{\text{s}}$
- Between the boxes is “glue”.
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

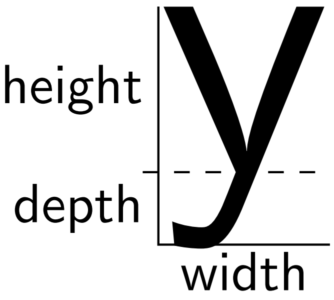
- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter

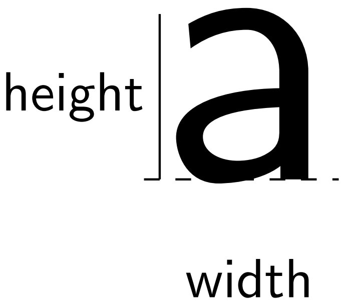


- More complicated boxes are made up of smaller boxes.
- Example: The phrase "cabbages and peas" is made up of 15 boxes: cabbages and peas
- Between the boxes is "glue".
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

- Everything on a page can be broken down into boxes.
- Each box has an associated width, height and depth.
- The simplest form of box is a single letter





- More complicated boxes are made up of smaller boxes.
- Example: The phrase “cabbages and peas” is made up of 15 boxes: $\boxed{c}\boxed{a}\boxed{b}\boxed{b}\boxed{a}\boxed{g}\boxed{e}\boxed{s}\ \boxed{a}\boxed{n}\boxed{d}\ \boxed{p}\boxed{e}\boxed{a}\boxed{s}$
- Between the boxes is “glue”.
- The job of the typesetter is to fix the boxes together according to typographical rules.

Boxes

- Boxes are treated as a single object.
- They can occur in the middle of a line.
- They can never be broken across a line.
- They can be vertically aligned.
- You have already used some boxes.

Boxes

- Boxes are treated as a single object.
- They can occur in the middle of a line.
- They can never be broken across a line.
- They can be vertically aligned.
- You have already used some boxes.

Boxes

- Boxes are treated as a single object.
- They can occur in the middle of a line.
- They can never be broken across a line.
- They can be vertically aligned.
- You have already used some boxes.

Boxes

- Boxes are treated as a single object.
- They can occur in the middle of a line.
- They can never be broken across a line.
- They can be vertically aligned.
- You have already used some boxes.

Boxes

- Boxes are treated as a single object.
- They can occur in the middle of a line.
- They can never be broken across a line.
- They can be vertically aligned.
- You have already used some boxes.

Example of a Box: The tabular Environment

```
Baseline
\begin{tabular}[c]{l}line 1\\line 2\\line 3\end{tabular}
\begin{tabular}[b]{l}line 1\\line 2\\line 3\end{tabular}
\begin{tabular}[t]{l}line 1\\line 2\\line 3\end{tabular}
```

INPUT

```
                    line 1
          line 1    line 2
Baseline  line 2    line 3    line 1
          line 3              line 2
                              line 3
```

OUTPUT

Basic Types of Boxes

- `\mbox{`*contents*`}`
  Simplest type of box.
  - Prevents text inside it from being broken across a line
  - Provides normal text inside a maths environment.
  - Dimensions of the box automatically computed to fit the contents of the box.
- `\makebox[`*width*`] [`*alignment*`] {`*contents*`}`
  Like `\mbox`, but you can specify the width of the box, and how the text is justified within it: left, right or centred.

Basic Types of Boxes

- `\mbox{`*contents*`}`
  Simplest type of box.
  - Prevents text inside it from being broken across a line
  - Provides normal text inside a maths environment.
  - Dimensions of the box automatically computed to fit the contents of the box.
- `\makebox[`*width*`][`*alignment*`]{`*contents*`}`
  Like `\mbox`, but you can specify the width of the box, and how the text is justified within it: left, right or centred.

Basic Types of Boxes

- `\mbox`{*contents*}

  Simplest type of box.

  - Prevents text inside it from being broken across a line
  - Provides normal text inside a maths environment.
  - Dimensions of the box automatically computed to fit the contents of the box.

- `\makebox`[*width*] [*alignment*] {*contents*}

  Like `\mbox`, but you can specify the width of the box, and how the text is justified within it: left, right or centred.

Basic Types of Boxes

- `\mbox`{*contents*}
  Simplest type of box.
  - Prevents text inside it from being broken across a line
  - Provides normal text inside a maths environment.
  - Dimensions of the box automatically computed to fit the contents of the box.
- `\makebox`[*width*] [*alignment*] {*contents*}
  Like `\mbox`, but you can specify the width of the box, and how the text is justified within it: left, right or centred.

Basic Types of Boxes

- `\mbox`{*contents*}
  Simplest type of box.
  - Prevents text inside it from being broken across a line
  - Provides normal text inside a maths environment.
  - Dimensions of the box automatically computed to fit the contents of the box.
- `\makebox`[*width*] [*alignment*] {*contents*}
  Like `\mbox`, but you can specify the width of the box, and how the text is justified within it: left, right or centred.

Examples using \mbox

```
\begin{displaymath}
y = x \mbox{ and }
z = x + y
\end{displaymath}
```

INPUT

$$y = x \text{ and } z = x + y$$

OUTPUT

```
Now some in-line
$x = 1, \ldots, n$ maths.
```

INPUT

Now some in-line $x =$
$1, \ldots, n$ maths.

OUTPUT

```
Now some in-line
\mbox{$x = 1, \ldots, n$}
maths.
```

INPUT

Now some in-line
$x = 1, \ldots, n$ maths.

OUTPUT

Examples using `\makebox`

```
Here is \makebox[1in][r]{\em a 1in} box
```

Input

Here is *a 1in* box

Output

```
\makebox[0pt][l]{/////}Hello!
```

Input

/////Hello!

Input

Boxes with Frames

`\fbox` and `\framebox` : These are the same as `\mbox` and `\makebox`, but they put a rectangular frame around the box.

```
Here is a \fbox{box}
```

INPUT

Here is a $\boxed{\text{box}}$

OUTPUT

```
Here is \framebox[1in][r]{\em a 1in} box
```

INPUT

Here is $\boxed{\qquad\qquad\quad \textit{a 1in}}$ box

OUTPUT

Lengths Associated with `\fbox` and `\framebox`

- `\fboxsep` : This is the gap between the frame and the contents of the box. For example:

```
\setlength{\fboxsep}{10pt}
\fbox{Some Text}
```

Some Text

- `\fboxrule` : This is the width of the frame. For example:

```
\setlength{\fboxrule}{4pt}
\fbox{Some Text}
```

Some Text

Lengths Associated with `\fbox` and `\framebox`

- `\fboxsep` : This is the gap between the frame and the contents of the box. For example:

  ```
  \setlength{\fboxsep}{10pt}
  \fbox{Some Text}
  ```

  Some Text

- `\fboxrule` : This is the width of the frame. For example:

  ```
  \setlength{\fboxrule}{4pt}
  \fbox{Some Text}
  ```

  Some Text

Lengths Associated with `\fbox` and `\framebox`

- `\fboxsep` : This is the gap between the frame and the contents of the box. For example:

```
\setlength{\fboxsep}{10pt}
\fbox{Some Text}
```

Some Text

- `\fboxrule` : This is the width of the frame. For example:

```
\setlength{\fboxrule}{4pt}
\fbox{Some Text}
```

Some Text

Lengths Associated with `\fbox` and `\framebox`

- `\fboxsep` : This is the gap between the frame and the contents of the box. For example:

```
\setlength{\fboxsep}{10pt}
\fbox{Some Text}
```

Some Text

- `\fboxrule` : This is the width of the frame. For example:

```
\setlength{\fboxrule}{4pt}
\fbox{Some Text}
```

Some Text

fancybox package

The fancybox *package* provides four commands, analogous to `\fbox`:

Input	Output
`\ovalbox{An oval frame}`	An oval frame
`\Ovalbox{A thicker oval frame}`	A thicker oval frame
`\doublebox{A double frame}`	A double frame
`\shadowbox{A shadow frame}`	A shadow frame

fancybox package

The fancybox *package* provides four commands, analogous to `\fbox`:

Input	Output
`\ovalbox{An oval frame}`	An oval frame
`\Ovalbox{A thicker oval frame}`	A thicker oval frame
`\doublebox{A double frame}`	A double frame
`\shadowbox{A shadow frame}`	A shadow frame

fancybox package

The fancybox *package* provides four commands, analogous to `\fbox`:

Input	Output
`\ovalbox{An oval frame}`	An oval frame
`\Ovalbox{A thicker oval frame}`	A thicker oval frame
`\doublebox{A double frame}`	A double frame
`\shadowbox{A shadow frame}`	A shadow frame

fancybox package

The fancybox *package* provides four commands, analogous to `\fbox`:

Input	Output
`\ovalbox{An oval frame}`	An oval frame
`\Ovalbox{A thicker oval frame}`	A thicker oval frame
`\doublebox{A double frame}`	A double frame
`\shadowbox{A shadow frame}`	A shadow frame

fancybox package

The fancybox *package* provides four commands, analogous to `\fbox`:

Input	Output
`\ovalbox{An oval frame}`	An oval frame
`\Ovalbox{A thicker oval frame}`	A thicker oval frame
`\doublebox{A double frame}`	A double frame
`\shadowbox{A shadow frame}`	A shadow frame

Examples

```
Here is a \ovalbox{box}
```

INPUT

Here is a box

OUTPUT

```
Here is \ovalbox{\makebox[1in][r]{\em a 1in}}
box
```

INPUT

Here is *a 1in* box

OUTPUT

Examples

```
Here is a \Ovalbox{box}
```

INPUT

Here is a box

OUTPUT

```
Here is \Ovalbox{\makebox[1in][r]{\em a 1in}}
box
```

INPUT

Here is *a 1in* box

OUTPUT

Examples

```
Here is a \doublebox{box}
```

INPUT

Here is a box

OUTPUT

```
Here is \doublebox{\makebox[1in][r]{\em a 1in}}
box
```

INPUT

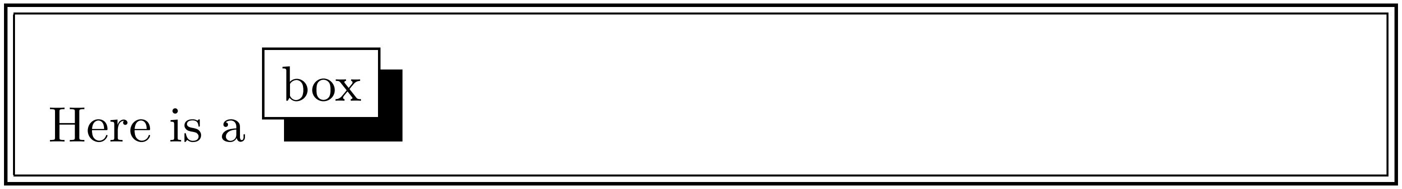
Here is *a 1in* box

OUTPUT

Examples

```
Here is a \shadowbox{box}
```

INPUT

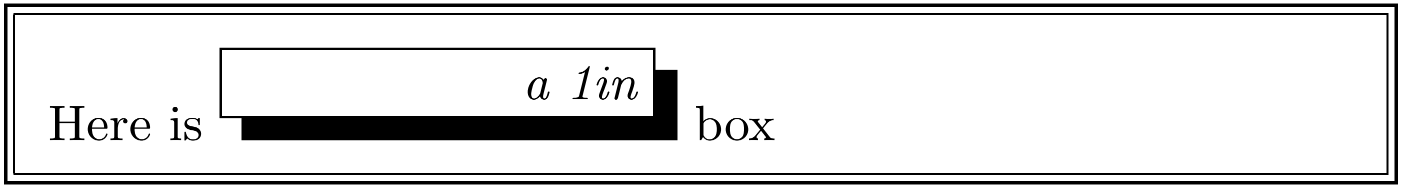


Here is a box

OUTPUT

```
Here is \shadowbox{\makebox[1in][r]{\em a 1in}}
box
```

INPUT



Here is *a 1in* box

OUTPUT

Typesetting a Paragraph Inside a Box

`\parbox`[*alignment*] [*height*] {*width*}{*contents of box*}

For example:

```
A paragraph within a box : \parbox{0.75in}{This box is
three quarters of an inch wide} so there!
```

INPUT

```
                           This box is
                           three quar-
A paragraph within a box : ters of an   so there!
                           inch wide
```

OUTPUT

The minipage Environment

\begin{minipage}[*alignment*][*height*]{*width*}

```
Some text. \begin{minipage}{0.4\textwidth} The width of this
minipage is 0.4 times the width of the text
body\footnote{Note we can also have a footnote}.
\end{minipage} Some more text.
```

INPUT

Some text. The width of this minipage is 0.4 times the width of the text body[a]. Some more text.

[a] Note we can also have a footnote

OUTPUT

The shapepar Package (`\diamondpar`)

♢

With the shapepar package, you can create some rather fancy effects. There are four predefined paragraph shapes: diamond, square, heart and nut shaped. It is possible to define other shapes using `\shapepar`. The argument must be a whole paragraph.

♢

```
\diamondpar{With the ... }
```

The shapepar Package (\squarepar)

With the shapepar package, you can create some rather fancy effects. There are four predefined paragraph shapes: diamond, square, heart and nut shaped. It is possible to define other shapes using `\shapepar`. The argument must be a whole paragraph.

```
\squarepar{With the ... }
```

The shapepar Package (`\heartpar`)

With the shapepar package, you can create some rather fancy effects. There are four predefined paragraph shapes: diamond, square, heart and nut shaped. It is possible to define other shapes using `\shapepar`. The argument must be a whole paragraph.

♡

```
\heartpar{With the ... }
```

The shapepar Package (\heartpar)

With the shapepar package, you can create some rather fancy effects. There are four predefined paragraph shapes: diamond, square, heart and nut shaped. It is possible to define other shapes using \shapepar. The argument must be a whole paragraph.

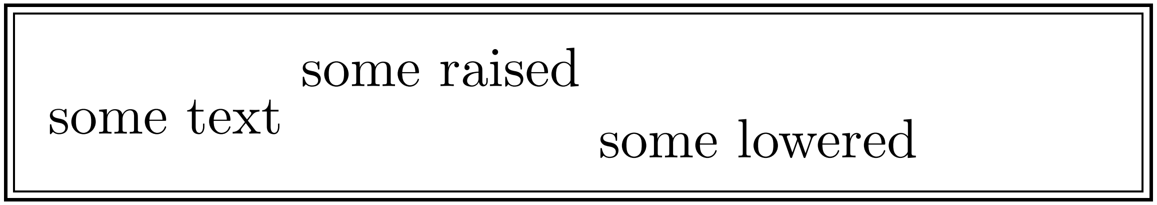
```
\shapepar\nutshape{With the ... }
```

Raising and Lowering Boxes

- Boxes can be raised or lowered
- Syntax: `\raisebox`{*lift*}[*depth*][*height*]{*contents*}

```
some text \raisebox{2ex}{some raised}
\raisebox{-1ex}{some lowered}
```

INPUT



some text some raised some lowered

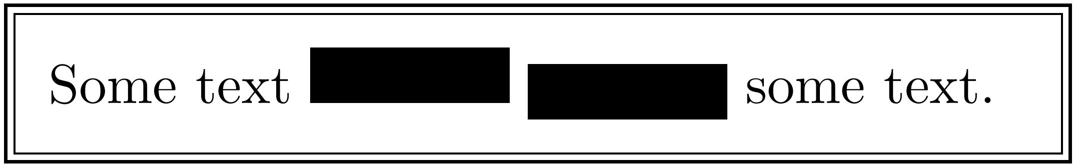
OUTPUT

Rules

- A rule is a rectangular blob of ink
- Syntax: `\rule[`*lift*`]{`*width*`}{`*height*`}`

```
Some text
\rule{0.5in}{10pt}
\rule[-3pt]{0.5in}{10pt}
some text.
```

INPUT



OUTPUT

Example

This example uses \settowidth, \makebox and \rule:

```
\newlength\mylen
\settowidth{\mylen}{Some Text}%
\makebox[0pt][l]{\rule[0.5ex]{\mylen}{1pt}}%
Some Text
```

INPUT

~~Some Text~~

OUTPUT

That's a bit fiddly. Time to define a new command!

Example

This example uses \settowidth, \makebox and \rule:

```
\newlength\mylen
\settowidth{\mylen}{Some Text}%
\makebox[0pt][l]{\rule[0.5ex]{\mylen}{1pt}}%
Some Text
```

INPUT

~~Some Text~~

OUTPUT

That's a bit fiddly. Time to define a new command!

Example

```
\newlength\mylen
\newcommand{\strikethrough}[1]{%
\settowidth{\mylen}{#1}%
\makebox[0pt][l]{\rule[0.5ex]{\mylen}{1pt}}%
#1}

\strikethrough{Some More Text}
```

INPUT

~~Some More Text~~

OUTPUT

This example won't allow for line breaks. Better to use ulem package.

Example

```
\newlength\mylen
\newcommand{\strikethrough}[1]{%
\settowidth{\mylen}{#1}%
\makebox[0pt][l]{\rule[0.5ex]{\mylen}{1pt}}%
#1}

\strikethrough{Some More Text}
```

INPUT

~~Some More Text~~

OUTPUT

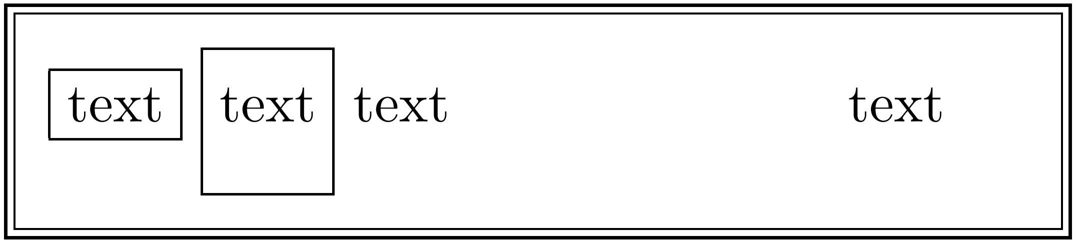
This example won't allow for line breaks. Better to use ulem package.

Struts

A zero width rule is called a *strut*:

```
\fbox{text}
\fbox{\rule[-10pt]{0pt}{20pt}text}
text\rule{1in}{0pt}text
```

INPUT



OUTPUT

Saveboxes

- A savebox allows you to save some typeset text for later use.
- Define new savebox: \newsavebox{*cmd*}.
- To save text to a savebox:
  - command:

    \sbox{*cmd*}{*text*}
  - lrbox environment:

    \begin{lrbox}{*cmd*}
    text
    \end{lrbox}
- To display the contents of a savebox: \usebox{*cmd*}

Saveboxes

- A savebox allows you to save some typeset text for later use.
- Define new savebox: \newsavebox{*cmd*}.
- To save text to a savebox:
  - command:
    \sbox{*cmd*}{*text*}
  - lrbox environment:
    ```
    \begin{lrbox}{cmd}
    text
    \end{lrbox}
    ```
- To display the contents of a savebox: \usebox{*cmd*}

Saveboxes

- A savebox allows you to save some typeset text for later use.
- Define new savebox: `\newsavebox{`*cmd*`}`.
- To save text to a savebox:
  - command:
    `\sbox{`*cmd*`}{`*text*`}`
  - lrbox environment:
    `\begin{lrbox}{`*cmd*`}`
    text
    `\end{lrbox}`
- To display the contents of a savebox: `\usebox{`*cmd*`}`

Saveboxes

- A savebox allows you to save some typeset text for later use.
- Define new savebox: \newsavebox{*cmd*}.
- To save text to a savebox:
  - command:

    \sbox{*cmd*}{*text*}
  - lrbox environment:

    \begin{lrbox}{*cmd*}
    text
    \end{lrbox}
- To display the contents of a savebox: \usebox{*cmd*}

Saveboxes

- A savebox allows you to save some typeset text for later use.
- Define new savebox: `\newsavebox{`*cmd*`}`.
- To save text to a savebox:
  - command:

    `\sbox{`*cmd*`}{`*text*`}`
  - lrbox environment:

    `\begin{lrbox}{`*cmd*`}`

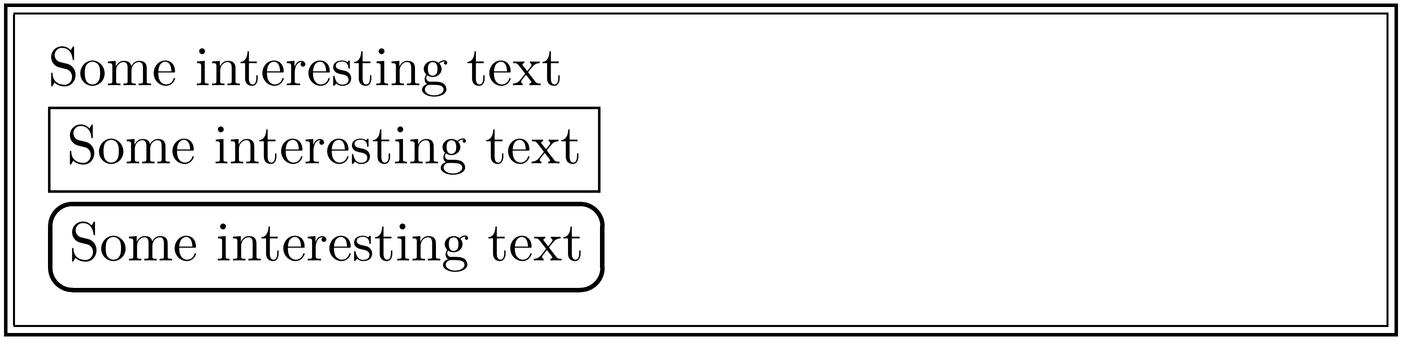
    text

    `\end{lrbox}`
- To display the contents of a savebox: `\usebox{`*cmd*`}`

Example Using \sbox

```
\newsavebox{\mysbox}
\sbox{\mysbox}{Some interesting text}

\usebox{\mysbox}\\
\fbox{\usebox{\mysbox}}\\
\Ovalbox{\usebox{\mysbox}}
```

Input



```
Some interesting text
Some interesting text
Some interesting text
```

Output

Example Using lrbox

```
\newsavebox{\mysbox}
\begin{lrbox}{\mysbox}
Some more interesting text.
\end{lrbox}

\usebox{\mysbox}\\
\fbox{\usebox{\mysbox}}
```

INPUT

Some more interesting text.

$\boxed{\text{Some more interesting text.}}$

OUTPUT

Macros verses Saveboxes

Using a macro (or *command*):

```
\newcommand{\sometext}{Some text}

\sometext.\\
\sffamily \sometext.\\
\ttfamily \sometext.
```

INPUT

```
Some text.
Some text.
Some text.
```

OUTPUT

TEX has to work out how to typeset “Some text” three times.

Macros verses Saveboxes

Using a savebox:

```
\newsavebox{\mysbox}
\sbox{\mysbox}{Some text}

\usebox{\mysbox}.\\
\sffamily \usebox{\mysbox}.\\
\ttfamily \usebox{\mysbox}.
```

INPUT

```
Some text.
Some text.
Some text.
```

OUTPUT

TEX only has to work out how to typeset “Some text” once.

Exercise 17 (Page 19)

ⓔ Try reproducing the output shown in Figure 17 on Page 19 of the handout.

⚠ Try reproducing the output shown in Figure 18 on Page 19 of the handout.

- Try changing the vertical alignment of the minipage.
- Try experimenting with footnotes insides and outside of the minipage environment.
- Try using a `\parbox` instead of a minipage.
- Try experimenting with different frames around the minipage.

Incorporating Images

- It is possible to generate images using LaTeX picture drawing commands. The pstricks or pgf set of packages can create very complex graphics, however most people find this too complicated.

- A more convenient approach is to create the image using a graphics application, and import it into your LaTeX document using the command:

  ```
  \includegraphics[options]{filename}
  ```

  which is defined in the graphicx *package*.

- Acceptable file types depend on the driver you are using. Commonly accepted file types are PS or EPS (with dvips) and PDF or PNG (with PDFLaTeX).

Incorporating Images

- It is possible to generate images using LATEX picture drawing commands. The pstricks or pgf set of packages can create very complex graphics, however most people find this too complicated.
- A more convenient approach is to create the image using a graphics application, and import it into your LATEX document using the command:

  ```
  \includegraphics[options]{filename}
  ```

  which is defined in the graphicx *package*.
- Acceptable file types depend on the driver you are using. Commonly accepted file types are PS or EPS (with dvips) and PDF or PNG (with PDFLATEX).

Incorporating Images

- It is possible to generate images using LaTeX picture drawing commands. The pstricks or pgf set of packages can create very complex graphics, however most people find this too complicated.
- A more convenient approach is to create the image using a graphics application, and import it into your LaTeX document using the command:

  ```
  \includegraphics[options]{filename}
  ```

  which is defined in the graphicx *package*.
- Acceptable file types depend on the driver you are using. Commonly accepted file types are PS or EPS (with dvips) and PDF or PNG (with PDFLaTeX).

Syntax

\includegraphics[*options*]{*filename*}

Some of the more common options are:

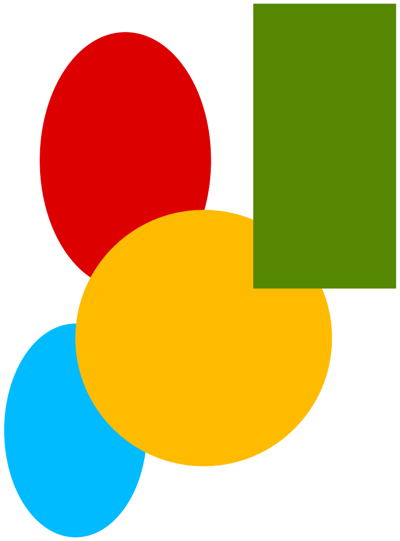
angle=*x*	rotate the picture by x°
width=*len*	scale the picture so that the width is *len*.
height=*len*	scale the picture so that the height is *len*.
scale=*x*	scale the picture.
trim=*lx by rx ty*	trim the picture so that the bottom left co-ordinate is (*lx*, *by*) and the top right co-ordinate is (*rx*, *ty*).
draft	don't display the image, just draw the bounding box with the filename inside.

An Example

- You don't need to specify the extension.

```
\includegraphics[angle=90,width=1in]{shapes}
```

INPUT



OUTPUT

- If LaTeX it uses `shapes.ps`, if PDFLaTeX it uses `shapes.pdf`.

Other graphicx Commands

- `\rotatebox`{*angle*}{*text*}

  Rotate *text* by *angle*.

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

  Rescales *text*.

- `\reflectbox`{*text*}

  Reflect *text*

- `\resizebox`{*h length*}{*v length*}{*text*}

  Resizes *text* so that it has width is *h length* and height *v length*. An exclamation mark `!` can be used to maintain the aspect ratio if only one length is specified.

Examples

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

INPUT

Some text

OUTPUT

```
\scalebox{1.5}{Some text}
```

INPUT

Some text

OUTPUT

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

INPUT

Some text

OUTPUT

Examples

```
\resizebox{6mm}{1cm}{Some
text}
```

Input

Some text

Output

```
\resizebox{6mm}{!}{Some
text}
```

Input

Some text

Output

Exercise 18 (Page 21)

- Copy the file shapes.ps (or shapes.pdf if you want to use PDFLaTeX) to your directory:

  ▶`cp /home/sys/gcc/insecure/shapes.ps .`

  and include it in a document.

- Try to centre the image, using `\centerline`.

- Try putting a frame around it.

- Try scaling and rotating it.

- Try passing the option `draft` to the graphicx package and see what happens.

Figures and Tables

- Figures and Tables are *floats* — they are *floated* to the nearest convenient location according to certain typographical rules.
- A figure or table has a caption and an associated number. Captions are produced using the command: `\caption[`*short caption*`]{`*caption text*`}`
- LaTeX handles numbering automatically. Floats can be cross-referenced using `\label` and `\ref`.
- Figures are created using the figure environment.
- Tables are created using the table environment.
- figure and table environments can not have a page break in them.

Figures and Tables

- Figures and Tables are *floats* — they are *floated* to the nearest convenient location according to certain typographical rules.
- A figure or table has a caption and an associated number. Captions are produced using the command: `\caption[`*short caption*`]{`*caption text*`}`
- LaTeX handles numbering automatically. Floats can be cross-referenced using `\label` and `\ref`.
- Figures are created using the figure environment.
- Tables are created using the table environment.
- figure and table environments can not have a page break in them.

Figures and Tables

- Figures and Tables are *floats* — they are *floated* to the nearest convenient location according to certain typographical rules.
- A figure or table has a caption and an associated number. Captions are produced using the command:
  `\caption[`*short caption*`]{`*caption text*`}`
- LaTeX handles numbering automatically. Floats can be cross-referenced using `\label` and `\ref`.
- Figures are created using the figure environment.
- Tables are created using the table environment.
- figure and table environments can not have a page break in them.

Figures and Tables

- Figures and Tables are *floats* — they are *floated* to the nearest convenient location according to certain typographical rules.
- A figure or table has a caption and an associated number. Captions are produced using the command: `\caption[`*short caption*`]{`*caption text*`}`
- LaTeX handles numbering automatically. Floats can be cross-referenced using `\label` and `\ref`.
- Figures are created using the figure environment.
- Tables are created using the table environment.
- figure and table environments can not have a page break in them.

Figures and Tables

- Figures and Tables are *floats* — they are *floated* to the nearest convenient location according to certain typographical rules.
- A figure or table has a caption and an associated number. Captions are produced using the command:
  `\caption[`*short caption*`]{`*caption text*`}`
- LaTeX handles numbering automatically. Floats can be cross-referenced using `\label` and `\ref`.
- Figures are created using the figure environment.
- Tables are created using the table environment.
- figure and table environments can not have a page break in them.

Figures and Tables

- Figures and Tables are *floats* — they are *floated* to the nearest convenient location according to certain typographical rules.
- A figure or table has a caption and an associated number. Captions are produced using the command: `\caption[`*short caption*`]{`*caption text*`}`
- LaTeX handles numbering automatically. Floats can be cross-referenced using `\label` and `\ref`.
- Figures are created using the figure environment.
- Tables are created using the table environment.
- figure and table environments can not have a page break in them.

An Example Figure

```
\begin{figure}[tbh]
\centerline{\includegraphics[height=1.25cm]{shapes.ps}}
\caption{Some shapes}
\label{fig:shapes}
\end{figure}
```

Input

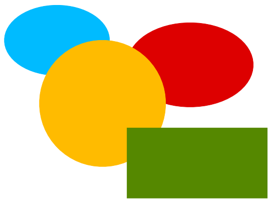


Figure 1: Some shapes

Output

An Example Table

```
\begin{table}[tbh]
\caption{An example table}
\label{tab:example}
\vspace{10pt}
\centerline{
\begin{tabular}{l|ll}
 & A & B\\\hline
I & 0.5 & 1.0\\
II & 12 & 14
\end{tabular}
}
\end{table}
```

Input

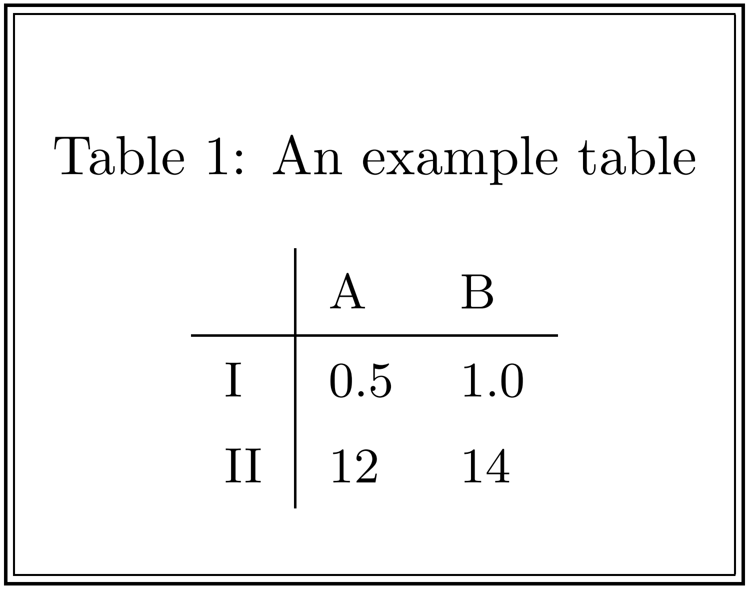


Table 1: An example table

	A	B
I	0.5	1.0
II	12	14

Output

Adjacent Figures

Two figures can be placed side by side in one figure environment:

```
\begin{figure}[tbh]
\begin{minipage}{0.4\textwidth}
\centerline{\includegraphics{circle.ps}}
\caption{A Circle}\label{fig:circ}
\end{minipage}
\begin{minipage}{0.5\textwidth}
\centerline{\includegraphics{rectangle.ps}}
\caption{A Rectangle}\label{fig:rect}
\end{minipage}
\end{figure}


Figure~\ref{fig:circ} shows a circle.
Figure~\ref{fig:rect} shows a rectangle.
```

INPUT

Adjacent Figures

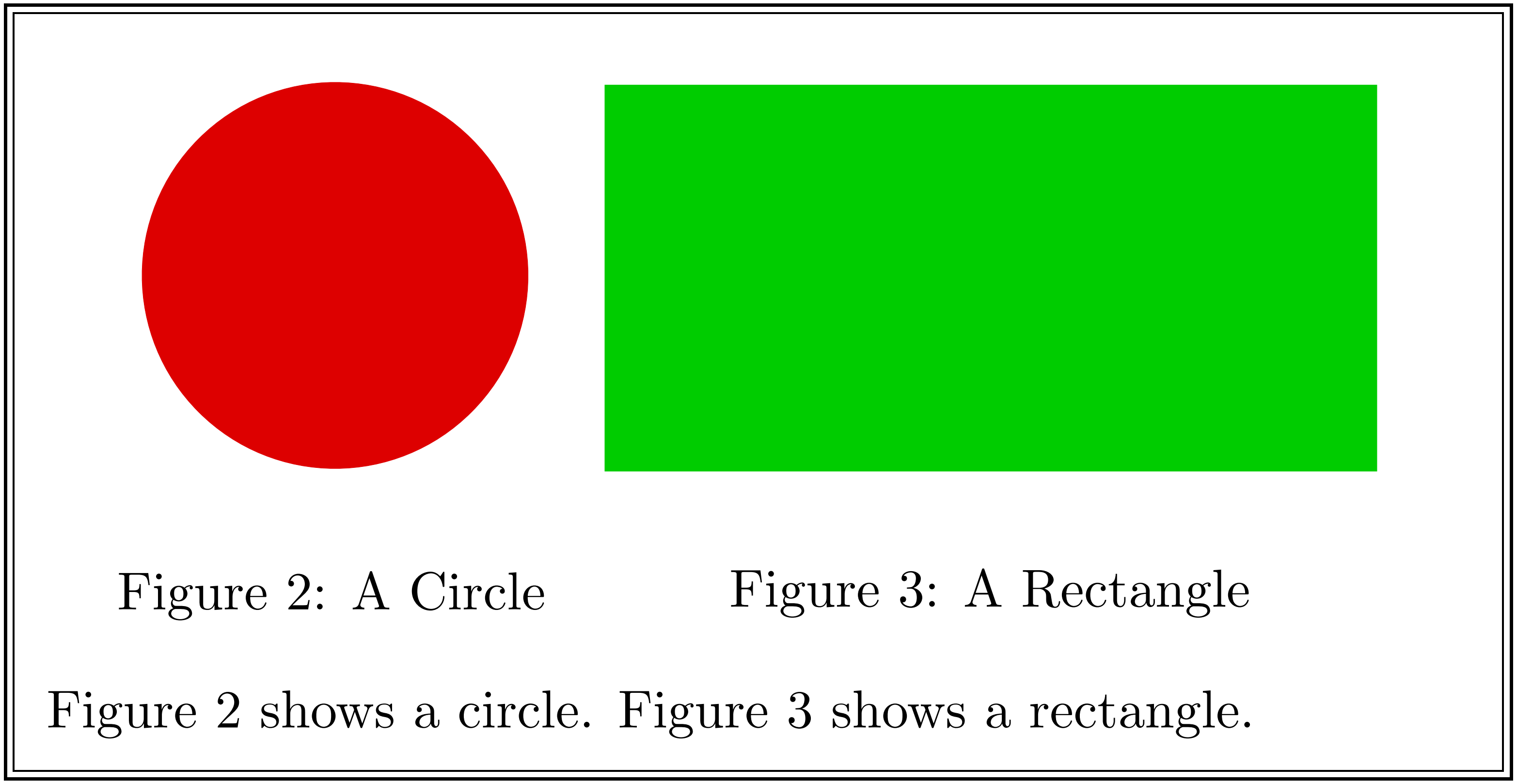


Figure 2: A Circle

Figure 3: A Rectangle

Figure 2 shows a circle. Figure 3 shows a rectangle.

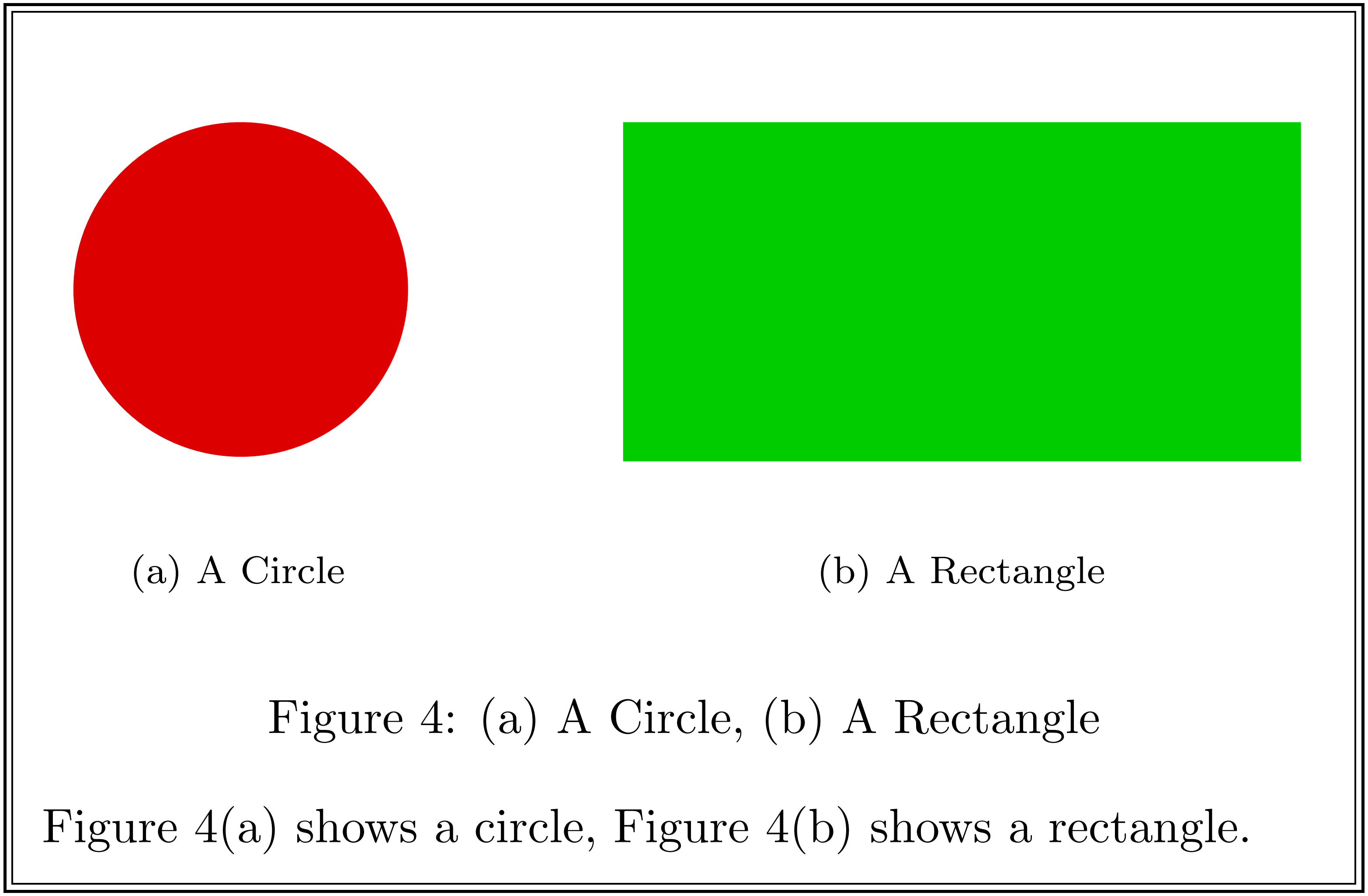
OUTPUT

Subfigures

Subfigures can be created using the command `\subfigure[`*caption*`]{`*contents*`}` which is defined in the package subfigure.

```
\begin{figure}[tbh]
\centering \subfigure[A Circle]{\label{fig:circle}%
\includegraphics[height=1in,clip]{circle.ps}}
\hspace{0.5in}
\subfigure[A Rectangle]{\label{fig:rectangle}%
\includegraphics[height=1in,clip]{rectangle.ps}}

\caption{(a) A Circle, (b) A Rectangle}
\label{fig:subfigex}
\end{figure}
Figure~\ref{fig:circle} shows a circle,
Figure~\ref{fig:rectangle} shows a rectangle.
```

Input



(a) A Circle

(b) A Rectangle

Figure 4: (a) A Circle, (b) A Rectangle

Figure 4(a) shows a circle, Figure 4(b) shows a rectangle.

OUTPUT

List of Figures/Tables

- A list of figures can be produced using the command:
  `\listoffigures`
- A list of tables can be produced using the command:
  `\listoftables`
- These commands should be placed at the start of the document, after the table of contents.
- The document should be LaTeXed twice to ensure that the list of figures and list of tables are up-to-date.

List of Figures/Tables

- A list of figures can be produced using the command: `\listoffigures`
- A list of tables can be produced using the command: `\listoftables`
- These commands should be placed at the start of the document, after the table of contents.
- The document should be LaTeXed twice to ensure that the list of figures and list of tables are up-to-date.

List of Figures/Tables

- A list of figures can be produced using the command: `\listoffigures`
- A list of tables can be produced using the command: `\listoftables`
- These commands should be placed at the start of the document, after the table of contents.
- The document should be LaTeXed twice to ensure that the list of figures and list of tables are up-to-date.

List of Figures/Tables

- A list of figures can be produced using the command: `\listoffigures`
- A list of tables can be produced using the command: `\listoftables`
- These commands should be placed at the start of the document, after the table of contents.
- The document should be LaTeXed twice to ensure that the list of figures and list of tables are up-to-date.

Exercise 19 (Page 21)

- Copy the files `circle.ps`, `rectangle.ps` and `shapes.ps` to your directory (or `circle.pdf`, `rectangle.pdf` and `shapes.pdf`).

  ```
  ▶cp /home/sys/gcc/insecure/circle.ps .
  ▶cp /home/sys/gcc/insecure/rectangle.ps .
  ```

- Make a document that contains Figures 19 and 20 and Table 11 in the handout.

- Add a list of figures and list of tables at the start of the document.

Creating Slides using LaTeX

- There are a number of class files available to produce slides.
- The simplest is slides
- There are far more advanced packages, such as beamer and prosper, which produce very professional looking presentations.
- We will be looking at the seminar class file.

The seminar Package

- Each slide is contained in a slide (landscape) or slide* (portrait) environment.
- To change the page layout to portrait, use the option `portrait`: `\documentclass[portrait]{seminar}`
- To use A4 paper, instead of the default US letter, use the sem-a4 package.
- To display only the landscape or only portrait slides, use the command `\landscapeonly` or `\portraitonly` in the *preamble*.

Title Slides

As with the other class files we have looked at, we can use the `\title`, `\author`, `\date` and `\maketitle` commands.

```
\title{\LARGE Introductory \LaTeX}
\author{Dr N.L.C. Talbot\\
\mdseries\slshape for\\
\mdseries\slshape UEA Centre for Staff and Educational
Development}
\date{}

\begin{slide}
\maketitle
\end{slide}
```

INPUT

Notes

- Any text that appears outside a slide or slide* environment will be treated as a note.
- You can specify only slides or only notes (e.g. if you want to print the slides on transparencies and the notes on paper):
  - `\documentclass[slidesonly]{seminar}`
  - `\documentclass[notesonly]{seminar}`
- A set of slides, and their corresponding notes can be turned into an article (for handouts, say) by using the `article` option:

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

Notes

- Any text that appears outside a slide or slide* environment will be treated as a note.
- You can specify only slides or only notes (e.g. if you want to print the slides on transparencies and the notes on paper):
  - `\documentclass[slidesonly]{seminar}`
  - `\documentclass[notesonly]{seminar}`
- A set of slides, and their corresponding notes can be turned into an article (for handouts, say) by using the `article` option:

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

Notes

- Any text that appears outside a slide or slide* environment will be treated as a note.
- You can specify only slides or only notes (e.g. if you want to print the slides on transparencies and the notes on paper):
  - `\documentclass[slidesonly]{seminar}`
  - `\documentclass[notesonly]{seminar}`
- A set of slides, and their corresponding notes can be turned into an article (for handouts, say) by using the `article` option:

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

Slide Frames

- Slide frames can be changed using the `\slideframe{`*style*`}` command. There are two standard styles:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional styles:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{none}`

Slide Frames

- Slide frames can be changed using the `\slideframe{`*style*`}` command. There are two standard styles:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional styles:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{none}`

Slide Frames

- Slide frames can be changed using the `\slideframe{`*style*`}` command. There are two standard styles:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional styles:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{none}`

Slide Frames

- Slide frames can be changed using the `\slideframe`{*style*} command. There are two standard style:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional frames:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{plain}`

Slide Frames

- Slide frames can be changed using the `\slideframe{`*style*`}` command. There are two standard style:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional frames:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{Oval}`

Slide Frames

- Slide frames can be changed using the `\slideframe`{*style*} command. There are two standard style:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional frames:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{shadow}`

Slide Frames

- Slide frames can be changed using the `\slideframe`{*style*} command. There are two standard style:
  - `none` (no frame)
  - `plain` (plain rectangle).
- The fancybox package defines the additional frames:
  - `shadow`
  - `double`
  - `oval`
  - `Oval`.
- Example: `\slideframe{double}`

Defining New Page Styles

- The seminar class file allows new page styles to be defined using the command:

  \newpagestyle{*name*}{*header*}{*footer*}

- Example:

```
\newpagestyle{csedlatex}{}{%
\textsc{Introductory \LaTeX}\hfill\thepage}

\pagestyle{csedlatex}
```

Exercise 20 (Page 22)

- Try to produce some of the slides used during this course.
- Try experimenting with different slide frames, and different page styles.
- Try including some notes about the slides.
- Try using the `article` option.
- With the `article` option, the slide caption is displayed according to the style specified by the command `\slidestyle{`*style*`}`. Available options are: `empty`, `left`, `bottom`. Try experimenting with the slide style.

Defining New Environments

New environments can be defined using:

\newenvironment{*env-name*}[*n*][*default*]{*begin-code*}{*end-code*}

```
\newenvironment{bfitemize}%
{\begin{bfseries}\begin{itemize}}%
{\end{itemize}\end{bfseries}}

\begin{bfitemize}
\item First item
\item Second item
\end{bfitemize}
```

INPUT

- **First item**
- **Second item**

OUTPUT

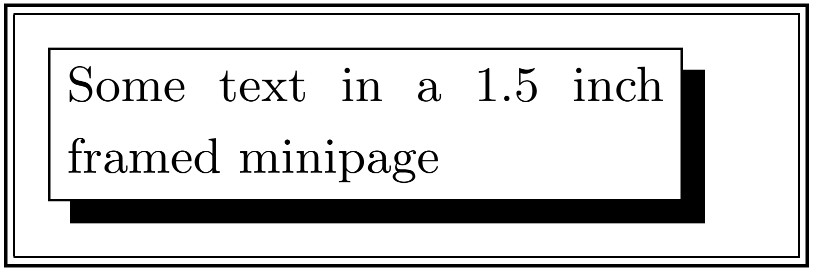
Example Environment with Arguments

```
\newsavebox{\fminibox}

\newenvironment{fminipage}[2][c]%
{\begin{lrbox}{\fminibox}\begin{minipage}[#1]{#2}}%
{\end{minipage}\end{lrbox}%
\shadowbox{\usebox{\fminibox}}}
```

Input

```
\begin{fminipage}{1.5in}
Some text in a 1.5
inch framed minipage
\end{fminipage}
```

Input



Some text in a 1.5 inch framed minipage

Output

Exercise 21 (Page 23)

- Create an environment called exercise that draws a horizontal line at the start and at the end of the environment. So, for example, the following code:

  ```
  \begin{exercise}
  Some text.
  \end{exercise}
  ```

  would produce the following output:

  Some text.

- Note that if the environment is preceded by a paragraph break, there will be a small space at the start of the horizontal line caused by paragraph indentation, this can be suppressed using `\noindent`.

Counters

- Counters contain integers that can be incremented or decremented.
- We have already used commands that have associated counters: \chapter, \section, \footnote, \caption.
- We have also used environments that use counters: equation, enumerate.
- To define a new counter:

  \newcounter{*ctr-name*}[*outer-ctr*]

  For example: \newcounter{exercise}.
- To reset the counter every time another counter is incremented: \newcounter{exercise}[chapter]

Counters

- Counters contain integers that can be incremented or decremented.
- We have already used commands that have associated counters: `\chapter`, `\section`, `\footnote`, `\caption`.
- We have also used environments that use counters: equation, enumerate.
- To define a new counter:

  `\newcounter{`*ctr-name*`}[`*outer-ctr*`]`

  For example: `\newcounter{exercise}`.
- To reset the counter every time another counter is incremented: `\newcounter{exercise}[chapter]`

Counters

- Counters contain integers that can be incremented or decremented.
- We have already used commands that have associated counters: `\chapter`, `\section`, `\footnote`, `\caption`.
- We have also used environments that use counters: equation, enumerate.
- To define a new counter:

  `\newcounter{`*ctr-name*`}[`*outer-ctr*`]`

  For example: `\newcounter{exercise}`.
- To reset the counter every time another counter is incremented: `\newcounter{exercise}[chapter]`

Counters

- Counters contain integers that can be incremented or decremented.
- We have already used commands that have associated counters: `\chapter`, `\section`, `\footnote`, `\caption`.
- We have also used environments that use counters: equation, enumerate.
- To define a new counter:

  `\newcounter{`*ctr-name*`}[`*outer-ctr*`]`

  For example: `\newcounter{exercise}`.
- To reset the counter every time another counter is incremented: `\newcounter{exercise}[chapter]`

Counters

- Counters contain integers that can be incremented or decremented.
- We have already used commands that have associated counters: `\chapter`, `\section`, `\footnote`, `\caption`.
- We have also used environments that use counters: equation, enumerate.
- To define a new counter:

  `\newcounter{`*ctr-name*`}[`*outer-ctr*`]`

  For example: `\newcounter{exercise}`.
- To reset the counter every time another counter is incremented: `\newcounter{exercise}[chapter]`

Changing the Value of a Counter

- `\stepcounter{`*ctr*`}` increment the counter by 1.
- `\refstepcounter{`*ctr*`}` as above, but allows you to reference the counter using `\ref` and `\label`.
- `\setcounter{`*ctr*`}{`*value*`}` set the counter to *value*
- `\addtocounter{`*ctr*`}{`*value*`}` add *value* to the counter
- `\value{`*ctr*`}` This command produces the value for use in the *value* part of `\setcounter` and `\addtocounter` commands.

Displaying the Value of a Counter

- The command `\the`*ctr* prints a representation of the value associated with *ctr*. Examples:
  - `\thepage` displays the value of the page counter.
  - `\thesection` displays the value of the `section` counter.
  - `\theslide` displays the value of the `slide` counter.

```
This is slide number
\theslide
```

INPUT

This is slide number 216

OUTPUT

- By default `\the`*ctr* will display the counter as an Arabic number.
- `\the`*ctr* can be defined using `\renewcommand` so that it uses a different format.

Displaying the Value of a Counter

- The command \the*ctr* prints a representation of the value associated with *ctr*. Examples:
  - \thepage displays the value of the page counter.
  - \thesection displays the value of the section counter.
  - \theslide displays the value of the slide counter.

```
This is slide number
\theslide
```

INPUT

This is slide number 216

OUTPUT

- By default \the*ctr* will display the counter as an Arabic number.
- \the*ctr* can be defined using \renewcommand so that it uses a different format.

Displaying the Value of a Counter

- The command `\the`*ctr* prints a representation of the value associated with *ctr*. Examples:
  - `\thepage` displays the value of the `page` counter.
  - `\thesection` displays the value of the `section` counter.
  - `\theslide` displays the value of the `slide` counter.

```
This is slide number
\theslide
```

Input

This is slide number 216

Output

- By default `\the`*ctr* will display the counter as an Arabic number.
- `\the`*ctr* can be defined using `\renewcommand` so that it uses a different format.

Displaying the Value of a Counter

- The command `\the`*ctr* prints a representation of the value associated with *ctr*. Examples:
  - `\thepage` displays the value of the `page` counter.
  - `\thesection` displays the value of the `section` counter.
  - `\theslide` displays the value of the `slide` counter.

```
This is slide number
\theslide
```

INPUT

This is slide number 216

OUTPUT

- By default `\the`*ctr* will display the counter as an Arabic number.
- `\the`*ctr* can be defined using `\renewcommand` so that it uses a different format.

Displaying the Value of a Counter

- The command `\the`*ctr* prints a representation of the value associated with *ctr*. Examples:
  - `\thepage` displays the value of the `page` counter.
  - `\thesection` displays the value of the `section` counter.
  - `\theslide` displays the value of the `slide` counter.

```
This is slide number
\theslide
```

Input

This is slide number 216

Output

- By default `\the`*ctr* will display the counter as an Arabic number.
- `\the`*ctr* can be defined using `\renewcommand` so that it uses a different format.

Displaying the Value of a Counter

- The command `\the`*ctr* prints a representation of the value associated with *ctr*. Examples:
  - `\thepage` displays the value of the `page` counter.
  - `\thesection` displays the value of the `section` counter.
  - `\theslide` displays the value of the `slide` counter.

```
This is slide number
\theslide
```

INPUT

This is slide number 216

OUTPUT

- By default `\the`*ctr* will display the counter as an Arabic number.
- `\the`*ctr* can be defined using `\renewcommand` so that it uses a different format.

Standard Counter Formats

- `\arabic{`*ctr*`}` print *ctr* as an arabic numeral
- `\roman{`*ctr*`}` print *ctr* as a lowercase roman numeral
- `\Roman{`*ctr*`}` print *ctr* as an uppercase Roman numeral
- `\alph{`*ctr*`}` print *ctr* as a lowercase letter (value of counter must be less than 26)
- `\Alph{`*ctr*`}` print *ctr* as an uppercase letter (value of counter must be less than 26)
- `\fnsymbol{`*ctr*`}` print *ctr* as a footnote symbol. (This command may only be used in maths mode)

These commands should only go in the definition of `\the`*ctr*

Examples

- `\thechaper` displays the value of the `chapter` counter.
- `\renewcommand{\thechapter}{\Roman{chapter}}`
  redefines `\thechapter` so that it displays the chapter number as an uppercase Roman numeral.
- `\renewcommand{\thefootnote}{\alph{footnote}}`
  will display the footnote counter as a lowercase letter.
- `\newcounter{lemma}[section]`
  defines a new counter called lemma that will be reset at the start of each section.
- `\renewcommand{\thelemma}{\thesection.\arabic{lemma}}`
  If the section number is 4 and the lemma number is 3, `\thelemma` will display 4.3

Examples

- `\thechaper` displays the value of the `chapter` counter.
- `\renewcommand{\thechapter}{\Roman{chapter}}`
  redefines `\thechapter` so that it displays the chapter number as an uppercase Roman numeral.
- `\renewcommand{\thefootnote}{\alph{footnote}}`
  will display the footnote counter as a lowercase letter.
- `\newcounter{lemma}[section]`
  defines a new counter called lemma that will be reset at the start of each section.
- `\renewcommand{\thelemma}{\thesection.\arabic{lemma}}`
  If the section number is 4 and the lemma number is 3, `\thelemma` will display 4.3

Examples

- `\thechaper` displays the value of the `chapter` counter.
- `\renewcommand{\thechapter}{\Roman{chapter}}` redefines `\thechapter` so that it displays the chapter number as an uppercase Roman numeral.
- `\renewcommand{\thefootnote}{\alph{footnote}}` will display the footnote counter as a lowercase letter.
- `\newcounter{lemma}[section]` defines a new counter called lemma that will be reset at the start of each section.
- `\renewcommand{\thelemma}{\thesection.\arabic{lemma}}` If the section number is 4 and the lemma number is 3, `\thelemma` will display 4.3

Examples

- `\thechaper` displays the value of the `chapter` counter.
- `\renewcommand{\thechapter}{\Roman{chapter}}`
  redefines `\thechapter` so that it displays the chapter number as an uppercase Roman numeral.
- `\renewcommand{\thefootnote}{\alph{footnote}}`
  will display the footnote counter as a lowercase letter.
- `\newcounter{lemma}[section]`
  defines a new counter called lemma that will be reset at the start of each section.
- `\renewcommand{\thelemma}{\thesection.\arabic{lemma}}`
  If the section number is 4 and the lemma number is 3, `\thelemma` will display 4.3

Examples

- `\thechaper` displays the value of the `chapter` counter.
- `\renewcommand{\thechapter}{\Roman{chapter}}`
  redefines `\thechapter` so that it displays the chapter number as an uppercase Roman numeral.
- `\renewcommand{\thefootnote}{\alph{footnote}}`
  will display the footnote counter as a lowercase letter.
- `\newcounter{lemma}[section]`
  defines a new counter called lemma that will be reset at the start of each section.
- `\renewcommand{\thelemma}{\thesection.\arabic{lemma}}`
  If the section number is 4 and the lemma number is 3, `\thelemma` will display 4.3

Additional Counter Formats (datetime Package)

The datetime package also provides the following commands for displaying the value of a counter:

`\ordinal{`*counter*`}`	Display the value of *counter* as an ordinal
`\ordinalstring{`*counter*`}`	Display the value of *counter* as an ordinal written out in full
`\Ordinalstring{`*counter*`}`	As above, but with the initial letters in uppercase
`\numberstring{`*counter*`}`	Display the value of *counter* as a string
`\Numberstring{`*counter*`}`	As above but with the initial letter in uppercase

Examples

```
\ordinal{slide}
```

INPUT

220th

OUTPUT

```
\ordinalstring{slide}
```

INPUT

two hundred and twentieth

OUTPUT

```
\Ordinalstring{slide}
```

INPUT

Two Hundred and Twentieth

OUTPUT

```
\numberstring{slide}
```

INPUT

two hundred and twenty

OUTPUT

```
\Numberstring{slide}
```

INPUT

Two Hundred and Twenty

OUTPUT

Additional Counter Formats (datetime Package v2.4)

Version 2.4 of the datetime package also provides the following commands for displaying the value of a counter:

`\binary{`*counter*`}` Display the value of *counter* as a binary number

`\octal{`*counter*`}` Display the value of *counter* as an octal number

`\hexadecimal{`*counter*`}` Display the value of *counter* as a hexadecimal number

`\aaalph{`*counter*`}` Display the value of *counter* in the form: a . . . z aa . . . zz aaa . . .

`\abalph{`*counter*`}` Display the value of *counter* in the form: a . . . z aa ab . . . az ba bb . . .

Examples

```
\binary{slide}
```

Input

```
11011110
```

Output

```
\octal{slide}
```

Input

```
336
```

Output

```
\hexadecimal{slide}
```

Input

```
de
```

Output

```
\aaalph{slide}
```

Input

```
nnnnnnnnn
```

Output

```
\abalph{slide}
```

Input

```
hn
```

Output

Enumeration Counters

Up to four nested enumerate environments are permitted. Each level has an associated counter:

Counter	Representation	Default	Example
enumi	\theenumi	\arabic{enumi}	1
	\labelenumi	\theenumi.	1.
enumii	\theenumii	\alph{enumii}	a
	\labelenumii	\theenumii)	a)
enumiii	\theenumiii	\roman{enumiii}	i
	\labelenumiii	\theenumiii.	i.
enumiv	\theenumiv	\Alph{enumiv}	A
	\labelenumiv	\theenumiv.	A.

Changing enumerate Counter Format

```
\renewcommand{\theenumi}{\Roman{enumi}}
```

INPUT

```
\begin{enumerate}
\item\label{itm:first} First item
\item Second item
\end{enumerate}
Item~\ref{itm:first} \ldots
```

INPUT

I. First item

II. Second item

Item I . . .

OUTPUT

Changing enumerate Label

```
\renewcommand{\labelenumi}{\#\theenumi}
```

INPUT

```
\begin{enumerate}
\item\label{itm:first} First item
\item Second item
\end{enumerate}
Item~\ref{itm:first} \ldots
```

INPUT

#1 First item

#2 Second item

Item 1 . . .

OUTPUT

Changing enumerate Format

```
\renewcommand{\theenumi}{\Numberstring{enumi}}
\renewcommand{\labelenumi}{\theenumi:}
```

INPUT

```
\begin{enumerate}
\item\label{itm:first} First item
\item Second item
\end{enumerate}
Item~\ref{itm:first} \ldots
```

INPUT

One: First item

Two: Second item

Item One . . .

OUTPUT

Exercise 22 (Page 24)

- Modify the environment you created in Exercise 21 so that for example, the following code:

  ```
  \begin{exercise}
  Some text.
  \end{exercise}
  ```

  would produce the following output:

  Exercise 1

  Some text.

- The value of the counter will need to be incremented at the start of each exercise.

- Try referencing it using `\label` and `\ref`.

Conditionals

- As with all programming languages, TEX has conditionals.
- TEX conditionals are of the form:
  `\if`*type* . . . `\else` . . . `\fi`
- We have already encountered a conditional (ifpdf package):
  `\ifpdf` . . . `\else` . . . `\fi`
- The `\ifcase` command is similar to the switch statement found in some languages such as C. Example:

```
\ifcase\month \or Jan\or Feb\or Mar\or Apr\or May\or
Jun\or Jul\or Aug\or Sep\or Oct\or Nov\or Dec\fi
```

Conditionals

- As with all programming languages, TEX has conditionals.
- TEX conditionals are of the form:
  `\if`*type* . . . `\else` . . . `\fi`
- We have already encountered a conditional (ifpdf package):
  `\ifpdf` . . . `\else` . . . `\fi`
- The `\ifcase` command is similar to the switch statement found in some languages such as C. Example:

```
\ifcase\month \or Jan\or Feb\or Mar\or Apr\or May\or
Jun\or Jul\or Aug\or Sep\or Oct\or Nov\or Dec\fi
```

Conditionals

- As with all programming languages, TeX has conditionals.

- TeX conditionals are of the form:
  \if*type* . . . \else . . . \fi

- We have already encountered a conditional (ifpdf package):
  \ifpdf . . . \else . . . \fi

- The \ifcase command is similar to the switch statement found in some languages such as C. Example:

```
\ifcase\month \or Jan\or Feb\or Mar\or Apr\or May\or
Jun\or Jul\or Aug\or Sep\or Oct\or Nov\or Dec\fi
```

Conditionals

- As with all programming languages, TEX has conditionals.
- TEX conditionals are of the form:

  ```
  \iftype ... \else ... \fi
  ```
- We have already encountered a conditional (ifpdf package):

  ```
  \ifpdf ...\else ...\fi
  ```
- The `\ifcase` command is similar to the switch statement found in some languages such as C. Example:

  ```
  \ifcase\month \or Jan\or Feb\or Mar\or Apr\or May\or
  Jun\or Jul\or Aug\or Sep\or Oct\or Nov\or Dec\fi
  ```

Example (\ifpdf and \pdfinfo)

- \ifpdf is quite often used in conjunction with \pdfinfo.
- The \pdfinfo command is only defined in PDFLaTeX not LaTeX.

```
\ifpdf
   \pdfinfo{
      /Title (A Sample Document)
      /Author (Nicola Talbot)
      /CreationDate (D:20040930140000)
      /ModDate (D:\pdfdate)
      /Subject (LaTeX Example)
   }
\fi
```

The ifthen Package

- The ifthen package (provided by Leslie Lamport and extended by David Carlisle) defines the conditional commands
  - `\ifthenelse`{*test*}{*then text*}{*else text*}
  - `\whiledo`{*test*}{*do text*}
- The argument *test* is a boolean statement.
- If *test* is true, *then text* or *do text* will be executed.
- If *test* is false, *else text* is executed, or in the case of `\whiledo`, the loop is terminated.
- Easier to use than TEX conditionals.

The ifthen Package

- The ifthen package (provided by Leslie Lamport and extended by David Carlisle) defines the conditional commands
  - `\ifthenelse`{*test*}{*then text*}{*else text*}
  - `\whiledo`{*test*}{*do text*}
- The argument *test* is a boolean statement.
- If *test* is true, *then text* or *do text* will be executed.
- If *test* is false, *else text* is executed, or in the case of `\whiledo`, the loop is terminated.
- Easier to use than TEX conditionals.

The ifthen Package

- The ifthen package (provided by Leslie Lamport and extended by David Carlisle) defines the conditional commands
  - `\ifthenelse`{*test*}{*then text*}{*else text*}
  - `\whiledo`{*test*}{*do text*}
- The argument *test* is a boolean statement.
- If *test* is true, *then text* or *do text* will be executed.
- If *test* is false, *else text* is executed, or in the case of `\whiledo`, the loop is terminated.
- Easier to use than TEX conditionals.

The ifthen Package

- The ifthen package (provided by Leslie Lamport and extended by David Carlisle) defines the conditional commands
  - `\ifthenelse`{*test*}{*then text*}{*else text*}
  - `\whiledo`{*test*}{*do text*}
- The argument *test* is a boolean statement.
- If *test* is true, *then text* or *do text* will be executed.
- If *test* is false, *else text* is executed, or in the case of `\whiledo`, the loop is terminated.
- Easier to use than TEX conditionals.

The ifthen Package

- The ifthen package (provided by Leslie Lamport and extended by David Carlisle) defines the conditional commands
  - `\ifthenelse`{*test*}{*then text*}{*else text*}
  - `\whiledo`{*test*}{*do text*}
- The argument *test* is a boolean statement.
- If *test* is true, *then text* or *do text* will be executed.
- If *test* is false, *else text* is executed, or in the case of `\whiledo`, the loop is terminated.
- Easier to use than TEX conditionals.

Boolean Operations

- Boolean variables can be defined using the command:

  ```
  \newboolean{name}
  ```

  where *name* (no backslash) is the name of the new variable.

- The variable can be assigned a value using

  ```
  \setboolean{name}{value}
  ```

  where *name* is the name of the boolean variable and *value* is either `true` or `false`.

- The value of a boolean variable can be tested using

  ```
  \boolean{name}
  ```

Boolean Operations

- Boolean variables can be defined using the command:

  ```
  \newboolean{name}
  ```

  where *name* (no backslash) is the name of the new variable.

- The variable can be assigned a value using

  ```
  \setboolean{name}{value}
  ```

  where *name* is the name of the boolean variable and *value* is either `true` or `false`.

- The value of a boolean variable can be tested using

  ```
  \boolean{name}
  ```

Boolean Operations

- Boolean variables can be defined using the command:

  ```
  \newboolean{name}
  ```

  where *name* (no backslash) is the name of the new variable.

- The variable can be assigned a value using

  ```
  \setboolean{name}{value}
  ```

  where *name* is the name of the boolean variable and *value* is either `true` or `false`.

- The value of a boolean variable can be tested using

  ```
  \boolean{name}
  ```

Example

- Suppose I have to teach the same course to two separate classes. The majority of the notes will be the same, with minor differences regarding dates and times.
- Suppose one class is on Fridays at 9:00am and the other is on Mondays at 10:00am.
- Define a boolean variable:

  ```
  \newboolean{friday}
  ```
- Specify whether or not this variable is true. e.g.:

  ```
  \setboolean{friday}{true}
  ```
- Can now use `\ifthenelse` and `\boolean`:

  ```
  Classes are on
  \ifthenelse{\boolean{friday}}{Fridays}{Mondays} at
  \ifthenelse{\boolean{friday}}{9:00am}{10:00am}.
  ```

Example

- All the TEX conditionals, such as `\ifpdf` have equivalent boolean variables, such as `pdf`.
- Better to use `\ifthenelse` instead of the lower-level `\ifpdf` etc.
- Our earlier example can now be written:

```
\ifthenelse{\boolean{pdf}}{%
A PDF\LaTeX\ document
}{%
A \LaTeX\ document
}
```

INPUT

A PDFLATEX document

OUTPUT

Testing Text

- To test whether two strings are equal, use:
  `\equal{`*string 1*`}{`*string 2*`}`

- Example:

  ```
  The work is written in
  \ifthenelse{\equal{\lang}{English}}
              {English} {another language}
  ```

  If the command `\lang` has been defined to be `English`, the following output will appear:

  ```
  The work is written in English.
  ```

  Output

  If `\lang` has been defined as something else:

  ```
  The work is written in another language.
  ```

  Output

Testing Numbers

- Two numbers can be compared using $<$, = and $>$
- The value of a counter can be compared using \value{*name*}
- Example:

```
This
\ifthenelse{\value{page}=42}{is}{isn’t}
my favourite page.
```

- To test whether a number is odd or even use \isodd{*value*}
- Example:

```
This page is an
\ifthenelse{\isodd{page}}{odd}{even}
numbered page.
```

\whiledo **example**

```
\newcounter{lines}
\whiledo{\value{lines}<6}
         {I will hand my homework in on time.\par
          \stepcounter{lines}}
```

Input

```
I will hand my homework in on time.
I will hand my homework in on time.
I will hand my homework in on time.
I will hand my homework in on time.
I will hand my homework in on time.
I will hand my homework in on time.
```

Output

Testing Lengths

- Lengths can be tested using
  `\lengthtest{`*relation*`}`

- Example:

```
This page is in
\ifthenelse{\lengthtest{\paperwidth > \paperheight}}
          {landscape} {portrait}
```

INPUT

This page is in portrait

INPUT

Writing a LaTeX Package

- Filename should have `.sty` extension.
- All LaTeX 2_ε packages should start with the line

  ```
  \NeedsTeXFormat{LaTeX2e}
  ```

- You must specify the name of the package using the command `\ProvidesPackage{`*name*`}[`*version*`]`

  For example, if your file is called, say `example.sty`, then you must have the line

  ```
  \ProvidesPackage{example}
  ```

  You can also specify the version in the optional argument:

  ```
  \ProvidesPackage{example}[2004/05/21 v1.0 (A.N. Other)]
  ```

- The last line of the file should have the command `\endinput`

Writing a LaTeX Package

- Filename should have `.sty` extension.
- All LaTeX 2_ε packages should start with the line
  `\NeedsTeXFormat{LaTeX2e}`
- You must specify the name of the package using the command
  `\ProvidesPackage{`*name*`}[`*version*`]`

  For example, if your file is called, say `example.sty`, then you must have the line

  `\ProvidesPackage{example}`

  You can also specify the version in the optional argument:

  `\ProvidesPackage{example}[2004/05/21 v1.0 (A.N. Other)]`
- The last line of the file should have the command `\endinput`

Writing a LaTeX Package

- Filename should have `.sty` extension.
- All LaTeX 2_ε packages should start with the line

  `\NeedsTeXFormat{LaTeX2e}`
- You must specify the name of the package using the command

  \ProvidesPackage{*name*}[*version*]

  For example, if your file is called, say `example.sty`, then you must have the line

  ```
  \ProvidesPackage{example}
  ```

  You can also specify the version in the optional argument:

  ```
  \ProvidesPackage{example}[2004/05/21 v1.0 (A.N. Other)]
  ```
- The last line of the file should have the command `\endinput`

Writing a LaTeX Package

- Filename should have `.sty` extension.
- All LaTeX 2_ε packages should start with the line

  ```
  \NeedsTeXFormat{LaTeX2e}
  ```

- You must specify the name of the package using the command

  \ProvidesPackage{*name*}[*version*]

  For example, if your file is called, say `example.sty`, then you must have the line

  ```
  \ProvidesPackage{example}
  ```

  You can also specify the version in the optional argument:

  ```
  \ProvidesPackage{example}[2004/05/21 v1.0 (A.N. Other)]
  ```

- The last line of the file should have the command `\endinput`

Writing a LaTeX Package

- Filename should have `.sty` extension.
- All LaTeX 2_ε packages should start with the line

  `\NeedsTeXFormat{LaTeX2e}`
- You must specify the name of the package using the command

  `\ProvidesPackage{`*name*`}[`*version*`]`

  For example, if your file is called, say `example.sty`, then you must have the line

  ```
  \ProvidesPackage{example}
  ```

  You can also specify the version in the optional argument:

  ```
  \ProvidesPackage{example}[2004/05/21 v1.0 (A.N. Other)]
  ```
- The last line of the file should have the command `\endinput`

Writing a LaTeX Package

- Filename should have `.sty` extension.
- All LaTeX 2$_\varepsilon$ packages should start with the line
  `\NeedsTeXFormat{LaTeX2e}`
- You must specify the name of the package using the command
  `\ProvidesPackage{`*name*`}[`*version*`]`

  For example, if your file is called, say `example.sty`, then you must have the line

  ```
  \ProvidesPackage{example}
  ```

  You can also specify the version in the optional argument:

  ```
  \ProvidesPackage{example}[2004/05/21 v1.0 (A.N. Other)]
  ```

- The last line of the file should have the command `\endinput`

Example

This is a very simple package. It redefines \today to produce the date in the form 21/5/2004.

```
\NeedsTeXFormat{LaTeX2e}

\ProvidesPackage{vardate}[2004/05/21 v1.0 (N.L.C. Talbot)]

\renewcommand{\today}{\the\day/\the\month/\the\year}

\endinput
```

VARDATE.STY

Package Options

- We have already come across packages that can have options passed to them (e.g. graphicx)
- Options can be defined using

  ```
  \DeclareOption{option}{code}
  ```

  where *option* is the option name and *code* is what LaTeX should do if this option is specified.
- The default action for any option not defined is given by

  ```
  \DeclareOption*{code}
  ```

  Within *code*, the following commands may be used:

`\CurrentOption`	name of current option
`\OptionNotUsed`	Marks this option as being unprocessed.

Package Options

- We have already come across packages that can have options passed to them (e.g. graphicx)
- Options can be defined using

  ```
  \DeclareOption{option}{code}
  ```

  where *option* is the option name and *code* is what LaTeX should do if this option is specified.
- The default action for any option not defined is given by

  ```
  \DeclareOption*{code}
  ```

  Within *code*, the following commands may be used:

`\CurrentOption`	name of current option
`\OptionNotUsed`	Marks this option as being unprocessed.

Package Options

- We have already come across packages that can have options passed to them (e.g. graphicx)
- Options can be defined using

  ```
  \DeclareOption{option}{code}
  ```

  where *option* is the option name and *code* is what LaTeX should do if this option is specified.
- The default action for any option not defined is given by

  ```
  \DeclareOption*{code}
  ```

  Within *code*, the following commands may be used:

`\CurrentOption`	name of current option
`\OptionNotUsed`	Marks this option as being unprocessed.

Package Options

- The options are then processed using the commands

`\ExecuteOptions{`*options*`}`	list of default options
`\ProcessOptions`	process in order defined
`\ProcessOptions*`	process in order specified.

- It is also possible to pass options to another package using

  `\PassOptionsToPackage{`*option list*`}{`*package name*`}`

- The named package must later be loaded using

  `\RequirePackage{`*package name*`}`

Package Options

- The options are then processed using the commands

\ExecuteOptions{*options*}	list of default options
\ProcessOptions	process in order defined
\ProcessOptions*	process in order specified.

- It is also possible to pass options to another package using

\PassOptionsToPackage{*option list*}{*package name*}

- The named package must later be loaded using

\RequirePackage{*package name*}

Package Options

- The options are then processed using the commands

`\ExecuteOptions{`*options*`}`	list of default options
`\ProcessOptions`	process in order defined
`\ProcessOptions*`	process in order specified.

- It is also possible to pass options to another package using

```
\PassOptionsToPackage{option list}{package name}
```

- The named package must later be loaded using

```
\RequirePackage{package name}
```

Extending the datetime Package — `vardate.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — `vardate.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — `vardate.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — `vardate.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — `vardate.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — `vardate.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Extending the datetime Package — vardate.sty

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{vardate}

\RequirePackage{ifthen}

\newboolean{dashdate}

\DeclareOption{dashdate}{\setboolean{dashdate}{true}}
\DeclareOption{nodashdate}{\setboolean{dashdate}{false}}
\DeclareOption*{\PassOptionsToPackage{\CurrentOption}{datetime}}

\ExecuteOptions{dashdate}
\ProcessOptions

\RequirePackage{datetime}

\newdateformat{dashdate}{\THEDAY-\THEMONTH-\THEYEAR}
\ifthenelse{\boolean{dashdate}}{\dashdate}{}
\endinput
```

Writing Class Files

Writing a class file is very similar to writing a package, except:

- Use `\ProvidesClass` instead of `\ProvidesPackage`
- Use the command

  `\PassOptionsToClass{`*options*`}{`*class name*`}`

  to pass *options* to the named class. This class file should later be loaded using

  `\LoadClass{`*class name*`}`

Extending the report Class File — `myrep.cls`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesClass{myrep}

\DeclareOption*{\PassOptionsToClass{report}}

\ProcessOptions
\LoadClass{report}
\RequirePackage[margins=1in]{geometry}

\renewcommand{\thechapter}{\Roman{chapter}}
\endinput
```

@ Commands

- The `@` character behaves differently depending whether it is in a class or package (`.cls`/`.sty`) file or whether it is in the document (`.tex`) file.

- In a `.tex` file, `@` is treated as a symbol. The symbol can not occur within a command name.
  E.g. \c@page will produce: @̧page.

- In a `.cls` or `.sty` file, `@` is treated as a letter. The letter can occur within a command name.
  E.g. \c@page is a command name (an internal representation of the page counter.)

- Commands containing the `@` symbol are internal commands, and should only be used in a class or package file.

@ Commands

- The `@` character behaves differently depending whether it is in a class or package (`.cls`/`.sty`) file or whether it is in the document (`.tex`) file.
- In a `.tex` file, `@` is treated as a symbol. The symbol can not occur within a command name.
  E.g. `\c@page` will produce: @̧page.
- In a `.cls` or `.sty` file, `@` is treated as a letter. The letter can occur within a command name.
  E.g. `\c@page` is a command name (an internal representation of the page counter.)
- Commands containing the `@` symbol are internal commands, and should only be used in a class or package file.

@ Commands

- The `@` character behaves differently depending whether it is in a class or package (`.cls`/`.sty`) file or whether it is in the document (`.tex`) file.

- In a `.tex` file, `@` is treated as a symbol. The symbol can not occur within a command name.
  E.g. `\c@page` will produce: @̧page.

- In a `.cls` or `.sty` file, `@` is treated as a letter. The letter can occur within a command name.
  E.g. `\c@page` is a command name (an internal representation of the page counter.)

- Commands containing the `@` symbol are internal commands, and should only be used in a class or package file.

@ Commands

- The `@` character behaves differently depending whether it is in a class or package (`.cls`/`.sty`) file or whether it is in the document (`.tex`) file.

- In a `.tex` file, `@` is treated as a symbol. The symbol can not occur within a command name.
  E.g. `\c@page` will produce: @̧page.

- In a `.cls` or `.sty` file, `@` is treated as a letter. The letter can occur within a command name.
  E.g. `\c@page` is a command name (an internal representation of the `page` counter.)

- Commands containing the `@` symbol are internal commands, and should only be used in a class or package file.

Redefining Page Styles

- The command `\pagestyle{`*style*`}` calls the command `\ps@`*style*, and it is this command that redefines the header and footer.
- The headers and footers given by the commands: `\@oddhead`, `\@evenhead`, `\@oddfoot` and `\@evenfoot`. It is these commands that need to be redefined to change the headers and footers.
- Example: define a new page style called, say, `example`:

```
\newcommand{\ps@example}{%
   \renewcommand{\@oddhead}{}
   \renewcommand{\@evenhead}{}
   \renewcommand{\@oddfoot}{\hfill-\thepage-\hfill}
   \renewcommand{\@evenfoot}{\hfill-\thepage-\hfill}
}
```

Redefining Page Styles

- The command `\pagestyle{`*style*`}` calls the command `\ps@`*style*, and it is this command that redefines the header and footer.
- The headers and footers given by the commands: `\@oddhead`, `\@evenhead`, `\@oddfoot` and `\@evenfoot`. It is these commands that need to be redefined to change the headers and footers.
- Example: define a new page style called, say, `example`:

```
\newcommand{\ps@example}{%
   \renewcommand{\@oddhead}{}
   \renewcommand{\@evenhead}{}
   \renewcommand{\@oddfoot}{\hfill-\thepage-\hfill}
   \renewcommand{\@evenfoot}{\hfill-\thepage-\hfill}
}
```

Redefining Page Styles

- The command `\pagestyle{`*style*`}` calls the command `\ps@`*style*, and it is this command that redefines the header and footer.

- The headers and footers given by the commands: `\@oddhead`, `\@evenhead`, `\@oddfoot` and `\@evenfoot`. It is these commands that need to be redefined to change the headers and footers.

- Example: define a new page style called, say, `example`:

```
\newcommand{\ps@example}{%
   \renewcommand{\@oddhead}{}
   \renewcommand{\@evenhead}{}
   \renewcommand{\@oddfoot}{\hfill-\thepage-\hfill}
   \renewcommand{\@evenfoot}{\hfill-\thepage-\hfill}
}
```

Changing the Section Headings

Sections, subsections etc headings can be changed by redefining `\section`, `\subsection` etc. These commands should use the command

`\@startsection`{*type*}{*level*}{*indent*}{*before*}{*after*}{*style*}

to format the heading.

type The sectioning type. (`section`, `subsection` etc)

level A number representing the sectioning level

indent A length, specifying indentation from the left margin

before The absolute value of this length gives the vertical distance before heading. If the value is negative, the first paragraph will not be indented.

after The absolute value of this length gives the vertical distance after heading. If negative, running heading used.

style Declarations for setting the style (e.g. `\bfseries`)

Example

Suppose you want to change the section headings so that they appear in a large italic font, you could do something like:

```
\renewcommand{\section}{\@startsection
{section}%                        % the name
{1}%                              % the level
{0mm}%                            % the indent
{-\baselineskip}%                 % the before skip
{0.5\baselineskip}%               % the after skip
{\normalfont\large\itshape}} % the style
```

Changing Chapter Headings

If you are using the report class file, or something similar, you can modify the chapter headings by redefining:

- `\@makechapterhead` for numbered chapters (produced using `\chapter`)
- `\@makeschapterhead` for unnumbered chapters (produced using `\chapter*`)

If you want to modify the part headings, you need to redefine:

- `\@part` for numbered parts (produced using `\part`)
- `\@spart` for unnumbered parts (produced using `\part*`)

The easiest way to do this is copy the code from the class file, and modify the appropriate formatting commands.

Example

This example changes the numbered chapter headings so that a line appears above and below the heading, and the heading itself appears in small capitals.

```
\renewcommand{\@makechapterhead}[1]{%
  \vspace*{50\p@}%
  {\parindent \z@ \raggedright \normalfont
    \hrule                                         % horizontal line
    \vspace{5pt}%                                  % add some vertical space
    \ifnum \c@secnumdepth >\m@ne
        \huge\scshape \@chapapp\space \thechapter % Chapter followed by number
        \par\nobreak
        \vskip 20\p@
    \fi
    \interlinepenalty\@M
    \Huge \scshape #1\par                          % chapter title
    \vspace{5pt}%                                  % add some vertical space
    \hrule                                         % horizontal rule
    \nobreak
    \vskip 40\p@
  }}
```

Exercise 24 (Page 26)

- Write a class file that loads the report class file and that:
  - modifies the chapter headings so that a line appears above and below the heading and the heading appears in small capitals centred.
  - modifies the section headings so that they appear in a large sans-serif font.
- Modify the document you used in Exercise 4 so that it uses your new class file instead of the report class file.

References

[1] “A Guide to LaTeX 2_ε: document preparation for beginners and advanced users”, Helmut Kopka and Patrick W. Daly, Addison-Wesley (1995).

[2] “The LaTeX Companion”, Michel Goossens, Frank Mittelbach and Alexander Samarin (Addison-Wesley, 1994).

[3] “The LaTeX Graphics Companion”, Michel Goossens, Sebastian Rahtz and Frank Mittelbach, Addison-Wesley (1997).

[4] “The LaTeX Web Companion”, Michel Goossens and Sebastian Rahtz, Addison-Wesley (1999).

Web Sites

- TEX archive site: `http://www.tex.ac.uk/`
- These slides are available at:
  `http://theoval.cmp.uea.ac.uk/~nlct/latex/csed/csed.html`

Index

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z