LaTeX, a Braindump

Edith Hodgen

December 5, 2000

Preface

This document, for lack of a better description, was written as a "brain-dump" of all the useful things that I learned about LaTeX in my three years in the office. Of course, it turned out to a brain-dump of all the useful things I learned in the three years in the office and, more particularly, while checking on bits and pieces during the construction of the document.
It was designed to:

1. serve as an approximation to an introduction to LaTeX;
2. provide a unified set of "most useful" commands, environments and other clever things;
3. supplement what is in the books (*LaTeX: A Document Preparation System* and *The LaTeX Companion*) with the odd truly useful thing that doesn't appear in them;
4. do all of the above in the local (Unix) environment — at least as I know it.

The first three drafts of the document resulted in several constructive suggestions that have been incorporated in this edition.
Thanks are due to those who placed sufficient faith in me to put me where I could and did "learn LaTeX"; those who, like Megan Clark, Tony Vignaux and David Vere-Jones, wanted a non-standard effect and talked down my insistence that it was "Just not possible" – to the point that I started reading the manuals; those who, like John and Justin and other graduate students, wanted to achieve an effect, and thought I might know how (but didn't, at least at first); and those, like Charles Semple and John Harper, who in the course of casual conversation shared some gem of arcane wisdom with me.
Any LaTeX-er will develop their own style and way of doing things (and as a corollary, any other LaTeX-er will probably do the same things in a different way). What is described here is my style of LaTeX, determined sometimes by what I learned first, and sometimes by what I judged to be best (and I'm not telling which is where).
Enjoy.

Edith Hodgen
Ex Statistical Computing Consultant

Contents

List of Tables

Part I

Getting Started

Chapter 1

Unix

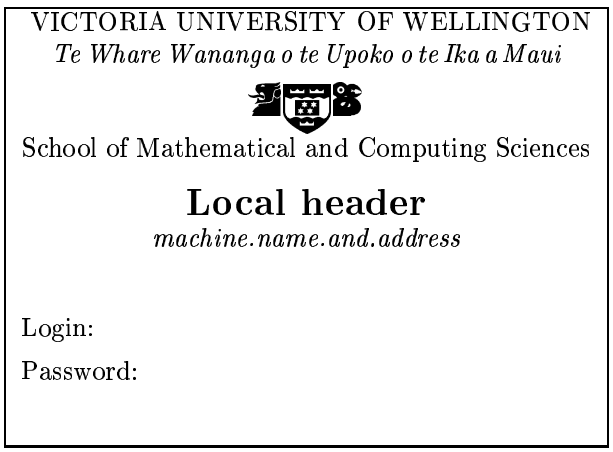
1.1 Conventions

The following notational conventions are used in these notes:

↩	"hit 'Enter' key"
`typewriter text`	type this in
`slanted typewriter text`	substitute what applies to you and type it in—for instance the name of a file or directory
[ ]	options—specification of whatever is between the brackets is optional
^a	Control key + `a` key—hold down the Control key and press `a`.

1.2 Logon

If the machine you are to use is ready for you to log on, when you move the mouse (or hit ↩ (Enter) key) you will see the logging on dialog box that looks something like this:



What you do is:

1. Type your "userid" next to the "login" prompt (if the cursor doesn't put itself there, click the left mouse button on that line). You can use ← (Back Space) to correct any mistakes as you type.
2. Hit ↩ (Enter).
3. Type your password (which does not appear on the screen). You can use ← (Back Space) to correct any mistakes as you type. Remember that the password is case sensitive.
4. Hit ↩ (Enter).

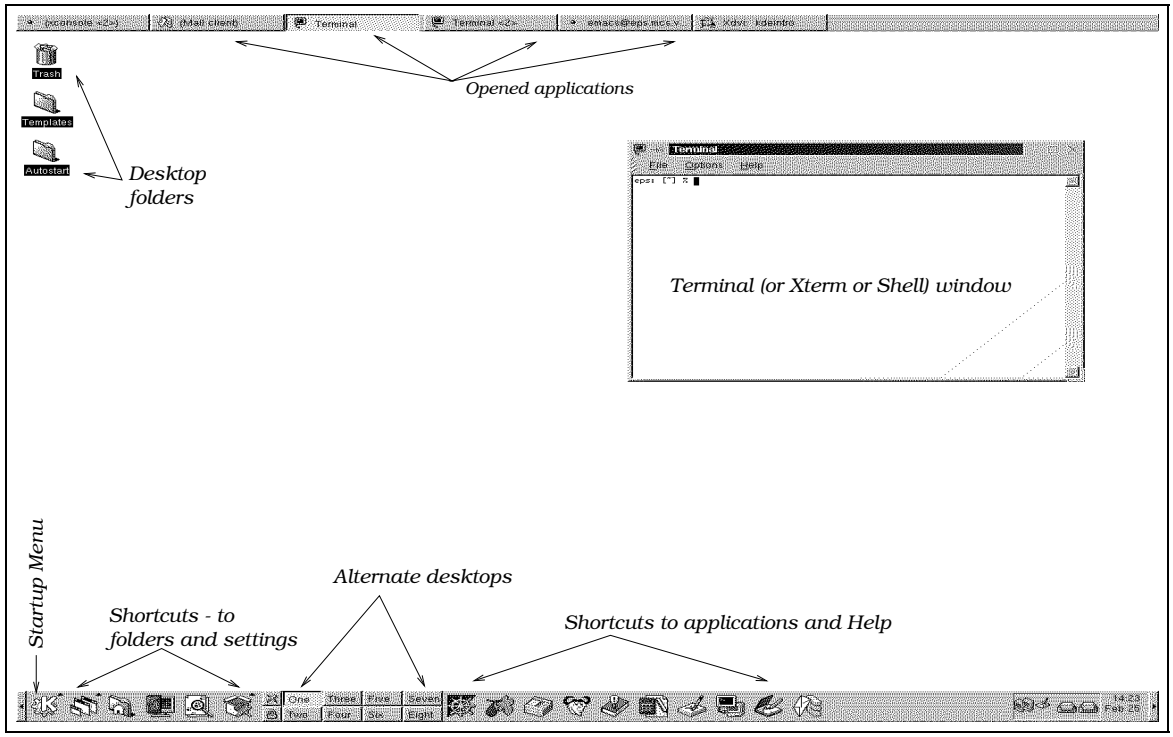
1.3 Desktop environments

At present there are, basically, two desktop environments in use in SMCS: KDE (in use since 1999) and Xwin. As they seem rather different to the user, they are discussed separately.

1.3.1 KDE

1.3.1.1 What You Get

This is the default KDE version of the desktop environment:



If you hold the mouse cursor over any one of the icons for a few seconds, a box describing (approximately) the purpose of the icon will appear (as it does in Windows or on a Mac).

KDE was designed to incorporate all the best features of Windows 95 (etc), Mac and OS/2 desktop environments, and to offer some features that are even better. And for all this to be within a Unix environment. Note that icons are activated by a **single** click with the left mouse button (1 click = 1 window opened; 2 clicks = 2 windows opened, and so on). If you have a double-click habit, you'll need to adjust it when using KDE.

The bar at the bottom of the screen can be made to appear and disappear by clicking on the extreme left or right, on the ◀ or ▶.

1.3.1.2 Logging off (Logout) and Locking The Screen

Before you log out, close all open applications (use the × in the top right-hand corner of the bar at the top of the window). It is particularly important that you close **Netscape**, and that an **SPlus (S+)** or R session is quit (type `q()` in the current command line) before closing the window (Terminal or Emacs) in which it is running.

You have at least two choices for logout:

- On the Startup Menu, select "Logout".
- Choose the icon on the bottom bar that looks sort of like a cross (x-shaped).

Either way, you will be asked whether you really are ready to log out, and after you click on the button to confirm that you are, *the screen goes back (after several seconds) to the login dialog box.* Only then do you **know** you're logged out.

Locking the screen (padlock below logout icon or on Startup Menu) is useful if you're in a public room, and you want to leave your terminal for a *short* while, but don't want to log out. Your files are protected and your place is kept. Enter your password to resume work. **Do not** leave a screen locked for more than 3 minutes—that would be very inconsiderate to other people sharing a public computer lab.

1.3.1.3 KDE Applications and Others

KDE perceives there to be two kinds of applications—KDE Apps and Non-KDE-Apps. KDE Apps are those applications that were designed specifically for KDE and are not found elsewhere. Non-KDE-Apps are applications (such as Emacs, Netscape, `Ghostview`, ...) that are found on many platforms.

How you use your mouse buttons depends on which type of application you are in. For instance, to move the scroll bar (on the right of a window) up and down in a KDE application window, (or Netscape), use the left mouse button; to do the same in a non-KDE application such as Emacs, use the middle mouse button.

1.3.1.4 Help

Help is available (and usually fairly helpful) on most features in KDE, and can be reached in the following ways:

1. If you ask for help from a KDE application, you get a menu offering help on
   - Contents (occasionally "Not written yet")—this gives access to the index of all the help available on that application,
   - About K*whatever* (a window giving the name of the programmer(s), and something about the development) or
   - *About KDE* (which gives the development details for KDE as a whole).
2. If you start by looking for help for a particular application, and then later want to look up something else, the way back to the main index is via the Help button on the menu bar at the top of the Help window (or the Startup menu, or the Help icon).
3. KDE Help (accessed from the Startup menu, or from the Help icon showing a book and light bulb containing a ! on the bar at the bottom of the screen) gives you access to all the help files.
4. Unix Manual Pages can be accessed from KDE Help.

1.3.1.5 Mouse

You will have the use of a mouse with either 2 or 3 buttons. A 2-button mouse will emulate a 3-button mouse, with a click of both buttons together having the same effect as the middle mouse button.

KDE-application window menus drop with the left mouse button (as they do in a Mac or Windows environment). The right (menu) button drops other menus (as it does in a Windows environment).

The mouse buttons may be used differently in KDE application windows and non-KDE application windows. For instance, in an Emacs window you use the left mouse button to select an item from a menu or to select an icon; the right button to drop a menu, either from a menu bar or from the top bar of a window, or from a background (all providing different options). The middle button is used to drop what's on the clip board at the point that was clicked on (and also works to drop menus from a menu bar).

It pays to spend a while pointing and clicking with all three buttons at different features on the screen, and within different types of window (KDE apps and non-KDE apps) to work out how to use the three mouse buttons.

1.3.1.6 Cut or Copy and Paste

Being able to copy (or cut) bits of text from here and drop them there is useful.

In KDE, if you click the left mouse button at the start of the text, hold the button down and drag the button to the end of the text, you can then drop the text in another window by clicking the middle

mouse button where the text is to be dropped. Drag and drop editing does not work in some (most) applications.

Highlighted text can be deleted or overwritten in a KDE window.

Note that any non-KDE application (like Emacs) may have its own cut, copy, delete and deal with highlighted text conventions. Editing may be a bit different in such applications, but the highlight and drop should work between any two windows.

1.3.1.7 Terminal (Xterm or Shell Window)

KDE gives you an environment in which you can ease gradually into learning unix commands. The unix shell can be compared to the command line facility provided by a DOS window in a Windows 95/98 environment, but is much more powerful (learn how, and you'll really *want* to use it).

The "Terminal Emulation (kvt)" window (icon on bar at the bottom of the screen, or "Terminal" on the Utilities subdirectory of the Startup Menu) allows you to give unix commands to do things like

- open applications, or open a specific file in an application;
- print a file (giving any options needed);
- display a unix manual page;
- show the contents of a (text) file;
- list files in a directory;
- move, copy, rename and delete files;
- change the protection mode of a file;
- change your password;
- make or delete directories;
- compress and decompress files

and a whole heap more.

1.3.1.8 File Browser

This can be opened from an icon on the bottom bar (icon of hut in front of folder, message "This folder contains all your personal files") or from the StartUp Menu ("Home Directory").

- Drag and drop works to move files—and so to throw them away in the Trash.
- On the View menu, select "View tree" to get a second window of directories if you want to move files between directories in this way (alternate method: use the unix command `mv`—look up the `man` page—in a terminal (Xterm) window).
- Click once on a file to open it in the best environment—Kedit or Emacs for text files, Postscript Viewer for postscript, Kview for gif, etc.—the environment is determined by the file extension. The applications used are, mainly, KDE applications, which may not actually be your preference. If you do not want the default application, open the application you want first (from a menu or icon, or by giving the appropriate command in an Xterm (Terminal) window), and then open the file from within the application (for instance, open an Emacs window, and then find the file—or give the command `emacs` *`filename`* `&` in an Xterm window).
- Depending on the default settings for your desktop environment, it is possible to get somewhat unexpected results when opening a directory (folder) that contains an `index.html` file—the file browser displays that web page, and can be used just like a web browser (to follow links, etc.). This can be changed on the View menu of the file browser. To edit (or locate) an html file it would, with such settings, be safer to use the xterm (Terminal) window and a text editor (Emacs, say).
- Backup copies are made automatically when files are edited—unix system backups have a `~` or other special character at the end of the file name. (In case you wondered where something like *`filename`*`~` (if the file was edited in Emacs) came from.)

1.3.1.9 Deleting Files

You can delete a file in several ways:

- As you would in a Windows or Mac environment, drag the icon(s) of the file(s) to be deleted from the File Browser window onto the Trash icon on the desk top.

Files in the Trash remain there, and can be rescued, until the Trash is emptied.
Click on the Trash icon with the right mouse button (menu button) and select **Empty Trash** to delete the Trash files permanently.

- From the Edit menu on the File Browser window, you have a choice of "Move to Trash" or "Delete". You can get back files from the Trash. Deleted files are gone forever.
- From an Xterm (Terminal) window, the command `rm mydoc.tex` will instantly and permanently remove the file `mydoc.tex` from the current directory. Use this command with care.

1.3.1.10 Text Editors and Emacs

KDE Text Editors KDE will make yellow stickies containing short messages or reminders that you can scatter around your desktops (icon of sticky-pad on bottom bar of screen, or Knotes on Utilities submenu of Startup Menu).

There is a simple text editor (Kedit) that has a few commands (see the Help file) for moving the cursor around and deleting anything between a single character and most of a line of text. It's not a bad place to start, or for editing short files (icon of open book and pencil on bottom bar of screen, or Text Editor on Applications Submenu of Startup Menu).

There is an "Advanced Editor" (StartUp Menu, Applications submenu) that is a "programmers' editor", in that it can be set to provide "colorized syntax" for different programming languages (for example C/C++, Java, Python, Perl, Bash, HTML, ...). This means that if, for instance, you are writing a Java program, you can have all keywords, character, string, comments, etc. displayed in different colours or fonts (so it's easy to identify them in the program and to see where each starts and stops).

There are commands for finding, replacing, copying, indenting, and unindenting (usual basic editing stuff) either on the edit menu or as a control sequence (open the application and have a look at the edit menu to see them) that are very similar to the equivalent commands on a Mac (except you use the Control key for KDE) or PC.

Emacs But if you like powerful toys and clever tricks (or are lazy enough to spend time learning how to save time) you should learn to use Emacs (**E**ditor with **MAC**ro**S**) which has

- macros (defined for repeated tasks)
- excellent search facility
- powerful search and replace, including the ability to use regular expressions
- spellcheck
- different modes for different editing tasks, for instance:
  - text
  - LaTeX or TeX
  - ESS for SAS or SPlus (statistical packages)

  that are invoked automatically, as determined by the file extension of the file being edited.
- Considerable possibilities for customisation: width of text on the screen, use of function keys, use of colours to make words, comments, commands and other structural features stand out on the screen (the same idea as in the "Advanced Editor"), etc.

Control commands for Emacs are given in section 1.9, page 17.

1.3.1.11 Virtual Screens or Multiple Desktops

KDE provides between 2 (your default) and 8 virtual desk tops.

- Buttons to change between the desk tops are on the bottom icon bar (buttons named "One", "Two", ... , "Eight").
- This means that you can have an uncluttered desktop dedicated to each of your usual unix activities, such as LaTeX, Emacs, SPlus, Mail, Netscape, ...
- Or you can have a desktop for each course you are involved with, or for ...
- Each opened application is "iconified" on a bar (at the top of the screen). Click on one of these buttons, and you are taken instantly to that application on its associated desktop.

1.3.1.12 Drawing Pins and Sticky

On the bar at the top of each window is a - and a drawing pin.

Click with the left mouse button on the -, and you get a menu offering you the choice to "Maximise", "Iconify", "Sticky", "Move to Desktop", "Close", etc. the current window.

"Move to Desktop" gives you a way to shift opened windows to where you want them.

"Sticky" has the same effect as clicking on the drawing pin (and causes that icon to change): the window "pinned to the desktop" will move to any desktop you have open (it's stuck, goes with you and is always visible). If you click on the drawing pin again, it's unstuck, and gets left on the desktop where you were when you "unstuck" it.

1.3.1.13 Printers

You are advised to read about the school printers, where they are and the printing options available for each (TechNote 101 at `http://www.mcs.vuw.ac.nz/technical/TechNotes/tn101`).

You will have been set up with one of the printers as your default printer, and usually that printer should be the most suitable for you to use.

But sometimes you may need to use one of the other printers. You can specify which by

- Giving an `lpr -P`*`printer`* command in an xterm (Terminal) window (look up the manual pages by typing `man lpr` in an xterm (Terminal) window). This enables you to give any printing options (see TechNote 101) you want, too.
- The print command (button) of some applications will let you specify options (and so printer name) in an `lpr` command, or by some other means specify the printer (this possibility is very application-dependent, so you'll have to "suck it and see").

1.3.1.14 Customisation

As you would expect, if you go into the KDE Control Center (icon on bottom bar, or on Startup menu) you can customise your KDE environment, much as you can in a Windows or Mac environment.

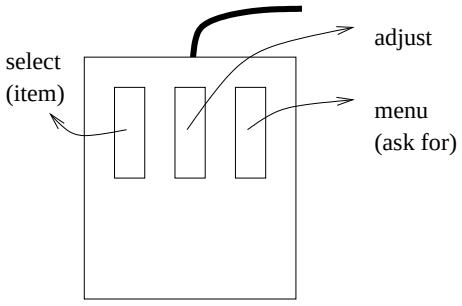
We strongly recommend:

- You use the default settings until you feel fully at home with KDE.
- Once you start changing things, you make your changes **one at a time** (then it's easy to change back to what you had), otherwise you may have some unexpected and unpleasant results that you have difficulty reversing.

1.3.2 Xwin

When you log on, you have a mainly blank desktop, with perhaps an open window, and some icons around the edges.

1.3.2.1 Mouse



Try clicking the **menu** button on:

- the background of the screen
- the background of an open window
- the top bar of an open window.

You should get one of a range of menus appearing, depending on where the mouse button was pointing when you clicked. You need to "play" with the menus to learn what you can do, and which facilities available are on which menu. To make the menus work, click once, firmly, on the right (menu) mouse button and release, and then click on the menu item you want with the left (select) mouse button. If you

want a sub-menu (menu items that have a ▶ have sub-menus), click, firmly, with the right (menu) mouse button on the ▶ to drop the sub-menu, and then click on the menu item you want with the left (select) mouse button. You can also depress the menu mouse button in the appropriate place, hold it down, and drag the highlight to the item you want to select, and then release the button. But you may find you get unexpected items less often if you use the first method described.

One-button-mouse-MacX users can set the arrow keys to simulate a three-button mouse. Exactly how this is done can be set (Edit menu, under Miscellaneous option), with the most convenient being to have Select ≡ click of mouse button; Adjust ≡← + click of mouse button; Menu ≡→ + click of mouse button.

`Emacs` makes considerable use of the Control and Meta buttons. Control is the usual control key for MacX users, and Meta can be simulated (as for the mouse buttons) using ↑.

1.3.2.2 What you get

xterm Window in which you type unix commands to

- start an application/process, eg `Emacs` or `Xfig`;
- give file commands, eg `mv, cp, lpr` (or "move", "copy" and "print").

Typically, one of these windows is opened as you log on, and you can open another any time you need to by clicking on the background with the menu (right) mouse button (once, firmly) and then selecting `xterm` from the Main Menu.

File Manager On "ISOR" machines (not necessarily on COMP). The File Manager may appear as an icon at the bottom of the screen when you first log on (otherwise it may be open on the desk-top). Double click with the select (left) mouse button to open it, and you should see a window with two sub-windows: the smaller upper one shows the path to your home directory/folder (or the sub-directory/folder that is currently open in the File Manager); the larger lower one shows the files and folders/sub-directories in your home directory (or the sub-directory/folder that is currently open).

- The file manager works rather like Mac/PC file managers for most basic file operations (copy, move, delete, create).
- Files that are deleted by dropping them from the File Manager into the Waste (which should be on the desk top) remain in the trash until the File Manager is `Quit`. Files that are deleted using the Unix instruction `rm` in an `Xterm` window are gone instantly and forever (until you've grovelled to a friendly system manager who may be able to access a backup tape if you're lucky).
- Use the menu (right) mouse button to drop menus; either drag highlight down to desired item and release button, or click menu button once to drop, and then click the select (left) button on the menu item you want.
- Double-click on a file with the select (left) button to open the file. The file extension will determine how/if this is done—eg a `.tex` file will be opened in the default text editor (not `Emacs`); a `.fig` file will be opened in `xfig` and a `.ps` or `.eps` file will be opened in `Ghostview`.
- Sometimes you don't want to actually *open* a file in the appropriate application, but just to look at its contents (maybe to check whether a PostScript file is Encapsulated or not, or to examine the internal workings of an `Xfig` file). This can be done by selecting `Open in Editor` on the `File` menu (either drag highlight down to desired item and release button, or click menu button once to drop, and then click the select button on the menu item).
- To select more than one file, click the first with the select button, and the rest with the adjust (middle) button.

You can customise the File Manager, by working through all the possibilities obtained from `Properties` on the `Edit` menu.

Main Menu To get this menu, click with the menu button anywhere on the background. The options depend on the system (COMP or ISOR—actually, on the contents of your `.openwin-menu` file), but typically include `Logout`, which logs you out, and `Xlock` which locks the screen while you're away from the computer—useful in the lab or Research Facility where there is public access to the machines.

Virtual Desktop Open windows tend to proliferate in a Unix working environment, and the Virtual Desktop provides a means for organising them. The desktop will either be open on the desktop, or will appear as an icon (of a sort of desk, with a "V" on it, called Desktop—double-click with the select (left) mouse button to open). When opened, it will show the current desktop, typically in the top left corner, littered with miniature windows; other possible desk-tops are shown as empty, and are demarcated by dashed lines.

The idea is that you use one of the desk-tops for all windows of one type (or associated with one application), and put all the open windows of another type (or associated with a different application) on a different desktop. To move windows from one desk-top to another, point and click-hold on the miniature version on the virtual desktop (select button) and drag the miniature window to the bit of desktop of your choice.

If your cursor is pointing at the **background** of your actual desk-top, you can shift between desk-tops using Shift+arrow key (by half a desk-top at a time — remember that if you want to move more than once you may need to move the cursor back onto the background before hopping on) or Control+arrow key (as far as possible in that direction at a time).

1.4 File structure considerations

LaTeX generates several files per document (at least four for a basic document that is printed), so a good file structure makes life easier. For instance something along the lines suggested in Figure 1.1:

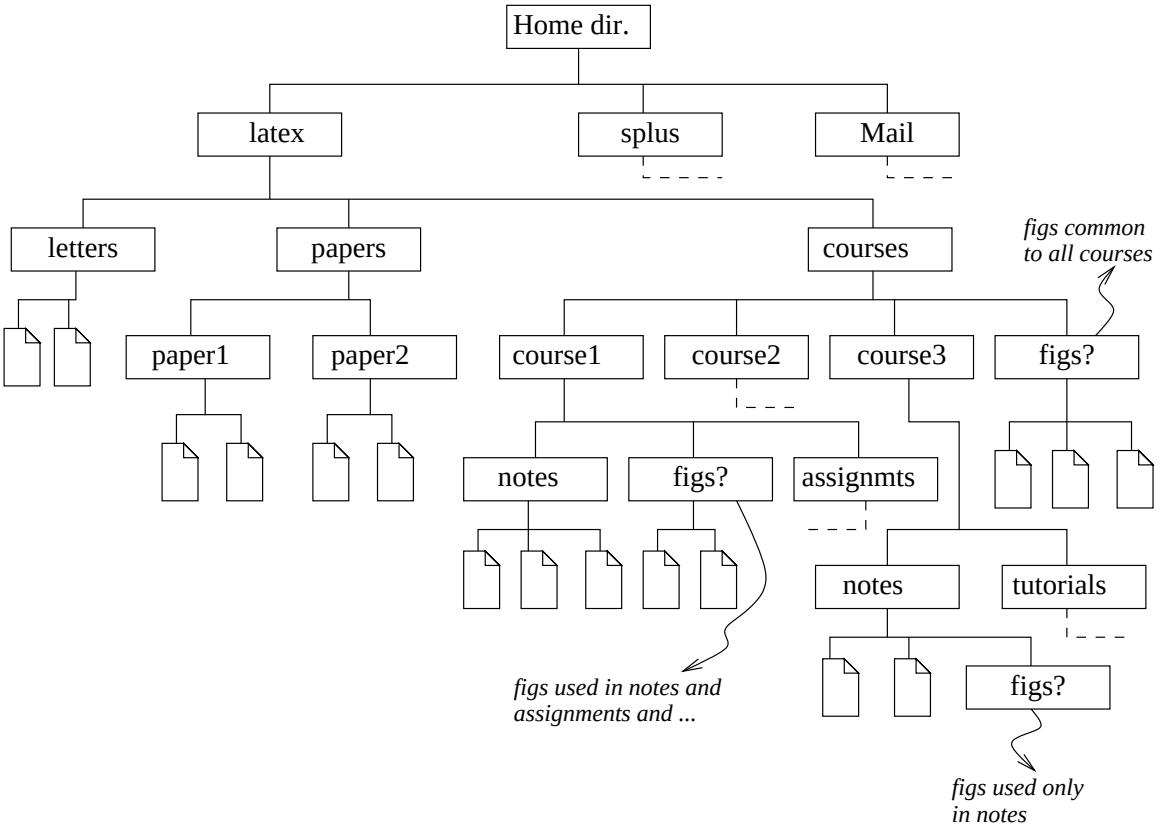


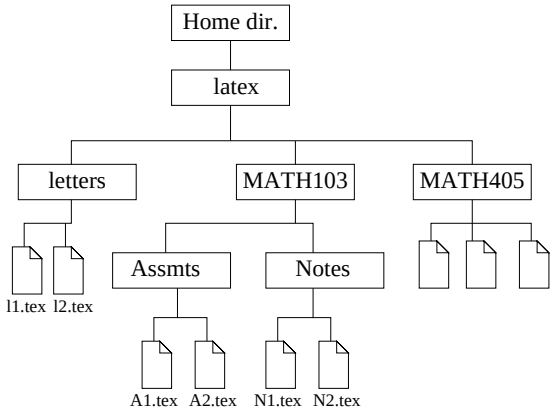
Figure 1.1: Suggested file structure.

Note that LaTeX has the requirement that file and directory names **cannot contain spaces**.

1.5 Useful shortcuts

In unix, life can be made a little easier if you use the following:

- **Completing filenames** —use the `Tab` key when specifying path or file names, and the system will supply any unique section of the path name that is in the pwd (present working directory). An example is given in Figure 1.2.



Starting in the home directory, to specify that you want A1.tex:

Type	Result
`Tab`	`~/latex` (as unique)
`M Tab`	`~/latex/MATH`
`1 Tab`	`~/latex/MATH103/`
`A Tab`	`~/latex/MATH103/Assmts`
`A1 Tab`	`~/latex/MATH103/Assmts/A1.tex`

Figure 1.2: Shortcuts for giving file names.

  Use of the Tab key is advisable because, firstly, it's quicker than typing out the path, and secondly, it confirms that the specified path is correct and the document is where you thought it would be (or even that you're in the directory in which you thought you were).

- It is possible to scroll through the series of unix commands given before in your working session. Use the ↑ key to move up the list and the ↓ key to move down the list.

- You can also enter `history` to get a numbered list of your previous commands on the screen. If you enter `!66` you will repeat command 66 in that list[1].

- ../ denotes the next directory up the tree.

  So if in the example tree in Figure 1.2, you are in `letters` and you wish to be in `MATH103/Assmts` you can change directory (folder) in one step by typing:

  ```
  cd ../MATH103/Assmts/
  ```

- . denotes the present working directory.

  So that, to copy all files from `MATH103/Notes` to the present working directory (pwd), you can specify:

  ```
  cp ~username/latex/MATH103/Notes/* .
  ```

  where the * is a wild card that signifies that all files be copied, and the . denotes that they be copied to the pwd.

  To copy only all the `.tex` files, say, you could specify

  ```
  cp ~username/latex/MATH103/Notes/*.tex .
  ```

  Note that you could have used ../../ as often as necessary to work up one branch of the directory tree before working down the MATH103 branch.

- `cd` ↩ will change you to your home directory.

[1] The `!` is a short-hand instruction to repeat the previous command, so that `history` ↩`!66` ↩is equivalent to `history 66` ↩.

1.6 Unix Commands

The following (by no means exhaustive) list of unix commands should be enough to get you started and satisfy most of your needs for some time to come. They're given in alphabetical order.

`applicationname &` — Start application and run console/xterm in background.
The `&` is very useful as it means that you do not need to shut the application window before giving the next instruction in the xterm window.

`cd pathname` — Change directory to that specified.
`cd`↩ changes to root directory
`cd .` changes to pwd (present working directory).
`cd ..` changes to parent directory of pwd.

`chmod g+w filename` — Give group write permission to that file (can only be done by owner of file).

`chmod g-w filename` — Remove group write permission for that file.
`chmod -R mode filename` Change mode recursively, descending through the directory, including subdirectories and symbolic links.

`chmod mode filename` — In general, *mode* can be given either as above (with all options listed below) or as an octal number:

`chmod` in *who opcode permission* form:

who	`u`	user
	`g`	group
	`o`	other
	`a`	all (the default)
opcode	`+`	add permission
	`-`	remove permission
	`=`	assign permission
permission	`r`	read
	`w`	write
	`x`	execute
	`s`	set user (or group ID)
	`u`	user's present permission
	`o`	other's present permission
	`g`	group's present permission
	`l`	mandatory locking

octal numbers — 3 digits given, the first specifying the owner's permission, the second that of the group, and the third that of others. The value of the digits given, is the sum of the values for the three possibilities:
4 = read 2 = write 1 = execute

eg: `chmod -R 740 myfiles/`

will (recursively) make all files (and directories?) readable, writable and executable for the user ($7 = 4 + 2 + 1$), readable for the group (4) and everyone else (others) has no access at all (0).

Equivalents:
`chmod 751 myfile` and
`chmod u=rwx, g=rx, o=x myfile`

`chmod =r myfile` and `chmod 444 myfile` and
`chmod a-wx, a+r myfile`

`chown newowner files` — Change ownership of one or more files to *newowner*, which is given either as user ID number or login name as specified in `/etc/passwd`.

`chown -h mode newowner files` changes owner on symbolic links.

`chown -R newowner files` changes owner recursively down directory.

`cp filename1 filename2` — Copy first file to second such that:

	`cp old old_dir`	Puts copy into existing directory *old_dir* without changing the name of the file.
	`cp -i old new`	Copies interactively (asking permission).
	`cp f1 f2...fn dir`	Copies the n files to *dir*.
	`cp ../* .`	Copies all files from parent directory (..) to pwd (.).

`date` — Displays current date and time.

`eject` — Eject a floppy disk. Some computers need the instruction given as `eject floppy`↩

`grep "string" files` — Prints occurrences of *"string"* in the named files.
`grep -v "strg" files` — Prints non-occurrences.
`grep ^"strg" files` — Prints lines beginning with *"strg"*.

If you anticipate many occurrences of *"string"*, you can pipe the results into `more`, as in:
`grep "string" filename(s) | more`

To list files that are not directories, use
`ls -lF | grep -v /`
To list files that are directories, use
`ls -lF | grep /`

`groups username(s)` — List groups that *username(s)* belongs to.

`less filename` — As `more`, but more powerful:

Help: Type `less` *filename* and then type `h` to get help, which displays possible commands for moving backwards and forwards through the file as well as doing all sorts of other things.

`ln -s trgtpath linknm` — Create a symbolic link to *trgtpath* called *linknm* in local directory. Can be between users.

`lpq` — **L**ist **p**rint **q**ueue on default printer—then look up your jobnumber (say to be able to remove job(s)).

`lpq -l`	for long display.
`lpq -Pprinter`	for print queue on named printer.

`lpr files` — Print *files* to default printer.

`lpr -l files`	for `.ps` files
`lpr -Pprntr files`	Print *files* to *prntr*.
`lpr -n files`	Print n copies of *files*
`lpr -Z2up -Pprinter files`	Print 2-up as specified.

See also technical information on WWW at
`http://www.mcs.vuw.ac.nz/technical/`

`lprm jobno` — Remove *jobno* from the print queue.

`lprm -a`	Removes all your jobs.

`ls` — List files in current directory.

`ls -a`	Lists all files, including hidden ones.
`ls -c`	List by creation/modification time.
`ls -F`	Put a slash (/) after each filename if the file is a directory, an asterisk (*) if the file is an executable, and an at-sign (@) if the file is a symbolic link.

`ls -g`	Long format, showing group name.
`ls -l`	List in long version.
`ls -r`	List in reverse order.
`ls -R`	List recursively, including subdirectories.
`ls -t`	List by modification time (newest to oldest).
`ls -u`	List by access time.

`man` *`command`* — Display **manual page** on *`command`* – to get far more detail than is given here.

`mdir a:`*`dirname`* — Display a directory on `a:` (floppy disc).

`mcopy` *`files`* `a:`*`files`*
or
`mcopy a:`*`files files`* — Copies one or more *`file(s)`* to or from a floppy disc.
`mcopy -t a:`*`file(s) file(s)`* converts a **text** file from MSDOS form to unix, changing carriage return/line feeds to line feeds.

`mkdir` *`pathname`* — Make a directory in the pwd.

`mkdir -m`*`mode path`*	Make directory and set the access mode, as in:
`mkdir -m444 mine`	Makes read only dir. called `mine`.
`mkdir -p` *`pathname`*	Create intervening parent directories if they do not exist. i.e. in one step make whole path.

`more` *`filename`* — Write out the contents of the file on the screen, one screenful at a time. ↩ shows the next line, the spacebar scrolls to the next screenful. See also `less`.

`mv` *`name1 name2`* — Move/rename first file or directory to/as second. Will overwrite an existing *`name2`*.

`mv -i` *`name1 name2`*	Move interactively (asking confirmation).
`mv` *`old old_dir`*	Move file to another (existing) directory. If *`old`* is a directory, it is moved to be a subdirectory of *`old_dir`*.

`ps` — Lists processes and gives PID numbers.

`pwd` — Gives path of **p**resent **w**orking **d**irectory.

`rm` *`files`* — Deletes *`files`*. Can use wildcards.

`rm -i` *`files`*	Work interactively (get permission).

`rmdir` *`dirname`* — Remove EMPTY directory(s).

`rm -r` *`dirname`*	Remove whole tree (recursively) from directory.

`sort` *`files`* — Sort lines of files.

`sort -n` *`files`*	Sort numerically.
`sort -r` *`files`*	Sort in reverse order.
`sort -f` *`files`*	Ignore case of letters.
`sort +`*`x files`*	Ignore first *`x`* fields (separated by spaces).

`ssh` *`host`* `-l` *`username`* — Secure login as *`username`* to the specified *`host`*. Allows x-apps to be run over the connection.

If the *`username`* is the same on both machines (hosts), and the entry in the `.rhosts` or `.shosts` files of the machines have been set up correctly, the command can be given as
`ssh` *`host`* ↩

`wc`	Count of lines, words and characters in a file. If a file name is sepcified, the word etc count is done on that file, otherwise the standard input will be used.
`who`	Lists current users.
`who am i`	Gives details of current session.
`;`	Separates commands.

Piped Output

Sometimes it is useful (or just convenient) to pipe the output of one unix command directly into another command. This can be a powerful process, but for a beginner the most obvious uses are:

- To display the contents of a directory containing many files (time to consider sub-directories):
  `ls -l | more` or `ls -l | less`
  which will display the list of files a screenful at a time.
- To display the results of `grep` if the string is found in many places:
  `grep "`*`string`*`" `*`files`*` |more` or `grep "`*`string`*`" `*`files`*` |less`
- To get an approximate word count in a LaTeX document:
  `detex `*`file`*`.tex | wc`

Control characters, different systems and conversions

Unix control characters

`^J`	Linefeed.
`^D`	Return to unix command level—eg end of input to `more`.
`^U`	Erase line.
`^S`	Pause output on screen.
`^Q`	Restart after `^S`.
Other systems	Dos and Mac files have different control characters (for instance, DOS uses `^M` for linefeed). There are instructions you can give that will change the file format from one of these to Unix, or vice versa:
`dos2unix `*`file1 file2`	Convert a DOS file (*`file1`*) to Unix (and store the result as *`file2`*).
`mac2unix `*`file1 file2`	Convert a Mac file to Unix.
`unix2dos` and `unix2mac`	Carry out the reverse process (use the same method of file specification).

Wildcards

`*`	Stands for anything, of any number of characters. `*.tex` means all files with `tex` extension; `*` means **all** files (with or without extensions); `*.*` means all files with an extension.
`?`	Any single character.
`[ ]`	Choice of characters, as in: `Type[Ab].*` — File called either `TypeA.*` (ie any extension) or `Typeb.*`. `Type[1-4]` — File called `Type1` or `Type2` or `Type3` or `Type4`.

FTP

`ftp `*`hostname`	Log onto *`hostname`*, a remote computer. Will be prompted for your password. Make sure that you are in the appropriate directories on both machines—done by starting `ftp` in the appropriate directory. Later changes can be made using:

`cd` *`pathname`*	Changes remote directory.
`lcd` *`pathname`*	Changes local directory (for duration of `ftp` session).
`put` *`filename`*	Puts/copies *`filename`* onto *`hostname`*.
`get` *`filename`*	Copies *`filename`* on the remote user to the local pwd.
`mget` *`dirname`*	Copies all files in the directory (y/n prompt for each one is given).
`mput` *`dirname`*	Puts all files in the directory.
`prompt`	Give this command **before** using `mput/mget`; turn the prompt **off** if it is certain that all files are to be got/put.
`dir`	Lists remote directory.
`lls`	Lists local directory.
`binary`	Copies file(s) to preserve pictures etc.
`help`	See list of `ftp` commands.
`quit`	End session.

1.7 File extensions

Unix filenames can be of any length. Extensions are used to show the type of file:

`.tex`	LaTeX file.
`.aux`	Auxiliary file, created by LaTeX; contains information such as the section headings (for table of contents), labels for cross-reference, etc. Is constructed/overwritten at the end of each successful compilation. See, for example, Figure 9.1 on page 151 of these notes.
`.log`	Record or log of LaTeX compilation—contains useful (and other) error messages, names of graphics files found, and the numbers of lines that were over or under-full.
`.fig`	Graphics file created by `xfig`.
`.eps`	Encapsulated post-script file (has information about the bounding box) that can be included in other files. Typically will have been exported from `xfig`.
`.bib`	Bibliography file.
`.html`	HTML file, used to construct WWW pages.
`.ps`	Postscript file—ready to print to a post-script printer.
`.dvi`	DeVice-Independent file—typeset LaTeX documents are typically written and viewed as `.dvi` files.
`~`	After an extension; denotes a backup file made automatically when the file was opened by an x-application.

The nature of the extension tells the operating system what the file is for. This in turn determines how the file is opened (and the icon used to represent it). If you double-click on the file in the file manager (single-click if you are a KDE user), a suitable application is selected to open the file. If you open the file in `Emacs`, the appropriate mode is selected.

1.8 Applications

There are many applications, and any one user is likely to use only a subset of them. Some that are relevant to these notes are:

Emacs	File editor
`Xfig`	Drawing (graphics)
`Splus`	Statistics package
`Ghostview`	View and print `.ps` files
Netscape	WWW

Invoke an application either from main menu, or by typing, in `xterm`:

appname [filename] &

- Specification of the optional *filename* causes the named file to be [created if necessary and] opened.
- & means that the xterm runs in the background while the application is running, and can therefore be used before the application is closed down.

Note:

You can bring a window to the front by clicking the select (left) mouse button on the (shaded) bar at the top of the window.

KDE users can set an option to bring a window to the front by clicking anywhere in the window (as for a Mac or PC), or to have a more Xwin-type of window environment.

Or, if you have one, by using the Front button on a keypad on the extreme left of the keyboard. Just make sure the cursor is somewhere in the window you want brought to the front before you press Front (this may not work in KDE).

1.9 Some useful Emacs control commands

In what follows, there is much mention of "buffers"—in any one editing session in Emacs, each time you open a file, or get a LaTeX log in a window, or a list of files in a window, you create another "buffer". For example, the buffer list for my editing session as I type (obtained by ^x^b) is:

```
 MR Buffer           Size  Mode         File
---------           ----  ----         ----
.*% *Buffer List*      90  Buffer Menu
  % latexdocs         252  Dired by name /u/staff/edith/latexdocs/
    unix.tex        22065  LaTeX        /u/staff/edith/latexdocs/latexcourse/unix.tex
    notes.tex         483  LaTeX        /u/staff/edith/latexdocs/latexcourse/notes.tex
 *  *~/latexdocs/latexcourse/notes output*  4249 LaTeX
    *scratch*           0  Lisp Interaction
 *  *Messages*       2051  Fundamental
    *ESS*               0  Fundamental
    *TeX background*    0  Fundamental
```

This list contains two buffers that are LaTeX code files (unix.tex and notes.tex); three buffers to do with running Emacs (*scratch*, *ESS*, *TeX background*); two buffers that are/were used in choosing files, the *Buffer List* and latexdocs, the former having been obtained as described above (^x^b) and the latter being a directory listing (like that obtained by typing ls -l in an xterm window—see under ^x d in what follows); two buffers that are messages to you, the user (*Messages*, which contains all the messages that Emacs writes to you and * /latexdocs/latexcourse/notes output*, which is the LaTeX log which you get if you type ^c^l during or after compiling a LaTeX document).

The directory listing is very useful as you can see the names of the files in a directory, as well as all the other information, like the date they were last altered and their size. You can also click (middle or adjust mouse button) on a file to open it in Emacs, or on the name of a sub-directory to change to that, or on the .. indicating the parent directory to change to that.

So you can, in a way, think of a buffer as a file you have been working on in a particular editing session, or some other collection of information that LaTeX or Emacs makes for you. Any file that you have opened remains on the buffer list until you end your Emacs session, or give an explicit "kill" command (^x k).

Customising the Emacs window To change the size of the font in an Emacs window (the default is rather small), hold down Shift and click the left mouse button inside the window, then have a look at the offered choices. You will need to do this each time you open an Emacs window.

More Control + Mouse button options An alternate way to change buffers (see below) is to hold down Control key and click the left mouse button in the Emacs window; slide the highlight down to the buffer of your choice and release the mouse button. Control and middle mouse button lets you change other properties of the window (also accessed on Text Properties on the Edit menu); Control and the

right mouse button enables you to drop in LaTeX commands, typeset or view a document (the same as offered on the `LaTeX` or `Command` menus).

Emacs Commands Most of the commands described here are also provided on one or other of the `Emacs` menus (which also contain some options not given here), and if they are, the corresponding control characters are usually given on the menu, which helps the learning process. In the long run, it is much quicker and easier to use the control characters than it is to depend on the mouse and the menus.

Notational conventions in what follows:

`^x^s`	Keep `^` (Control) down while typing both `x` and `s`.
`^x s`	Release `^` before typing `s`
`Mx` or `esc x`	Meta key (◆ on some keyboards, Alt on x-terminals) held down while `x` is typed.

File operations (`^x` commands)

`^x^f`	Find/create file. Can be used to get a directory listing by specifying the path of the desired (sub-)directory and typing ↩. Note that the final directory name in the path must **not** have the `/` following it. It is possible to open any of the buffers, or to change to any of the listed directories, by clicking on the appropriate name with the middle (adjust) mouse button.
`^x^s`	Save file.
`^x s`	Save some buffers—`Emacs` goes through your buffer list.
`^x^b`	List buffers (see above); it is possible to open any of the buffers in the list by clicking on its name with the adjust (middle) mouse button.
`^x b`	Change buffers (ie switch to another buffer)—the default buffer is the last one to be opened, but you can specify any one by typing its name in the box at the bottom of the screen.
`^x d`	Directory listing. It is possible to open any of the buffers, or to change to any of the listed directories, by clicking on the appropriate name with the middle (adjust) mouse button.
`^x 1`	Show only the `Emacs` window containing the cursor.
`^x 2`	Split the window into two horizontally (can then view two documents simultaneously or view two parts of the same document—good for copy and paste).
`^x 3`	Split the window into two vertically.
	These commands can be repeated in any sequence to make a whole checkerboard of little windows, if that is what you need.
`^x o`	(**o**ther) Move cursor to other window (if there are 2; moves cursor around each window in turn if there are more than 2).
`^x^c`	Quit `Emacs` (can also quit by using `menu` button on bar at top of window).
`^z`	Close `Emacs` window—icon drops to bottom of screen.
`^x^w`	Write (to file)—equivalent to "**save as**".
`^x k`	Kill buffer (ie remove from buffer list).
`^x^q`	Auto-check in and out files—this ensures that if 2 people have write access to a file, only one can edit the file at a time: the one who has checked the file out.
`^g`	Quit current command (if wrong)—very useful!

Editing commands

`^␣` (`^<space>`)	Mark the beginning of a **region** (text to be highlighted etc). Show the end of the text to be marked by positioning the cursor there. This method works well when the section marked is long (several screenfuls).

	Alternately, click at the one end of the text to be marked with the select (left) button and at the other end with the menu (right) button. This method works well for short sections to be marked (within a single screen).
`^w`	Delete marked section—the same as "cut", the text will later be available to "paste".
`Mw`	Write (copy) marked section to cut buffer.
`^p`	Move cursor one line u**p** (↑)
`^b`	Move cursor one character **b**ack (←).
`^n`	Move cursor one character dow**n** (↓).
`^f`	Move cursor one character **f**orward (→).
`Mf`	Move cursor one "word" forward.
`Mb`	Move cursor one "word" backward.
`^v`	Move down one screen (same as page down).
`Mv`	Move up one screen (same as page up).
`^l`	Put line containing cursor in the middle of the screen.
`^a`	Move cursor to beginning of line.
`Ma`	Move cursor to beginning of paragraph—sort of, anyway.
`^e`	Move cursor to end of line.
`Me`	Move cursor to end of paragraph.
`M>`	Move cursor to end of file.
`M<`	Move cursor to beginning of file.
`Back Space`	Delete one character to the left.
`MBack Space`	Delete one word to the left.
`^d`	Delete one character to the right.
`Md`	Delete one word to the right.
`^k` (**k**ill)	Deletes from cursor to end of line.
`^y` (**y**ank)	Drops killed text (or contents of cut buffer, which can contain copied text) at the cursor position.
	Equivalently, use the adjust button (middle one) which drops the text where the cursor is pointed when the button is clicked.
`My`	Drops previously killed text (next up the stack), and in fact, each time you give the `My` command, what was dropped on previous use of `My` is replaced by the text next highest up the stack.
`^x u` or `^_`	Undo. Giving the command repeatedly causes all earlier editing changes to be undone, one at a time, scrolling right up the stack.
`^q`	Puts in line breaks and indents—makes raw LaTeX look "pretty" and structured.
`^u 20 ^x f`	Sets width of `Emacs` window to 20 characters (brought into effect by `Mq`)—or any other no. of characters that suits you.
`^u 70 ^x f`	Sets the width to 70 characters, the usual number.

Commands useful for LaTeX

`^c^c`	Typeset current document (if you are correctly set up), or View current document (if it has been typeset correctly or a `*.dvi` file exists). Equivalent to selecting `LaTeX` or `View` on the `Command` menu. The action can be changed in the echo area at the bottom of the screen (or minibuffer) if the default is not what you want to do.
`^c ;`	Comment out highlighted text.
`^c :`	Uncomment highlighted text.
`^c^l`	See the abridged version of the `*.log` file for a file that has just been (or is being) typeset by LaTeX.
`^c^e`	Insert an environment.
`^u^c^e`	Change the environment in which the cursor is positioned.

Rectangles

In Emacs it is possible to operate on rectangles of space that span parts of several lines—a rectangle is marked, just like a region, at the beginning and end. The difference is that the special rectangle instructions are applied only to the column(s) and row(s) that lie between the beginning point and the end point (instructions for regions apply to all characters between the beginning and end points, across all "columns"). This is very useful when the file contains, say, data of some sort in regular columns, or even to eliminate or insert characters at the beginning (or end, if all lines are of the same length) of some lines in a file. Once the rectangle has been defined, it can be copied, deleted, or filled with space or text.

^␣	Mark the start of the rectangle.
Position of cursor (click with left mouse button)	Mark the end of the rectangle. Should be at least 1 column over and 1 line down from the start.
^x r d	Delete rectangle. (To get it back, must use Undo (^_).) Text to the right of the rectangle is shifted left by the width of the rectangle.
^x r k	Kill to buffer—in other words, delete and copy contents to where they can be yanked back (next instruction).
^x r y	Yank last (last rectangle that was killed to buffer). Position cursor at top left corner of where buffer contents are to be placed.
^x r t	Fill with string. No matter how wide the rectangle is defined to be, the number of columns inserted will be equal to the length of the string that is inserted in each line, left justified, in the rectangle. Text to the right of the rectangle is shifted to the right by the length of the string.
^x r o	Fill with blanks. The number of blanks inserted is determined by the width of the rectangle defined. Text to the right of the rectangle is shifted to the right by the width of the rectangle.
^x r c	Clear rectangle. Replace characters in rectangle with spaces. Text to the right of the rectangle does not shift.

Changing case

Mc	Put text from the cursor, to the right, up to the next space, into sentence case (eg Xvse).
Ml	Put text from the cursor, to the right, up to the next space, into lower case.
Mu	Put text from the cursor, to the right, up to the next space, into upper case.

Changing mode

Mx	Cursor goes to echo area, at the bottom of the window. Type the required mode (can use Tab to complete):

text mode	Good for text.
fundamental	OK for most Emacs applications.
auto-fill-mode	Turns on wrap, breaking between words.
LaTeX-mode	Best for LaTeX–can compile and view documents from within Emacs.

^h m	Get help on modes.
	Emacs selects the mode from the extension used in the file name, so that in fact it is seldom necessary to change modes.

Searching

^s	Forward search—searches forwards on each character, as it's typed in.

`^r`	Reverse (backwards) search. See also the `Search` options on the `Search` menu.
Control characters	Sometimes you have a file that contains control characters inserted by another operating system (DOS file) or application (SAS output) that contains control characters (like `^M` or `^L`). You can tell that the `^M` that appears in your file is a control character, as a single `BackSpace` will delete both the `^` and the `M`. You can search for these characters by entering `^q` before `^M` (or whatever), so that to request a forward search for `^M`, you would enter the sequence: `^s ^q ^M`. You can replace, say, `^M`s with ↩s, using the `Query Replace` option on the `Search` menu, in much the same way.

For Help on any topic, explore the possibilities offered by the `Help` menu in `Emacs`.

Chapter 2

Overview of LaTeX

This chapter covers broad issues (What is TeX or LaTeX? What are packages for? What do you do to typeset a LaTeX document? To see what it looks like? To print it? And, fairly critically, if there is something wrong with it, what might that be, and how to find where and what it is?). The next chapter deals with the actual commands, declarations and environments available in LaTeX, and how to get basic control over the appearance of a document.

2.1 The TeX family

The document preparation software that can be considered to make up the TeX family consists, in broad terms, of:

- TeX–not very user-friendly.
- LaTeX, built on top of TeX, is more user-friendly. It is a "structured" mark-up language, in which LaTeX **commands** and **declarations** (both of which operate on text enclosed in { ... }s) and **environments** (`\begin{`*env*`} ... \end{`*env*`}`) specify the formatting desired.
- Packages (`*.sty` documents) expand the options, by providing additional commands or environments, or control of the appearance of the typeset text in some way (typically by changing the values of the various space and measurement variables, or the default settings for fonts, or sectioning etc. specifications).
- Among the packages, special enough to warrant an individual mention, are amsmath (which used to be called "amstex"[1]) and amssymb, which are the result of the American Mathematical Society project to extend the "vanilla" LaTeX range of mathematical commands and environments. They aim to provide almost all the symbols you could dream of, and to enable you to typeset mathematics more easily and in an aesthetically pleasing way.

  A fair amount of "on-line" documentation is available, giving greater and more technical detail than is provided here. See page 161.

2.2 Documents

Documents consist, on the whole (certainly the "simple" ones) of a preamble and a body.

The preamble contains specification of:

- document class;

[1] But (in a way dear to the heart of every British comediperson) this was a misnomer, as there was a different version of TeX with the same name ($\mathcal{AMS}$-TeX), and use of the "amstex" package would actually give you $\mathcal{AMS}$-LaTeX. This confusion is slightly reduced with the new name.

- packages (these are add-on bits that enable you to do more than you can in "vanilla" LaTeX) to be used (in the `\usepackage{...}` command);
- changes to the default page dimensions, such as: `\textwidth`, `\textheight`, `\parskip`, `\parindent`
- declarations of new commands and environments that will be useful for your document.

The body (everything inside `\begin{document}...\end{document}`) contains the text and formatting commands and environments of the actual document being produced.

2.3 Document Classes

For full description, see page 176 of Lamport[2]; more details are given in Section 7.2, page 110 in these notes.

Each different document class is set up to be most suitable for a particular type of document—a journal article, a book, a letter, overhead transparencies, even an exam paper.

The types of things that vary from class to class are the sizes of the pages, the "decoration" in the header, the position of the page number, the types of sectioning commands (if any) and any standard bits of text that will appear in such a document with a particular format (for instance, the title and author, or address and closing in a letter) and default settings that are most suitable for a particular class of document.

- Locally available document classes include:
  `article`, `book`, `report`, `letter`, `slides`, `seminar`, `vuwletter`, `vuwexam`, `isorexam`
- The declaration has the form

  `\documentclass[`*`options`*`]{`*`class`*`}` where the *`options`* include:

  - `10pt, 11pt, 12pt`—specification of the font size for the document, with default of `10pt`.
  - paper size—not really necessary
  - `landscape`
  - controls of how books are formatted for printing—1-sided or 2-sided printing; whether chapters are constrained to start on right-hand pages
  - `leqno`—put equation numbers on the left of the page (default is on the right)
  - `fleqn`—left-align displayed equations (default is that they are centred).

  See also Section 7.2.

2.4 Packages

Some of the more useful packages include:

amsmath	Provides access to more mathematics environments, fonts and commands.
amssymb	More mathematical symbols than in "vanilla" LaTeX.

[2]Leslie Lamport, *LaTeX User's Guide and References Manual*, Addison-Wesley Publishing Company, 1994.

dateonbottom	Prints an identifier string (that is specified by a `\setftnotestring{`*string*`}` command in the preamble) on the LHS of the page footer, the page number in the centre, and the date on the RHS. Useful to distinguish between versions of the document, and to remind you where you saved the file (if the string is the path) or whatever else you need to record. Is "activated" by the command `\pagestyle{plain}` in the preamble.
graphics or graphicx	Allows geometric transformations of text (scale and rotate) and graphics (`eps`) files. Alternative (older) packages for placing graphics files are epsf and epsfig.

Many more packages are detailed in Goossens, Mittelbach and Samarin (GMS)[3] and in documentation files that are available, see page 161.

2.5 LaTeX using Emacs editing environment

`Emacs` provides a very helpful environment for editing raw LaTeX files. The `LaTeX` menu at the top of the window offers many LaTeX-specific things that you may want to do, and the `Command` menu offers the means to compile and view the document from within the `Emacs` environment.

Adding and changing LaTeX environments

On the `LaTeX` menu (use right mouse button to drop) are the options to insert an environment, or to change an existing environment. This obviates the need to type out, say, `\begin{description} ... \end{description}` laboriously each time. And avoids spelling mistakes. If you prefer, use control characters, as shown on the drop-down menu:

`^c^e`	Insert an environment. The default is `itemize`; type the one you want in the echo area (box at the bottom of the screen). You can use `Tab` to complete the environment name.
`^u^c^e`	Change the environment in which the cursor is positioned. The name of the new environment is entered as above. Very useful when you change your mind about the effect you wanted to create in a document.

2.6 Typesetting and viewing

This can be done either from within `Emacs` or from the `xterm` window. You'll soon find which way suits you best.

2.6.1 From Emacs

1. On the `Command` menu (use right button to drop), choose `LaTeX` (or `LaTeXInteractive`, which stops at each error and then starts again—you'll find which suits you the best). This will cause the document to be typeset, and the `*.aux` and `*.log` files to be put in the same directory as the `*.tex` file, together with the `*.dvi` file if at least one page was typeset successfully.

   Or use the `^c^c` shortcut, which should cause `Command: (default LaTeX)` to appear in the echo area. If one of the other options (say, `View`) appears as the default, type `L` in the echo area and use TAB to complete `LaTeX`.

2. It's a good idea to watch for errors in the `*.log` file. Type `^c^l` to see an abridged version of the log.

3. To **view**, select `View` from the `Command` menu. The name of the `*.dvi` file just created (or associated with the current `*.tex` document) appears in the echo area (box at bottom of window). Press ↩ to accept this file (or if you wish to see the `*.dvi` file of another document, specify the appropriate name).

[3] Goossens, Mittelbach and Samarin, *The LaTeX Companion*, Addison-Wesley Publishing Company, 1994.

Again, the short cut `^c^c` should have the same effect as `View` on the menu. If there were errors in the compilation, the default action offered on `^c^c` will be `LateX`, but if you want to see the file anyway, type `V` in the echo area and use TAB to complete `View`.

2.6.2 From `xterm`:

1. To keep all the files associated with the file you're working on in the same directory, make sure your pwd is that of the `*.tex` file (use `cd` to change directory).
2. Type `latex` *`filename`* ↩.
3. The abridged log appears in the `xterm` window.
4. To view the typeset document, type `xdvi` *`file`*`.dvi &` ↩. Use of `&` means that you can give more commands in the `xterm` while the `Xdvi` window is still open.

If you leave the `Xdvi` window open while you make any improvements or additions necessary to your document, the updated version can be seen by clicking on the `Xdvi` window after you have edited and retypeset your document.

2.7 Error Messages

It is almost certain that the first time you try to run LaTeX your file will contain some errors. LaTeX will tell you about them in the `.log` file. The message will give an indication (it may not be particularly good or useful) of what may be wrong, and will give the line number of the line where the defect was noticed (although in errors such as "missing }" or "missing $", the actual offence may have been committed several lines earlier).

Errors are discussed in some detail in Lamport on pg 28–32 (what might have gone wrong) and pg 134–147 (Error Messages), and a detailed discussion is not repeated here.

The most common errors (which seem catastrophic at first, but which become really easy to find and correct with the inevitable practice) are:

- Missing }—you started to demarcate some part of the document and forgot to reach an end. A missing } in a sentence at the end of, say, a declaration (such as `\bf` or `\em`) may give you more emphasised or bold text than you anticipated, and may give a warning of

  ```
  (\end occurred inside a group at level 1)
  (see the transcript file for additional information)
  ```

  at the end of the `.log` file, but will not cause the LaTeX run to be halted. The kind of missing } that causes problems is that in a command, say, in a maths environment, as in `$\frac{ab}{c^2$`, where one of the pairs of braces required for the command (`\frac`) is incomplete. In fact, any required braces (containing the argument(s) for a command or environment) that are not closed will cause problems.

- An unknown command or environment—usually a result of a mis-spelling, or of a failure to specify the use of a package [for instance, neither `amsmath` nor `amssymb` is specified in the `\usepackage` command, but one of the commands (say `\mathbb`) or environments (say, `align`) is used in the document], or a failure to remember to `\input{gems}` (see page 73).

- Use of any one of `# $ % & _ { } ^ \` as if it were an ordinary printing symbol (see page 30). Use of `$` is fairly easy to pick up as it makes the typeset version look funny, and use of `%` is equally clear (but more puzzling) as it causes whole bits of a sentence to "disappear". Typically just after mention of a percentage ...

  It's especially important to note that `^` and `_` can only be used in a mathematical environment; and `&` in a `tabular` or `array` or other aligned environment. And it's useful to note that `>`, common in email replies, can also only be used in a mathematical environment; `> Dear` will appear as "¿ Dear" in the typeset version.

- Improperly nested environments: the latest environment named in a `\begin{env}` command must be `\end{env}`-ed before any other previously "begun" environment is ended. In the same way, delimiters, like `\( ... \)` and `\[ ... \]` (used to delimit maths environments) must be properly matched.
- A missing command argument—most commands (and many environments) have required arguments, which will cause an error if they are not supplied or are incorrectly supplied, mainly because LaTeX assumes the next character to be the argument, and, if expecting a number, gets terminally confused by a letter or special character. [There are also, often, optional arguments that give no problems if omitted.]

2.8 Printing the typeset document

Printing can be done from the application `Ghostview`, or `gv`, or directly from the `xterm` window. Sometimes the one method is more suitable than the other – each has some particular advantage(s). `Ghostview` and `gv` can be used to print any postscript file.

2.8.1 To prepare a `.dvi` file for printing:

1. To keep all the files associated with the file you're working on in the same directory, make sure your pwd is that of the `*.tex` file (use `cd` to change directory).
2. Before a file can be printed, it can be helpful to convert from `.dvi` form to postscript (`.ps`). This is particularly true if you want to print out some pages and not others, or if you've used special effects not seen in the `.dvi` file (rotated text, some changes of scale, etc.)—it is safer to check the `.ps` file to see that they worked before printing. In the `xterm` window, type `dvips` *`filename`*`.dvi`↩.
3. Alternately, choose Print on the Command menu in Emacs. This prints directly from the `.dvi` file (and does not cause a `.ps` file to be created).

2.8.2 To print using `Ghostview` or `gv`:

1. In the `xterm` window, type: `ghostview` *`filename`*`.ps` ↩
   (or `gv` *`filename`*`.ps` ↩). In fact, it is better to type `ghostview` *`filename`*`.ps &` ↩(or `gv ... &` ↩) as you can then leave the `Ghostview` (or `gv`) window open with the `xterm` window running in the background—see page 17.
2. In the `Ghostview` window that will open, you can view any page by clicking on the number of that page with the adjust (middle) mouse button (in `gv` it's the select (left) mouse button, and you can scroll through the document using ≪ and ≫).
3. It is possible to select a subset of the pages for printing: highlight these pages with the select (left) mouse button; on the `Page` menu select `Mark`; on the `File` menu, select `Print marked pages...` (in `gv` the adjust (middle) button highlights marked pages, then click the left button on Print Marked).
4. To print the whole document, on the `File` menu, select `Print ...` (Print All in `gv`).
5. From "ISOR" machines, it's possible to control, using the `iprint` command (`iprint` *`filename`*`.ps` ↩ in an `xterm` window):
   (a) the printer used
   (b) how the pages are printed [Portrait (full page) or 2-up or 4-up]
   (c) other options [Draft (a large DRAFT is printed across the page); a landscape-printed identifier specified by the user printed in the top right and bottom left side margins; provision for Name and Tut. Gp. No. to be handwritten by students on the first page of test or tutorial papers.]

   These options are not all available on "COMP" machines.

2.8.3 To print using the `lpr` command:

This command can be used to print files of any type: `lpr` *`filename`* ↩ is the basic command to print the named file to the default printer. The command can be varied along the following lines (this is not an exhaustive list):

1. `lpr -P`*`printer filename`* ↩ prints the file to the named printer.
2. `lpr -d -P`*`printer file`*`.dvi` will convert the `.dvi` file to postscript and print it (without making a `.ps` file).
3. `lpr -Z2up -Ppress` *`file`*`.ps` ↩ will print the named postscript file to the printer "press", and it will be printed out 2 pages per page (2-up) and duplexed (backed), which is the default on Press.
4. `lpr -Z2up,noduplex -Ppress` *`file`*`.ps` ↩ does the same, but prints singled-sided pages.

Printing in this way gives control of formatting from "COMP" machines, and gives control of default duplexing on Press.

Duplexed printing on Press is either "longbind", ie the pages should be stapled or punched on the long side of the page (left if 1-up, top if 2-up), or "shortbind", ie the pages should be stapled or punched on the short side of the page (top if 1-up, left if 2-up). The default is longbind for 1-up printing and shortbind for 2-up printing. So a longbind, 2-up job would be specified:

`lpr -Z2up,longbind -Ppress` *`filename`*`.ps`↩

2.8.4 Printing as a booklet

A clever trick from "COMP" to print a postscript document in page-order suitable for reproduction as a booklet is achieved by typing the following in an `xterm` window:

`psbook "`*`name`*`.ps" > `*`newname`*`.ps` ↩
`lpr -Ppress -Zshortbind,2up` *`newname`*`.ps` ↩.

Note that Press is the only printer that can be used to do this as duplexing is assumed in the formatting process.

Chapter 3

First steps in LaTeX

3.1 Introduction

The first thing to note about LaTeX is how it treats (multiple) spaces and carriage returns ("Enter"s). It ignores the second and following spaces between words, and all spaces within a mathematical environment (it has its own rules for spacing "maths", see Section 4.6). It treats the first carriage return as a space; the second is taken to indicate a new paragraph (equivalent to `\par`) and the third and following carriage returns are ignored.

If you want longer spaces within lines or between them, you need to specify this using one of the several suitable commands (see Section 3.8).

3.2 LaTeX Commands and Declarations

LaTeX commands produce some output, like `\quad`, which leaves a certain sized space, or `\,`, which leaves a much smaller space, or `\LaTeX` (LaTeX), `\LaTeXe` (LaTeX 2_ε) and `\today` (December 5, 2000), or `\textbf{bold}` which produces **bold** (note that the appearance of the contents of { ... } is changed).

Declarations do not produce output, but do affect the appearance of the output (like `{\em emphasized}` which produces *emphasized*). See section 3.7.

Commands can be given as *control words*, which consist of a backslash followed by one or more letters (no numbers or symbols), optionally followed by one or more arguments, each in {}. Or they can be given as a *control symbol*, which consists of a backslash and a non-alphabetic character. For instance, `\quad` and `\LaTeX` have no arguments, `\textbf{bold}` has one argument, and `$\frac{10}{11}$` has 2 arguments, `\,` is a control symbol.

Declarations look like control words (but are **not** followed by arguments).

Declarations can be either global or local. A global declaration applies to the whole document from the point at which it appears, or to the page on which it appears. LaTeX's global declarations are `\addtocounter`, `\hyphenation`, `\newcounter`, `\newlength`, `\newsavebox`, `\newtheorem`, `\pagenumbering`, `\setcounter`, `\thispagestyle` (only `\thispagestyle` is restricted to the page on which it appears).

A local declaration applies to only a local "bit" of the document, with a "bit" being any one of:

- Text within {}. For instance: the {`\sl slanted text`} (*slanted text*) is {`\bf not \rm the same as \em emphasized text`} (**not** the same as *emphasized text*). Is it?
- Text within an environment (see Section 3.3)—the effect of any local declaration will cease at the next `\end{`*env*`}`.
- Text within a "cell" of an `array` or `tabular` environment (see page 80)—ie text between an `\\` and `&` (in either order), or two `&`'s, in such an environment.

Note that it is impossible to nest declarations that change the appearance of the text—each new declaration replaces that which went before.

Spaces are ignored after a command so that any punctuation mark following the command will be

correctly placed, but this means that care must be taken to ensure that the typeset version, using LaTeX, is correct. Compare `\TeX and \TeX\ and` (TEXand TEX and)—the space after the command must be put in when it is needed. See Section 3.8.

LaTeX users can define their own commands to make their life easier. Easy general examples that spring to mind are `\nz`, New Zealand, and `\maori`, Māori, which can be defined as:

```
\newcommand{\nz}{New Zealand}
\newcommand{\maori}{M\={a}ori}
```

where the first argument (in the first { ... }s) is the name of the command, and the second contains what must appear in the typeset version.

3.3 LaTeX environments

An environment is a (usually longish) section of text to be formatted in some special way. An environment has its beginning and end shown by a `\begin{`*env*`}` and `\end{`*env*`}`, respectively.

Have a look in the Index (under "environment") to see a fairly full list of environment names, many of which are self-explanatory.

A simple example of an environment is the `center` environment:

Centred text with several different faces.

Note that a local declaration inside an environment does not affect any text outside the environment.

```
\begin{center}
\Large\bf Centred text with several
different faces.

\large\em Note that a local declaration
inside an environment does not affect
any text outside the environment.
\end{center}
```

The text here is not affected by any changes made within the section of text above.

3.4 Dashes, quotes, special characters and accents

Text often needs embellishment in some way—short or long dashes between words or numbers (or even within words), other authors need to be quoted, and LaTeX has its own ideas about what is or is not a special character, and where such things can be used (*in* a maths environment, or *outside* one?).

3.4.1 Dashing and quoting

See Lamport pg 170–171 for details, but, typically, a single `-` is used as a hyphen (or for subtraction in a mathematical environment), a double hyphen, `--`, is used for a fairly long hyphen or to indicate a range of pages, as in 10–20 (compare with 10-20), and a triple hyphen, `---` is used for a really long hyphen—in a sentence.

Usually, LaTeX hyphenates words really well, but occasionally you need to use a word that baffles its set of rules, or you use a long word in a place where the algorithm is particularly reluctant to hyphenate (like near the beginning of the statement of a theorem) and needs some encouragement. An optional hyphen is indicated by `\-`.

For quotes, note that ` is used for left quote, and ' for right quote. Typically `` gives double left, and '' double right, but `Emacs` will insert double quotes correctly if you use " (≡ `Shift '`) preceded by a space and followed by a character at the start of the quotation, and a character followed by " followed by punctuation or a space at the end of the quotation.

3.4.2 Special characters

The characters `% \ $ # { } _ ^ ~ &` are all special characters, and have special uses. @ can also be tricky, as it's used extensively in TEX.[1]

`%`	Comment—ignore all material to the right of the sign. `Emacs` has very useful block comment/uncomment functions, which allow you to comment out several lines of LATEX coding (either to edit it out, or to help you to locate a mistake). Highlight the lines to be commented out, then look on the LaTeX menu, then on the Miscellaneous sub-menu (or type `^c ;`).
`\`	Used to show that a LATEX command is beginning.
`$`	Used to define a text-style mathematics environment (an in-line formula).
`#`	Used when specifying variables in new LATEX commands, as in: `\newcommand{\mc}[1]{\ensuremath{\mathcal{#1}}}`.
`{ ... }`	Used to define a section of text which is to be given some special treatment.
`_`	Forms subscripts in a mathematical environment.
`^`	Forms superscripts in a mathematical environment.
`~`	A non-breaking space.
`&`	Used to separate columns in `tabular` and `array` and similar (mathematical) environments of aligned columns.

|, <, > can only be used in a mathematical environment (or as in `|, <, >`).

```
$|, <, >$ can only be used in a
mathematical environment (or as
in \verb1|, <, >1).
```

```
The  special characters ....
by a backslash:

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

The special characters that are not specifically mathematical can be used in text if they are preceded by a backslash:
% $ { } _ # &

Also useful are:

`\dag`	(†)	`\ddag`	(‡)	`\S`	(§)
`\P`	(¶)	`\pounds`	(£)	`\copyright`	(©)
`\textperiodcentered`	(·)	`\textvisiblespace`	(␣)	`\textbullet`	(•)
`\textcircled{a}`	(ⓐ)				

And `\-` defines an optional hyphen.

Note that `\bullet` works only in math mode, but `\textbullet` works in text (LR or paragraph) mode (see section 3.10). `\textcircled` provides a circle of fixed size, so it may be necessary to make a capital letter or number small enough to fit in the circle as in `\textcircled{\scriptsize{A}}` or `\textcircled{\scriptsize{9}}` (Ⓐ and ⑨ compared with the full-sized equivalents of Ⓐ and ⑨).

3.4.3 Accents and non-English letters

The commands to make accents and non-English letters are shown in Table 3.1.

These commands only work in paragraph or LR mode (see Section 3.10 for description of "modes").

Note that the brackets are not really necessary where the argument consists of a single character, particularly where a control symbol is used. In the case of a control word, there must be a space between the control word and the (single-character) argument. For instance, `\=a` and `\={a}` are equivalent, as are `\c{s}` and `\c s` (ā, ā, ş, ş).

For maths mode (in a mathematical environment), the available accents are shown in Table 3.2:

[1] In vanilla LATEX, @ works just fine (although it has a special use in an `array` or `tabular` environment). It also works fine when using `amssymb` or `amsmath` packages. But when using `amstex` package, to get @, you need to type `@@`. See also Section 8.6.

Table 3.1: Accents and Non-English Letters

\‘{a}	à	\’{a}	á	\^{o}	ô	\"{o}	ö
\~{n}	ñ	\=a	ā	\.o	ȯ	\u{c}	č
\v{c}	č	\H{f}	f̋	\t{oo}	o͡o	\c{s}	ş
\d{o}	ọ	\b{a}	a̲	\oe	œ	\OE	Œ
\ae	æ	\AE	Æ	\aa	å	\AA	Å
\o	ø	\O	Ø	\l	ł	\L	Ł
\ss	ß	?‘	¿	!‘	¡		

Table 3.2: Accents in Mathematics

\hat{a}	$\hat{a}$	\widehat{2-x}	$\widehat{2-x}$	\check{a}	$\check{a}$
\breve{a}	$\breve{a}$	\acute{a}	$\acute{a}$	\grave{a}	$\grave{a}$
\tilde{a}	$\tilde{a}$	\widetilde{a+5}	$\widetilde{a+5}$	\bar{X}	$\bar{X}$
\overline{X-Y}	$\overline{X-Y}$	\overline{X}	$\overline{X}$	\underline{x+4}	$\underline{x+4}$
\vec{y}	$\vec{y}$	\dot{a}	$\dot{a}$	\ddot{a}	$\ddot{a}$

`\widehat`, `\widetilde` and `\overline` all produce wider “accents” than their narrower counterparts (`\hat`, `\tilde`, `\bar`), and will “spread” to cover a short expression.

`\underline` is the only one of these commands to work in paragraph and LR mode, too.

See also Section 4.5, page 46 of these notes.

If you need to use one of the accents in Table 3.1 in math mode, enclose the command in an `\mbox{...}`. For instance, `$x+\mbox{\r{a}}$` ($x+$ å).

Note that in math mode, if you need an accent over i or j, you can suppress the dot by using `\imath` or `\jmath`, as in `$\vec{\imath}$` ($\vec{\imath}$).

3.5 Justification

The default is for text to be left and right justified. This can be changed using either a declaration (`\raggedright`, `\centering`, `\raggedleft`) or an equivalent (list-like) environment (`flushleft`, `center`, `flushright`). The biggest difference between the effect of the declaration and its matching environment is in the amount of white space before and after the affected text (there is typically more white space before and after the environment). A blank line before and/or after the declaration will increase the amount of white space.

`\raggedright`, `\centering`, and `\raggedleft` don't always work as you'd expect: they work best used inside other environments, such as one of the parboxes like `minipage` or in a `p` column in an `array` or `tabular` environment.

Some text in a `minipage`
that is `\raggedleft`.

Centered text (and note that the environment is spelled the American way).

Next line still ragged left.

```
\begin{minipage}{0.45\linewidth}
\raggedleft Some  text in a
{\tt minipage}\\
that is \verb1\raggedleft1.
\begin{center}
  Centered text (and note that
  the environment is spelled
  the American way).
\end{center}
Next line still ragged left.
\end{minipage}
```

```
\begin{flushright}
  Flushed right and ragged left.
  As can be observed from these few
  lines, so long as there are
  more than two lines of text.
\end{flushright}
```

Flushed right and ragged left. As can be observed from these few lines, so long as there are more than two lines of text.

Flushed left and ragged right. As can be observed from these few lines, so long as there are more than two lines of text.

```
\begin{flushleft}
  Flushed left and ragged right.
  As can be observed from these few
  lines, so long as there are
  more than two lines of text.
\end{flushleft}
```

3.6 Sectioning commands

All text headings of parts, chapters, sections, etc. should be made with sectioning commands (not "by hand") to gain the following advantages:

- All headings of the same level have exactly the same format (which, for some or all of the document, can be changed, see Section 8.6).
- The same amount (almost) of extra white space is added above and below all the headings of each level.
- A new section (or whatever) will never be started at the very bottom of a page (if there is not enough room for a certain number of lines of text after the heading, a new page is started).
- There is (optional) automagic numbering of the sections.
- Section headings can be used in page headers (especially in `book` class documents).
- There is automatic inclusion of a (specified) number of levels of section heading in a Table of Contents.

The sectioning commands are:

```
\part       (optional)
\chapter    (not available in article class)
\section
\subsection
\subsubsection
\paragraph
\subparagraph
```

NOTE that if the headings are numbered you should use a higher-level command before using a lower-level command. For instance, you can't have a `\subsection` without having `\section` (otherwise, you might get a subsection called "Introduction" that is numbered: **0.1 Introduction**).
But you can use the starred versions in any order (no numbering to get confused!).

A `*`, as in `\subsection*{ ... }`, causes the section title to be displayed in the same font as the unstarred version, with the same space above and below it, but without a number, and without a table of contents entry being made.

These commands have the general format of, for instance: `\section[`*`listentry`*`]{`*`heading`*`}` where *`heading`* appears in the text at the start of the section, and *`listentry`* appears in the page header (if this is used) and in the table of contents (typically, *`listentry`* is used if *`heading`* is too long to fit into a single line). Both *`listentry`* and *`heading`* should contain only robust commands (see Table 8.1 for a list of fragile and robust commands).

For example, this section was headed: `\section{Sectioning commands}`, the next was headed: `\section{Special Effects}`, with subsection `\subsection{Face, shape and family}`. Chapter 8, which has a long heading, was specified as:
`\chapter[Fragility, Counters and Modifying Appearances]{Fragility, Counters,%`
`Footnotes, Landscaping and Modifying Appearances}`.

3.7 Special Effects

All of these special effects apply only to paragraph-mode text. Equivalent effects in a mathematical environment can be obtained using different commands and environments (see page 73).

3.7.1 Face, shape and family

Text delimited by `{ }`s can be made to be **boldface** series (`{\bf boldface}`), or the default of medium series. And to have either an *italic* shape (`{\em italic}`), *slanted* shape (`{\sl slanted}`), SMALL CAPS shape, (`{\sc Small Caps}`) or the default, upright shape. And can be of the `typewriter` family (`{\tt typewriter}`), sans serif family (`{\sf sans serif}`), or default roman family.

The commands (but not the declarations) determining the face, shape and family can be nested to get combined effects, such as ***bold slanted***, or **bold sans serif**, or *`typewriter slanted`*.

LaTeX offers two ways of specifying the series, face and family (see Section 3.2 for the difference between commands and declarations, and the scope of the text affected by a declaration):

1. As a **command**:

`\textbf{`*text*`}`	`\textnormal{`*text*`}`
`\textmd{`*text*`}`	`\textsl{`*text*`}`
`\textit{`*text*`}`	`\textup{`*text*`}`
`\textrm{`*text*`}`	`\texttt{`*text*`}`
`\textsc{`*text*`}`	`\textsf{`*text*`}`

   Use the commands to nest the effects. (`\texttt{\textsl{typewriter slanted}}`)

2. As a **local declaration**:

`{\bf ...}`	`{\normalfont ....}`
`{\mdseries ...}`	`{\sl ... }`
`{\em ....}`	`{\it ....}`
`{\rm ... }`	`{\tt ....}`
`{\sc ...}`	`{\sf ...}`

3.7.2 Size

Text size is changed using the local declarations:

`\tiny`	`\scriptsize`	`\footnotesize`	`\small`
`\normalsize`	`\large`	`\Large`	`\LARGE`
`\huge`	`\HUGE`		

The sizes used are proportional to the chosen font size (declared as an option in `\documentclass`). This is tiny (`{\tiny tiny}`) and this Large (`{\Large Large}`).

The segment of text to be of the specified size is delimited by `{ ... }`.

If the delimiters (`{ ...}`) are omitted, the special effect applies to all text from that point onwards, until it is changed again.

Superscripts in text can be made using `\textsuperscript{`*text*`}` as in `4\textsuperscript{th}` (4^{th}). The superscript is formatted in the current text font, not in a maths font (compare 4^{th} with `$4^{th}$` which gives 4^{th}).

There is **no** corresponding text subscript.

3.8 Considerations of space

3.8.1 Control of horizontal space

A number of control words and control symbols can be used to make a space (of somewhat variable size) within a line:

`\␣` Inter-word space. Should be used (`...␣with␣e.g.\␣etc.\␣in␣...`) with e.g. etc. in a sentence (compare that with the use of etc. in the same way—the second etc (`...␣of␣etc.␣in␣...`) should be followed by more space, but exactly how much will depend on what else is in the line).

`\,` Small space (as in " ... at last he said 'I won't'" ``‘‘␣\ldots␣at␣last␣he␣said␣‘I␣won't'\,'' `` compared with " ... he said 'I won't"' ``‘‘\ldots␣he␣said␣‘I␣won't''' ``).

`\:` Medium space (thicker than a thin space, but thinner than a thick space).

`\;` Thick space (a bit smaller than a typical inter-word space).

`\!` Negative thin space (moves text to the left by about the same amount as `\,` moves text to the right; about 2 such spaces are almost equivalent to a typical inter-word space).

`\quad` Space equal to the em value of the font (see page 36).

`\qquad` Space twice as wide as a `\quad`.

`~` Non-breaking space. Particularly useful in references, especially at the end of a paragraph, as in: "page 40" (`page~40`).

`\@` Suppose a sentence ends this way: ... Section B. The problem is that LaTeX believes that sentences cannot end with a capital letter. Compare with: ... Section B. There should be a noticeably bigger space the second time (`... Section~B\@. There ...`).

The commands `\␣`, `\,`, `\:`, `\:`, `\!`, `\quad` and `\qquad` can be used in paragraph (ie in text) and maths (ie in a mathematical formula) mode.

A larger chunk of text, or an in-line formula that should not be broken can be enclosed in `\mbox{ ...}` to prevent it being broken across two lines. This may cause the "box" created to stick out into the right margin, but this can be prevented putting `\linebreak` before the `\mbox`.

3.8.2 Space adjustment in and between sentences

LaTeX is almost perfect most of the time, but there are times when the default spacing can seem suboptimal to the purist who wants to do something out of the ordinary. These cases, and the corresponding fixes, are summarised in Table 3.3.

Table 3.3: Special-case spaces

Do what?	Fix	Example of Default	Example of Improvement
Command in sentence.	`\␣`	LaTeXis wonderful	LaTeX is wonderful (`\LaTeX\␣is␣...`)
Lots of quotes	`\,`	" ... 'I won't' "	" ... 'I won't' " (``‘‘\ldots‘I␣won't'\,''``)
Lonely numbers (non-breaking space)	`~`	... page 40	... page 40 (`page~40`)
Sentence ending in uppercase letter	`\@`	... Part II. Next ...	... Part II. Next ... (`\ldots␣Part␣II\@.␣Next␣\ldots`)
Unwanted linebreak	`\mbox`	put all this on one line	put all this on one line (`put \mbox{all ... line}`)
	`\linebreak`	May be needed to break the line before the `\mbox{...}` to prevent the latter sticking into the margin. Add to **final** final draft of document.	
Italicised word in sentence	`\/`	without *italic* correction	with *italic* correction (`{\em␣italic\/}␣correction`)
		This correction is not needed when the italicised text is followed by `.` or `,`. It is needed when italicised text is followed by `:` or `;` or a quotation mark or `---` (emdash) or `?`. This correction is needed for italicised text that is made by	

Continued on next page

Continued from previous page			
Do what?	Fix	Example of Default	Example of Improvement
		a **declaration** (`\em` or `\it`), but is *not* needed for italicised text made by a command (`\textit{ ... }`).	
Midsentence `. ? ! : also .) .'' :''`	`\␣`	For eg. in this case (and etc.) where " ... words b.t.w." is a quote.	For eg. in this case (and etc.) where " ... words b.t.w." is a quote. `eg.\␣in ... etc.)\␣in ... b.t.w.''\␣is a ...`

3.8.3 Control of vertical space (between paragraphs)

The normal amount of rubber space between paragraphs is `\parskip`, which has a natural value of 0 inches, and can be stretched or compressed a bit (see Section 6.2.1 on page 89, in particular the discussion on `\setlength`).

So if you wanted to change its length (say, to have a document with no indent at the start of paragraphs, but space between paragraphs) the command should be given along the lines of

```
\setlength{\parskip}{4pt plus 2pt minus 2pt}
```

which would ensure that all paragraphs are separated by between 2pt and 6pt, depending on what else is on that page. If the change is to apply to the whole document, it should be made in the preamble. If it is to apply to only part of a document, it should be made within { ... }s or the environment in which it is to apply.

The rubber space between items in a list-type environment (see Sections 3.9.7 and 8.8, pages 39 and 141) is determined by the `\parsep` length parameter.

Sometimes it is desirable to have more, or less, space between two paragraphs than is usual. This can be done using:

`\bigskip` which produces `\vspace{\bigskipamount}` of vertical space. `\bigskipamount` is a rubber length command that is set (no matter what size option [10pt, 11pt or 12pt] is in force for the document) to `12pt plus 4pt minus 4pt`.

`\medskip` which produces `\vspace{\medskipamount}` of vertical space. `\medskipamount` is a rubber length command that is set (no matter what size option is in force for a document) to `6pt plus 2pt minus 2pt`.

`\smallskip` which produces `\vspace{\smallskipamount}` of vertical space. `\smallskipamount` is a rubber length command that is set (no matter what size option is in force for a document) to `3pt plus 1pt minus 1pt`.

```
The space between this paragraph
and the next is the default.

Now we start to make changes.

{\setlength{\parskip}{8pt plus 2pt minus 2pt}
Between this line

and this there should be a
larger-than-default space.
}

After this line is a \verb1\bigskip1.\bigskip

After this line a \verb1\medskip1.\medskip

After this a \verb1\smallskip1.\smallskip

End of demonstration.
```

The space between this paragraph and the next is the default.

Now we start to make changes.

Between this line

and this there should be a larger-than-default space.

After this line is a `\bigskip`.

After this line a `\medskip`.

After this a `\smallskip`.

End of demonstration.

3.9 New lines and pages

3.9.1 Breaking a line that is too long

Use `\linebreak[`*num*`]` to break a line and preserve the justification (left and right).

Use `\nolinebreak[`*num*`]` to prevent a linebreak at a point.

num takes the values 0, 1, 2, 3, or 4, with default of 4. 0 is a very gentle suggestion that a break would(n't) be a good idea; 4 indicates that a break is (isn't) compulsory. In fact, 0–3 seem to be ignored under the default line-breaking settings.

Remember that `~` (non-breaking space), `\-` (optional hyphen) and `\mbox{ ... }` can also be used to control line-breaking.

3.9.2 Sticking out text and overfull boxes

LaTeX has two local declarations that influence line breaks: `\fussy` (the default) and `\sloppy`.

`\fussy` avoids "ugly" excess white space within a line, with at times the result that text will stick out into the right margin (particularly a textstyle maths environment, or a word that doesn't hyphenate easily).

`\sloppy` does the opposite, and will move the "box" that stuck out to the next line, even if there is then too much white space in the line.

In the `log`, you get a warning of both cases, as either an overfull box (one rule of thumb is that all boxes overfull by 5pt or more need attention) or an underfull box (in which case a measure of "badness" is given).

To avoid big overfull boxes, and keep the document `\fussy` you have the option of using `\linebreak` (but if the text is edited later you may get a *very* empty line) or enclosing the paragraph in `{\sloppy ...}` or using the equivalent environment `sloppypar`, which typesets the material within it in paragraph mode with the `\sloppy` declaration in force.

Linebreaking is controlled by the declaration in effect at the blank line or `\par` command that ends the paragraph.

If you find use of the above to be less than all you expected, you can alter the TeX parameters that they affect directly. There are several of these parameters, but the most important two are `\tolerance` and `\emergencystretch`. `\tolerance` is a measure for how much the inter-word space in a paragraph is allowed to diverge from its optimum value (which is font defined). The default value is often 200, and so `\tolerance=50` will make TeX try harder to stay near the optimum, and so cause more overfull boxes; `\tolerance=900` allows for looser typesetting, and fewer overfull boxes. Sensible values are in the range of 50 to 9999.

`\emergencystretch` when set to a positive value will add this length as stretchable space to every line, thereby accepting line breaking solutions that have been rejected before (and the line is typeset loosely, with more underfull boxes).

3.9.3 Breaking a line, but not starting a paragraph

The equivalent of a hard carriage return (the line ended is typeset "ragged right") is `\newline` or `\\[`*len*`]`, where *len* adds extra vertical space of length *len* above the new line.

`\\*[`*len*`]` forms a new line, like `\\[`*len*`]`, but inhibits a page break right before the new line. How to specify *len* is discussed next.

3.9.4 Specifying length in LaTeX

There are several different places in LaTeX where you will need to specify a measure of length, and in most places you can do so in inches or centimetres. You can also use points (as in `12pt` or `3pt`; 1 inch = 72.27 points). All of these are of fixed length, and do not change if you change the font size for the document or part of the document.

What is sometimes better, where the amount of space should be proportional to the "non-space" on either side of it, is to use a space measure that is defined a little more variably. For the moment, the best such

measures are ems and exs, which are, respectively, approximately the width of M in the current font and the height of x. Thus *len* could be expressed as 1ex or 0.5em or 2.1ex—whatever seems appropriate. Compare the first and second columns of the tables in Table 3.4.

Table 3.4: Effects of size change and space measures

All the tables below are made using the coding:

```
\hfill\begin{tabular}[t]{l}\hline
  default \\
  distance\\[7pt]
  {\tt 7pt} extra space above,\\[14pt]
  {\tt 14pt} extra space.\\\hline
\end{tabular}\hfill\
```

```
\hfill\begin{tabular}[t]{l}\hline
  default \\
  distance\\[.7em]
  {\tt .7em} extra space above,\\[1.4em]
  {\tt 1.4em} extra space.\\\hline
\end{tabular}\hfill\
```

\normalsize

default distance 7pt extra space above, 14pt extra space.

default distance .7em extra space above, 1.4em extra space.

\huge

default distance 7pt extra 14pt extra

default distance .7em extra 1.4em extra

\scriptsize

default distance 7pt extra 14pt extra

default distance .7em extra 1.4em extra

Use of ems and exs means that any changes you make to the overall font size will not need to be tidied up later. In math mode, there is the mu measure of length (**m**ath **u**nit), where 18mu = 1 em.

In other places, like in a minipage (see page 93), or tabular (see page 80) environment where a p column format is used (and there are examples of both environments in this document), the width of the column or minipage could be specified in centimetres or inches. However, it can be safer to define these widths relative to the width of the page—or rather, to the width of the text on the page—and that is precisely what is done in statements such as

```
\begin{minipage}{0.45\linewidth}
\hrulefill width of {\tt minipage}
\hrulefill\\
\hspace*{-1ex}
\begin{tabular}{lp{0.65\linewidth}}
  1 & The contents of the
  {\tt minipage} in which this
  {\tt tabular} environment
  is placed is 45\% the width
  of the text on the page.\\
  2 & The text in the table is
  65\% the width of the minipage.
\end{tabular}
\rule{\linewidth}{0.4pt}
\end{minipage}
```

width of `minipage`

1	The contents of the `minipage` in which this `tabular` environment is placed is 45% the width of the text on the page.
2	The text in the table is 65% the width of the minipage.

which state that the minipage should be 45% of the width of the line, and that the paragraph column of the table should be 65% of the width of the line (and in this case, as it is within the `minipage` environment, it is 65% of the 45% of the text on the page). The advantage of this method is that two minipages so defined, designed to make two columns of text on a page, will always fit into the page width (as their widths total 90% of the width of the text line) and if the width of the text is changed, this will still be true; moreover, the `tabular` environment will still fit inside the minipage if the text width is changed. And if the `tabular` is taken outside the `minipage` it will still fit well on the page (which would not be the case had the definition been `\hspace*{-1ex}\begin{tabular}{lp{3cm}}`).

There are two widths that can be used: `\linewidth` and `\textwidth`. If you are always using single column format on a page, it makes no difference which you use (as the two are equal). If you will ever use 2-column format on a page, it is safer to use `\linewidth` as that is the width of the text in a column; `\textwidth` is the full width of the text on the page (so its use can cause the `minipage` or `tabular` environments to be much wider than a column). The actual width of a column is `\columnwidth`, and in a 2-column document it would probably be best to use this width.

The `\hspace*{-1ex}`? It's a compulsory (because of the `*`) horizontal space "to the left"—backspacing—as its length is < 0. A `tabular` environment automatically indents (by about `1ex` in a `minipage`) and this command preserves the original left margin (ie left align the `tabular`). See also page 81ff for another way (better?) to prevent the indent.

3.9.5 Stretchy space

If you've been watching the raw code closely (say, in Table 3.4), you'll have seen some `\hfill`s. This means: fill the space between (say, the three pairs of `tabular` environments in Table 3.4 and the left and right margins) in a stretchy way; divide all the horizontal white space into the same number of equal pieces as there are `\hfill`s, and replace each `\hfill` by one of these pieces of space. This is a useful way of achieving a proportional spread of tables or minipages or figures across a page, or of two or more blocks of text in, say, a document header line.

There is a vertical equivalent called, surprisingly enough, `\vfill`, that works in exactly the same way on spare vertical space. This is useful when spacing out subquestions on a test or exam paper, where questions are to start on a new page, and each page looks better with the white spread between question parts (with, perhaps, most of it at the bottom of the page), rather than as a single block at the bottom of the page.

Sometimes, a `\vfill` (or, more rarely, `\hfill`) is ignored—in particular, it will be ignored if it is at the beginning or end of a line. The way around this is to use `\vspace` (or `\hspace`), see Section 6.2.1, page 89. A quick and dirty solution (used in Table 3.4) is to make LaTeX think the `\hfill` or `\vfill` is not at the end of the line. This can be done by adding a space (which may unbalance the space slightly), as in `\hfill\␣`, or by adding an empty box (better) as in `\hfill\mbox{}`.

3.9.6 Page breaks

`\pagebreak[num]` encourages a page break (*num* is as above for `\linebreak`) while preserving the text height in the page – short pages are prevented by increasing vertical white spaces between paragraphs and before and after tables, figures and sections on the page.

`\nopagebreak[num]` discourages a page break.

`\enlargethispage{len}` causes the height of the page to be increased by *len*. Can cause problems if the page has a footer, as the text in the page may end on top of the footer text.

`\enlargethispage*{len}` as above, except all rubber space is squeezed as much as it can be.

`\newpage` stops the current page at that point and starts a new page, so it can result in a shorter page.

`\clearpage` and `\cleardoublepage` (for 2-sided printing, in which case the next page with text is right-handed, odd-numbered) has an effect like `\newpage`, but in addition causes *all* figures and tables (see page 106) that are in memory to be output.

3.9.7 List environments

There are three kinds of obviously "listy" environments, and several list-like environments.

3.9.8 `enumerate, description, itemize`

The three "listy" environments are:

enumerate where each item is numbered

description which is being used here—although the full effect of this environment is difficult to see without longer entries on each item

itemize where each item is bullet pointed.

```
\begin{description}
 \item[enumerate] where
   each item is numbered
 \item[description] which
   is being used here ...
 \item[\mdseries\sl itemize]
   where each item is ...
\end{description}
```

Note that you can change the appearance of the item labels (the default is **bold**) by inserting one or more declarations in the square brackets—as above for *itemize*.

Further considerations:

```
\begin{enumerate}
 \item In each such
   environment the
   \verb1\begin{list}1 must
    be followed ...
 \item Nesting is possible:
   \begin{enumerate}
     \item within similar ...
     \item within different ..
     \item up to four levels.
       \begin{enumerate}
         \item In  ...
         \item In  ...
         \item More than ...
       \end{enumerate}
   \end{enumerate}
\end{enumerate}
```

1. In each such environment the `\begin{list}` must be followed immediately by a `\item`. Failure to do so results in an error message.
2. Nesting is possible:
   (a) within similar environments
   (b) within different environments
   (c) up to four levels.
      i. In `enumerate` a different form of numbering is used, by default, at each level.
      ii. In `itemize` a different special character is used, by default, at each level.
      iii. More than one level of `description` is unlikely to be used.

Note that a "structured look" to our raw LaTeX coding will

- aid legibility—it is easier to see what's going on
- make it easier to track errors, and find the beginning or end of an environment.

```
\begin{itemize}
  \item aid legibility---
    it ...
  \item make it easier ...
\end{itemize}
```

3.9.9 List-like environments

These include three environments for including quotations or writing poetry, as well as the three environments of Section 3.5 which control the justification of text.

`quote` is intended for short quotations and has
- wider-than-usual left and right margins (there is an equal indentation on the left and the right);
- no paragraph indent;
- vertical space (small) between paragraphs.

A `quote` is intended to be short.

And may even hold quotes from two or more sources in its separate paragraphs.

```
\begin{quote}
A {\tt quote} is intended to be short.

And may even hold quotes from two or
more sources in its separate
paragraphs.
\end{quote}
```

`quotation` is intended for longer quotations and has
- wider left and right margins;
- indented paragraphs (suppress this with `\noindent`) even if `\parindent` is set to 0pt;
- the default space (none) between paragraphs.

A `quotation` is intended to be longer than a `quote`.

It is typically a longer quotation from a single source.

```
\begin{quotation}
A {\tt quotation} is intended to
be longer than a {\tt quote}.

It is typically a longer quotation
from a single source.
\end{quotation}
```

`verse` for typesetting poems has
- wider left and right margins;
- lines ended by `\\` or `\\*` (to prevent a page break at that point);
- stanzas separated by a blank line in the LaTeX code.

```
\begin{verse}
  Any verse\\
  made by me\\
  must be blank.

  This matches\\
  the state \\
  of my head.
\end{verse}
```

Any verse
mmade by me
must be blank.

This matches
the state
of my head.

These environments can be nested, with each level being further indented.

3.10 Modes

The mode that LaTeX is in determines how it typesets a particular part of a document—how it treats spaces, does(n't) break lines, and so on.

There are four modes:

- **Paragraph mode** for typesetting prose by the paragraph. Some commands can only be used in paragraph mode.
- **Mathematics mode** for typesetting a mathematical formula that can be from a single character to several lines of mathematics in length. All the commands to produce mathematical symbols and some to produce other effects necessary in typesetting mathematics can only be used in math mode. Few paragraph mode commands can be used in math mode. Spacing in math mode is not the same as in paragraph mode.
- **LR mode** is the mode used in LR boxes (see page 92) which are typically less than a line long. For instance, `\mbox` (page 34 and 92) which can be used to ensure some text, or a mathematical formula, is not broken across lines.
- **Picture mode** (see page 97) is used inside a picture environment. It is a restricted form of LR mode.

Chapter 4

Basic Mathematics

This chapter covers the basic concepts of LaTeX in math mode, the different "styles", types of symbol, types of mathematical environment, and fonts. Understanding these differences is important, because this gives control over how the typeset product looks. The next chapter puts the basic principles into practice, and covers some ideas that I think are useful—including some that are not strictly speaking mathematics, but it seems an appropriate place for them to be discussed.

You will have to browse the textbooks for the bits that apply to your kind of maths, but here are some generally useful features.

4.1 "Styley" Maths

Most mathematical characters and commands can be used only in a mathematical environment (that is, in math mode).

For further details on the symbols, log-like operators, variable-sized symbols, see Lamport pg 40 ff., and GMS pg. 216 ff. (the latter is more complete, as it includes all the ams-stuff—for which you need the packages amsmath and/or amssymb). Most symbols and operators are listed in Section 4.6 of these notes. Further documentation on amsmath is available locally, see page 161.

4.1.1 "Big maths"

There are two basic kinds of mathematical environment: those where the formulae are **displayed** (in `displaystyle`) and those where they aren't (in `textstyle`). Maths within a sentence, like $x^2+2x-4\log y$ is "`textstyle`".

Displayed maths has the characteristics:

1. Starts on a new line, as does following text, with a big skip on either side of the maths environment.
2. Fractions have their numerator and denominator in the same-sized font as the text.
3. Sums, products, limits etc. have their limits "under" and "over" the symbol, which is large; integral signs are large.
4. The environment is centered (the default; will be left justified if `fleqn` option is used in the `\documentclass` statement).
5. Can be numbered for reference.

Textstyle maths has the characteristics:

1. Starts where the next word would start on the current line; following text starts after a normal inter-word gap.
2. Fractions have their numerator and denominator in smaller font than the text.

3. Sums, products, limits etc. have their limits as super- and sub-scripts to the symbol, which is small; integral signs are small.
4. The environment is not centered.
5. Is never numbered.

Sometimes it's preferable to mix styles, and this can be done by putting a `\displaystyle` (local) declaration in a `textstyle` maths environment, or a `\textstyle` (local) declaration in a displayed maths environment.

If we compare the textstyle $\frac{1}{2}x^2 + \frac{\sin\theta}{\cos\theta}$ or $\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$ with the displayed equivalents:

$$\frac{1}{2}x^2 + \frac{\sin\theta}{\cos\theta}$$

or

$$\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

which are the result of the coding:

```
If we compare the textstyle $\frac{1}{2}x^2 + \frac{\sin\theta}{\cos\theta}$ or
$\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} +
    \frac{1}{n_2}}}$ with the displayed equivalents:
$$\frac{1}{2}x^2 + \frac{\sin\theta}{\cos\theta}$$ or $$\frac{(\bar{x}_1 -
  \bar{x}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} +  \frac{1}{n_2}}}$$
```

some of the differences between `textstyle` and `displaystyle` become apparent.

We can achieve a mix of the styles as follows:

We may be happy with the first `textstyle` expression, $\frac{1}{2}x^2 + \frac{\sin\theta}{\cos\theta}$ but maybe not with the second, preferring $\displaystyle\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$ and in the displayed equivalents, we may prefer a smaller fractional coefficient of x^2:

$$\textstyle\frac{1}{2}x^2 + \frac{\sin\theta}{\cos\theta}$$

and a larger denominator in the second expression

$$\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\displaystyle s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

which is the result of the coding:

```
We may be happy with the first {\tt textstyle} expression, $\frac{1}{2}x^2 +
  \frac{\sin\theta}{\cos\theta}$ but maybe not with the second, preferring
$\displaystyle\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 -
  \mu_2)}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$ and in  the displayed equivalents,
 we may prefer a smaller fractional coefficient of $x^2$, :
$${\textstyle\frac{1}{2}}x^2 + \frac{\sin\theta}{\cos\theta}$$
and a larger denominator in the second expression
$$\frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}
  {\displaystyle s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$
```

Hint `\displaystyle` and `\textstyle` are long names to type; it is really useful to give the declarations shorter names (aliases), such as:

```
\newcommand{\D}{\displaystyle}
\newcommand{\T}{\textstyle}
```

These definitions should appear in the preamble, and would be used as in

```
$\frac{1}{2}x^2 + \D\frac{\sin\theta}{\cos\theta}$
```

or

```
$${\T\frac{1}{2}}x^2 + \D \frac{\sin\theta}{\cos\theta}$$
```

4.1.2 "Little maths"

The other two style declarations, `\scriptstyle` and `\scriptscriptstyle` are used to produce smaller symbols in a formula, and are discussed further in Section 5.1.10 on page 73.

4.2 Mathematical one-liners

There are several ways of defining (one-line) mathematical environments. For simplicity, maths that is to be (basically) `textstyle` is best delimited by `$ ... $` (as in the previous section), for instance, `$x^2 + 3x$` which gives $x^2 + 3x$. An alternative is to use `\( ... \)`, as in `\(x^2 + 3x\)`, but this, firstly, involves more typing, and secondly is fragile (see page 123). So usually `$ ... $` is "better". One time when `\( ... \)` has to be used is when using `alltt` environment (page 147).

The two most convenient methods of specifying an un-numbered displayed maths environment are: `$$ ... $$` (as in the previous section), or `\[ ...\]`, for instance

```
\[\sin x_1^2 - \cos\alpha_2^2\]
```

which gives

$$\sin x_1^2 - \cos \alpha_2^2$$

Single-line numbered equations are generated using the environment

```
\begin{equation}
  \label{eq:label}
  .
  .
  .
\end{equation}
```

where `{eq:label}` is the identifier of that equation used to reference it. The `eq:` is not an essential part of the *`label`*, but if all your equation labels start with `eq:`, all your figure labels start with `fig:`, all your section labels with `sec:`, tables with `tab:` and so on, you have more choices for labels, as `eq:1` is distinct from `fig:1`, etc.

Should you later decide not to number the equation, change the environment to the starred version:

```
\begin{equation*}
  %\label{eq:label}   <----  can delete or comment out reference label
  .
  .
  .
\end{equation*}
```

The `*` suppresses numbering.

As an example of `equation`, the code

```
\begin{equation}
  \label{eq:1}
  y = \int\exp\{-t^2/2\}\,dt.
\end{equation}
but
\begin{equation*}
  z = \int\exp\{-x^2/2\}\,dx.
\end{equation*}

And in the integral in
(\ref{eq:1}) we see \ldots
```

typesets as:

$$y = \int \exp\{-t^2/2\}\, dt. \tag{4.1}$$

but

$$z = \int \exp\{-x^2/2\}\, dx.$$

And in the integral in (4.1) we see ...

Note the use of the equation label. Referencing labelled equations is discussed further on page 69.

4.3 Multiline formulae

The environments provided by the package amsmath (insert a `\usepackage{amsmath}` in the preamble) are far superior to `eqnarray` (provided in "vanilla" LaTeX), as there is less space on either side of the alignment character, only one & need be typed at (before) the alignment character (`eqnarray` requires an & both before and after the alignment character), there are several alignment options, and very long formulae that must be split across at least 2 lines are well catered for. Each environment (except `split`) can be used in either starred (no equation numbers generated) or un-starred (numbered) form and there are some options on how the lines are labelled (pg 240 in GMS). To get a proper coverage of the options, you need to read GMS, Pg. 235–239, 244–251.

Some examples are given in this chapter, and a more detailed discussion is given in Section 5.1.7 on page 64 of these notes.

4.4 Text in a maths environment and vice versa

It is possible, *buthighlyundesirable* (`$but highly undesirable$`) to use a math environment to generate italicised text.

In the same, but opposite way, the command `\mathrm{text in a formula}` can be used as in

$$x = y\mathrm{textinaformula}y = z$$

to get roman "writing" in a formula.

As you can see, spaces are not handled very well either way.

The `$ ... $` environment should only include letters that are part of "maths", and `\mathrm{ ... }` should only be used to generate non-space-separated bits of text that are actually part of a formula (unless you put the spaces in by hand, as in `\mathrm{text\␣in\␣...}`. For instance, in $\mathbf{A}^{\mathrm{T}}$ (`$\mathbf{A}^{\mathrm{T}}$`) to denote the transpose of a matrix $\mathbf{A}$.

In addition to `\mathbf` and `\mathrm`, there are the commands `\mathsf`, `\mathit` and `\mathtt` to produce sans serif, italics and typewriter fonts in math mode. Note the effect on the mathematical symbols in `$x + \mathsf{z - 2\theta + y}$` ($x + \mathsf{z - 2\theta + y}$).

The correct ways to insert text into a maths environment are:

1. In "vanilla" LaTeX, use `\mbox{text in here}`, as in

   $$x = y \text{ text in a formula } y = z$$

   (`\[x = y \mbox{ text in a formula } y = z\]`).

2. amsmath package has the command `\text{...}` (which is easier to type!), as in

   $$x = y \text{ text in a formula } y = z$$

   (`\[x = y \text{ text in a formula } y = z\]`).

`\ensuremath{ ...}` is very useful, as it ensures that what is in the { ... }s is in math mode. It is particularly useful for commands that you write to facilitate your own typing. For instance, if you type a lot of statistics, the following would be useful:

```
\newcommand{\Xb}{\ensuremath{\overline{X}}}
```

Without the `\ensuremath{ ...}`, a similar command would generate an $\overline{X}$, but would always have to be enclosed in a maths environment, so that to use an $\overline{X}$ in a sentence (as was done here) you would need to type `$\Xb$`.

But with the `\ensuremath{ ...}`, math mode is "ensured", and $\overline{X}$ is created as `\Xb`—you just need to watch that you insert a hard space (`\␣`) before the next word if $\overline{X}$ is not followed some form of punctuation.

4.5 Decorations and accents

Text mode accents and math mode accents were discussed in section 3.4.3 (page 30). Note that

1. The accents are a "single size" to fit over one character, unless they are one of those discussed below.
2. `\widehat` and `\widetilde` are
   - always wider than `\hat` or `\tilde`;
   - one of three sizes, depending on whether they are over one, two or three characters.

   Examples: $\widehat{x}$ $\widehat{xy}$ $\widetilde{xyz}$ (`$\widehat{x}\ \widehat{xy}\ \widetilde{xyz}$`).
3. - `\overline` and `\underline` draw lines long enough to fit over/under any number of characters (and `\underline` can be used in text mode).
   - `\overrightarrow` and `\overleftarrow` draw arrows (pointing right and left, respectively), long enough to fit over the specified number of characters (`\underleftarrow` and `\underrightarrow` are available in the `amsmath` package, which also modifies the implementation of the over/under arrows so that they scale properly in the subscript styles).
     Examples (`amsmath` used): $\overrightarrow{xyzw}$ $\underleftarrow{abcdefg}$ $\overleftarrow{BDE}$ $\underrightarrow{UVWXYZ}$ (`$\overrightarrow{xyzw}` `\underleftarrow{abcdefg}` `\scriptstyle\overleftarrow{BDE}` `\scriptscriptstyle{\underrightarrow{UVWXYZ}}$`).
   - `\overbrace` and `\underbrace` draw braces over/under the specified number of characters. These two commands have the option of labels over/under the brace—see the example on page 63.

4.6 Symbols and Operators

In math mode, the amount of space between any two characters or symbols is determined by the kind of characters or symbols that they are. The categories of symbol, and some examples of each are given below; more complete listings of the symbols are given in Tables 4.2–4.7.

Table 4.1: Types of mathematical symbol

Kind of symbol	Examples		Spacing

Ordinary symbol: see Tables 4.2 and 4.3.	α
	θ
	|
	x, Y
	$1, 4, 8$
	$\forall$
	/, $\|$
	$\exists$
	%
		`\alpha`
		`\theta`
		`|`
		`x, Y`
		`1, 4, 8`
		`\forall`
		`/, |`
		`\exists`
		`\% ...`
			No space between adjacent symbols.

Unary operators: see Table 4.4	There are three types, depending on how super/subscripts are typeset.		Typically, a thin space (`\,` or $\frac{1}{6}$em) between the operator and argument; no space between a function and an opening symbol (eg `\sin(A+B) = \sin A ...` giving $\sin(A+B) = \sin A \ldots$).

	$\sum\sum$
	$\prod\prod$
	$\int\int$
	$\bigcap\bigcap$
	$\bigcup\bigcup$
		`\sum`
		`\prod`
		`\int`
		`\bigcap`
		`\bigcup ...`
			These come in two sizes—small for textstyle ($\sum$) and large for displaystyle $\left(\sum\right)$. Subscripts and superscripts appear as limits (over and under symbol—except for `\int`) in displaystyle $\left(\sum_{i=1}^{n}\right)$, and as sub/superscripts in textstyle ($\sum_{i=1}^{n}$).

Continued on next page

Continued from previous page

Kind of symbol	Examples		Spacing
	det lim max min ...	`\det \lim` `\max \min`	Log-like operators in which sub/superscripts appear as limits in displaystyle ($\lim_{n\to\infty}$ $\lim_{n\to\infty}$).

	arccos
	tan log
	deg
		`\arccos`
		`\tan \log`
		`\deg ...`
			Log-like operators in which sub/superscripts *always* appear as such (never as limits).

Binary operators: see Tables 4.5 and 4.7	$+, -$
	$\cdot$
	$\circ$
	$\div, \pm$
	...
		`+, -`
		`\cdot`
		`\circ`
		`\div \pm`
			Symbols that make a (bigger) term out of two terms. Medium space (`\:` or $\frac{2}{9}$em) before and after a binary operator when it is between two operands (eg in $x + y$). However, when not between two operands, a binary operator is treated as an ordinary symbol (no space), as in $-x$.

Binary relation symbols	$<, \ll$
	$>, \gg$
	$=, \doteq$
	$\sim$
	$\simeq$
	$\parallel$
	$\therefore$
	$\propto$
	$\approx$
	$\equiv$
	$\leq, \geq, :$
		`< \ll`
		`> \gg`
		`= \doteq`
		`\sim`
		`\simeq`
		`\parallel`
		`\therefore`
		`\propto`
		`\approx`
		`\equiv`
		`\le \ge :`
			Symbols that make a formula out of two terms. Thick space (`\;` or $\frac{5}{18}$em) before and after the symbol. Many exist in negated form: `\neq \nless \nleq \notin ...`, producing $\neq \not< \not\leq \notin \ldots$. Otherwise, use `\not`, as in `\not\sim` ($\not\sim$).

	All arrows, eg
	$\to, \leftarrow$
	$\Leftarrow$
	$\Longleftarrow$
	$\nearrow$
	$\searrow$
	$\nwarrow$
	$\swarrow$
	$\leftrightarrow$
	...
		`\to \gets`
		`\Leftarrow`
		`\Longleftarrow`
		`\nearrow`
		`\searrow`
		`\nwarrow`
		`\swarrow`
		`\leftrightarrow`
			Some arrows have negated versions:
			`\nleftarrow \nrightarrow \nLeftarrow`
			`\nRightarrow \nleftrightarrow`
			`\nLeftrightarrow` which give
			$\nleftarrow \nrightarrow \nLeftarrow \nRightarrow \nleftrightarrow \nLeftrightarrow$

	Some vertical arrows:
	$\uparrow$
	$\downarrow$
	$\Uparrow$
	$\Downarrow$
	$\updownarrow$
	$\Updownarrow$
		`\uparrow`
		`\downarrow`
		`\Uparrow`
		`\Downarrow`
		`\updownarrow`
		`\Updownarrow`
			Are relational symbols when used alone. Are opening or closing symbols when, for instance, used as in: `\left\uparrow` or `\right\uparrow` or `\bigl\uparrow`, etc. $\left\uparrow \sum_{i=1}^{n} x_i \right\downarrow$ shows the effect, even if it lacks meaning.

Opening and closing symbols	( ) [ ] { }
	$\lfloor \rfloor$
	$\lceil \rceil$
	$\langle \rangle$
		`( ) [ ] \{ \}`
		`\lfloor \rfloor`
		`\lceil \rceil`
		`\langle \rangle`
			Not much space is left before or after these symbols, although it may vary.

Punctuation	, ; :	`, ; \colon`	These symbols are followed by a thin (`\,` or $\frac{1}{6}$em) space. These are the only supplied punctuation symbols.

Continued on next page

Continued from previous page		
Kind of symbol	Examples	Spacing
	: | :	Binary relation symbol, not punctuation symbol.
	. | .	Ordinary symbol, will not cause extra space to be inserted after it.

When a bit of maths is typeset with `\scriptstyle` or `\scriptscriptstyle` in force, very little space in inserted, as can be seen in the limits in $\sum_{i=1}^{n}$ and in $\sin\theta + y = 4$ $\sin\theta + y = 4$ $\sin\theta+y=4$ $\sin\theta+y=4$
(`\displaystyle\sin\theta + y = 4` `\quad\textstyle \sin\theta + y = 4 \quad \scriptstyle\sin\theta + y = 4` `\quad\scriptscriptstyle \sin\theta + y = 4`).

The ordinary symbols available are listed in Tables 4.2 and 4.3; the unary operator symbols in Table 4.4; the binary operator symbols in Table 4.5 and the binary relation symbols in Tables 4.6 to 4.7.

Table 4.2: Greek letters

Lower case	Upper case	Upper case italics Vanilla LaTeX	Upper case italics amsmath	Lower case variants
α `\alpha`	A `\mathrm{A}`	A `A`	`A`	
β `\beta`	B `\mathrm{B}`	B `B`	`B`	
γ `\gamma`	Γ `\Gamma`	Γ `\mathit{\Gamma}`	`\varGamma`	
δ `\delta`	Δ `\Delta`	Δ `\mathit{\Delta}`	`\varDelta`	∂ `\partial`
ϵ `\epsilon`	E `\mathrm{E}`	E `E`	`E`	ε `\varepsilon`
ζ `\zeta`	Z `\mathrm{Z}`	Z `Z`	`Z`	
η `\eta`	H `\mathrm{H}`	H `H`	`H`	
θ `\theta`	Θ `\Theta`	Θ `\mathit{\Theta}`	`\varTheta`	ϑ `\vartheta`
ι `\iota`	I `\mathit{I}`	I `I`	`I`	
κ `\kappa`	K `\mathrm{K}`	K `K`	`K`	
λ `\lambda`	Λ `\Lambda`	Λ `\mathit{\Lambda}`	`\varLambda`	
μ `\mu`	M `\mathrm{M}`	M `M`	`M`	
ν `\nu`	N `\mathrm{N}`	N `N`	`N`	
ξ `\xi`	Ξ `\Xi`	Ξ `\mathit{\Xi}`	`\varXi`	
o `o`	O `\mathrm{O}`	O `O`	`O`	
π `\pi`	Π `\Pi`	Π `\mathit{\Pi}`	`\varPi`	ϖ `\varpi`
ρ `\rho`	P `\mathrm{P}`	P `P`	`P`	ϱ `\varrho`
σ `\sigma`	Σ `\Sigma`	Σ `\mathit{\Sigma}`	`\varSigma`	ς `\varsigma`
τ `\tau`	T `\mathrm{T}`	T `T`	`T`	
υ `\upsilon`	Υ `\Upsilon`	Υ `\mathit{\Upsilon}`	`\varUpsilon`	
ϕ `\phi`	Φ `\Phi`	Φ `\mathit{\Phi}`	`\varPhi`	φ `\varphi`
χ `\chi`	X `\mathrm{X}`	X `X`	`X`	
ψ `\psi`	Ψ `\Psi`	Ψ `\mathit{\Psi}`	`\varPsi`	
ω `\omega`	Ω `\Omega`	Ω `\mathit{Omega}`	`\varOmega`	

amsmath also provides $\digamma$ (`\digamma`) and $\varkappa$ (`\varkappa`).

When amsmath package is used, `\mathit` does not cause capital Greek letters to be italicised. It is necessary to use `\varGamma` etc. to achieve this effect.

Table 4.3: Ordinary symbols

Symbols available in vanilla LaTeX:			
$\forall$ \forall	$\exists$ \exists	$\neg$ \neg or \lnot	$\top$ \top
$\bot$ \bot	$\emptyset$ \emptyset	∞ \infty	$\aleph$ \aleph
$\hbar$ \hbar	/ /	| | or \vert	\ \backslash
¶ \P	§ \S	! !	# \#
% \%	_ _	$ \$	& \&
$\imath$ \imath	$\jmath$ \jmath	ℓ \ell	$\wp$ \wp
$\Re$ \Re	$\Im$ \Im	$\prime$ \prime	∇ \nabla
$\surd$ \surd	$\flat$ \flat	$\sharp$ \sharp	$\natural$ \natural
† \dag	‡ \ddag	? ?	$\angle$ \angle
♠ \spadesuit	♡ \heartsuit	♣ \clubsuit	♢ \diamondsuit
△ \triangle	$\int$ \smallint	∂ \partial	@ @
. .	… \ldots	⋯ \cdots	⋮ \vdots
⋱ \ddots			

Symbols provided by amssymb package:			
$\nexists$ \nexists	$\varnothing$ \varnothing	$\beth$ \beth	$\gimel$ \gimel
$\daleth$ \daleth	$\hslash$ \hslash	$\diagup$ \diagup	$\diagdown$ \diagdown
$\Bbbk$ \Bbbk	$\backprime$ \backprime	$\eth$ \eth	★ \bigstar
Ⓢ \circledS	$\Finv$ \Finv	$\measuredangle$ \measuredangle	$\sphericalangle$ \sphericalangle
◊ \lozenge	⧫ \blacklozenge	△ \vartriangle	▲ \blacktriangle
▽ \triangledown	▼ \blacktriangledown	□ \square	■ \blacksquare
∁ \complement	$\mho$ \mho	$\Game$ \Game	

Symbols provided by latexsym and amssymb packages:			
□ \Box	◊ \Diamond	$\mho$ \mho	

Table 4.4: Unary operator symbols

Large operators—larger version for displaystyle, smaller version for textstyle:							
$\bigcap$	\bigcap	$\bigcup$	\bigcup	$\bigodot$	\bigodot	$\bigoplus$	\bigoplus
$\bigotimes$	\bigotimes	$\bigsqcup$	\bigsqcup	$\biguplus$	\biguplus	$\bigvee$	\bigvee
$\bigwedge$	\bigwedge	$\coprod$	\coprod	$\int$	\int	$\oint$	\oint
$\prod$	\prod	$\sum$	\sum				

Vanilla LaTeX Log-like operators—super/subscripts as limits in displaystyle:							
det	\det	gcd	\gcd	inf	\inf	lim	\lim
lim inf	\liminf	lim sup	\limsup	max	\max	min	\min
Pr	\Pr	sup	\sup				
Extras provided by amsmath package:				inj lim	\injlim	proj lim	\projlim

Vanilla LaTeX Log-like operators—super/subscripts never as limits: All names as for arccos and det (\arccos \det).							
arccos	arcsin	arctan	arg	cos	cosh	cot	coth
csc	deg	dim	exp	hom	ker	lg	ln
log	sec	sin	sinh	tan	tanh		
Extras provided by amsmath package							
$\varinjlim$	\varinjlim	$\varliminf$	\varliminf	$\varlimsup$	\varlimsup	$\varprojlim$	\varprojlim

Table 4.5: Binary operator symbols

Symbols available in vanilla LaTeX:			
$+$ +	$-$ -	$\pm$ \pm	$\mp$ \mp
$\times$ \times	$\div$ \div	mod \bmod	$*$ \ast or *
$\star$ \star	$\cup$ \cup	$\cap$ \cap	$\setminus$ \setminus
$\cdot$ \cdot	$\bullet$ \bullet	$\circ$ \circ	$\bigcirc$ \bigcirc
$\oplus$ \oplus	$\ominus$ \ominus	$\otimes$ \otimes	$\oslash$ \oslash
$\odot$ \odot	$\dagger$ \dagger	$\ddagger$ \ddagger	$\wr$ \wr
$\wedge$ \wedge or \land	$\vee$ \vee or \lor	$\uplus$ \uplus	$\sqcup$ \sqcup
$\sqcap$ \sqcap	$\amalg$ \amalg	$\diamond$ \diamond	$\triangleleft$ \triangleleft
$\triangleright$ \trianlgeright	$\triangle$ \bigtriangleup	$\bigtriangledown$ \bigtriangledown	
Symbols provided by amssymb package:			
$\ltimes$ \ltimes	$\rtimes$ \rtimes	$\leftthreetimes$ \leftthreetimes	$\rightthreetimes$ \rightthreetimes
$\divideontimes$ \divideontimes	$\Cup$ \Cup	$\Cap$ \Cap	$\smallsetminus$ \smallsetminus
$\centerdot$ \centerdot	$\circledast$ \circledast	$\circleddash$ \circleddash	$\circledcirc$ \circledcirc
$\boxplus$ \boxplus	$\boxminus$ \boxminus	$\boxtimes$ \boxtimes	$\boxdot$ \boxdot
$\intercal$ \intercal	$\barwedge$ \barwedge	$\doublebarwedge$ \doublebarwedge	$\curlywedge$ \curlywedge
$\veebar$ \veebar	$\curlyvee$ \curlyvee	$\dotplus$ \dotplus	
Symbols provided by latexsym and amssymb packages:			
$\lhd$ \lhd	$\unlhd$ \unlhd	$\rhd$ \rhd	$\unrhd$ \unrhd

Table 4.6: Binary relation symbols—general

Symbols available in vanilla LaTeX:			
$<$ <	$>$ >	$\ll$ \ll	$\gg$ \gg
$\leq$ \le or \leq	$\geq$ \ge or \geq	$\prec$ \prec	$\succ$ \succ
$\preceq$ \preceq	$\succeq$ \succeq	$\in$ \in	$\ni$ \ni or \owns
$\subset$ \subset	$\supset$ \supset	$\subseteq$ \subseteq	$\supseteq$ \supseteq
$\sqsubseteq$ \sqsubseteq	$\sqsupseteq$ \sqsupseteq	$\smile$ \smile	$\frown$ \frown
$\parallel$ \parallel	$\mid$ \mid	$\dashv$ \dashv	$\vdash$ \vdash
$\models$ \models	$\bowtie$ \bowtie	: :	$=$ \=
$\doteq$ \doteq	$\equiv$ \equiv	$\sim$ \sim	$\simeq$ \simeq
$\cong$ \cong	$\approx$ \approx	$\perp$ \perp	$\asymp$ \asymp
$\propto$ \propto			
Negation symbols:		$\notin$ \notin	$\neq$ \neq or \ne
Symbols provided by amssymb package:			
$\lll$ \lll or \llless	$\ggg$ \ggg or \gggtr	$\leqslant$ \leqslant	$\geqslant$ \geqslant
$\leqq$ \leqq	$\geqq$ \geqq	$\eqslantless$ \eqslantless	$\eqslantgtr$ \eqslantgtr
$\lessdot$ \lessday	$\gtrdot$ \grtdot	$\preccurlyeq$ \preccurlyeq	$\succcurlyeq$ \succcurlyeq
$\curlyeqprec$ \curlyeqprec	$\curlyeqsucc$ \curlyeqsucc	$\lesssim$ \lesssim	$\gtrsim$ \gtrsim
$\lessapprox$ \lessapprox	$\gtrapprox$ \gtrapprox	$\precsim$ \precsim	$\succsim$ \succsim
$\precapprox$ \precapprox	$\succapprox$ \succapprox	$\Subset$ \Subset	$\Supset$ \Supset
$\subseteqq$ \subseteqq	$\supseteqq$ \supseteqq	$\sqsubset$ \sqsubset	$\sqsupset$ \sqsupset
$\smallsmile$ \smallsmile	$\smallfrown$ \smallfrown	$\shortparallel$ \shortparallel	$\shortmid$ \shortmid
$\between$ \between	$\Vdash$ \Vdash	$\vDash$ \vDash	$\Vvdash$ \Vvdash
$\therefore$ \therefore	$\because$ \because	$\pitchfork$ \pitchfork	$\circeq$ \circeq
$\eqcirc$ \eqcirc	$\risingdotseq$ \risingdotseq	$\doteqdot$ \doteqdot or \Doteq	$\fallingdotseq$ \fallingdotseq
$\trianglelefteq$ \triangleq	$\bumpeq$ \bumpeq	$\Bumpeq$ \Bumpeq	$\thicksim$ \thicksim
$\backsim$ \backsim	$\backsimeq$ \backsimeq	$\thickapprox$ \thickapprox	$\approxeq$ \approxeq
$\blacktriangleleft$ \blacktriangleleft	$\vartriangleleft$ \vartriangleleft	$\trianglelefteq$ \trianglelefteq	$\blacktriangleright$ \blacktriangleright
$\vartriangleright$ \vartriangleright	$\trianglerighteq$ \trianglerighteq	$\ni$ \backepsilon	$\varpropto$ \varpropto

Continued on next page ...

Binary relation symbols continued			
⩽ \lessgtr	⋚ \lesseqgtr	⪋ \lesseqqgtr	⩾ \gtrless
⋛ \gtreqless	⪌ \gtreqqless		
Negation symbols:			
≮ \nless	⩽̸ \nleqslant	≰ \nleq	≨ \lneq
≦̸ \nleqq	≨ \lneqq	≰ \lvertneqq	⊀ \nprec
⋠ \npreceq	⪵̸ \precneqq	⋦ \lnsim	⪉ \lnapprox
⋨ \precnsim	⪹̸ \precnapprox	⊈ \nsubseteq	⫋ \varsubsetneq
⊊ \subsetneq	⫅̸ \nsubseteqq	⫋ \varsubsetneqq	⫋ \subsetneqq
∦ \nparallel	∦ \nshortparallel	⊬ \nvdash	⊭ \nvDash
⊯ \nVdash	≯ \ngtr	≱ \ngeqslant	≱ \ngeq
≩ \gneq	≧̸ \gneqq	≩ \gneqq	≩ \gvertneqq
⊁ \nsucc	⪰̸ \nsucceq	⋩ \succneqq	⪊ \gnsim
⪊ \gnapprox	⪺̸ \succnsim	⪊ \succnapprox	⊉ \nsupseteq
⫋ \varsupsetneq	⊋ \supsetneq	⫆̸ \nsupseteqq	⫋ \varsupsetneqq
⫌ \supsetneqq	∤ \nmid	∤ \nshortmid	≁ \nsim
≇ \ncong	⋪ \ntriangleleft	⋬ \ntrianglelefteq	⋫ \ntriangleright
⋭ \ntrianglerighteq			
Symbols provided by latexsym and amssymb packages:			⋈ \Join

Table 4.7: Binary relation symbols—arrows

Symbols available in vanilla LaTeX:		
↗ \neqrrow	↙ \swarrow	↘ \searrow
↖ \nwarrow	← \gets or \leftarrow	⇐ \Leftarrow
→ \to or \rightarrow	⇒ \Rightarrow	↔ \leftrightarrow
⇔ \Leftrightarrow	↦ \mapsto	⟵ \longleftarrow
⟸ \Longleftarrow	⟶ \longrightarrow	⟹ \Longrightarrow
⟷ \longleftrightarrow	⟺ \Longleftrightarrow	⟼ \longmapsto
↼ \leftharpoonup	⇀ \rightharpoonup	↽ \leftharpoondown
⇁ \rightharpoondown	↩ \hookleftarrow	↪ \hookrightarrow
⇌ \rightleftharpoons		
Symbols provided by amssymb package:		
⇃ \downharpoonleft	⇂ \downharpoonright	↿ \upharpoonleft
↾ \upharpoonright	⇈ \upuparrows	⇊ \downdownarrows
↢ \leftarrowtail	↣ \rightarrowtail	↞ \twoheadleftarrow
↠ \twoheadrightarrow	⇚ \Lleftarrow	⇛ \Rrightarrow
⇇ \leftleftarrows	⇉ \rightrightarrows	⇆ \leftrightarrows
⇄ \rightleftarrows	⇝ \rightsquigarrow	↭ \leftrightsquigarrow
↶ \curvearrowleft	↺ \circlearrowleft	⊸ \multimap
↫ \looparrowleft	↰ \Lsh	↬ \looparrowright
↷ \curvearrowright	↻ \circlearrowright	↱ \Rsh
Symbols provided by latexsym and amssymb packages:		⇝ \leadsto

Vertical arrows are named and discussed in Table 4.1.

Negated arrows are listed in Table 4.1.

Opening and closing symbols (delimiters) are a little different from other symbols, and are of four basic types:

- the six vertical arrows (and < >) that are binary relation symbols unless they are preceded by \left

or `\right` or one of the `\big...` series of commands, in which case they are delimiters that extend to a suitable size;
- the the twelve opening and closing symbols listed in Table 4.1, that have a "native size" (ie they can be used without being preceded by `\left` or `\right` or one of the `\big...` series of commands;
- `|` or `\vert`, `\|` or `\Vert`, `/` and `\backslash` which are ordinary symbols unless preceded by `\left` or `\right` or one of the `\big...` series of commands;
- the seven large delimiters given in Table 4.8:

Table 4.8: Large delimiters

⟮ `\rmoustache`	⟯ `\lmoustache`	⟮ `\rgroup`	⟯ `\lgroup`
⎪ `\arrowvert`	⎪⎪ `\Arrowvert`	⎪ `\bracevert`	

These symbols **must** by preceded by `\left` or `\right` or one of the `\big...` series of commands, and some of them have a larger-than-usual minimum size (typically `\Big`).

Most delimiters have no maximum size, but the sloping ones (`/ \backslash \langle < \rangle >`), do.

Setting the size of symbols is discussed further in Section 5.1.1.

It is possible to define new symbols and operators (from old ones, typically), or to change how a particular symbol or operator is spaced in a one-off useage.

4.6.1 Defining new log-like symbols

The three types of log-like symbols can be made as shown below. Assume that we want a log-like symbol called "name" that appears as name x:

1. **Super-/sub-scripts as limits in displaystyle and super-/sub-scripts in textstyle**:
   - Vanilla LaTeX: To make a new function called `\name`
     `\newcommand{\name}{\mathop{\mathrm{name}}\displaylimits}`
     which would be used: `\name x`.
   - Using $\mathcal{AMS}$-LaTeX: (the `*` is important in the command name)
     `\newcommand{\name}{\operatorname*{name}}`
   - Alternative using $\mathcal{AMS}$-LaTeX: (the `*` is necessary again)
     `\DeclareMathOperator*{\name}{name}`
2. **Super-/sub-scripts always as limits** (in both text- and display-style): To make a new function called `\lname`
   `\newcommand{\lname}{\mathop{\mathrm{name}}\limits}`
3. **Super-/sub-scripts always as super-/sub-scripts**: To make a new function called `\nname`
   - Vanilla LaTeX:
     `\newcommand{\nname}{\mathop{\mathrm{name}}\nolimits}`
   - $\mathcal{AMS}$-LaTeX (no `*` this time)
     `\newcommand{\nname}{\operatorname{name}}`
   - Alternative using $\mathcal{AMS}$-LaTeX: (no `*` this time)
     `\DeclareMathOperator{\nname}{name}`

Note that `\mathop` and `\operatorname(*)` can be used to make one-off new operators in the body of your document (ie not defined first in a `\newcommand`) although this would not be efficient if the new operator was to be used several times in the document.

For instance, instead of using `$x=\rm{name}␣y$`, which may not be spaced correctly, use `$x=\mathop{\mathrm{name}}␣y$`; the results are $x = \mathrm{name}y$ and $x = \mathrm{name}\,y$, respectively. Using `\operatorname`, in its two versions, we have `$\D␣x=\operatorname{name}_a^b␣y$` ($x = \mathrm{name}_a^b\, y$) and `$\Dx=\operatorname*{name}_a^b␣y$` ($x = \underset{a}{\overset{b}{\mathrm{name}}}\, y$).

4.6.2 Defining new binary operators

To define the new binary operator $+\!\!+$ (`\con`):

`\newcommand{\con}{\mathbin{+\mkern-8mu +}}`

where 18mu (**m**ath **u**nit) = 1em, and `\mkern - 8mu` shifts the second + 8 mus to the left. We can use this as `$x␣\con␣y$`, which gives $x +\!\!+ y$.

`\mathbin`, too, can be used outside a `\newcommand` to make a one-off binary operator, as in:

`$x␣\mathbin{+\mkern␣-␣8mu␣+}y$` which gives $x +\!\!+ y$.

4.6.3 Forcing a binary operator to be treated as an ordinary symbol

There are two possibilities, illustrated on `\sim`, which may be used as a symbol for negation:

- Enclose the name of the binary operator in `{ }`s. For instance, compare:

  $\sim p \cap \sim q$ (`\sim p \cap\sim q`) and $\sim p \cap \sim q$ (`{\sim} p\cap{\sim} q`).

- Define a new command (or use in any mathematical environment) using `\mathord`:

  `\newcommand{\lneg}{\mathord{\sim}}` which gives

  $\sim p \cap \sim q$ (`\lneg p \cap \lneg q`), which could also have been created as:
  `$\mathord{\sim}p␣\cap\mathord{\sim}␣q$` which appears as $\sim p \cap \sim q$.

4.6.4 Defining new binary relations

There are two possibilities to consider, both of which can be used in a `\newcommand` or any mathematical environment:

1. Use `\mathrel` to define a new binary relation:

   `\newcommand{\cuphat}{\mathrel{\widehat{\sqcup}}}`

   which gives $x \mathrel{\widehat{\sqcup}} y$ (`\x\cuphat y`).

2. Use `\stackrel` to stack one symbol (`\scriptstyle` and of any kind) over another (`\textstyle` and of any kind), and have the compound spaced as a binary relation.

   Say you wanted to stack a variety of symbols over an $=$:

   `\newcommand{\steq}[1]{\stackrel{#1}{=}}`

   gives $x \steq{\wedge} y \steq{a} z \steq{\mathrm{b}} xx$ (`x\steq{\wedge} y \steq{a} z \steq{\mathrm{b}}xx`).

   You can specify that the two symbols are the same size (style) as in:

   $x \stackrel{x}{y} y$ $x \stackrel{\textstyle x}{y} y$ (`x\stackrel{x}{y} y \qquad x\stackrel{\textstyle x}{y} y`).

4.6.5 Defining mathematical punctuation symbols

Any "punctuation" mark can be defined as such using `\mathpunct`, either as in

`\newcommand{\fullstop}{\mathpunct{.}}`

or on a one-off basis as in $f{:}x$ $f:x$ (`f\mathpunct{:}x \qquad f:x`).

4.6.6 When one symbol may be two things

Some symbols, particularly : $\backslash$ $|$ $\|$ $<$ $>$ $\perp$ $\dagger$ $\ddagger$ may sometimes need to be used as a binary relation (surrounded by a thick space) and sometimes as an ordinary symbol or opening symbol (surrounded by no space) or punctuation mark (followed by a thin space). In each case, LaTeX provides two commands, one of each kind (which at first sight is confusing in the LaTeX texts).

In the examples above we have:

The symbol	Binary relation name	Punctuation/ordinary symbol	Opening/delimiter symbol
:	:	\colon	
\	\setminus	\backslash	
|	\mid	| or \vert	\left | or \right\vert
‖	\parallel	\Vert or \\|	\left\Vert or \right\\|
<	<	\langle	\left\langle and \left<
>	>	\rangle	\right\rangle and \right>.
⊥	\perp	\bot	
†	\dagger	\dag	
‡	\ddagger	\ddag	

4.7 Some of the real stuff

Commonly used mathematical constructions include the following (**Note** that use of the package amsmath is assumed throughout, unless otherwise specified):

```
\begin{gather*}
 \int_0^\infty f(x)\,dx=\int\limits_0^\infty f(x)\,dx\\
      % note use of \, to create space before dx
 \lim_{n\to\infty}\left(a+\frac{x}{n}\right)^n\\
 \sum_{r=0}^n\binom{n}{r}p^r(1-p)^{n-r}\\
 \sqrt[n]{a+b} \neq \sqrt{b-x}\\
 \vect{A}\vect{x} = \vect{b} \\
 (\mat{X}\trans\mat{X})\inv\mat{X}\trans \mat{y}
\end{gather*}
```

which gives:

$$\int_0^\infty f(x)\,dx = \int\limits_0^\infty f(x)\,dx$$

$$\lim_{n\to\infty}\left(a+\frac{x}{n}\right)^n$$

$$\sum_{r=0}^n \binom{n}{r} p^r(1-p)^{n-r}$$

$$\sqrt[n]{a+b} \neq \sqrt{b-x}$$

$$\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b}$$

$$(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Note:

1. The use of `gather*`—a centred, non-aligned unnumbered (as `*` form) multiline displaystyle environment.
2. Use of `{ .. }` to put things "up in the air" or "underfoot", i.e. following `^` or `_`. Brackets are *not necessary* around single characters or the names of LaTeX-defined commands, but *are necessary* around any user-defined commands or multiple characters.
3. The `\limits` command forces subscripted limits (here of integration) to appear as limits (like the limits of summation). A command to do the opposite is `\nolimits`. For instance to make the limits of summation in a displaystyle environment appear in textstyle, but with a bigger Σ use:

$$\sum_{i=1}^{k+1} x_i = \sum\nolimits_{i=1}^{k+1} x_i = \textstyle\sum_{i=1}^{k+1} x_i$$

   (`\sum_{i=1}^{k+1}x_i=\sum\nolimits_{i=1}^{k+1}x_i=\textstyle\sum_{i=1}^{k+1}x_i`)

4. That `\sqrt` does for both nth roots and square roots, with the n bit being an optional (`[ .. ]`) argument.
5. That most "log-like" functions that you need, like sin, cos, exp, ln and log have been defined (appear in roman type in a formula, and, where necessary, things that should be under them appear so—eg for lim—if entered as a subscript). Try just typing `\name` for one you want and see if it works. See Section 4.6.1 of these notes, and GMS pg 228 *Operator Names* for how you would define any that are not supplied.

6. The commands used above (and below) include some that are user-defined: `\vect`, `\mat`, `\trans`, and `\inv`. These commands, and their definition, are discussed further on pages 60 and 73.

Compare with the textstyle equivalents—note the default line spacing between the first two lines, and the increased amounts of space between the successive lines (achieved using `\\[len]`):

$\int_o^\infty f(x)\,dx \qquad \lim_{n\to\infty}\left(a+\frac{x}{n}\right)^n$
$\sum_{r=0}^{n}\binom{n}{r}p^r(1-p)^{n-r}$

$\sqrt[n]{a+b} \neq \sqrt{b-x}$

$\underset{\sim}{A}\underset{\sim}{x} = \underset{\sim}{b} \qquad (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$

```
$ \int_o^\infty f(x)\,dx\qquad
    \lim__{n\to\infty}\left(a + \frac{x}{n}\right)^n\\
\sum_{r=0}^n\binom{n}{r} p^r(1-p)^{n-r}\\[2ex]
\sqrt[n]{a+b} \neq \sqrt{b-x}\\[1ex]
\vect{A}\vect{x} = \vect{b} \qquad
  (\mat{X}\trans\mat{X})\inv\mat{X}\trans \mat{y}$
```

Another possibility is to have several lines of equations (as opposed to a single `gather*` environment)—note the unattractively large white spaces between equations, and the effect of a blank line (new paragraph) between the last two equations (exaggerated by increasing the size of `\parskip`, the gap between paragraphs):

$$\int_o^\infty f(x)\,dx$$

$$\lim_{n\to\infty}\left(a+\frac{x}{n}\right)^n$$

$$\sum_{r=0}^{n}\binom{n}{r}p^r(1-p)^{n-r}$$

```
\parskip=2em
\[  \int_o^\infty f(x)\,dx\]
\[  \lim_{n\to\infty}\left(a
    + \frac{x}{n}\right)^n\]

\[  \sum_{r=0}^n\binom{n}{r}
          p^r(1-p)^{n-r}\]
```

4.8 Calligraphy, Blackboard Bold, and Euler

One font that is less than entirely perfect is the maths calligraphy:

$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

(`$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$`) which is available only for the 26 upper case letters and only in math mode. The letters should really be a little more ornate. But not even LaTeX is quite perfect. A slight improvement and two further choices is offered by the eucal package:

1. `\usepackage{eucal}` redefines `\mathcal` to produce $\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ (the Euler Script alphabet).
2. `\usepackage[mathscr]{eucal}` does not redefine `\mathcal`, and the Euler Script alphabet letters are made with the `\mathscr` command:

$\mathcal{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathcal{A B ... Z}`
$\mathscr{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$ `\mathscr{A B ... Z}`

Another useful special effect font is "Blackboard Bold", which is available using amsmath:

$\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$

(`$\mathbb{ABCDEFGHIJKLMNOPQRSTUVWXYZ}$`)—also upper case only, math mode only.
And then there is (also using amsmath)

$\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234}$

(`$\mathfrak{ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234}$`) — also math mode, but available for numbers and upper and lower case letters.

Chapter 5

Useful Mathematics

Given an understanding of what is meant by the different "styles" in mathematics, and the implications of the different types of mathematical symbol, and how to make a basic mathematical environment, we can look at controlling the appearance of a bit of mathematics, and at some of the bells and whistles available, particularly when using the amsmath package.

Full documentation on the amsmath package is available, see page 161.

5.1 Some of my favourites

In no particular order, here are some useful things and some of the "neat" effects achieved either painstakingly by first principles, or else quickly (and better) using the amsmath commands and environments. (If the truth must be told, these include several things about which I have said to some hopeful person "Sorry, that's just not possible!!" and then discovered, in some obscure place or other much, much later, that they really existed and were easy to effect.)

5.1.1 Delimiters

The most commonly used ones (see also Tables 4.1 and 4.8, page 46–52 of these notes) are:

{ \{	} \}	[and]	(and)
⟨ \langle	⟩ \rangle	| |	‖ \\|

Delimiters (opening and closing symbols) can be made suitably large by:

- In vanilla LaTeX: `\left`*delim* ... `\right`*delim* in a formula, for instance

```
\left[\frac{\sin x}{\cos x}\right]
      \quad \text{or}\quad
\left\{\frac{a}{b} +
\left[\frac{c^2d}{ef}
       - 1\right]\right\}
```

Displaystyle:

$$\left[\frac{\sin x}{\cos x}\right] \quad \text{or} \quad \left\{\frac{a}{b} + \left[\frac{c^2d}{ef} - 1\right]\right\}$$

Textstyle:

$\left[\frac{\sin x}{\cos x}\right]$ or $\left\{\frac{a}{b} + \left[\frac{c^2d}{ef} - 1\right]\right\}$

where LaTeX chooses the appropriate sizes of the delimiters, and you have no control. Note that these delimiters **must be matched**, and that `.` denotes a "missing" delimiter, as in $(x - 1$ which is made using `$\left(x - 1\right.$`. Failure to match delimiters results in an error message.

One place where such error messages are common is in aligned mathematical environments (see Section 5.1.7, page 64 of these notes) or an `array` environment where an alignment character, `&`, will "break" a `\left ... \right` pair, so that

```
\left(\sum_{i=1}^n x_i/n\right. &  = \left.\frac{120}{5}\right)
```

will work and

```
\left(\sum_{i=1}^n x_i/n &  = \frac{120}{5}\right)
```

will cause an error message.

- `\big \Big \bigg \Bigg` give you control over the size of the delimiter, and can enforce larger sizes than would otherwise be used, e.g.

  `$\bigl((x-1)^2 - 4\bigr)$` $\bigl((x-1)^2 - 4\bigr)$

  or slightly undersized brackets where that seems appropriate, e.g.:

```
\begin{gather}
\frac{1}{(\frac{1}{n_1} + \frac{1}{n_2})}\\
 \intertext{or}
  \frac{1}{\left(\frac{1}{n_1}
+ \frac{1}{n_2}\right)}\\
 \intertext{or}
  \frac{1}{\bigl(\frac{1}{n_1}
+ \frac{1}{n_2}\bigr)}.
\end{gather}
```

$$\frac{1}{(\frac{1}{n_1} + \frac{1}{n_2})} \tag{5.1}$$

or

$$\frac{1}{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)} \tag{5.2}$$

or

$$\frac{1}{\bigl(\frac{1}{n_1} + \frac{1}{n_2}\bigr)}. \tag{5.3}$$

  These delimiters do not need to be matched: $\biggl(a - 1$ (`$\biggl(a - 1$`) does not result in an error message. This means that these commands will probably be safer to use in an aligned or `array` environment.

  These commands are in plain TEX, and so can be used in a LATEX document. In a vanilla LATEX document they work correctly for 10 point type (and may be too small in a document with the 12pt option in force). However, in $\mathcal{AMS}$-LATEX the family of commands is redefined so that they work for all the size options.

  The commands work only on delimiters, but within that there are variations that give the user control over the spacing around the larger delimiter. Each of the four basic size commands (`\big \Big \bigg \Bigg`) is available in four "flavours": `l r m ␣`, which will space a left delimiter, a right delimiter, a relation symbol and an ordinary symbol, respectively (for instance `\bigl, \bigr, \bigm, \big`). The effect of this can be seen in:

  $x\Big(x\Big(x\ \Big(\ x\Big(x\Big)x\ \Big)\ x\Big)x\Big)x$ (`$x\Bigl(x\Bigr(x\Bigm(x\Big(x\Bigr)x\Bigm)x\Big)x\Big)x$`).

- Delimiters around fractions (only) can be made to look a little neater by the slightly-more-clumsy-to-specify means shown in the section below.

5.1.2 Fractions and friends

5.1.2.1 Basic controls

Compare

```
$\frac{a}{b}\quad \dfrac{a}{b}
    \quad \tfrac{a}{b}$
$$\frac{a}{b}\quad \dfrac{a}{b}
    \quad \tfrac{a}{b}$$
```

$\frac{a}{b}\quad \dfrac{a}{b}\quad \tfrac{a}{b}$

$$\frac{a}{b}\quad \dfrac{a}{b}\quad \tfrac{a}{b}$$

typeset in `textstyle` and `displaystyle`.

`\binom{n}{r}`, too, exists in the additional `\dbinom` and `\tbinom` forms:

$\binom{n}{r}$ $\dbinom{n}{r}$ $\tbinom{n}{r}$

```
\binom{n}{r}\quad
\dbinom{n}{r}\quad
\tbinom{n}{r}
```

$\binom{n}{r}$ $\dbinom{n}{r}$ $\tbinom{n}{r}$

5.1.2.2 Continued fractions

The amsmath package provides `\cfrac`:

$$\cfrac{1}{\sqrt{2}+\cfrac{1}{\sqrt{3}+\cfrac{1}{\sqrt{4}+\cfrac{1}{\sqrt{5}+\dots}}}}$$

```
\[ \cfrac{1}{\sqrt2 +          %by default the
      \cfrac{1}{\sqrt3 +       %numerator is centered
        \cfrac[l]{1}{\sqrt4 + %this numerator
                              %is left justified
          \cfrac[r]{1}{\sqrt5 + \dots
                              %this numerator
                              %is right justified
   }}}}  \]
```

5.1.2.3 Obscure bells and whistles

The option to enclose fractions in delimiters, and to control the thickness of the fraction line has been removed from amsmath (it still appears in GMS, as it was in the package amstex, but it won't work). However, by prodding around in the actual code (in `amsmath.sty`) I found the following:

`\genfrac`*`ldelim rdelim`*`{`*`thickness`*`}`*`style`*`{`*`numerator`*`}{`*`denominator`*`}`

Where the two delimiters must be specified (use `.` to denote no delimiter), *`thickness`* is the thickness of the line (if specified as `0pt` there will be no line, as in `\binom` and variations), measured in `pt` or any other suitable measure (default used in `\frac` seems to be about `0.08ex`), and *`style`* = 0, 1, 2, 3 if the end result is to be `displaystyle`, `textstyle`, `scriptstyle`, `scriptscriptstyle`, respectively.

```
\begin{gather*}
 \binom{n}{r} = \genfrac(){0pt}0{n}{r}
    = \left(\begin{array}{c}
              n\\r
            \end{array}\right)
   \quad\left(\frac{n}{r}\right)\\[1ex]
 \genfrac..{0.2ex}0{\frac{x^2 - 1}{x-1}}%
   {\frac{x^2 - 1}{x + 1}} =
   \frac{\frac{x^2 - 1}{x-1}}%
   {\frac{x^2 - 1}{x + 1}}\\[1ex]
 {\genfrac[|{0.08ex}0{\sin2y}{4}}_0^{\pi/6}
   = \left[\frac{\sin2y}{4}\right|_0^{\pi/6}
\end{gather*}
```

$$\binom{n}{r} = \binom{n}{r} = \left(\begin{array}{c} n \\ r \end{array}\right) \quad \left(\frac{n}{r}\right)$$

$$\frac{\frac{x^2-1}{x-1}}{\frac{x^2-1}{x+1}} = \frac{\frac{x^2-1}{x-1}}{\frac{x^2-1}{x+1}}$$

$$\left[\frac{\sin 2y}{4}\right|_0^{\pi/6} = \left[\frac{\sin 2y}{4}\right|_0^{\pi/6}$$

The main difference, and sometimes advantage, of `\genfrac` is that the fraction is tucked into the delimiters, with no white space between.

5.1.3 Cases

```
\[ \f_X(x) =
     \begin{cases}
       \frac12,       & \frac12 \le x < 1\\
       \frac{x}{2}, & 1 \le x < 2\\
       0            & \text{elsewhere}
      \end{cases}
\]
```

$$f_X(x) = \begin{cases} \frac{1}{2}, & \frac{1}{2} \le x < 1 \\ \frac{x}{2}, & 1 \le x < 2 \\ 0 & \text{elsewhere} \end{cases}$$

Very easy to use; can use as many "cases" lines as necessary. The brace seems to resize correctly each time (which doesn't always happen using some of the alternative methods to achieve the same effect).

5.1.4 Decorating symbols

$\mathcal{AMS}$-LaTeX (the amsmath package) provides a number of improvements to vanilla LaTeX.

5.1.4.1 Multiple integral signs

$$\iint_V f(x,y)\,dx\,dy$$
$$\iiint_V f(x,y,z)\,dx\,dy\,dz$$
$$\iiiint_V f(u,v,w,x)\,du\,dv\,dw\,dx$$
$$\int\cdots\int_V f(x_1,\ldots,x_k)\,dx_1\,dx_2\,\ldots\,dx_k$$

```
\begin{gather*}
  \iint\limits_V f(x, y)\,dx\,dy\\
  \iiint\limits_V f(x, y, z)\,dx\,dy\,dz\\
  \iiiint\limits_V f(u, v, w, x)\,du\,dv\,dw\,dx\\
  \idotsint\limits_V f(x_1, \ldots, x_k)\,dx_1\,dx_2
                                        \,\dots\,dx_k
\end{gather*}
```

The example shows the four commands to create multiple integral signs (compare the above with $\int\int f(x,y)\,dx\,dy$ which is made by `$\D \int\int f(x, y)\,dx\,dy$`). Note the use of `\,` to create thin spaces where necessary, and the use of `\limits` to cause the subscript (V) to be displayed as a limit.

5.1.4.2 Dots

$\mathcal{AMS}$-LaTeX will usually position the dots correctly if the command `\dots` is used (on the base line if the next character is, say, a comma; in the middle of the line if it is a plus sign, etc.).

If the dots are at the end of a mathematical formula, it may be necessary to use one of:

`\dotsc` for "dots with comma", which has the same effect as `\ldots`
`\dotsb` for "dots with binary operators", which has the same effect as `\cdots`
`\dotsi` for "dots with integrals", which also has the same effect as `\cdots`.

Examples:

$a,b,c,\ldots$ (`$a, b, c, \dots$`), $a,b,c,\ldots$ (`$a, b, c, \dotsc$`),
$a+b+\cdots+x$ (`$a+b+\dots+x$`), $a+b+\ldots$ (`$a+b+\dots$`), $a+b+\cdots$ (`$a+b+\dotsb$`)

5.1.4.3 Roots

All root signs are made using `\sqrt`, and the commands `\leftroot` and `\uproot` move the root index to the left and up, respectively (or right and down, if negative) as can be seen:

$$\sqrt[n]{a} = \sqrt[n]{a} = \sqrt[n]{a} = \sqrt[n]{a}$$

```
$\sqrt[n]{a}=\sqrt[\leftroot{2}\uproot{4}n]{a}
 =\sqrt[\leftroot{-2}\uproot{4}n]{a}
=\sqrt[\leftroot{-2}\uproot{-4}n]{a} $
```

5.1.4.4 Boxed formulae

The command `\boxed` puts a box around its argument, much like `\fbox`, except that it is used in math mode.

```
\[ \boxed{\bar{X} = \sum_{i=1}^n X_i}\]
```

$$\boxed{\bar{X} = \sum_{i=1}^{n} X_i}$$

5.1.4.5 Stacking and not quite super/subscripting

`\stackrel` in LaTeX is designed to place a superscript over a binary relation. The commands `\overset` and `\underset`, provided by amsmath are more general, and will place a superscript-size symbol over

and/or under any other symbol, respectively.

$\overset{a}{X}\quad \underset{*}{B}\quad \underset{\sim}{x}\quad \overset{1}{\underset{2}{\alpha}}$

```
\[ \overset{a}{X}\quad \underset{*}{B}
\quad \underset{\widetilde{}}{x}
 \quad \overset{1}{\underset{2}{\alpha}}\]
```

The command `\sideset` allows the correct positioning of super/subscripts on large unary operators (like $\sum$ and $\prod$) when it would otherwise be difficult:

$$\sideset{_1^2}{_3^4}\prod_{i=j} \qquad \sideset{}{'}\sum_{i<k\in A} X_{ik}$$

```
\[ \sideset{_1^2}{_3^4}\prod_{i=j} \qquad
\sideset{}{'}\sum_{i < k \in A} X_{ik}\]
```

To get two or more rows of a limit on a large unary operator typeset correctly, use `\substack` as in:

`$\displaystyle\sum_{\substack{i,j=0\\i < j}}^nx_{ij}$` giving $\displaystyle\sum_{\substack{i,j=0\\i<j}}^{n} x_{ij}$.

5.1.4.6 Trimming things

The plain TEX command `\smash` keeps the contents of a box but cuts its height (above the baseline) and depth (below the baseline); amsmath introduces optional arguments `t` and `b` that enable the user to specify if only the height (`t`) should be cut, or only the depth.

```
\begin{gather*}
\sqrt{a}\ \sqrt{a_j} \ \sqrt{B}\ \sqrt{B_j^2}\\ % "as is"
\sqrt{\smash{a}} ... \\            % smash top and bottom
... \sqrt{\smash[t]{a_j}}... \\    % smash top only
...\sqrt{\smash[b]{B_j^2}}         % smash bottom only
\end{gather*}
```

$$\sqrt{a}\ \sqrt{a_j}\ \sqrt{B}\ \sqrt{B_j^2}$$
$$\sqrt{a}\ \sqrt{a_j}\ \sqrt{\smash{B}}\ \sqrt{\smash{B_j^2}}$$
$$\sqrt{a}\ \sqrt{\smash[t]{a_j}}\ \sqrt{\smash{B}}\ \sqrt{\smash[b]{B_j^2}}$$
$$\sqrt{a}\ \sqrt{a_j}\ \sqrt{B}\ \sqrt{B_j^2}$$

5.1.5 Matrices made easy

5.1.5.1 Naming them

It's useful to have a command (say, `\vect`) defined to write the name of a matrix or vector. Then, if you decide to change from, for instance, (obsolete, dare I say it) under-tildes to bold letters, you need only change your definition of the command `\vect` to change the appearance of all matrix/vector names in your document. Some possibilities are:

```
\newcommand{\utildev}[1]{\mathrel{\mathop{#1}\limits_{^{\sim}}}} %vanilla LaTeX
\newcommand{\utildea}[1]{\underset{\widetilde{}}{#1}}           %needs amsmath
\newcommand{\vect}[1]{\ensuremath{\utilde{#1}}}
\newcommand{\mat}[1]{\ensuremath{\mathbf{#1}}}
\newcommand{\gmat}[1]{\mbox{\boldmath${#1}$}}
\newcommand{\utsubv}[2]{\utildev{#1}_{\raisebox{1.4ex}{${\scriptstyle{#2}}$}}}
\newcommand{\utsuba}[2]{\utildea{#1}_{\raisebox{1.4ex}{${\scriptstyle{#2}}$}}}
```

For bold names, you have the options `\mathbf` (LaTeX) and `\boldsymbol` (amsmath only), which have the following effects on letters, Roman and Greek, and symbols:

`$\mathbf{aB\Gamma\alpha + \times}$\\` $\mathbf{aB\Gamma\alpha + \times}$

`$\boldsymbol{aB\Gamma\alpha + \times}$` $\boldsymbol{aB\Gamma\alpha + \times}$

It is then necessary to have one command (`\mat` above) for Roman-lettered matrices, and another for Greek-lettered matrices (`\gmat`).[1]

Under-tildes are tricky, particularly if subscripted, and two workable options (one for subscripted vectors) are given. It is worth comparing

[1] The `\gmat` shown is defined using `\boldmath`, which is available in "vanilla" LaTeX. An equivalent definition, assuming that the amsmath package will always have been loaded, is `\newcommand{\gmat}[1]{\ensuremath{\boldsymbol{#1}}}`.

```
$\utildev{x}, \utildev{x}_1, \utsubv{x}{1},
\text{ and } \utildev{x_1}$
and
$\utildea{x}, \utildea{x}_1, \utsuba{x}{1},
\text{ and } \utildea{x_1}$
```

$\underset{\sim}{x}, \underset{\sim}{x}_1, \underset{\sim}{x}_1,$ and $x_{\sim 1}$
and
$\underset{\sim}{x}, \underset{\sim}{x}_1, \underset{\sim}{x}_1,$ and $x_{\sim 1}$

5.1.5.2 Making them

You can make matrices and related arrays with up to 10 centred columns (and any number of rows) very easily. See GMS pg. 232 for what to do in the case of > 10 columns. The main types will be covered by one of the following:

- As in

```
\[ \mat{A} =
\begin{matrix}
  1 & 0\\
  0 & 1
\end{matrix} \]
```

$$\mathbf{A} = \begin{matrix} 1 & 0 \\ 0 & 1 \end{matrix}$$

The other such kinds of matrix are:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \quad \begin{bmatrix} -1 & 2 \\ 3 & 4 \end{bmatrix}$$
$$\begin{vmatrix} \alpha & -\alpha \\ \beta & \beta \end{vmatrix} \quad \begin{Vmatrix} 3 & 1 \\ 99 & -10 \end{Vmatrix}$$

```
\begin{gather*}
  \begin{pmatrix}a&b\\c&d\end{pmatrix}\quad
  \begin{bmatrix}-1&2\\3&4\end{bmatrix}\\
  \begin{vmatrix}\alpha&-\alpha\\
                 \beta&\beta\end{vmatrix}
  \quad \begin{Vmatrix}3&1\\99&-10\end{Vmatrix}
\end{gather*}
```

- From TEX (NB: hence we use `\cr`, not `\\`) we get a bordered matrix with named rows and columns:

```
\[ \bordermatrix{%
    & \text{col 1} & c_2 & {\mathrm C3}\cr
  r_1 & 1          & 1   & 0\cr
\text{Row 2}& 3     & 1   & 0} \]
```

$$\bordermatrix{ & \text{col 1} & c_2 & \mathrm{C3}\cr r_1 & 1 & 1 & 0\cr \text{Row 2} & 3 & 1 & 0}$$

- A textstyle "mini-matrix" is provided by $\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$ which fits entirely within a line. The effect of this should be compared with that of one of the other matrix environments, used textstyle, say `pmatrix` as in $\begin{pmatrix} e & f \\ g & h \end{pmatrix}$.

```
$\left(\begin{smallmatrix}
        a & b \\c & d
    \end{smallmatrix}\right)$
```

- All those dots (horizontal) to show possible rows and columns can be made using `\hdotsfor[`*spacing-factor*`]{`*number*`}`.

  The default value for *spacing-factor* is 1; *number* = number of columns dotted.

$$\det(\mathbf{\Phi}) = \begin{vmatrix} \phi_1 & 0 & \ldots\ldots\ldots & 0 \\ \phi_2 & \phi_3 & \ldots\ldots\ldots & 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \phi_4 & \phi_5 & \ldots & \phi_{k-1} & \phi_k \end{vmatrix}$$

```
\[\det(\gmat{\Phi}) =
 \begin{vmatrix}
   \phi_1 & 0 & \hdotsfor{2} & 0\\
   \phi_2 & \phi_3 & \hdotsfor[1.5]{2} & 0\\
   \hdotsfor[2]{5}\\
   \phi_4 & \phi_5 & \dots & \phi_{k-1} & \phi_k
 \end{vmatrix}  \]
```

"Handmade" dots can be achieved using `\cdots`, `\ddots` and `\vdots` as in

```
$\begin{pmatrix}
  1 & 0 & \cdots & 0\\
  0 & 1 & & 0\\
  \vdots & &\ddots & \vdots\\
  0 & 0 & \cdots & 1
\end{pmatrix}$
```

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

5.1.5.3 Making lines in them

Sometimes it is desirable to partition a matrix with vertical and/or horizontal lines in some way.

The environments `matrix`, `pmatrix`, etc. have the body of the matrix tucked well into the delimiters, so that lines drawn at the first and last column can overlap the delimiters (brackets) which looks ugly.

It is therefore probably better to draw partitioned matrices "from first principles" using an array environment, or to use `matrix` and specify delimiters (but you need to add space inside the delimiters).

The vertical lines are made easier if you define two new commands (see Section 5.2.1), say `\mca` (multicolumn after) and `\mcb` (multicolumn before) to insert a vertical line after or before, respectively, the argument of the command:

```
\newcommand{\mca}[1]{\multicolumn{1}{c|}{#1}}
\newcommand{\mcb}[1]{\multicolumn{1}{|c}{#1}}
```

These can be used in an array:

```
\[A = \left[
    \begin{array}{cccc}
      1 & 0 & * & *\\ \cline{1-1}
      0 & \mcb{1} & * & *\\ \cline{2-4}
      0 & 0 & 0 & 0
    \end{array}
    \right] =
    \begin{bmatrix}
      1 & 0 & * & *\\ \cline{1-1}
      0 & \mcb{1} & * & *\\ \cline{2-4}
      0 & 0 & 0 & 0
    \end{bmatrix}\]
```

$$A = \left[\begin{array}{c|ccc} 1 & 0 & * & * \\ \hline 0 & 1 & * & * \\ 0 & 0 & 0 & 0 \end{array}\right] = \begin{bmatrix} 1 & 0 & * & * \\ \hline 0 & 1 & * & * \\ \hline 0 & 0 & 0 & 0 \end{bmatrix}$$

where the overlap is obvious in the second matrix, or

```
\[A = \left[\
    \begin{matrix}
      1 & 0 & * & *\\ \cline{1-1}
      0 & \mcb{1} & * & *\\ \cline{2-4}
      0 & 0 & 0 & 0
    \end{matrix} \
    \right]\]
```

$$A = \left[\begin{array}{c|ccc} 1 & 0 & * & * \\ \hline 0 & 1 & * & * \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Another possibility is:

```
\[\det\left[
  \begin{array}{cccc}
    b & 0 & \cdots & 0\\ \cline{2-4}
    a_2 & \mcb{} & & \mca{}\\
    \vdots & \mcb{} & D & \mca{}\\
    a_n & \mcb{} & & \mca{}\\ \cline{2-4}
  \end{array}\right] = b(\det D).\]
```

$$\det \left[\begin{array}{c|ccc} b & 0 & \cdots & 0 \\ \hline a_2 & & & \\ \vdots & & D & \\ a_n & & & \end{array}\right] = b(\det D).$$

5.1.5.4 Linear Equations

Closely related to matrices, but to line everything up beautifully you're back to basics, using the "vanilla" LaTeX `array` environment (which is discussed more fully later in Chapter 6, together with its (usually) text sibling `tabular`).

The end result is neater if the "independent variable" columns are right justified, the symbols are centred, and the RHS is left justified. You need a column for each variable and one for each arithmetic operator between variables. That makes for a lot of columns, but if the column formats are repeated, you can give a short-hand (and easily altered) version.

For instance:

Explicit column statement	Shorthand version
`{rrrrrrrrrrr}`	`{*{11}{r}}` (Need `{11}` as 2 digits)
`{rcrcrc}`	`{*3{rc}}` Need not be `{3}`, as 1 digit.
`{|l|rrr|l}`	`{|l|*3{r}|l}`
`{|l|l|l|l|l|}`	`{|*5{l|}}`

Suppose we have a system of 5 equations in 5 unknowns, then we will need a column definition of `{r*4{cr}cl}`:

$$\underset{\texttt{r}}{a_1x_1}\ \underbrace{+a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5}_{\texttt{*4cr}}\ \underset{\texttt{c}}{=}\ \underset{\texttt{l}}{y}$$

[The line above was made using:

```
\[\underset{\text{\tt r}}{a_1x_1} \underbrace{+ a_2x_2 +a_3x_3 + a_4x_4 + a_5x_5}
   _{\text{\tt *4\{cr\}}}} \underset{\text{\tt c}}{=} \underset{\text{\tt l}}{y}\]
```

where `\underset` and the corresponding `\overset` will set character(s) under/over another, and `\underbrace` has a corresponding `\overbrace` which has a `^` instead of the `_` to put text or symbols above the overbrace.]

For instance:

$$\mathbf{Ax = b}\qquad \begin{array}{rcrcrcrcrcl} x_1 & + & x_2 & - & 3x_3 & + & 2x_4 & - & x_5 & = & a \\ x_1 & & & & & - & 3x_4 & + & 2x_5 & = & b \\ & & 2x_2 & + & x_3 & & & + & x_5 & = & c \\ & & & - & x_3 & + & x_4 & & & = & d \\ 3x_1 & & & & & & & + & 4x_5 & = & e \end{array}$$

```
\[\mat{Ax} = \mat{b}\qquad
    \begin{array}[t]{r*4{cr}cl}
       x_1 & + &  x_2 & - & 3x_3 & + & 2x_4 & - &  x_5 & = & a\\
       x_1 &   &      &   &      & - & 3x_4 & + & 2x_5 & = & b\\
           &   & 2x_2 & + &  x_3 &   &      & + &  x_5 & = & c\\
           &   &      & - &  x_3 & + &  x_4 &   &      & = & d\\
     3x_1  &   &      &   &      &   &      & + & 4x_5 & = & e
    \end{array}  \]
```

The columns are unattractively far apart. We'll look at reducing the intercolumn gap in Section 6.1.4.

5.1.6 Commutative diagrams

The amscd package provides the building blocks for rather nice "just" commutative diagrams, or, with a bit of ingenuity, more complicated flow-chart-type diagrams.

A commutative diagram is made in a `CD` environment, for instance:

```
\[
\begin{CD}
  A @ >{f}>> B\\
  @V{f'}VV @ VV{g}V\\
  C@>>{g'}> D
\end{CD}\]
```

$$\begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle f'}\downarrow & & \downarrow{\scriptstyle g} \\ C & \xrightarrow[g']{} & D \end{array}$$

where `@>>>` (or `@))))`) produces a long right arrow; `@<<<` (or `@((((`) produces a long left arrow; `@VVV` produces a long down arrow; `@AAA` produces a long up arrow; `@.` shows there is nothing between two nodes; `@=` produces a long horizontal double line and `@|` or `@\vert` a long vertical double line.

Mathematical text can be placed to the left or right of vertical arrows by positioning the text between the first and second, or second and third As or Vs, respectively.

In the same way, mathematical text can be placed above or below horizontal arrows by positioning the text between the first and second, or second and third <s or >s, respectively.

It is safest to enclose the text in `{...}`. The diagrams in Figure 9.1 and 9.2 were created using commutative diagrams, with the commands for the former being:

```
\[\newcommand{\file}{\text{\opt{file}}}
        % already have \newcommand{\opt}[1]{\texttt{\textsl{#1}}}
\newcommand{\extn}[1]{\text{\texttt{.#1}}}
\newcommand{\lrun}[1]{\text{\fbox{%
      \begin{tabular}{c}
        \LaTeX\\Run #1
      \end{tabular}}}}
\begin{CD}
  \file\extn{tex} @>>> \lrun 1 @. \file\extn{bib}\text{ and
    }\opt{style}\extn{bst}\\
@VVV @V{\text{contains}}V{\text{cited refs}}V @V{\text{\shortstack{data\\base}
    and}}V{\text{\shortstack{bibliography\\style}}}V\\
  @.\file\extn{aux} @>>> \text{\fbox{\bibtex}}\\
    @VVV @VVV
    @V{\text{\shortstack{formatted\\references}}}V\Bigl({\text{
                \shortstack{and log\\ file}}}\Bigr)V\\
  \qquad\longrightarrow@>>> \lrun 2 @<{\text{formatted}}<{\text{references}}<
  \begin{array}[t]{c}
    \file\extn{bbl}\\
    \left(
      \begin{array}{c}
        \text{and \file\extn{blg},} \\\text{not reused}
      \end{array}\right)
  \end{array}\\
@VVV  @V{\text{updated}}V{\text{file}}V @.\\ %no arrow between middle nodes
 @. \file\extn{aux} @.\\
  @VVV @VVV\\
   \qquad\longrightarrow @>>> \lrun 3 @>{\text{output}}>{\text{file}}>\file\extn{dvi}
\end{CD}
\]
```

Note the use of local commands; because they are defined in the equation environment, they are temporary and not available outside that environment.

Note also the rather useful command `\shortstack`, which has the form

`\shortstack[`*pos*`]{`*col*`}`

where *pos* takes the values `l`, `r` or `c` (with the last being the default), and *col* is the text to be typeset in a narrow column. The lines are separated by `\\`, and the space between the lines is less than that left in a `tabular` or `array` environment. This compactness makes it ideal for use in diagrams, such as commutative-type diagrams, or in a `picture` environment. If used in a `picture` environment, the reference point is at the bottom left hand corner.

5.1.7 Aligned lines

You should read GMS, at least to check out the examples, because you can combine the options to fairly dramatic effect. A summary of the aligning environments is given here.

A point to bear in mind in all of these environments is that a pair of delimiters enlarged by `\left` and `\right` **cannot** have the alignment character, &, between the `\left` and the `\right` (see Section 5.1.1).

5.1.7.1 The "outer environments"

These environments "are" mathematical environments—they turn on maths mode at their start, and turn it off at their end. They cannot be nested within other mathematical environments (but can contain environments that must be in maths mode, such as `array` or `split`).

Each of the environments exists in both the unstarred (lines can be numbered) and starred (unnumbered) form.

The environments can be grouped as follows:

1. **Single-column environments**: `gather` `multline`

   The lines in these environments are not aligned; in the former the lines are centred separately, and the latter has its first line left-justified, its last line right-justified, and any middle lines are individually centered (except when the `fleqn` option is in effect). There is a provision for the first and last lines in a `multline` environment to be indented from the margins by an amount equal to `\multlinegap`, which can be set using `\setlength` and `\addtolength` commands.

   The other big difference between the two environments is that the former can have each line numbered, but the latter has a single number for the whole equation-structure—note the position of the `\label`s in the examples that follow.

```
\begin{gather}
  x = 2              \label{eq:2}\\
  x^2 - 4xy + 6 = 7  \nonumber\\
  y \le 6            \notag
\end{gather}
```

$$\begin{gathered} x = 2 \\ x^2 - 4xy + 6 = 7 \\ y \le 6 \end{gathered} \tag{5.4}$$

`\nonumber` is the LaTeX command to suppress equation numbering; `\notag` is provided by amsmath.

```
\begin{multline}\label{eq:5}
  x = 2              \\
  x^2 - 4xy + 6 = 7  \\
  y \le 6
\end{multline}
\addtolength{\multlinegap}{3em}
\begin{multline}\label{eq:6}
  x = 2              \\
  x^2 - 4xy + 6 = 7  \\
  y \le 6
\end{multline}
```

$$\begin{multline} x = 2 \\ x^2 - 4xy + 6 = 7 \\ y \le 6 \end{multline} \tag{5.5}$$

$$\begin{multline} x = 2 \\ x^2 - 4xy + 6 = 7 \\ y \le 6 \end{multline} \tag{5.6}$$

Remember that the starred version of any displayed maths environment will suppress numbering.

2. **Paired columns, any number of:** `align` `flalign`

   These environments will align any number of columns in pairs, with the space between pair members determined by the amount of space usually between such mathematical characters, as is shown in the example below:

$$12 \quad 34 \quad 56 \quad (5.7)$$
$$12 \quad 34 \quad 56 \quad (5.8)$$

$$1 = 2 \quad 3 = 4 \quad 5 = 6 \quad 7$$
$$1 = 2 \quad 3 = 4 \quad 5 = 6$$

$$12 \qquad 34 \qquad 56$$
$$12 \qquad 34 \qquad 56$$

$$1, 2 \quad 3, 4 \quad 5, 6 \quad 7 \quad (5.9)$$
$$1, 2 \quad 3, 4 \quad 5, 6 \quad (5.10)$$

```
\begin{align}
  1 & 2 & 3 & 4 & 5 & 6\\
  1 & 2 & 3 & 4 & 5 & 6
\end{align}
\begin{align*}
   1 &= 2 & 3 &= 4 & 5 &= 6 & 7\\
   1 &= 2 & 3 &= 4 & 5 &= 6
\end{align*}
\begin{flalign*}
  1 & 2 & 3 & 4 & 5 & 6\\
  1 & 2 & 3 & 4 & 5 & 6
\end{flalign*}
\begin{flalign}
   1 &, 2 & 3 &, 4 & 5 &, 6 & 7\\
   1 &, 2 & 3 &, 4 & 5 &, 6
\end{flalign}
```

As can be seen from the example, `align` has a equal amounts of space between the pairs of columns and between the first and last columns and the margins, while `flalign` has no space between the first and last columns and the margins and equal amounts of space between the pairs of columns.

Both environments have an unstarred, numbered form and starred, un-numbered form.

There appears to be no difference between the two environments if there is a single pair of columns (both shown with equation numbers):

$$(a+b)^2 = (a+b)(a+b) \quad (5.11)$$
$$= a^2 + 2ab + b^2 \quad (5.12)$$

$$(a+b)^2 = (a+b)(a+b) \quad (5.13)$$
$$= a^2 + 2ab + b^2 \quad (5.14)$$

```
\begin{align}
  (a+b)^2 &= (a+b)(a+b)       \\
          &= a^2 + 2ab + b^2
\end{align}
\begin{flalign}
  (a+b)^2 &= (a+b)(a+b)       \label{eq:3}\\
          &= a^2 + 2ab + b^2 \label{eq:4}
\end{flalign}
```

3. ***k*-column (*k* must be specified) environments** `alignat` `xalignat` `xxalignat`

These all have the form shown for `alignat`:

```
\begin{alignat}{3}
  x &= x & y &= y & z &= z\\
  x &= x & y &= y & z &= z
\end{alignat}
```

where in this example, $k = 3$, so there are three aligned pairs of columns (xs, yx and zs), and in each line in the environment $2k - 1 = 5$ alignment characters (&s) are required.

The difference between the three environments is in the space between columns: in `alignat` the columns are grouped fairly close together (the space between all adjacent columns is determined by the character after the alignment &—as is the case between the columns in each pair in `align` and `flalign`); in `xalignat` there are equal spaces at the margins and between the pairs of columns (and the space between column-pair members is as in `align` and `flalign`); and in `xxalignat` there is no space at the margins and in consequence this environment is always un-numbered (unlike `flalign`, which *can* be numbered).

$$C = 2\pi r C = 2\pi \times 10 \tag{5.15}$$
$$A = \pi r^2 \, A = \pi \times 10 \tag{5.16}$$

$$C = 2\pi r \qquad C = 2\pi \times 10 \tag{5.17}$$
$$A = \pi r^2 \qquad A = \pi \times 10 \tag{5.18}$$

$$C = 2\pi r \qquad\qquad C = 2\pi \times 10$$
$$A = \pi r^2 \qquad\qquad A = \pi \times 10$$

```
\begin{alignat}{2}
    \label{eq:al1}
  C &= 2\pi r & C &= 2\pi\times 10\\
  A &= \pi r^2 & A &= \pi\times 10
\end{alignat}
\begin{xalignat}{2}
    \label{eq:al2}
  C &= 2\pi r & C &= 2\pi\times 10\\
  A &= \pi r^2 & A &= \pi\times 10
\end{xalignat}
\begin{xxalignat}{2}
  \label{eq:al3}
  C &= 2\pi r & C &= 2\pi\times 10\\
  A &= \pi r^2 & A &= \pi\times 10
\end{xxalignat}
```

5.1.7.2 The "inner environments"

These environments are designed to be used *inside* a mathematical environment. The outer environment would be the one to be numbered, so these environments exist only in an un-starred, un-numbered form. There are two groupings of these environments:

1. The environment `split` which will split and align, say, an `equation` into two or more lines.

   As the lines of the `split` environment cannot be numbered, the possible number of line numbers is determined by the "outer" environment—if it is `equation` there can only be one; if it is `align` there can be more than one, but it would be necessary to use a separate `split` environment for each section of the total structure that is to be numbered.

   The default position of the line numbers is in the (vertical) centre of the `split` environment, as in:

```
\begin{align}     %All  splits are aligned
  \begin{split}
    2(a+b)^2 &= 2(a^2 + 2ab + b^2)\\
             &= 2a^2 + 4ab + 2b^2
  \end{split}\\   %need \\ for new line
  \begin{split}
    2(a-b)^2 &= 2(a^2 - 2ab + b^2)\\
             &= 2a^2 - 4ab + 2b^2
  \end{split}\\
\notag            %last split unnumbered
  \begin{split}
    a &= c+d\\
    b &= d+e
  \end{split}
\end{align}
```

$$\begin{aligned} 2(a+b)^2 &= 2(a^2 + 2ab + b^2) \\ &= 2a^2 + 4ab + 2b^2 \end{aligned} \tag{5.19}$$

$$\begin{aligned} 2(a-b)^2 &= 2(a^2 - 2ab + b^2) \\ &= 2a^2 - 4ab + 2b^2 \end{aligned} \tag{5.20}$$

$$a = c + d$$
$$b = d + e$$

2. Environments that form "blocks" in a display, with options for varying vertical alignment (top, center or bottom), two of the block environments having the possibility of being (vertically) internally aligned. The three environments are `aligned`, `gathered` (no internal alignment possible) and `alignedat`. The documentation on `alignedat` is exceedingly sketchy. It appears, from trial and error, that `alignedat` has 3 possible arguments: the optional vertical alignment argument, and two numerical arguments, the *second* being the number of pairs of columns. The value of the first appears to make no difference to the result—the braces can be empty, even—but two arguments, at least, are required. The spacing between columns in `alignedat` is as in `alignat` (ie no extra space between pairs of columns).

$$Z = 2X + 3Y \tag{5.21}$$
$$\begin{aligned} X + Y &\le 10 \\ 2X - Y &\le 8 \end{aligned} \tag{5.22}$$
$$X, Y > 0 \tag{5.23}$$

```
\begin{gather}
  Z = 2X + 3Y\\   %need \\ for new line
  \begin{aligned}
    X + Y & \le 10\\
    2X - Y & \le 8
  \end{aligned}\\ %need \\ for new line
  X, Y > 0
\end{gather}
```

$$(a \pm b)^2 \qquad (a+b)^2 = a^2 + 2ab + b^2 \qquad (a-b)^2 = a^2 - 2ab + b^2 \tag{5.24}$$
$$(3x+1)^2 = 9x^2 + 6x + 1 \qquad (3x-4y)^2 = 9x^2 - 24xy + 16y^2$$

```
\begin{equation}
  (a \pm b)^2 \quad
  \begin{aligned}dat}[t]{1}{2}  %why {1} ????? {2} pairs cols
    (a + b)^2 &= a^2 + 2ab + b^2 & (a - b)^2 &= a^2 - 2ab + b^2\\
    (3x+1)^2 &= 9x^2 + 6x + 1 &\quad (3x - 4y)^2 &= 9x^2 - 24xy + 16y^2
                              %\quad to add space between pairs of columns
  \end{aligned}
\end{equation}
```

$$(a \pm b)^2 \qquad \begin{aligned} (a+b)^2 &= a^2 + 2ab + b^2 \\ (3x+1)^2 &= 9x^2 + 6x + 1 \end{aligned} \qquad \begin{aligned} (a-b)^2 &= a^2 - 2ab + b^2 \\ (3x-4y)^2 &= 9x^2 - 24xy + 16y^2 \end{aligned}$$

```
\[  (a \pm b)^2 \quad
  \begin{aligned}[t]          % top aligned
    (a + b)^2 &= a^2 + 2ab + b^2 \\
    (3x+1)^2 &= 9x^2 + 6x + 1
  \end{aligned}\qquad         %make space between "blocks" on same line
  \begin{gathered}[b]         % bottom aligned
    (a - b)^2 = a^2 - 2ab + b^2\\
    (3x - 4y)^2 = 9x^2 - 24xy + 16y^2
  \end{gathered}
\]
```

One place where these environments are really useful/necessary is when an aligned environment is to be the first thing in an `\item` of a list environment. The "outer environments" are `displaystyle`, which means that they start on a new line. For instance:

1. $$\begin{aligned} f(x) &= \int_{-\infty}^{\infty} f(x)\,dx \\ &= \int_{-\infty}^{\infty} e^{-2x}\,dx \end{aligned}$$

```
\begin{enumerate}
\item
  \begin{align*}
    f(x) &= \intinf f(x)\,dx\\
    &= \intinf e^{-2x}\,dx
  \end{align*}
\end{enumerate}
```

An option is to use one of the "inner environments", remembering to top align the environment:

1. $\begin{aligned} f(x) &= \int_{-\infty}^{\infty} f(x)\,dx \\ &= \int_{-\infty}^{\infty} e^{-2x}\,dx \end{aligned}$

```
\begin{enumerate}
\item $
  \begin{aligned}[t]
    f(x) &= \intinf f(x)\,dx\\
    &= \intinf e^{-2x}\,dx
  \end{aligned} $
\end{enumerate}
```

Some other crude solutions to this problem are given on page 87.

5.1.7.3 Vertical spacing and page breaks in amsmath equation structures

Extra vertical space: A command `\\[2ex]` will leave an extra `2ex` between lines in one of the environments discussed above (as it would in a `tabular` or `array` environment or anywhere else)—and of course any length given in any type of LaTeX length measure can be substituted for the 2ex above, including negative values, which will decrease the space between the lines.

Page breaks: To force or allow page breaks in the middle of one of these environments, there are commands
`\displaybreak` and `\allowdisplaybreaks`. The former can take (like `\pagebreak`) an optional argument where `\displaybreak[0]` suggests very gently that such a break might be possible, and `\displaybreak[4]`, or, equivalently, `\displaybreak` will force a page break at that point.

The `\displaybreak` command should go before the `\\` at which the new page is to be started.

The scope within which a pagebreak would be allowed should be shown by

`{\allowdisplaybreaks ... lines that can be split across pages ...}`.

If there are a number of lines except one or two at which a page can be broken, the scope within which the page breaks can occur is defined, and `\\*` is used to prohibit a break at the critical line(s).

5.1.7.4 Interjections

Mathematicians (and statisticians) have irresistible urges to say "and so", "thus", "hence", "finally", and (my personal favourite, because of its questionable veracity) "and hence it is obvious", and they expect the interrupted mathematics to maintain its alignment.

amsmath provides for this by `\intertext{ ... }` as in

```
\begin{align*}
  f(x) &= \intinf f(x)\,dx\\
       &= \intinf e^{-2|x|}\,dx\\
  \intertext{but $x > 0$, so}
       &= \int_0^\infty e^{-2|x|}\,dx.
\end{align*}
```

$$
\begin{aligned}
f(x) &= \int_{-\infty}^{\infty} f(x)\,dx \\
&= \int_{-\infty}^{\infty} e^{-2|x|}\,dx \\
\text{but } x > 0, \text{ so} & \\
&= \int_{0}^{\infty} e^{-2|x|}\,dx.
\end{aligned}
$$

`\intinf` is not a standard command, but was defined:

```
\newcommand{\intinf}{\int_{-\infty}^\infty}
```

The interjection will be left-aligned to the left margin of the page (shown by the box) rather than the current environment, and so will sometimes need to be moved (by the insertion of space) to the right, for instance as in `\intertext{\hspace*{1cm}but }`.

There can be more than one `\intertext` in an environment.

In the LaTeX code above, you may have noticed that the limits were defined as `{\int_{-\infty}^\infty}`, not `{\int_{-\infty}^{\infty}}`. The latter version is "safer" until you are really confident that you understand how LaTeX works. The braces are not essential if the super-/sub-script is a single letter, number, keyboard character or predefined LaTeX command to produce a symbol. Braces **are** essential if the super-/sub-script consists of **more than one** letter, number, keyboard character or predefined LaTeX command to produce a symbol **or** consists of a user-defined command (which would need to be given in braces).

5.1.8 Equation numbering and reference

You can do many things with equation numbers, and the pages to read all about it are GMS pg 240–241.

amsmath provides the `\eqref` command: compare the effect, if we refer to one of the equations on page 65, of: 5.4 (`\ref{eq:2}`) and (5.4) (`\eqref{eq:2}`).

amsmath also provides `\tag` and `\tag*`, which allow you to give special (formatted) text labels (note that if you want a mathematical symbol as a label, you would have to put it in a maths environment, eg

`\tag{$\star$}`), or out-of-sequence numeric labels, and even to use a reference to another equation in a tag. `\tag*` causes the label in the equation to be typeset without any annotation (such as parentheses) that otherwise would be inserted automatically.

Both these commands can be used in both the starred and unstarred versions of the amsmath environments (so it is possible to insert a single equation "number" in an unnumbered multi-line environment):

```
\begin{align*}
  x &= 2\\
  y &= 2 \tag{\dag}\label{eq:new}
\end{align*}
```

$$x = 2$$
$$y = 2 \qquad (\dagger)$$

and the reference is (†)(`\eqref{eq:new}`) as usual.

`\notag` causes a line to be un-numbered (in an otherwise numbered environment).

For instance:

$$2x + 3y = 7 \qquad \text{(i)}$$
$$3x - y = 6 \qquad \text{(ii)}$$
$$6x + 9y = 21 \qquad \text{(3*i)}$$
$$6x - 2y = 12 \qquad \text{(2*ii)}$$
$$7y = 9$$
$$y = \tfrac{9}{7} \qquad \text{A}$$

```
\begin{align}
  2x + 3y &= 7  \tag{i} \label{i}\\
  3x - y &=  6  \tag{ii}\label{ii}\\ \hline
  6x + 9y &= 21 \tag{3*\ref{i}} \label{3}\\
  6x - 2y &= 12 \tag{2*\ref{ii}}\label{4}\\\\\hline
  7y &= 9       \notag\\
  y &= \tfrac 97       \tag*{\textsf{A}} \label{A}
\end{align}
```

And we can refer to (i) (`\eqref{i}`) or i (`\ref{i}`), (3*i) (`\eqref{3}`) or 3*i (`\ref{3}`) or (A) (`\eqref{A}`) or A (`\ref{A}`) and see the effect of each.

Automatic numbering of subequations is possible:

```
\begin{subequations}
 .....
\end{subequations}
```

causes all numbered equation environments within that scope to be numbered (4.9a), (4.9b), etc. assuming that the previous numbered equation was (4.8). The `subequations` environment can contain both text and mathematics environments, with the mathematics environments being either numbered or unnumbered.

A `\label` command immediately after the `\begin{subequations}` command will produce a `\ref` of the parent number "4.9", not the subequations (4.9a, etc.). The counters used by the subequations environment are `parentequation` and `equation` and standard uses of `\addtocounter`, `\setcounter`, etc. are possible with those counter names.

```
\begin{equation}
   y = b_0 + b_1x \label{eq:last}
\end{equation}
\begin{subequations}\label{eq:subs}
Some text
\begin{align}
  y &= x + 2 \label{eq:sub1}\\
  y &= 2x - 1\label{eq:sub2}
\end{align}
More text
\begin{equation}
  y = ax^2 + bx + c \label{eq:sub3}
\end{equation}
\end{subequations}
More text
\begin{equation}
  y = 2x^2 -5x + 3 \label{eq:next}
\end{equation}
```

$$y = b_0 + b_1 x \qquad (5.25)$$

Some text

$$y = x + 2 \qquad (5.26a)$$
$$y = 2x - 1 \qquad (5.26b)$$

More text

$$y = ax^2 + bx + c \qquad (5.26c)$$

More text

$$y = 2x^2 - 5x + 3 \qquad (5.27)$$

Then references can be made to (5.25) (`\eqref{eq:last}`), (5.26) (`\eqref{eq:subs}`), (5.26a) (`\eqref{eq:sub1}`), ... (5.26c) (`\eqref{eq:sub3}`) and (5.27) (`\eqref{eq:next}`).

How equation numbers are represented depends, in part, on the class of the document: in `article` class documents, equations are numbered from 1 to n across the whole document. In `book` and `report` classes, equations are numbered within chapters. If you want to change this default numbering system, you can insert a line along the lines of `\numberwithin{equation}{section}` in the preamble to cause equations to be numbered within sections in an `article` class document (the counter `equation` is reset at the start of each section, and the representation of the equation counter, `\theequation`, is redefined to be `\thesection.\arabic{equation}`).

It is possible to use `\numberwithin` for other counters too, although you may get some unexpected results.

5.1.9 Theorems and other animals

5.1.9.1 "Vanilla" LaTeX

The commands provided give control of:

- the name (used in `\begin{name}`)
- naming (ie what is typeset in the numbered heading) of the environment,
- options for the numbering to be either within a section (or other sectioning "bit") or
- in sequence with another theorem-like environment (but **not** both—the idea being that all those environments numbered in sequence would have the same numbering style as the first environment of that type).

The default shape for the theorem body is italic (emphasized shape), and the default theorem header font is boldface.

So

`\newtheorem{thm}{Theorem}`	Defines environment `thm` with label **Theorem #**. All n `thm`s numbered in sequence from 1 to n

The above is the most straightforward definition. More complicated are:

`\newtheorem{thm}{Theorem}[section]`	As above, but the theorems will be numbered within a section, so that the first theorem in §6 is **Theorem 6.1**, etc.
`\newtheorem{defn}[thm]{\sc Definition}`	Defines environment `defn` labelled in sequence with (and just like) the environment `thm` (say, in the sequence **Theorem 6.1**, DEFINITION 6.2, DEFINITION 6.3, **Theorem 6.4** ...), but with a different shape for the name.
`\newtheorem{eg}{Example}[chapter]`	Numbered separately from `thm`, and within chapter, not section.

The definitions should appear in the preamble, and are used as in:

DEFINITION 5.1.1 (OPTIONAL NAME) *A theorem-like environment is one that is given a numbered header, and has the option to be numbered in sequence with similar environments.*

```
\begin{defn}[Optional Name]
A theorem-like environment is one that
is given a numbered header, and has
the option to be numbered in sequence
with similar environments.
\end{defn}
```

and then later

Theorem 5.1.2 *A theorem is something non-trivial and useful that can be proved to be true.*

and before and after one of these it often helps to have

```
and then later
\begin{thm}
  A theorem is something .... be true.
\end{thm}
and before and after one of
these it often helps to have
```

Example 5.1 *An example to illustrate the concepts discussed abstractly, as most people reach understanding of the abstract through exposure to the concrete.*

```
\begin{eg}
  An example to .... the concrete.
\end{eg}
```

It is possible to have a different shape for the body of the theorem, by putting in the shape of choice (and remember that the `\em` declaration works like a toggle switch, turning the emphasised shape on and off alternately):

Example 5.2 (Use of `\em`) An example to illustrate an upright shape (any of the other shapes could have been invoked by use of the appropriate declaration) body font.

```
\begin{eg}[Use of
  \texttt{\symbol{'134}em}]
  {\em An example to illustrate
.... body font.}
\end{eg}
```

5.1.9.2 theorem package

This improves on the above by offering control over:

1. Space above and below the environment (see GMS, pg 253).
2. Font of the body of the theorem, determined by `\theorembodyfont{`*`size, shape, series and family declarations`*`}` command.
3. "Style" of the theorem—explained below, determined by `\theoremstyle{`*`style`*`}` command.
4. Font of the theorem header, determined by `\theoremheaderfontfont{`*`size, shape, series and family declarations`*`}` command.

You should read GMS pg. 252, but some of the possibilities are explored below.

It is important to note that you can have several `\theorembodyfont` commands in a document (a new one will replace the definition of an earlier one, or else use { ... } to confine the effect of any one of them to what lies within the parentheses), and several `\theoremstyle` commands in a document, with the effects controlled in the same way. However, the `\theoremheaderfont` command is global, so that there can be only **one** per document, and it will affect the font of **all** the theorem headers in the document (presumably because it's assumed that the same style would be wanted for all of them) although it is possible to change some, as was done for `defn` in the previous subsection.

The three main *`style`*s for theorems are:

`plain`	Like the "vanilla" LaTeX, but with control over the space. This is the default.
`break`	The heading and number appear alone on the first line, the body starts on a new line.
`change`	The number and header are interchanged (number appears first); in most other respects like `plain`.
`margin`	Like `change`, but the number is in the left margin.

They are also offered in combination in styles `marginbreak` and `changebreak`.

Note that the default theorem bodyfont of `plain` is italics (as in LaTeX), but for all other styles is slanted (`\slshape`).

If we make the definitions (in the preamble):

```
{\theorembodyfont{\rmfamily\footnotesize}\newtheorem{rmk}{Remark}
{\theoremstyle{changebreak}\newtheorem{nt}[rmk]{Note}}}
{\theoremstyle{margin}\newtheorem{ex}[eg]{Exercise}}
```

we can see the effects:

Remark 1 (Hopeful) This remark should have a footnotesized roman body, and the default kind of numbering for a `book` class document (from 1 to n across the document).

```
\begin{rmk}[Hopeful]
   This remark should ...
 \end{rmk}
```

2 Note
This note should have a footnotesized roman body font, like the Remark above, displayed heading, with the number before the heading name instead of after, and Notes and Remarks are numbered in the same sequence.

```
\begin{nt}
   This note should ...
 \end{nt}
```

5.3 Exercise *Exercise (*ex*) has* margin *style, which puts the number in the margin, and uses normalsized* **slanted** *body font, and is defined to be numbered in sequence with* eg*, and hence, like* eg*, within chapter.*

```
\begin{ex}
  Exercise ({\tt ex})
  has ..
\end{ex}
```

5.1.10 Size and "bolding" declarations in maths

5.1.10.1 Size

Size is affected by the local declarations: `\displaystyle`, `\textstyle`, `\scriptstyle` and `\scriptscriptstyle`. Their effect can be seen in:

$$x\ x\ x\ x.$$

```
\[\displaystyle{x}\ \textstyle{x}\ %
\scriptstyle{x}\ \scriptscriptstyle{x}.\]
```

and more clearly in

$$\frac{x}{y}x\ \frac{x}{y}x\ \frac{x}{y}x\ \frac{x}{y}x.$$

```
and more clearly in
\[\displaystyle\frac{x}{y}{x}\ %
\textstyle\frac{x}{y}{x}\ %
\scriptstyle\frac{x}{y}{x}\ %
\scriptscriptstyle\frac{x}{y}{x}.\]
```

5.1.10.2 Bold characters

`\boldmath` is a local declaration (see Section 3.2, page 28) that can only be used in LR or paragraph mode (see Section 3.10, page 41) . It causes every mathematical expression in its scope to be typeset in bold.

One possible use would be in slides if the text is to appear in bold (the maths will then match the text).

`\unbold` has the opposite effect.

The `\boldsymbol{`*arg*`}` command (amsmath only) produces *arg* in bold (this affects all letters and mathematical symbols of all types).

The `\mathbf{`*arg*`}` command will cause all letters, numbers and uppercase Greek letters (well, maybe) to appear upright and bold. Any other symbols (of any type) are not affected in any way.

5.2 Not quite maths

5.2.1 New commands

LaTeX and $\mathcal{AMS}$-LaTeX provide many commands and environments, but they tend to be fairly basic and general ones that are used to construct more complex formulae, and they also tend to have long and readily understandable names. You'll find that you type some things over and over, and it can be really useful to construct a file, kept in your root directory of LaTeX files, that contains a collection of your own labour-saving commands, and cleverly written special effects. Say you call your file `gems.tex`. If you put an `\input` statement:

```
\input{gems} % or \input{/your whole path .../gems} if necessary
```

in the preamble of each document you make, all your "gems" are immediately available for use.

An alternative is to save your "gems" in `gems.sty`, and, so long as your latexinputs path[2] includes the directory containing this file, list `gems` amongst the packages in a `\usepackage` command.

Some obvious possibilities for this file (i.e. some of my "gems") include

```
\newcommand{\half}{\ensuremath{\frac{1}{2}}}
\newcommand{\D}{\displaystyle}
\newcommand{\T}{\textstyle}
\newcommand{\utilde}[1]{\mathrel{\mathop{#1}\limits_{^{\sim}}}}
```

[2] Add a line like
`setenv TEXINPUTS /u/staff/edith/texinputs::.`
to your `.cshrc` file, among any other `setenv` lines in the file.

```
\newcommand{\utsub}[2]{\utilde{#1}_{\raisebox{1.4ex}{${\scriptstyle {#2}}$}}}
\newcommand{\vect}[1]{\ensuremath{\utilde{#1}}}
                                    %make vects and matr. with tildes
\newcommand{\mat}[1]{\ensuremath{\mathbf{#1}}} %vectors/matrices A, B, x, etc
\newcommand{\gmat}[1]{\mbox{\boldmath${#1}$}}  %   "         with Greek names
\newcommand{\trans}{^{\rm T}}
\newcommand{\Xson}{\ensuremath{X_{1},\,\ldots \, , X_{n}}}%sample of X1 to Xn
%sample where var name(#1), starting (#2) and ending (#3) subscripts are given:
\newcommand{\sample}[3]{\ensuremath{#1_{#2},\,\dots,\,#1_{#3}}}
%sample where var name(#2),  and ending (#3) subscripts are given, if starting
%subscript (#1) us omitted, the default value is 1:
\newcommand{\dsample}[3][1]{\ensuremath{#2_{#1},\,\dots,\,#2_{#3}}}
\newcommand{\hn}{\ensuremath{H_{0}}}               % H0
\newcommand{\ha}{\ensuremath{H_{1}}}               % H1
\newcommand{\inv}{\ensuremath{^{-1}}}
\newcommand{\nms}{\ensuremath{N(\mu,\,\sigma^2)}}
\newcommand{\var}{\ensuremath{\mathop{\mathrm{Var}}}}
\newcommand{\cov}{\ensuremath{\mmathop{\mathrm{Cov}}}}
\newcommand{\fxx}{\ensuremath{f_X(x)}}
\newcommand{\ph}{\ensuremath{\hat{p}}}
\newcommand{\mc}[1]{\ensuremath{\mathcal{#1}}}
\newcommand{\pr}{\ensuremath{\mathrm{Pr}}}
\newcommand{\mbie}{\text{i.e.}}
\newcommand{\dydx}{\frac{dy}{dx}}
\newcommand{\pdydx}{\frac{\partial y}{\partial x}}
\newcommand{\intinf}{\int_{-\infty}^\infty}
\newcommand{\bR}{\ensuremath{\mathbb{R}}}    % R for Real #s
\newcommand{\bI}{\ensuremath{\mathbb{Z}}}    % Z for integers
```

To formalise what can be observed from the examples, the `\newcommand` statement has the form:

`\newcommand{`*cmd*`}[`*args*`][`*opt*`]{`*def*`}`

If you are changing an existing command, use `\renewcommand`, which has the same form.

cmd names the command: it starts with a `\`, followed by either a unique series of upper and/or lower case letter(s) or by a single non-letter (e.g. `\!`, `\|`, `\=`). The names are case sensitive. Note that numbers **cannot** be used as part of a command name.

args Number between 1 and 9, giving the number of arguments for the command. Default is *args* = 0.

opt Optional—if absent, all arguments are mandatory. If present, it gives the default value of the first argument, which is optional. (ie, if you want an optional argument, it must be the first, and the default value must be given in *opt*.)

def Definition of the command; the arguments are denoted by `#1`, `...`, `#n`.

5.2.2 Braces and command arguments

It is worth mentioning at this point the conventions for using commands. Let's illustrate this with `\frac`, which takes two arguments.

1. If the numerator and denominator (the 2 arguments) are single digits, the { }s are not necessary, so that `\frac23` and `\frac{2}{3}` are equivalent.
2. If the numerator and denominator are single letters, there must be a space between "`frac`" and the letters, so that `\frac{a}{b}` and `\frac ab` are equivalent (`\fracab` would be interpreted as the name of a command).
3. If either the numerator, or the denominator (or both) involve more complicated expressions or longer numbers, braces should be used.

The same principles apply to commands with any other number of arguments. The above notwithstanding, the safe rule for the new LaTeX user is "when in doubt, brace", in that too many matched braces

never hurt anyone, but too few can at best have unanticipated effects and at worst cause baffling error messages and a lack of output.

5.2.3 New environments

If you wish to provide your documents with a unique (or just consistent) look, it can be useful to define you own proof, or example, or whatever, environments.

When would an environment be better than a command? Mainly when you want to insert formatting or text or instructions both before and after some text or other characters. Or you want to build a new environment on the basis of an existing environment (like a new kind of `list` environment, or array-type environment).

General structure of a definition of an environment is:

`\newenvironment{`*name*`}[`*narg*`][`*opt*`]{`*begdef*`}{`*enddef*`}`

where *narg* ≤ 9, *opt* is as for `\newcommand`; *begdef* is what sets up the start of the environment; *enddef* is what ends it off. Note that all the *narg* parameters must be used in *begdef*.

To change an existing environment, use the `\renewenvironment` command, which has exactly the same structure.

Example Say, you wanted unnumbered examples[3] in notes to appear with a line ruled before and after the example and a header of *Example*.

```
\newenvironment{exaa}                %name of env.
  {\rule{\linewidth}{0.02cm}\\[0.5ex]  %def of start
    {\large\em Example}\qquad}
    %def of end, with line on new page discouraged
  {\nopagebreak[4]\\*[-0.5ex] \rule{\textwidth}{0.02cm}}

\begin{exaa}
  This is an example of an example,
   .....and over the lines, resp.
\end{exaa}
```

Used:

Example This is an example of an example, according to the specifications given above. There should be adequate space under and over the lines, resp.

Proof MkI A simple (and occasionally problematic) example of a proof environment is

```
\newenvironment{proof}
  { {\em Proof}\\[.5ex]}
  {\hfill $\square$}

  \begin{proof}
    The proof of a theorem ... the line.
  \end{proof}
```

Proof
The proof of a theorem is usually indicated by a small, empty square, at the end of the line. □

Which illustrates the occasional problem.

Proof MkII A better version, also illustrating a run-in header (no new line), is

```
\renewenvironment{proof}
  %\renew... as "proof" has been
  % defined in this document.
 {{\em Proof}\qquad}
 {\hspace*{\fill}\nolinebreak[1]%
   \hspace*{\fill}$\square$}

 \begin{proof}
   The proof ... line-end.
 \end{proof}
```

Proof The proof of a theorem is usually indicated by a small, empty square, at line-end. □

[3] Numbered examples would most easily be made with a theorem-like environment, see page 71. There is an example of a new, numbered, referenced list-type environment in Section 8.8.4, page 144.

Proof MkIII If you wanted to be picky, you might want your proofs indented (on the left) and this could be achieved using `description` in the new environment. The default label is in `bfseries`, so our proof environment will include a redefinition of the labels for `description`, which will not affect any other `description` environments used (as it's within `proof`)—as can be seen by the typeset version of this document.

```
\renewenvironment{proof}
  {\renewcommand{\descriptionlabel}[1]%
   {\hspace{\labelsep}\textit{##1}}
\begin{description}\item[Proof]}
    {\hspace*{\fill}\nolinebreak[1]%
      \hspace*{\fill}$\square$%
     \end{description}}

 \begin{proof}
    The proof ... too full.
  \end{proof}
```

Proof The proof of a theorem is usually indicated by a small, empty square, at the end of the line, on the next line if the first is too full. □

This example has an interesting feature: the redefinition of a command inside the definition of an environment. This gives the possibility of two sets of optional arguments, between which LaTeX must be able to distinguish. The means decided on is that the outer definition's arguments (in this case for `proof`, and in fact there are none) would be denoted in the definition by `#1, #2` etc., and the inner definition's arguments would be denoted by `##1, ##2` etc.

Remember If you wanted all the most important points, scattered through a set of notes, to appear in a box, with the heading **Remember**, one way to achieve this is an environment `nb`, which is based on a tabular environment of one paragraph. Inter-paragraph spaces will then have to be entered using `\\[`*`len`*`]`.

```
 \newenvironment{nb}
  {\begin{center}
    \begin{tabular}{|p{0.95\linewidth}|}
          \hline  \\[-1.5ex]
      {\bf\Large Remember}\\[1ex]}
  {\\[.8ex]\hline
     \end{tabular}\end{center}}

  \begin{nb}
     The ... ``paragraphs''.

     Double carriage returns
     (don't) work as above.\\[1ex]
     \verb2\\[1ex]2 was used the second time.
   \end{nb}
```

Remember

The main thing to remember in this environment is to add more space between paragraphs. This box contains three "paragraphs".
Double carriage returns (don't) work as above.

`\\[1ex]` was used the second time.

"Template" And finally, you could construct an environment to make a new-look school tutorial-header, along the lines of:

```
\newenvironment{mcshead}[3]
% Need to supply 3 bits of information for a header line. Say:
% Course, Title for the document, and Date.
 {
  \begin{center}
    {\large\sc School of Mathematical and Computing Sciences}\\[.5ex]
    {\sc Te Kura P\=angarau, Rorohiko}
```

```
  \end{center}
\rule[2mm]{\linewidth}{0.02cm}
{\bf #1} \hfill {\bf #2} \hfill {\bf #3} \\
% The 3 bits of information positioned against the margins (#1 & #3)
%  and in the centre of the line (#2).
\rule{\linewidth}{0.02cm} \\
}
{
  \begin{center}
    * * * * *
  \end{center}
}
```

Which would start your document with the header, and end it with a row of * * * * *s. For example:

```
\begin{mcshead}{MATH 104}{Assignment 3}{1998}
  Answer all questions. Hand in to the boxes.
  \begin{enumerate}
  \item Prove \ldots
  \item If \ldots
  \end{enumerate}
  \begin{center}
    {\bf     Tutorial Questions}
  \end{center}
  \begin{enumerate}
  \item etc.
  \end{enumerate}
\end{mcshead}
```

Which produces:

School of Mathematical and Computing Sciences
Te Kura Pāngarau, Rorohiko

MATH 104 **Assignment 3** **1998**

Answer all questions. Hand in to the boxes.

1. Prove ...
2. If ...

Tutorial Questions

1. etc.

* * * * *

If you had no closing text, the same effect could be obtained using a command with 3 arguments.

Part II

Gaining Control

Chapter 6

Arrays, boxes, space, fragility and pictures

6.1 Arrays and tables

Lining things up in columns can be done using `tabbing` or `tabular` environments in any mode, and an `array` environment in maths mode. Column-like effects (for short sections of text) can also be achieved using a `minipage` environment. How to typeset all or at least large sections of a document in two or more columns is discussed in Section 8.5.

6.1.1 `tabbing`

Characteristics:

1. Starts on a new line, as does the text following the environment.
2. Starts flush left against the current left margin.
3. Good for aligning 1 or more columns where no line wraps are needed (all text fits within a single line).
4. Can only be used in paragraph mode.
5. Cannot be nested (but new tab stops can be set on any line).
6. Can be split across 2 (or more) pages (pagebreaks inserted by LaTeX).

A full discussion of all the options is given in Lamport pg 203.

The most simple (and therefore frequently used) features are:

`\=`	Sets a tab stop; need not be in the first line of text in `tabbing` environment.
`\\`	Ends line. Space between lines can be elongated with `\\[len]` (see page 36).
`\>`	Move to next tab-stop.
`\kill`	Throw away the text in this line. Use to set the first tabstop(s) using text that is not printed—perhaps the longest text on any line between pairs of stops.

For example: A simple table can be constructed, along the following lines:

... end of preceding paragraph.

Cost	**Cause**	**Effect**
New	Unknown	Increase of $5,000
Carried over	Indeterminate	Decrease of $1,000
Net	Hopeful	This amount can only be determined at year end.

Following paragraph ...

```
\ldots end of preceding paragraph.
```

```
\begin{tabbing}
  Carried over \quad\= Indeterminate \quad\= \kill
  {\bf Cost}        \> {\bf Cause}        \> {\bf Effect}\\[.5ex]
  New               \> Unknown            \> Increase of \$5,000\\
  Carried over      \> Indeterminate      \> Decrease of \$1,000\\
  Net               \> Hopeful            \> This amount can only be ... year end.
\end{tabbing}
Following paragraph \ldots
```

Note that if you want to convert your document to HTML, you should **not** use `tabbing` as there is no equivalent in HTML with the result that the environment is either ignored or mangled (depending on the translator).

6.1.2 `tabular` and `array` environments

Characteristics:

1. Do not start on a new line. Following text continues on the same line. Often irritating, occasionally very useful.
2. Start slightly indented from the left margin. This can be overcome by putting an `\hspace*{-1ex}` —the exact length is half the intercolumn space—in front of the `\begin` statement, or by starting the column format with `@{}` (another `@{}` at the end of the column format will suppress an equal space to the right of the `tabular` environment).
3. Can cope with text that is longer than one line (using `p` column type).
4. Very flexible—can achieve many effects; vertical and horizontal lines easy to draw between the rows and columns.
5. `tabular` can be used in any mode; `array` can only be used in maths mode.
6. Can be nested.
7. Cannot be split across 2 pages (see longtable for the solution to this potential problem).
8. - The scope of a local declaration, such as `\em`, is the "cell" in which it is made, so to get a row (or column) typeset in bold, a `\bf` is needed in each cell of the row (or column).
   - Similarly, in `tabular`, any textstyle mathematical environment must be opened and closed in the cell in which it appears.
   - Any "cell" in an `array` environment that contains a `\left` delimiter-enlarging command **must** also contain a matching `\right`—use `\right.` if neccessary and put a `\left.` in the "cell" that contains the `\right` associated with the actual right delimiter.

6.1.3 Basic `tabular` and `array` environments

The only limit to what you can achieve with these environments is your imagination. Well, nearly, anyway. Some of the difficulties, like decimal alignment and really long tables are addressed by packages, which are discussed later.

`tabular` has a starred version which is of fixed width, but requires stretchy space between the columns (`tabular` is of flexible width, determined by the column entries, with, by default, fixed space between columns although you can change the space between any pair of columns).

The general form of the environments is:

```
\begin{array}[pos]{cols} ...  \end{array}
\begin{tabular}[pos]{cols} ...  \end{tabular}
\begin{tabular*}{wdth}[pos]{cols} ...  \end{tabular*}
```

- In the body of the table, columns are delimited by `&` , and lines are ended by `\\[`*`len`*`]`.

  Note that any formatting used (like `\bf` or `\em`—local declarations—or `$ ... $`) applies only within a single cell on the table. This means that braces are not necessary around text formatting declarations (but the declaration is required in each cell to which it is to apply), and also that both `$ ... $`s must be within the same cell.

  As mentioned above, any "cell" in an `array` environment that contains a `\left` delimiter-enlarging command **must** also contain a matching `\right`—use `\right.` if neccessary and put a `\left.` in the "cell" that contains the `\right` associated with the actual right delimiter.

- For an example and discussion on `tabular*`, see Lamport, pg 204.

- *`pos`* is the vertical positioning—`c` is the default, with the other possibilities being `t` and `b`. May not work too well just after `\item` in a list-type of environment, particularly if paired with a `minipage` containing graphics. Get around that problem most simply by inserting a leading line of text, or with some trial and error, by using `\raisebox` where the height is measured in terms of `\height` or `\totalheight` (see Section 6.2.2.1).

- *`cols`* specifies column formatting, with the most common options being `l`, `r` and `c` (left, right and centre justified, respectively). `|` causes a vertical line to appear between columns, and `||` produces a double vertical line.

  A useful alternative for *`cols`* is `p{`*`wd`*`}`, particularly if *`wd`* is defined as a multiple of `\linewidth`. Text is typeset in a parbox of the specified width, so, if you want to start a new line *within* the column, you may find that there are circumstances where this can only be done using a `\newline` (`\\` may not do what you want as it will end the box and start the next line in the table). Can contain `minipage, array`, or `tabular` environments.

  Repeated formats can be specified using `*{`*`num`*`}{`*`cols`*`}`.

  `@{ ...}` replaces the intercolumn space with what is in the brackets. If you wanted two columns separated by = signs, with a normal inter-word space between columns, this would be achieved by `@{\␣=\␣}`—but this will produce some unexpected effects if you want an = between columns in *almost* but not exactly every line. A wider inter-column space is achieved by something like `@{\qquad}` or `@{\hspace{2em}}`. See Lamport Pg 205 for other usages.

  To suppress the (automatic) indentation, start and/or end *`cols`* with `@{}`. For instance, `\begin{tabular}{@{}ll}` defines a table of two left-aligned columns that will be flush with the left margin.

- `\multicolumn{`*`num`*`}{`*`col`*`}{`*`text`*`}` places *`text`* across *`num`* columns;

  - *`col`* is `l`, `r` or `c`, and can contain `@`-expressions and `|` characters.
  - *`num`* = 1 can change formatting in a single column.
  - A `|` in the column format of the preceding column (for the whole table) will usually be observed (except when the multiple columns start at the beginning of a line), giving a vertical line before the `\multicolumn`, but all `|`s in the original column format that would be within or immediately after the multiple columns will be suppressed, so if you want a `|` to follow the multi-column, you must say so in `{`*`col`*`}`, e.g. as `{c|}`.
  - `\multicolumn` must appear at the start of a row, or immediately after "`&`" within a row.

- **Lines**. `\vline` is equivalent to `|`, but causes a vertical line within a column.

  `\hline` gives a horizontal line right across the table.

  `\hline\hline` gives a double horizontal line.

  `\cline{`*`c1-c2`*`}` gives a horizontal line from the beginning of column *`c1`* to the end of column *`c2`*. Double lines are again specified by repeated commands.

- A final `\\` is *only* needed if a horizontal line is to be drawn across any of the table, under the last line of text. Compare the raw LaTeX for the tables in these pages with, say, that for the arrays on page 63.

- **Blank lines**. Leave **NO** blank lines in an `array` or `tabular` environment. They can cause errors associated with some very unclear error messages.

To illustrate the above:

Top aligned table

Heading	
Row 1	value
Row 2	value

Default (center) aligned

Col	Values	Text
1	20	Each of these cells in the table should be typeset in parboxes.
2	30	They will have the same width, but may have differing heights, depending on the length of the text.

Bottom aligned table

Col	Values	
1	20	30
2	30	10
	50	40
3	15	25
Total	65	65

```
\hrulefill\begin{tabular}[t]{|*2{c|}}
  \multicolumn{2}{c}{Top aligned table}\\\hline
  \multicolumn{2}{|c|}{\bf Heading}\\\hline
  Row 1 & value\\
  Row 2 & value\\ \hline
\end{tabular}\hrulefill
\begin{tabular}{||*2{c|}p{0.2\linewidth}||}
  \multicolumn{3}{c}{Default}\\
  \multicolumn{3}{c}{(center) aligned}\\\hline\hline
  \bf Col & \bf Values &  Text\\ \hline
  1 & 20 & Each of these cells in the table should be typeset in parboxes.\\[1ex]
  2 & 30 & They will have the same width, but may have  differing heights, depending on
           the length of the text.\\\hline\hline
\end{tabular}\hrulefill
\begin{tabular}[b]{||*3{c|}|}
  \multicolumn{3}{c}{Bottom aligned table}\\\hline\hline
  {\bf Col} & \multicolumn{2}{c||}{\bf Values}\\ \hline
  1 & 20 & 30\\
  2 & 30 & 10\\ \cline{2-3}
    & 50 & 40\\\hline
    3 & 15 & 25\\\hline
  Total & 65 & 65\\ \hline\hline
\end{tabular}\hrulefill
```

The lines drawn outside and between the tables indicate the position of the base line (the base line of the "text" for that line) that passes along the bottom of the first line in the first table, through the centre of the second, and along the bottom of the third table. See further discussion on the reference points of boxes in Section 6.2.2 on page 92.

6.1.4 Changing spacing in `array` and `tabular` environments

Occasional changes in vertical spacing are made by `[len]`, and some control over inter-column space is given by `@{ ... }`.

For instance, the body of the third table on the previous page looks fairly cramped, and one solution, giving extra space above and below each horizontal ruled line, is as follows:

Col	Values	
1	20	30
2	30	10
	50	40
3	15	25
Total	65	65

```
\begin{tabular}{||c|c@{\quad}|@{\quad}c||}\hline\hline
    & \multicolumn{2}{c||}{} \\[-2ex]
  {\bf Col} & \multicolumn{2}{c||}{\bf Values}\\[.2ex]\hline
    &    & \\[-2.3ex]
  1 & 20 & 30\\
  2 & 30 & 10\\[.2ex] \cline{2-3}
    &    & \\[-2.3ex]
    & 50 & 40\\[.2ex]\hline
    &    & \\[-2.3ex]
  3 & 15 & 25\\[.2ex]\hline
    &    & \\[-2.3ex]
  Total & 65 & 65\\[.2ex] \hline\hline
\end{tabular}
```

Changes that are to apply to all vertical or horizontal spaces are best made using style parameters, which are given outside the environment (but style parameter changes that are to apply to a nested environment would be made inside the outer environment, outside the inner environment). If the changes are to apply to a limited number of environments, put a { before the first change and } after the last environment to be affected.

`\arraycolsep` or `\tabcolsep` have length = half the distance between columns in an `array` or `tabular` environment, respectively. The value is changed by a statement such as:

```
{\arraycolsep = 2em
.....
}
```

The default values are 5pt (for `array`) and 6pt (for `tabular`).

`\arrayrulewidth` controls the width of vertical and horizontal lines. Its value would be changed as above, and the default width of lines is 0.4pt.

`\doublerulesep` controls the space between double vertical and horizontal lines. Its value would be changed as above, and the default space between lines is 2pt.

`\arraystretch` gives a proportional control over space between lines in both `array` and `tabular` environments. The default is 1 (ie the space between lines is the usual (1×)`\baselineskip`).
A new value is set as in:

```
{\renewcommand{\arraystretch}{1.8}
.....
}
```

which makes the distance between lines 1.8 times that of the default.

An alternative version for the table above, modifying the spaces globally, is:

Col	Values	
1	20	30
2	30	10
	50	40
3	15	25
Total	65	65

```
{\tabcolsep=12pt
\renewcommand{\arraystretch}{1.4}
\begin{tabular}{||*3{c|}|}\hline\hline
  {\bf Col} & \multicolumn{2}{c||}{\bf Values}\\\hline
  1 & 20 & 30\\
  2 & 30 & 10\\\cline{2-3}
    & 50 & 40\\\hline
  3 & 15 & 25\\\hline
  Total & 65 & 65\\ \hline\hline
\end{tabular}}
```

6.1.5 Packages

See GMS pg. 104 ff. and for local "on-line" documentation, see page 161 of these notes. A sample of my favourites (there are others) is:

- array (GMS pg. 105). Put `array` in your list of `\usepackage`s. The package offers:
  - `m{`*`wdth`*`}` and `b{`*`wdth`*`}` which are like the `p{`*`wdth`*`}`, but the three column specifications have vertical alignments that are **m**iddle, **b**ottom and to**p**, respectively.
  - Automatic insertion of *`decl`* before (`>{`*`decl`*`}`), after (`<{`*`decl`*`}`) or between (`!{`*`decl`*`}`) columns. Clever use of this would be to achieve a particular special effect on a column, say, large, or bold, or italics, as in

    ```
    \begin{tabular}{|>{\large}c|>{\bfseries}l|>{\itshape}c|}
    ....
    \end{tabular}
    ```

    which would give a table consisting of 3 columns, the first centred, with all its entries one size bigger than usual, the second left justified, with all its entries in boldface, and the third centred, with all its entries in italics, and having vertical lines on the edges of the table, and between columns.

    Another useful application of this is to construct a column that is in math mode (in a text `tabular` environment), as in

    ```
    \begin{tabular}{>{$}c<{$}l}
    .....
    \end{tabular}
    ```

    where maths mode is turned on before the centred column, and off after it (so the left justified column is back in text mode).
  - A clever way of re-setting narrow columns to be typeset ragged right is shown in GMS on page 108.

    An alternative is discussed on page 92 of these notes.
- longtable (supertabular is not installed on our machines) will typeset a table across 2 or more pages. This environment is a combination between `tabular` and `table` environments, doing, in a sense, the work of both (but does not exist in *-ed form). The latest version (on the machines) is not quite according to GMS pg 122 ff.

  longtable allows you to:
  - have a really long table.
  - Specify a caption, which will be added to the `\listoftables` list (unless you specify `\caption*{...}`, in which case the caption is unnumbered and not listed) and which can be referenced (using `\label{...}` and `\ref{ ... }`). It uses the same counter as `table`, so the correct sequence of table numbers is maintained.
  - Specify the text/formatting that appears at the head (first line) of each page-section of the table—say, the column titles and various `\hline`s.
  - Specify the text/formatting that appears at the foot (last line) of each page-section of the table—say "continued".
  - Specify the text/formatting to appear at the very end of the table—say the final `\hline`.
  - Use `\\*` as well as `\\`; the former inhibits a page break before the new line.
  - Use `\pagebreak`, which controls where the page breaks occur; otherwise they occur where LaTeX sees fit.

  A useful alternative for the `l r c` options specifying the justification of the columns in the table, is `p{`*`wd`*`}`, particularly if *`wd`* is defined as a multiple of `\linewidth`. Text in that "cell" is typeset in a parbox of the specified width, so, if you want to start a new line *within* the column, you may find

that there are circumstances where this can only be done using a `\newline` (`\\` may not do what you want as it will end the box and start the next line in the table). Columns specified in this way can contain `minipage`, `array`, or `tabular` environments.

There is a nice example on pg. 128 of GMS, and Table 8.1 on page 123 of these notes was made using longtable (as were several other tables). The table in this document (and its footnotes) is defined:

```
\begin{center}
  \begin{longtable}{|l|l|}
% Make the caption and provide a label. \caption* gives an unnumbered caption,
% but the counter table is still incremented.
    \caption{Fragile and Robust Commands}\label{tab:long}\\ \hline
% Make the heading for the first line of the table.
      Fragile & Robust \\ \hline
    \endfirsthead
% Make the heading to be used on all continuation pages of the table.
      \hline\multicolumn{2}{|l|}{\em Continued from previous page}\\ \hline
      Fragile &  Robust \\ \hline
    \endhead
% Make the footer for the end of all pages within the table.
      \hline\multicolumn{2}{|r|}{\em Continued on next page}\\ \hline
    \endfoot
% Make the end of the table.
      \hline
    \endlastfoot
% Start the actual table.
      \verb1\(1 ....
          ..............
      & \verb1\symbol1
% The final horizontal rule is provided by the \endlastfoot.
\end{longtable}
% There were 2 footnotes, made with \footnotemark. Now correct the counter and
% specify the text to go in the footnotes. See Note 3 below.
{\addtocounter{footnote}{-1}
\footnotetext{{\tt $\backslash$value\{}\opt{ctr}{\tt \}} gives current value of
    \opt{ctr}---will print it or assign it to another counter.}
\addtocounter{footnote}{1}
\footnotetext{But {\tt $\backslash$label} can be used in a caption or any of
                the sectioning commands.}}
\end{center}
```

Note:

1. If you change entries to the table, thus affecting column widths (using one of `l r c`), you may need several compilations to get the changes across all pages (column widths are stored in `.aux` files, so it takes several (up to 3) runs to update the information—you should get a reminder in the `log` during typesetting) and some strange effects in the table in the interim.
2. Use of `longtable` environment increments the `table` counter whether a caption is created or not. One way around this is to place an

   `\addtocounter{table}{-1}`

   after each un-numbered `longtable` environment.
3. See Section 8.3 for details on making footnotes in a box (like a `longtable`).

- dcolumn offers a more elegant way of decimal-aligning a column than the obvious first-principles method in LaTeX of using `{r@{.}l}` which requires 2 columns to represent one number, and prints a . between the columns on every line. GMS pg. 129.

- hhline handles double lines more neatly than LaTeX, if you like tables with multiple double lines.

- rotating enables you to print individual figures and tables "sideways", as well as to rotate text or other boxes. See Section 8.4.2, page 129 of these notes, and GMS page 325.

6.1.6 Making your own `textstyle` aligned mathematics environments

As discussed on page 68, neither LaTeX nor amsmath provide `textstyle` aligned environments, although the "inner" aligned environments, such as `aligned` can be used in a `textstyle` environment.
If you really wanted to have a one-step mathematical environment, the simplest option would be to define one using the `array` environment:

```
\newenvironment{daln}{$\arraycolsep=.2ex\begin{array}[t]{rl}%
                      {\end{array}$}
```

This environment:

1. has a small gap between columns (just like `align*`—as `\arraycolsep=.2ex`);

2. has a default of `\baselineskip` between successive rows (`\arraystretch` is not altered);

3. is a `textstyle` mathematics environment (with all that that implies).

This means that it works fine in

1. $3x^2 + 4x = 1$
   $\therefore \quad 3x^2 + 4x + 1 = 0$
   $\therefore \quad (3x+1)(x+1) = 0$
   $\therefore \quad x = -\frac{1}{3} \quad \text{or} \quad x = -1$

```
\begin{enumerate}
  \item
    \begin{daln}
       3x^2 + 4x                        &= 1\\
       \therefore\qquad 3x^2 + 4x + 1 &= 0\\
       \therefore\qquad (3x+1)(x+1)   &= 0\\
       \multicolumn{2}{r}{\therefore \qquad
         x = -\frac13 \quad $or$ \quad x = -1}
    \end{daln}
\end{enumerate}
```

Notice, also, that you can toggle into and out of maths mode by inserting text inside dollar signs, and that you can use `\multicolumn`.
This environment, however, needs some tweaking where the lines are such that `displaystyle` is more suitable:

1. $S_n = \sum_{i=1}^{n} ar^{i-1}$
   $= \frac{a(r^n-1)}{r-1}$

```
\begin{enumerate}
  \item
    \begin{daln}
      S_n &= \sum_{i=1}^n ar^{i-1}\\
          &= \frac{a(r^n - 1)}{r-1}
    \end{daln}
\end{enumerate}
```

This example may look better if

1. the RHSs are in `displaystyle`;

2. the gaps between the lines is increased.

Here are 2 possibilities:

1. $S_n = \sum_{i=1}^{n} ar^{i-1}$
   $= \frac{a(r^n - 1)}{r - 1}$

2. $S_n = \sum_{i=1}^{n} ar^{i-1}$
   $= \frac{a(r^n - 1)}{r - 1}$

```
\begin{enumerate}
  \item
    \begin{daln}
      S_n &= \D\sum_{i=1}^n ar^{i-1}\\[2.5ex]
          &= \D\frac{a(r^n - 1)}{r-1}
    \end{daln}
  \item {\renewcommand{\arraystretch}{2}
    \begin{daln}
      S_n &= \D\sum_{i=1}^n ar^{i-1}\\
          &= \D\frac{a(r^n - 1)}{r-1}
    \end{daln}}
\end{enumerate}
```

If you're happy tweaking whenever and however necessary, this is all you need. But if you'd prefer a second environment that has `displaystyle` and increased spacing built in, then use of the array package gives the option of:

```
\newenvironment{Daln}{\tabcolsep=.2ex\renewcommand{\arraystretch}{2}%
       \begin{tabular}[t]{>{$\D}r<{$}>{$\D}l<{$}}}%
       {\end{tabular}}
```

This environment has:

1. The same gap between columns as `daln`, `align`, etc.;

2. a default space of 2`\baselineskip` (which you can change either globally in your definition, or at any part of the document using `\renewenvironment`, where the scope of the change can be limited by `{ }`);

3. the LHS and RHS (the 2 "columns") are both (separately) in `displaystyle` maths mode; the environment itself is paragraph mode.

We can see the effect on the previous examples:

1. $3x^2 + 4x = 1$

   $\therefore \quad 3x^2 + 4x + 1 = 0$

   $\therefore \quad (3x + 1)(x + 1) = 0$

   $\therefore \quad x = -\frac{1}{3} \quad \text{or} \quad x = -1$

2. $S_n = \sum_{i=1}^{n} ar^{i-1}$
   $= \frac{a(r^n - 1)}{r - 1}$

```
\begin{enumerate}
\item \begin{Daln}
  3x^2 + 4x &= 1\\
    \therefore\qquad 3x^2+4x+1 &= 0\\
    \therefore\qquad (3x+1)(x+1) &= 0\\
  \multicolumn{2}{r}{$\therefore \qquad
      x = -\frac13 \quad $or$ \quad
      x = -1$}
 \end{Daln}
\item
  \begin{Daln}
     S_n &= \sum_{i=1}^n ar^{i-1}\\
         &= \frac{a(r^n - 1)}{r-1}
  \end{Daln}
\end{enumerate}
```

Note that as the `Daln` environment is now "outside" the mathematics environment, anything in a `\multicolumn` will need to be put into a maths environment.

The same examples typeset using `aligned` come out as:

```
\begin{enumerate}
\item $\begin{aligned}[t]
  3x^2 + 4x &= 1\\
    \therefore\qquad 3x^2+4x+1 &= 0\\
    \therefore\qquad (3x+1)(x+1) &= 0\\
  \multicolumn{2}{r}{\therefore \qquad
      x = -\frac13 \quad \text{or} \quad
      x = -1}
 \end{aligned}$
\item $
  \begin{aligned}[t]
     S_n &= \sum_{i=1}^n ar^{i-1}\\
         &= \frac{a(r^n - 1)}{r-1}
  \end{aligned}$
\end{enumerate}
```

1. $$\begin{aligned} 3x^2 + 4x &= 1 \\ \therefore \quad 3x^2 + 4x + 1 &= 0 \\ \therefore \quad (3x+1)(x+1) &= 0 \\ \therefore \quad x = -\tfrac{1}{3} \quad &\text{or} \quad x = -1 \end{aligned}$$

2. $$\begin{aligned} S_n &= \sum_{i=1}^{n} ar^{i-1} \\ &= \frac{a(r^n - 1)}{r - 1} \end{aligned}$$

6.2 Boxes and space

6.2.1 Space and double-spacing

We've covered the idea that space can be measured in fixed units (page 36), such as inches, cm or pt (points)[1], or font-dependent units (`em`, `ex` or `mu`—maths units, math mode only), or in terms of a parameter (a length parameter is a length command that affects the appearance of the output produced by LaTeX) such as `\textwidth` or `\linewidth`. And the idea that space can be stretchy (rubber space).

6.2.1.1 Filling space

We've met some space-commands like `\vspace` `\hspace` `\vfill` `\hfill` (and it is worth noting that `\vfill` is equivalent to `\vspace{\fill}` and `\hfill` is equivalent to `\hspace{\fill}` where `\fill` is a general stretchy space), and maybe noticed in passing that there are places, like the beginning and end of lines and pages, where LaTeX removes extra space and ignores spacing commands.

`\vspace*{`*space*`}` and `\hspace*{`*space*`}` forces space in these places, and if it's rubber space you want, `\vspace*{\fill}` and `\hspace*{\fill}` will provide it.

`\fill` has a natural length of zero and the ability to stretch to any arbitrary (positive) length.

`\stretch{`*dec_num*`}` has natural length zero and *dec_num* (a positive or negative decimal number) times the stretchability of `\fill`.

This means that

```
\newpage
\vspace*{\stretch{1}} Title \vspace*{\stretch{2}}
\newpage
```

will position Title a third of the way down the page.

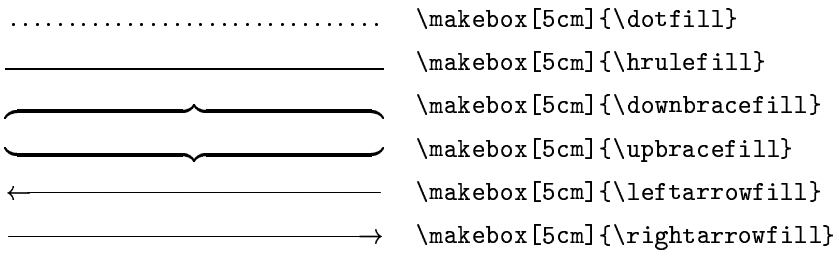
Used with `\hspace`, `\stretch` can position text horizontally.

A fixed length of *space* (like 3cm or 2em) will be left for a `\vspace*{`*space*`}` even if a page-break occurs within its "scope" (unlike the unstarred version which will leave space at the bottom of a page, but not at the top of the next page).

Sometimes dots or a line are needed to fill a stretchy space; the commands to do this are `\dotfill`, `\hrulefill`, `\downbracefill`, `\upbracefill`, `\leftarrowfill` and `\rightarrowfill`.

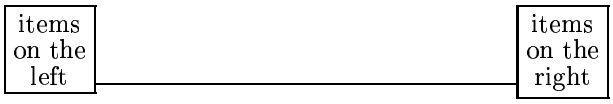
They can be used as in:

[1] In fact, TeX stores all lengths internally in scaled points, where 65536 scaled points = 1 pt and 72.27 pt = 1 inch.



or

```
\fbox{\shortstack{items\\on the\\left}}%
\hrulefill
\fbox{\shortstack{items\\on the\\right}}
```



6.2.1.2 Measuring space

We've also seen that the length of a space can be either + (to the right of the current position, or, typically, down) or − (to the left of the current position, or, typically, up).

We've met in passing some of the length parameters, listed together here for completeness:

`\textwidth` `\textheight`	Width and height of text on the page. These should not be changed outside the preamble.
`\parindent`	Amount of indentation of paragraphs; in any paragraph indentation can be suppressed by `\noindent`, or enforced by `\indent`.
`\linewidth`	The width of text in the current environment, calculated by LaTeX; should not be changed by the user.
`\parskip`	Stretchy space between paragraphs.
`\baselineskip`	Minimum distance between bottom of two successive lines.
`\baselinestretch`	A decimal number, default value of 1, to stretch proportionally the distance between lines. Changed by `\renewcommand`.

With the exception of the first two (which must be set in the preamble), the values of the parameters can be changed anywhere in the document.

Their values can be changed using the declarations:

- `\setlength{`*comm*`}{`*len*`}` for a rigid length or `\setlength{`*comm*`}{`*len*$_1$ `plus` *len*$_2$ `minus` *len*$_3$`}` for a stretchy length. *len*$_1$ is the natural length value of the parameter, *len*$_2$ is the maximum amount of space by which it can grow (be stretched) and *len*$_3$ is the maximum amount of space by which it can shrunk. Where *len*$_2$ and *len*$_3$ are not given (the "rigid" version), these lengths are set to `0pt`.

  For example: `\setlength{\parskip}{12pt plus 4pt minus 2pt}` specifies that `\parskip` is usually 12pt, but can be "stretched" or "shrunk" as appropriate to be between 10pt and 16pt.

- `\addtolength{`*comm*`}{`*len*`}` which adds a + or − amount to the *comm*. It is possible to add/subtract a multiple of the original command parameter, as in

  ```
  \addtolength{\parskip}{-0.1\parskip}
  ```

  which redefines `\parskip` to be 0.9 of what it was.

- `\settowidth{`*comm*`}{`*text*`}` that will measure the length of *text* and set *comm*, the length command, to have that length.

- `\settoheight{`*comm*`}{`*text*`}` that will measure the height from the baseline to the top of the tallest letter in *text*, and set *comm*, the length command, accordingly.

- `\settodepth{`*comm*`}{`*text*`}` that will measure the depth from the baseline to the bottom of the letter going furthest below the baseline in *text*, and set the length command accordingly.

The last three commands can be used to reset any length command, but are particularly useful when you need a space exactly the size of a piece of text. This is done using

- `\newlength{`*`comm`*`}` where the new length has default length of 0 units and is a rubber length.

Say, for some reason, all theorem proofs are to be set out

Proof:

With the words in the proof starting after the end of *Proof:*. If user-defined space of the correct length is defined, then the proof will be as desired. □

```
\newlength{\prf}
\settowidth{\prf}{\em Proof:}
  :
  :
{\em Proof:}\\[1ex]
\hspace*{\prf}With the words ....
as desired.
\hspace*{\fill}$\square$
```

Note the effect of the `*` in `\hspace*{\fill}$\square$`.

6.2.1.3 Double-spacing

- A crude form of double-spacing can be achieved by

  ```
  \renewcommand{\baselinestretch}{2}
  ```

  which will be brought into effect by changing the size of the font (say by having `\large\normalsize` immediately after the command, or wherever it is to take effect).

- `\linespread{`*`dec_num`*`}` in the preamble will affect all line spacing in the document. The default value of *`dec_num`* is 1; 1.3 gives $1\frac{1}{2}$-spacing and 1.6 gives "double"-spacing.

- A neater effect is achieved by defining a new command in the preamble:

  ```
  \newcommand{\doublespacing}{\baselineskip=26pt plus 0.5pt minus 1pt}
    :
    :
  \begin{document}
    \doublespacing
    :
  ```

  The command `\doublespacing` can be inserted at that point where such spacing is to begin.

  A companion command `\singlespacing` could be defined to revert to single spacing.

  The above is designed for 12pt font; minor changes can be made for 10 or 11pt documents.

- The package doublespace provides an environment in which the spacing for the text in the environment (which could well be the whole document) is set. It operates by taking the *`coef`* specified in:

  `\begin{spacing}{`*`coef`*`} ... `*`text`*` ... \end{spacing}`

  and resetting `\baselinestretch` to that value. The value of *`coef`* will depend on the point size of the document as well as the distance between lines desired, and will, for $1\frac{1}{2}$-spacing be about 1.25, 1.21 and 1.24 for 10pt, 11pt and 12pt respectively; and 1.67, 1.62 and 1.66 for double spacing (so that *`coef`* $= 2$ will give a very wide spacing).

6.2.1.4 Instant ragged right

Inside `minipage`, `tabular` and `list` environments and in the text of commands such as `\footnote`, `\parbox` and `\caption`, the `\rightskip` is automatically set to 0, at the start of the environment or command, so that text is flush right. It can be the case (particularly in narrow "columns" such as `minipage` that a ragged right block of text will look better, and this can be achieved by

```
\setlength{\rightskip}{0pt plus 1fil}   % which gives a very ragged right
or
\setlength{\rightskip}{0pt plus 1cm}    % which gives a less ragged right
```

`plus 1cm` means that there is at most 1cm of white on the right. A larger value (or the first length using rubber space) gives a more ragged right; a smaller value gives a less ragged right.

There is an example in a minipage on page 95, which is the result of the code:

```
\begin{minipage}{0.3\textwidth}
  \setlength{\rightskip}{0pt plus 1cm}
     Boxes to be filled in on forms, .... wherever necessary:
\end{minipage}
```

6.2.2 Boxes

LaTeX typesets a page of text by aligning a lot of boxes in what seems to be the most sensible way. The smallest boxes are single letters or other characters, and there are box-making commands and environments that construct larger boxes that are treated as if they were single letters. Hence they are never split across lines or pages, which can be a problem with very long or big boxes.

Each box has a reference point on its left-hand edge, and these reference points are aligned horizontally across the page. The default position of the reference point for text boxes is the (vertical) centre, and for graphics, the lower left-hand corner, which can cause some unusual effects on horizontal alignment.

There are three kinds of boxes: LR boxes, in which the contents are processed in LR mode, and that contain less than 1 line of text; parboxes, in which the contents are processed in paragraph mode, and that typically contain more than one line of text; and rule boxes, that consist of a rectangular blob of ink—and which are more useful than they might seem.

A box-making command or environment can be used in any mode.

6.2.2.1 LR boxes

`\mbox{`*`text`*`}`
`\makebox[`*`wdth`*`][`*`pos`*`]{`*`text`*`}`
`\fbox{`*`text`*`}`
`\framebox[`*`wdth`*`][`*`pos`*`]{`*`text`*`}`

The short forms, `\mbox` `\fbox`, make a box of size *`text`*, the latter with a frame around it.

The long forms make a box of specified *`wdth`* (can use `\newlength` plus some space here, if it is necessary to measure the contents to determine how big a box to make) and *`pos`* (`l` `r` `s`) gives control of the horizontal position of *`text`*: `l` and `r` are flush left and right respectively; `s` causes inter-word space to be stretchy, so that *`text`* fits the box exactly. By default *`text`* is centred.

`wdth` is usually given as a fixed length (as 3em, or 4cm, or 20pt, etc.), but sometimes it may be useful to give it in terms of the size of the box into which the text fits. The length parameters:

`\depth`	measures the distance from the baseline to the bottom of the box
`\height`	measures the distance from the baseline to the top of the box
`\totalheight`	= `\depth` + `\height`
`\width`	is the width of the box.

So to make a box 1.5 times as wide as the text it contains, give the command

`\framebox[1.5\width]{`*`text in box`*`}` *text in box*

Because of the way the boxes are made, you may not get exactly what you expect if you use one of the height measures to determine the width of the box, and it is probably better to use a strut made of a

rule box (see below) to make a framed box bigger.

6.2.2.2 Parboxes

`figure`, `table`, `\parbox`, `minipage` and `p` columns in `tabular` create parboxes.
`figure` and `table` will be discussed later (Section 6.4, page 106), but the other three may best be used:

- `\parbox` for a short piece of text;
- `minipage` for a longer piece of text;
- `tabular` using `{|p{ ... }|}` and `\hline` to create a framed parbox. Although it can also be done using a combination of a `\parbox` (or `minipage`) and `\fbox`.

The most commonly used form of the command and environment statements is:
`\parbox[`*pos*`]{`*wdth*`}{`*text*`}`
`\begin{minipage}[`*pos*`]{`*wdth*`} .. `*text*` .. \end{minipage}`
pos has a default of `c` and optional values `t, b`, and determines the vertical positioning of the reference points on the box, which affects the vertical alignment.
wdth is the width of the box, and it can be advisable to define it in terms of `\textwidth` or, better, `\linewidth`.

Note

1. `\parindent` is set to 0 in a parbox. If you want indentations, then reset `\parindent` using `\setlength`.
2. `minipage` can contain a wider variety of environments than `\parbox` (including `tabbing` and `tabular`).
3. Any `\footnotes` need special treatment—discussed later (page 127).

The more general form of the command and environment statements (giving more control of the height of the box) is:
`\begin{minipage}[`pos_1`][`len_2`][`pos_2`]{`len_1`}{`*text*`}`
where the arguments are:

len_1 a rigid length—the width of the box (as before). Can be given in terms of `\linewidth`

pos_1 vertical alignment (as before):
- `t` aligns the base line of the first line in the environment with the base line of the current line
- `b` aligns the base line of the last line of the environment with the base line of the current line
- `c` (default) centres the box vertically.

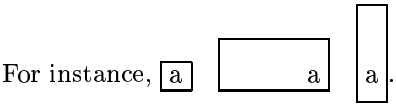
len_2 a rigid length—the height of the box which can be given in terms of `\height \depth \totalheight \width`, the natural dimensions of the box (see Section 6.2.2.1).

pos_2 determines where in the box of len_2 the text is positioned:
- `t` top of box
- `b` bottom of box
- `c` centred
- `s` stretched vertically to fill the box (this can be used to get a double-spaced effect within the box).

6.2.2.3 Rule boxes

Can be used to make an actual black box, or to make an invisible strut inside, say, an `\fbox`, to make it sufficiently wide and high.
`\rule[`*raiselen*`]{`*wdth*`}{`*hght*`}`

raiselen raises the blob that distance above the base of the line (if < 0, lowers it).
A *wdth* of 0 units will create a vertical strut (an invisible vertical bar of height *hght*).
A *hght* of 0 units will create a horizontal strut (an invisible horizontal bar or width *wdth*).

For instance, 

```
For instance, \fbox{a}\quad
\fbox{\rule[3ex]{3em}{0em}a}\quad
\fbox{\rule[-1ex]{0em}{3em}a}.
```

The first box is exactly the size of an "a". The second has a 3em-long space in front of the "a", and is deeper than the original box, because the invisible line was raised by 3ex. The third box starts below the baseline as it was lowered by 1ex, and is taller, because it has an invisible strut 3em high. See also the example on page 95.

6.2.2.4 Raising boxes

`\raisebox{`*raiselen*`}[`*hght*`][`*dpth*`]{`*text*`}`
will raise *text* a length *raiselen* above the base of the current line.
If *hght* (and *dpth*) are specified, then the *text* is treated as if its biggest character is *hght* above the baseline and lowest character is *dpth* below. This would be used to cause larger/smaller space to be left above and below the line than would be the default. If omitted, the actual dimensions of *text* are used.

6.2.2.5 Saving boxes

Boxes (and this includes graphics inserted by the `\includegraphics`) used several times over are best saved in `\savebox`, which saves the typeset box for repeated use, thus speeding up typesetting. (Defining the box with `\newcommand` won't do this; it just saves the instructions, which are typeset from scratch each time.)
Boxes are saved in three steps:

1. `\newsavebox{`*cmd*`}` declares the command name to be used.
2. `\sbox{`*cmd*`}{`*text*`}` or
   `\savebox{`*cmd*`}[`*wdth*`][`*pos*`]{`*text*`}` or
   `\begin{lrbox}{`*cmd*`}` ... *text* ... `\end{lrbox}`

   save *text* in the bin called *cmd* (which, being a command, starts with a \).

   The bin can be re-filled several times, and the latest contents can be reused as often as needed.

   wdth, pos, text are as before. `lrbox` is an environment, having the same effect as the command `\sbox`, but it ignores spaces at the beginning and end of *text*. It would be used, for instance, to define a framed environment (see GMS pg. 459).
3. `\usebox{`*cmd*`}` typesets the latest contents of the box at that point in the document.

Some simple examples:
Back to Proof. Suppose the latest effect to catch your eye was *Proof:*. This could be achieved by:

```
\newsavebox{\proof}
\savebox{\proof}{\fbox{\em Proof:}}

which would be used as in:

\usebox{\proof}
The proof of the pudding is here.
\hspace*{\fill}$\square$

and later, after the next Theorem:

\usebox{\proof} Once again.
\hspace*{\fill}$\square$
```

which would be used as in:

Proof: The proof of the pudding is here. □

and later, after the next Theorem:

Proof: Once again. □

`\usebox{\proof}` could be incorporated into a proof environment (if the box had first been defined and filled).

Boxes to be filled in on forms and standard-formatted test answer sheets can be defined in a variety of suitable sizes, and then used wherever necessary:

```
\newsavebox{\chkbox}
\newsavebox{\smlbox}
\newsavebox{\bgbox}
\sbox{\chkbox}{\fbox{\rule[-.5ex]{0ex}{2.5ex}%
                     \rule{2.5ex}{0ex}}}
\sbox{\smlbox}{\fbox{\rule[-0.5ex]{0ex}{2.5ex}%
                     \rule{4cm}{0ex}}}
\sbox{\bgbox}{\framebox[0.85\linewidth]{%
                     \rule{0ex}{4cm}}}
```

Which allows usage:

```
\begin{enumerate}
    \item The correct answer is:
       \begin{enumerate}
           \item This \hfill \usebox{\chkbox}
           \item This\hfill\usebox{\chkbox}
           \item Or this\hfill\usebox{\chkbox}
       \end{enumerate}

    \item The formula is \hfill\usebox{\smlbox}

          in most cases, but in exceptions is \usebox{\smlbox}.

    \item A brief description is

          \usebox{\bgbox}
\end{enumerate}
```

and gives the result:

1. The correct answer is:
   (a) This ☐
   (b) This ☐
   (c) Or this ☐
2. The formula is ☐
   in most cases, but in exceptions is ☐.
3. A brief description is
   ☐

A graphic (see Section 6.3.4, page 99) used repeatedly (for instance in a header or footer) is best saved

in a box (the advantages are that typesetting is faster and the final `.ps` file is smaller). For instance:

```
\newsavebox{\mygraphic}
\sbox{\mygraphic}{\includegraphics{mygraph.eps}}
   :
\usebox{\mygraphic}     % used as often as necessary
   :
```

Style parameters for framed boxes

- `\fboxrule` that determines the width of lines created by `\fbox` and `\framebox` (but not in a `picture` environment). Default is `0.4pt`.
- `\fboxsep` which is the amount of space between the frame and contents of the box (again, not in `picture`). Default is `3pt`.

The value of such a parameter is changed using `\setlength`, as in `\setlength{\fboxsep}{4pt}`.

6.3 Pictures

Pictures can be generated directly by LaTeX coding, if they consist of lines (straight or curved), arrowed lines ("vectors"), circles, ovals, framed boxes and text. This is fine for fairly simple diagrams, and if you have fairly standard pictures, you can store "template" code that is copied and edited each time it's needed. The advantage of this fairly laborious picture-making process is that you have only one file.

More complicated pictures are usually produced using a graphics package— `Xfig` for hand-made pictures, and the likes of `maple` or `Splus` for software-plotted pictures. Whatever software is used, an (`eps`) file is scaled, positioned and included using one of LaTeX's packages, typically graphics, epsfig (older) or epsf (even older still). It was intended that graphics would supersede the others, and it offers more features, like scale and reflect, and these features can be applied to any "box", including text.

6.3.1 Encapsulated PostScript

PostScript is a language for describing how images should be drawn on a page. PostScript data can be sent directly to a laser printer which supports PostScript, or can be displayed on a computer screen using a program such as `Ghostview` (or, from an application like Word, a document can be "printed" to a file, which will create a PostScript image of that document).

A PostScript file should begin with the characters `%!`, for example:

```
%!PS-Adobe-2.0
%%Creator: dvipsk 5.66a Copyright 1986-97 Radical Eye Software ...
%%Title: notes.dvi
%%Pages: 25
%%PageOrder: Ascend
%%BoundingBox: 0 0 596 842
%%DocumentFonts: CMBX12 CMR12 CMTT12 CMBX8 CMR8 CMR10 CMMI12 CMSY10
.......
```

A PostScript file is "encapsulated", for our purposes, if:

1. it meets some extra restrictions which allow it to be embedded in other documents;
2. it has a "bounding box" comment which specifies the rectangle within which the PostScript will place its output on the page—this gives the lower-left and upper-right corners of the box in points (`pt`);
3. usually, it will have additional text in the comment at the beginning of the file, such as:

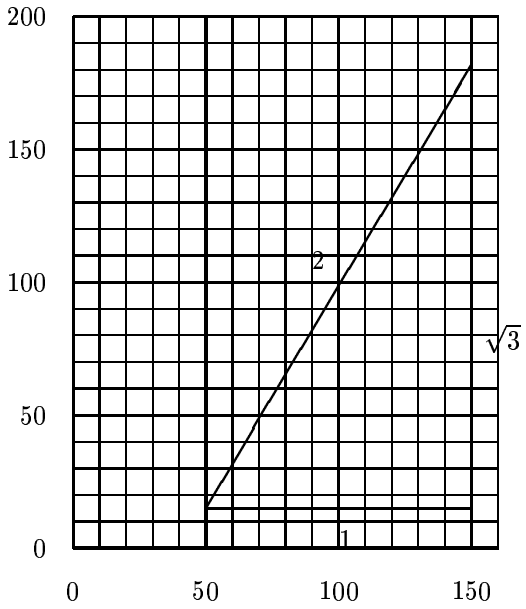
   ```
   %!PS-Adobe-2.0 EPSF-2.0
   %%Title: egdir.eps
   ```

When `Ghostview` or `gv` display an encapsulated file, the size of the window on the screen will be reduced to the bounding box limits.

6.3.2 LaTeX `picture` Pictures

Picture-making instructions are given inside a `picture` environment. By default, all measurements are made in points (but this can be changed by a `\setlength{\unitlength}{1mm}` command, say, to work in millimetres, given outside the `picture` environment; each `picture` environment in a document can have a different unit length; any LaTeX length measure can be used to specify the unit length), and the environment defines the size of the box containing the picture. A full discussion on making pictures is given in Lamport, pg. 118 ff and 219 ff., but an example, drawing a 1, 2, $\sqrt{3}$ triangle is given below.

```
\begin{picture}(200,240)(0,-20)
  \graphpaper(0,0)(160,200)
  \thicklines
   \put(50,15){\line(1,0){100}}
   \put(150,15){\line(0,1){167}}
   \put(50,15){\line(3,5){100}}
   \put(100,0){1}
   \put(155,75){$\sqrt{3}$}
   \put(90,105){2}
\end{picture}
```



The graph paper in the background is useful when designing the picture, i.e., determining the lengths and angles of the lines, and the positions of the text. It requires use of `graphpap` package, and, typically, would be removed from the final version.

Some of the information in the book is not quite true. For instance, `pict2e` package is installed, but currently is "empty" and so cannot be used (it is supposed to give greater functionality, for instance with respect to the slope of straight lines and vectors), and although it is stated that either `\thicklines` or `\thinlines` can be used, curves (including circles) are only produced properly if `\thicklines` is used.

The greatest restrictions are with respect to the slopes of lines (currently in terms of change-in-x and change-in-y values which must be integer, lie between -6 and 6 and have no common factors), and the sizes of circles (diameter ≤ 40 points). And `\scalebox` and `\resizebox` work on a `picture` environment, but their effect will only be seen on the `.ps` version (**not** on the `.dvi` version) of the file, which can be awkward. Even more awkward is the fact that while they work **within** a `picture` environment, the effect is only seen in the `.ps` version, and in addition, scaling of a `\put(20,30){\circle{40}}` by, say, a factor of 2 will not only cause the line to be twice as thick and the circle twice the diameter (i.e. to have a diameter of 80), but the coordinates of the `\put` (centre of the circle) will also be doubled (effectively equivalent to `\put(40,60)`). So considerable care would be needed to incorporate a scaled circle exactly where and how it's needed.

6.3.3 color package

This package is needed to achieve the effects:

- coloured text;
- coloured page;
- coloured box with or without a coloured frame, containing text (this can be particularly useful

when combined with psfrag see Section 6.3.9, page 104)—in effect, highlighting the text.

The package requires special support from the device driver; some of the colour models are supported by some drivers, some drivers have their own colour models.

6.3.3.1 Colour models

The colour models most typically available include:

`rgb` **R**ed, **G**reen, **B**lue: Colours are specified in a comma separated list of 3 numbers between 0 and 1 giving the components of the colour (1 corresponds to a full dose of the colour, 0 to none).

`cmyk` **C**yan, **M**agenta, **Y**ellow, blac**K**: Colours are specified in a comma separated list of 4 numbers between 0 and 1 giving the components of the colour (1 corresponds to a full dose of the colour, 0 to none).

`gray` Grey scale, a number between 0 and 1 denoting the degree of greyness (1 is white, 0 is black).

`named` Colours are accessed by naming one of the predefined named colours (such as JungleGreen) of the driver; the `dvips` driver has 68 named colours.

6.3.3.2 Colours

The package predefines the colours: `black white red green blue cyan magenta yellow`.

You can define your own colours using one of the above models with the command:

`\definecolor{`*`name`*`}{`*`model`*`}{`*`spec`*`}`

where *`name`*, any sequence of letters and numbers, is the name of the new colour, *`model`* is one of `rgb cmyk gray`, and *`spec`* is the number(s) specifying the colour. For instance:

```
\definecolor{lt-blue}{rgb}{0.8,0.85,1}
\definecolor{mygrey}{gray}{0.75}
```

6.3.3.3 Text colour

The colour of text can be determined either by the declaration (see Section 3.2 on page 28 for details on declarations)

`\color{`*`name`*`}`

which will affect the text colour from the point of the declaration to the end of its scope (such as the end of the current environment), or the command

`\textcolor{`*`name`*`}{`*`text`*`}`

which affects only the colour of *`text`*.

It is also possible to specify the text colour directly:

`\color[`*`model`*`]{`*`spec`*`}`	as in	`\color[rgb]{1,0.2,0.3}`
`\textcolor[`*`model`*`]{`*`spec`*`}{`*`text`*`}`	as in	`\textcolor[named]{BrickRed}{Red Text}`

6.3.3.4 Page colour

The background colour of the current page and all subsequent pages is set by the global declaration `\pagecolor`, in one of the forms:

`\pagecolor{`*`name`*`}`
`\pagecolor[`*`model`*`]{`*`spec`*`}`

where *`name`* and *`spec`* are as above.

A `\pagecolor{white}` declaration would be necessary to get back to "normal".

This page was typeset with

```
\pagecolor[gray]{0.85}
```

in force.

6.3.3.5 Coloured boxes

These are produced by the following commands, in one of two forms:

```
\colorbox{name}{text}                    % name is the colour of the background
                                         behind text
\colorbox[model]{spec}{text}             % specifies the background colour directly
\fcolorbox{name1}{name2}{text}           % puts frame of colour name1 around box of
                                         % colour name2 that contains text
\fcolorbox[model]{spec1}{spec2}{text}    % specifies the colours directly
```

The `\fbox` parameters `\fboxrule` and `\fboxsep` (see also page 96) are used to determine the thickness of the frame `{\setlength{\fboxrule}{.6pt} \fcolorbox[gray]{0}{.85}{frame}}` (if framed) and the size of the border `{\setlength{\fboxsep}{6pt}\colorbox[gray]{.85}{border}}` around *text* in the shaded area (for all colour boxes).

6.3.4 graphics and graphicx packages

These packages will scale, rotate, resize, reflect or include, the text or file specified. Changes to `.eps` files will show in the `.dvi` file; changes to text or `pictures` will only show in the .ps version. graphics is the "standard" package, and graphicx is an "enhanced" version with additional optional arguments for some commands.

Features of these packages are described briefly here—for greater detail see page 161 for the source of further documentation, and the document `epslatex.ps`, available from the `ctan` site (see page 161) which discusses various graphics-related issues in great detail, as well as the graphics packages themselves.

Bigger pictures that occupy most of the width of the text are best centred; small pictures that are to appear next to text are best put into a `minipage` environment (as should be the text) so that the two environments to be aligned have a similar reference point (the reference point for a text box is by default in the middle on the left edge, that of a graph is at the bottom left).

Pictures that are to have a numbered caption that can be referenced should be in a `figure` environment.

Examples and greater detail are in Lamport, pg 129 ff and 224 ff., but the basic commands and their options and parameters are:

- `\includegraphics{filename}` graphics, unclipped
  `\includegraphics*[llx,lly][urx,ury]{filename}` graphics, clipped
  `\includegraphics[keyval list]{filename}` graphicx, unclipped
  `\includegraphics*[keyval list]{filename}` graphicx, clipped

  Includes a graphics file.

  This file should be encapsulated postscript (`.eps`), and certainly `.eps` files exported from `Xfig` work well.

  Postscript files (`.ps`) produced by other means may give problems – for instance, depending on how Splus graphs were made, you may get a small graph occupying a whole page—and a possible solution is to run, in the `Xterm`, the program `ps2epsi` (usage: `ps2epsi` *oldfilename newfilename* ↩; if *newfilename* is not specified, a file with the same name as the existing file is created but with the extension `.epsi` if the existing file has a standard (postscript) extension) which will produce an `.epsi` file that has a bounding box and should be "better behaved". It's possible that the graphs or pictures made by other programs will need to be scaled (if they are far too big or small) or rotated (if they were produced in landscape orientation) and these effects are easily achieved using graphics or graphicx.

  The graphics version will include the file, with the option to "clip" the picture to show only that part inside a specified bounding box. The optional arguments `[llx,lly][urx,ury]` define (by default in points, in which case no units need be shown; otherwise in any units of length so long as the units are specified, eg `[1in,1in]`) the lower left and upper right coordinates of the image to be included (everything outside this rectangle will be "clipped"). In fact only the upper right coordinates need be specified, so long as the lower left coordinates are `[0,0]`.

The graphicx version has the option for a "key value list" to be specified. This key value list includes the possibility to specify the bounding box (in one of several different ways), to trim the picture, to rotate it, to specify the origin point for the rotation, and to scale the picture to have a specified width or height or to scale it proportionally.

Full details on the key value list are obtainable from the documentation (for the package, or the more general `epslatex.ps` document), but some of the more useful options are:

`height=`*`height`* The graphic is scaled to have a height of *`height`*, measured in any of the accepted units of length (or length parameter, such as `\linewidth`).

`totalheight=`*`height`* Similar to `height`, but including height both above and below the baseline.

`width=`*`width`* The graphic is scaled to have a width of *`width`*.

By default, if only one of the height and width is specified, the graphic is scaled so that the aspect ratio (width to height) is maintained.

If both are specified, by default the graphic is scaled anamorphically to have the specified width and height. The Boolean option `keepaspectratio`, if listed as one of the key values, will cause the aspect ratio to be maintained and the height and width values are taken to be the maximum possible (the graphic will have one dimension equal to one of the dimension values, but the other dimension will be less than the given value—whichever way around will maximise the size of the graphic).

`scale=`*`num`* The graphic will appear *`num`* times its natural size.

`angle=`*`num`* The graphic will be rotated in a counter-clockwise direction through *`num`* degrees (negative angle defines a clockwise rotation).

`origin=`*`ref`* By default, the graphic is rotated about its reference point (see Section 6.2.2, page 92). This command allows the user to define 12 other possible "origins" for rotation: the top, centre, Baseline, bottom (vertically) and left, centre, right (horizontally)—see also the discussion on `\rotatebox` below. Here these points are defined as in `origin=tl` (no square brackets).

In each of the following commands, *`text`* can be text, an `\includegraphics` statement, or a `picture` environment.

- `\scalebox{`*`hscale`*`}[`*`vscale`*`]{`*`text`*`}`

  This produces a box (LR box) in which *`text`* is scaled by the proportional factor *`hscale`* horizontally and *`vscale`* vertically. By default *`vscale`* is set to *`hscale`*.

- `\resizebox{`*`wdth`*`}{`*`ht`*`}{`*`text`*`}`
  `\resizebox*{`*`wdth`*`}{`*`ht`*`}{`*`text`*`}`

  Will re-scale *`text`* to fit the box of width *`wdth`* and height *`ht`*. If only one of *`wdth`* and *`ht`* are to be specified, and the rescaling of the other is to be proportional, give the "other" the value of `!`. Could be used to make all "pictures" of the same size, although the font sizes in any labelling text may then differ (some being scaled up, some scaled down).

  If *`text`* is literally a single line of text, the `*` form will take *`ht`* to specify the height + depth of the text (the other form measures only the height above the base-line).

- `\rotatebox{`*`ang`*`}{`*`text`*`}`

  Produces a box (LR) formed by rotating *`text`* anti-clockwise through *`ang`* degrees. The reference point of the rotated box is at the same vertical height as the reference point of the original *`text`*.

  The graphics version always rotates about the reference point (see Section 6.2.2, page 92).

  The graphicx version has the option for some key vales to be specified. This version of the command uses the syntax:

  `\rotatebox[`*`options`*`]{`*`ang`*`}{`*`text`*`}`

  If no *`options`* are given, the default values are the same as in the graphics version (rotation about the reference-point, angle of rotation counter-clockwise if positive and measured in degrees).

  The possible *`options`* are:

`origin=`*`label`* where *`label`* specifies the point on the box to be used as the origin for rotation. There are 12 possible points, defined by the use of one or two of the letters `t B b c l r`. The first four letters give the vertical position of the origin (top, Baseline, bottom, centre) and the last three the horizontal position (centre, left, right).

Note

- The order of the letters is not important, so that `[Bc]` and `[cB]` are equivalent.
- The meaning of `c` is determined by the second letter (if present) so that `[bc]` means bottom (horizontal) centre, and `[lc]` means left (vertical) centre.
- If only one letter is specified, the other is assumed to be `c`. So that `[c]` is the middle of the box, `[B]` is Baseline (horizontal) centre and `[r]` is right (vertical) centre.

`x=`*`xdim`*`,y=`*`rdim`* gives the x and y-coordinates of the centre of rotation, relative to the reference point of the box. The values *`xdim`* and *`ydim`* must be valid measures of length (mm, `em`, etc.).

`units=`*`num`* allow a change from the default units of degrees anti-clockwise. The value of *`num`* must be the number of (new) units in one full anti-clockwise rotation, so that `[units=-360]` specifies that the angle will be given in degrees clockwise, and `[units=6.283185]` specifies a rotation in radians (anti-clockwise).

- `\reflectbox{`*`text`*`}`

  Reflects the image about a vertical line. It is an abbreviation of `\scalebox{-1}[1]{`*`text`*`}`.

6.3.5 epsfig package

This package allows the inclusion of a graphics (`.eps`) file, and provides options to scale by resizing, rotate, clip and re-specify the bounding box. The full command is:

`\epsfig{file=`*`fn`*`,height=`*`ht`*`,width=`*`wd`*`,clip=,angle=`*`degrees`*`,silent=,%`
`bbllx=`*`blx`*`,bblly=`*`bly`*`,bburx=`*`brx`*`,bbury=`*`bry`*`}`

Note that there are no spaces around the =s.

`file`	Can also use the alias `figure=`. Specifies the name of the file (`.eps`).
`height`	Sets the desired height, if absent the natural height is used. If the `width` is specified but not the height, the height is determined proportionally from the original height.
`width`	As for height, except that if `height` and not `width` is specified it is the `width` that is determined proportionally.
`bbllx`	x-coordinate of lower left-hand corner of the BoundingBox.
`bblly`	y-coordinate of lower left-hand corner of the BoundingBox.
`bburx`	x-coordinate of upper right-hand corner of the BoundingBox.
`bbury`	y-coordinate of upper right-hand corner of the BoundingBox.
`clip`	Clips figure to values of BoundingBox. A switch (on if there, else off), that requires the = even though nothing follows it.
`angle`	Anti-clockwise angle of rotation.
`silent`	Turns off informative messages as figure is processed. Must have =.

`file=` (or `figure=`) is the only option that **must** be specified.

6.3.6 Positioning one or more pictures

Pictures usually look best centred (horizontally) on the page, and how this is best achieved depends on the circumstances:

- Graphic(s) embedded in the main text, with text above and below it is best centred as:

```
... text
```

```
\begin{center}
  \includegraphics{graphic2.eps}
\end{center}
next text ...
```

An extra line above and/or below the `center` environment will start a new paragraph and give more white space above and below the graphic(s) (although the amount of white space will depend on the settings used in the document).

- Graphic(s) in an environment (such as `figure` or `minipage`), with no uncentred text in the environment can be dealt with as in:

```
... text
\begin{figure}[!htbp]
  \centering
  \includegraphics{graphics1.eps}
  \caption{......}
  \label{fig:1}
\end{figure}
```

This is one option; the `\centering` declaration will apply to the picture and everything following it in the `figure` environment (the effect can be restricted within the environment by using `{}`).

- Graphic(s) in an environment (such as `figure` or `minipage`), with uncentred text in the environment are better positioned as in:

```
\begin{figure}[!htbp]
  Figure text first, say.
  \begin{center}
    \includegraphics{graphic2.eps}
    \caption{......}
    \label{fig:2}
  \end{center}
\end{figure}
```

The biggest difference between the last two methods is in the amount of white space left above and below the figure: the `center` environment will cause more white space to be left above and below the graphic(s)—which sometimes is an advantage and sometimes not.

Where two or more graphics are to appear on a line, by default an interword space is left between them. This can be increased by the use of `\quad` or `\qquad` or `\hspace` or using stretchy space. There are several commands to specify the amount of stretchy space: `\hfill` used as in

```
\includegraphics{graphic1.eps} \hfill \includegraphics{graphic2.eps}
```

will push the graphics against the margins, and `\hspace` or `\hspace*` used with `\fill` and/or `\stretch` and positioned before, between and after the graphics allow the available space to be divided up appropriately and (if desired) proportionately (see Section 6.2.1, page 89).

6.3.7 `Splus` to PostScript

There are two possibilities for directly producing a PostScript version of an `Splus` graphic:

1. **From a `motif()` graphics device.**

   The default print command (you can see it on the `Options` menu under `Printing`) is `lpr -r [-P`*`printer`*`]` which will

   (i) write a file `ps.out.ps.00`*`n`*`.ps` to your `.Data` directory;

   (ii) print the file (on the default printer if the `-P` option is not given);

(iii) and then delete the file from you directory (because of the `-r` option).

This gives you two options for getting a postscript file from the `motif` graphics device:

(a) To keep a copy of the file and print a hard copy:
   (i) Use the options menu of the graphics device to change the printing option.
   (ii) Read the `Help` to see the effect of hitting the `Reset`, `Apply`, etc. buttons on the window that opens.
   (iii) Change the `Command` to `lpr [-P`*printer*`]` (remove the `-r` option).
   (iv) Press the `Print` button.
   (v) Check in your working `.Data` directory for the presence of the `ps.out.ps.00`*n*`.ps` file. It should be an `eps` file (see page 96 for what the first line should be like). If not, or if it's a single graph you want to occupy part of a page, it may be helpful to run `ps2epsi`.

(b) To produce a file copy only:
   (i) As above.
   (ii) As above.
   (iii) Change the `Command` to `echo`.
   (iv) Press `Print` button.
   (v) In your `Splus` session the file name will appear at the prompt; type ↩ to get the next prompt.
   (vi) Check in your working `.Data` directory for the presence of the `ps.out.ps.00`*n*`.ps` file. It should be an `eps` file (see page 96 for what the first line should be like). If not, or if it's a single graph you want to occupy part of a page, it may be helpful to run `ps2epsi`.

2. **From a `postscript()` graphics device**

   Alternatively, as part of the S-plus code, use something like:

```
postscript(file="foo.ps", horiz=F, onefile=F, print.it=F)
plot.... #S+ code to make plot
dev.off()
```

   The output will be written to the specified file.

   This method has the advantage that if you are using fancy formatting such as subscripts on your plot, these appear as they should in your postscript file.

   Note that the file is written only after the `dev.off()` instruction is given.

It may be useful to process the PostScript file with the `ps2epsi` program (supplied with `ghostscript`) before including it into LaTeX, since Splus tends to leave large margins:

```
ps2epsi foo.ps foo.eps
```

This also adds a bitmap version of the image, which may be used by some Mackintosh software.

6.3.8 LaTeX in `Xfig`

If you make most of your figures in `Xfig`, and most of those require only small amounts of "maths", it may seem easier to fiddle with several fonts (Roman and Italic) and sizes of font (full and 2 sizes smaller for subscripts) to "make" "maths" that is "good enough", so that `Xfig` can be run in default mode and the resultant picture is exported as `.eps` and included as outlined above.

However, if you use `Xfig` to produce, typically, graphs that require a lot of carefully typeset "maths" (maybe import bits from something like Splus, and pretty it up with `Xfig`) it may be worthwhile to use `Xfig` just a little differently.

This process uses epsfig (sort of hidden from view) and so it needs to be included in your list of packages. Or you need an extra step in the process — see Section 6.3.10.

The command to run `Xfig` appears to depend on which bit of the system is being used. Before starting any experiments, type `man xfig` ↩ in an `Xterm` to get the manual entry (which is not short) and then persevere down the options until either `-sp` or `spec` appears—and use whichever you find in the following process:

1. Run `Xfig` with the `-sp` or `-spec` flag (SPECial text):

   `xfig -spec` *`filename`*`.fig &`

   This causes any text fields in the figure to be processed in LaTeX.

2. Make the figure as usual. Any PostScript fonts used in text will be converted to the default LaTeX font, so it makes sense to change the font to Use LaTeX Fonts on the fonts option, followed by Default on the next Font menu that is provided. Mathematics mode is delimited as usual, $ ... $ being the safest (as Robust), in any text object. LaTeX commands can be used in text.

3. Save and export your figure. On the Export menu, the default Language is Encapsulated Postscript and this must be changed. Two files are exported, one containing the PS file (*`filename`*`.pstex`), and the other (*`filename`*`.pstex_t`) containing a short section of LaTeX coding that is created to insert the picture using epsfig. The version of `Xfig` that requires the `-sp` specification may well need you to export both the

   Combined PS/LaTeX (PS part),
   Combined PS/LaTeX (LaTeX part)

   files separately. The version of `Xfig` that requires the `-spec` specification offers

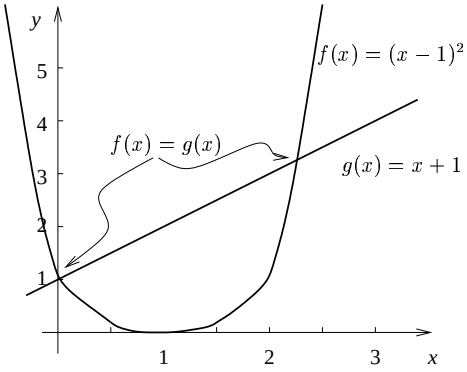
   Combined PS/LaTeX (both parts)

   so that in this case exporting is a single operation.

4. The picture is included in the LaTeX file by a `\input{`*`filename`*`.pstex_t}` command (which inputs the short file of LaTeX code that was created).

   This command can be in a `\figure` environment if it is to have a caption, or else in a minipage, or centred, much as discussed before.

```
\begin{center}
  \scalebox{0.7}{\input{test.pstex_t}}
\end{center}
```



Test graph, with LaTeX and PostScript fonts.

The use of `\scalebox` has a particularly interesting effect in this case. In the `.dvi` version, the graph is scaled, but the text is not. However, once converted to `.ps`, the text scaling takes visible effect.

If you want to be able to see the scaled effect in the `.dvi` version, you may prefer to run a driver file to typeset the file (only), and then run `dvips` with the necessary options to create an `.eps` file that can be included in your main LaTeX file. The process is discussed in Section 6.3.10.

6.3.9 The psfrag method of getting LaTeX in a PostScript file

The method described above causes a greater-than-usual proliferation of picture-associated files. An alternative, and one that would work well with `Splus` files too, is to use the package psfrag to replace text strings in the `.eps` file with LaTeX (formatted) text or equations.

The procedure is:

1. Include psfrag in the list of packages in a `\usepackage` command in the preamble.

2. In the document, use the `\psfrag` command to specify the text to be replaced and the text to replace it. This substitution will be made in any `\usegraphics` command that follows in the same environment (or within the same `{}`s).

The syntax of the `\psfrag` command is:

`\psfrag{`*`PStext`*`}[`*`posn`*`][`*`PSposn`*`][`*`scale`*`][`*`rot`*`]{`*`text`*`}`

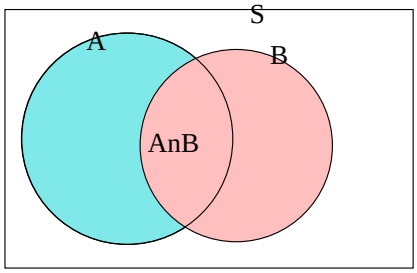
where the first and last arguments must be given, the rest are optional. The arguments are:

PStext	The text in the `eps` file to be replaced.
posn	Default value `[Bl]`. The position of the placement point relative to the new LaTeX text. See Section 6.3.4, under `\rotatebox` for more details on defining the 12 possible positions of the reference point.
PSposn	Default value `[Bl]`. Position of the placement point relative to the existing `eps` text. As for *posn*, 12 possible positions can be defined.
scale	Defaults to 1. Scaling factor for *text*. For best results, avoid using this, and use LaTeX type-size commands, such as `\small` or `\Large`.
rot	Defaults to 0. Counter-clockwise angle of rotation of the new text relative to the old text.
text	The text with LaTeX mark-up.

3. Use the `\includegraphics` command as usual

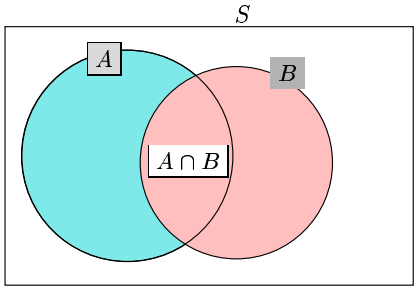
Note:

- `\psfrag` matches entire strings, so that `\psfrag{pi}{$\pi$}` will replace `pi` with π, but not affect `pi/2` or `2pi` (they'd need their own two `\psfrag` commands).
- `\shortstack` (see Section 5.1.6, page 64) is useful to replace a single line of text with a couple of closely-stacked lines.
- `\colorbox` (part of the color package, see Section 6.3.3) and `\fcolorbox` can be used to good effect to clear background lines and colours behind the text.




```
\includegraphics{/u/staff/edith/.../test2.eps}
```

```
\begin{center}
 \psfrag{AnB}{\fcolorbox{black}{white}{$A\cap B$}}
 \psfrag{A}{\fcolorbox[gray]{0}{0.85}{$A$}}
 \psfrag{B}{\colorbox[gray]{0.7}{$B$}}
 \psfrag{S}[br][tl]{$S$}
 \includegraphics{/u/staff/edith/.../test2.eps}
\end{center}
```



Note that the original text will disappear and the changes to be made will be listed, typeset, under the heading "PSfrag replacements" to the left of the picture on the `dvi` version of the document; the `ps` version will show the actual effect of the changes.

6.3.10 A one-page LaTeX document inside a longer one

The method will work for any **one-page** document (say a layout picture made using layout and `\layout`), but will probably be most useful with the combination of `Xfig` and LaTeX that has been discussed here.

Firstly, create a driver file, say called `test.tex`.

Continuing with the example of `test.pstex_t` above:

```
\documentclass{article}
\usepackage{epsfig}
\setlength{\textwidth}{100mm}     %Make the page the same size as the picture.
\setlength{\textheight}{10cm}
```

```
\begin{document}
\pagestyle{empty}                    %No page numbers, thanks!
\input{test.pstex_t}                 %Input the file containing the LaTeX code.
\end{document}
```

When typeset, this will create the typeset file `test.dvi`. Now (in `xterm`) run the command

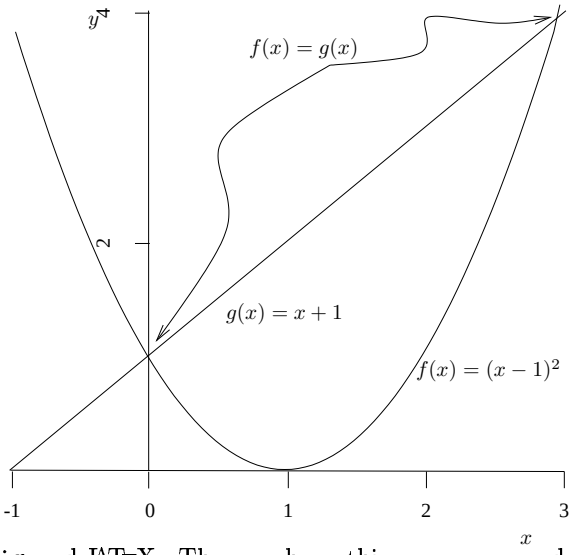
```
dvips -E test.dvi -o test.eps
```

The `-E` option says that the PostScript file must be Encapsulated, and the `-o` option that the name of the output file is to follow. The instruction will make a file `test.eps` that can be included (using `\includegraphics`) in another LaTeX document—and removes the need for epsfig in the "real" LaTeX document if graphics is the package of your choice. And rotation and scaling will work (that is, the effects will be seen) on this new `.eps` file in the `.dvi` version of your long document.

6.3.11 Splus to Xfig

This requires a special instruction within `Splus`, and then some combination of the methods outlined in the last two sections.

1. Create the graph in `Splus` with the Fig device driver active:
   ```
   fig("fig1.fig")
   plot...
   dev.off()
   ```
2. From outside `S` (in `xterm`) do:
   ```
   xfig -spec fig1.fig
   ```
3. Inside `Xfig`, add the LaTeX expressions to the figure, as above.
4. Export the figure, using "Combined PS/LaTeX (both parts)"—as above.
5. Finish off as described above.



The test graph a page or two back was made with `Xfig` and LaTeX. The graph on this page was made `Splus`, `Xfig` and LaTeX. And this time the `.eps` file used was made using a short driver file.

6.4 figure and table

These environments create a parbox (of width `\textwidth` or `\columnwidth`) that can have a numbered caption and that will "float" to a suitable position in the text (remember that parboxes cannot be split across lines or pages). They can only be used in outer paragraph mode (that is, not in any box such as a `parbox` or `minipage`).

This has the advantage that, if there is insufficient space on a page for a biggish table or picture, there is no great block of white space left at the bottom of the page. The float is placed at the top (or bottom) of the next page, and the text continues on the previous page, flowing around the "repositioned" float.

It also allows you to specify that all tables and diagrams be printed at the end of the document.

It has the disadvantage that the floats can appear, and really want to stay, where you don't think they should go.

If you are using two-column format, you need to see Section 8.5 in these notes and read further in Lamport, pg 197. What follows is true for one-column format.

`\begin{figure}[`*loc*`]` *body* `\end{figure}`
`\begin{table}[`*loc*`]` *body* `\end{table}`

body can contain text, and/or the figure(s) or table(s), processed in paragraph mode, and one or more captions.

loc contains a sequence of 1 to 4 letters, each specifying where the float may be placed, in order of preference.

- `h` **H**ere, i.e. at it's "natural" position in the text. `h!` is a stronger call to have it "Here!".
- `t` **T**op of (the next) page.
- `b` **B**ottom of (the next?) page.
- `p` **P**age of floats—a page that contains only figures and/or tables.

If none of the above is specified, the default is `[tbp]`.

If the results you get are not quite what you want or expect, read Lamport pg 198 for the rules of the positioning algorithm, and pg 199 ff for the style parameters that are defined to implement the rules. In fact, a good place to start if you have a page that contains a "float" and not much else, is to increase the proportion of picture/table on a page in relation to the proportion of text on the page. The default is 0.5. Taking a fairly dramatically high proportion (80%), you can try (it's always worked for me) `\renewcommand{\floatpagefraction}{0.8}`.

This can go anywhere in the document, but if you have many biggish floats, it may be best in the preamble of the document.

`\caption[`*listentry*`]{`*heading*`}`

This produces a numbered caption. If `\caption` is not used, an un-numbered floating environment is produced. You need to note:

- There can be more than one `\caption` in a `figure` or `table` environment. Each will produce a numbered caption that can be referenced. In other words, it is possible to include more than one numbered and captioned figure or table within a single environment, so long as the whole environment will fit on one page.

- `\caption` can be used only in paragraph mode. If necessary put it in a `\parbox` or `minipage`.

- The `label` command associated with the caption's number **must** be placed **inside** the `{`*heading*`}`, or **immediately after** the `\caption` command. Otherwise it will refer to the previous numbered environment (whatever that was), to your considerable confusion, as the label applies to the `\caption` and not the `figure` or `table` environment. This makes sense as it is then possible to have more than one labelled caption within any single `figure` or `table` environment.

- *listentry* would be used if the *heading* were very long, in which case it would specify a short version of the caption to go into the list of figures or tables. It should contain only robust commands.

- *heading* is the text of the caption. It should contain only robust commands, unless *listentry* is specified, in which case *heading* can contain commands of any type.

If for some reason you **don't** want additional floats on a particular page, use `\suppressfloats[`*pos*`]` where *pos* is either `b` (no additional floats at the bottom of the current page) or `t` (no additional floats at the top of the current page) and if *pos* is not specified, then additional floats are prevented from appearing at either the top or the bottom of the current page. This command will not affect floats for which `!` or `h` is used to specify their position. Commands to put the much-used test picture in a `figure` environment are:

```
\begin{figure}[h!tb]
 \begin{center}
  \scalebox{0.7}{\includegraphics{/u/..../fig1.eps}}
\caption{Example for the text.
         \label{fig:1}}

% Or that could be
%\caption{Example for the text.}
%\label{fig:1}

   \end{center}
\end{figure}
```

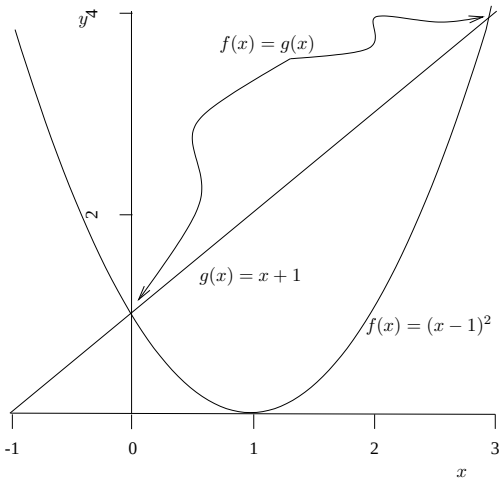


Figure 6.1: Example for the text.

The commands `\listoftables` and `\listoffigures` cause a list of tables and figures (respectively) to be generated at that point in the document.

Chapter 7

Classes of Document and Related Issues

The next issues about LaTeX are: documents that are typical of the different classes or types of document.

7.1 Preliminaries

7.1.1 Style and Class files

These files provide the definitions of the packages (`style` files) which as we have seen give Vanilla LaTeX considerably greater functionality, and different classes of document (`class` files) which determine the appearance of the document, and it can be helpful to look at them to see the default definitions of various parameters and commands.

They can be found, typically, at addresses such as

```
/usr/local/share/texmf/tex/latex/base/article.cls
/usr/local/share/texmf/tex/latex/tools/longtable.sty
```

A certain amount of investigation of the subdirectories of `/usr/local/share/texmf/tex/latex/` will soon turn up other possibilities. Documentation (available for some packages) is in the directory `/usr/local/share/texmf/doc/`.

Style and class documents tend to be written in TeX.

You can copy these files (your "root" directory for LaTeX files is a good place to put them; you may need to modify how and where your system searches for files), rename them and then modify them if you want to make you own equivalent classes or packages that are just a little bit different.

7.1.2 Saving trees, or changing default page size

Most of the LaTeX defaults for the page size are designed for books or journals rather than to optimise the use of an A4 sheet of paper. They also belong to the school of typesetting that believes that the eye/brain does not absorb the contents of long lines or wide pages.

To change the size of the text on the page, insert commands on the lines of:

```
\setlength{\textwidth}{16cm}
\setlength{\textheight}{24cm}
\hoffset=-2cm
\voffset=-2cm
```

in the preamble.

The second two lines are needed as the top left-hand corner remains the same, when the size of the page is enlarged, which pushes the (larger) text off-centre. Changing the vertical and horizontal offset by 2cm (up and to the left, respectively, as the values are negative) may correct this. The amount of change to

the offset will depend on the original default settings, and the new settings, and a certain amount of trial and error can be needed to get it right.

7.2 Classes

The options in the `\documentclass` command have default values that are usually determined by the class of the document, although a few have general default values. A summary of the default settings is given in Table 7.1.

To check the layout of a typical page under the various classes, and using one or more of the options, typeset a variety of documents of the form shown in Figure 7.1.

Sample document:

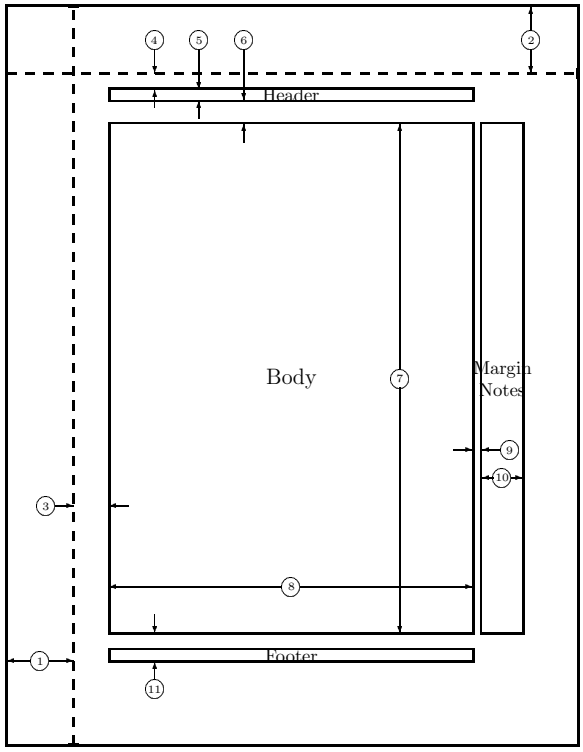
```
\documentclass[12pt]{article}
\usepackage{layout}
\begin{document}

\layout

\end{document}
```

In this document, you specify the class options that will be used in the final document.

Result:



1	one inch + \hoffset	2	one inch + \voffset
3	\oddsidemargin = 39pt	4	\topmargin = 17pt
5	\headheight = 12pt	6	\headsep = 25pt
7	\textheight = 548pt	8	\textwidth = 390pt
9	\marginparsep = 10pt	10	\marginparwidth = 44pt
11	\footskip = 30pt		\marginparpush = 7pt (not shown)
	\hoffset = 0pt		\voffset = 0pt
	\paperwidth = 614pt		\paperheight = 794pt

Figure 7.1: Layout for `article` class, with option `12pt`.

7.2.1 Page styles and page numbering

An output page consists of a "header", "body" and "footer" and the page style determines what is put in the header and footer.

Each document class has a default page style, but this is easily changed using the `\pagestyle{`*style*`}` (local) declaration.

The page style depends on whether the `twoside` (the left or even-numbered pages have different parameters to the right or odd-numbered pages) or `oneside` (all pages are treated as being right-hand) options are in force.

Table 7.1: Default options for the document classes

Option	General default	article	book	report	letter	slide
10pt etc	10pt					Not recognised
*type*paper	letterpaper					
draft/final	final					
landscape	portrait					
one/twoside		oneside	twoside	oneside	oneside	Can only use oneside
open*side*		—	openright	openany	—	—
*num*column	onecolumn				Only onecolumn	Only onecolumn
titlepage		notitlepage	titlepage	titlepage	—	titlepage

Details and More Options:

10pt, 11pt, 12pt—the three options for the size of the body font.

letterpaper, legalpaper, executivepaper, a4paper, a5paper, b5paper. letterpaper is 21.6cm by 27.9cm; A4 paper is 21cm by 29.7cm.

final, draft—draft uses black boxes to mark lines containing overfull boxes.

fleqn—causes all displayed mathematical environments to be indented from the left margin by a distance of \mathindent

fleqno—causes all equation numbers to be put flush against the left margin (instead of right).

landscape, portrait —the landscape option effectively interchanges the width and height dimensions of the text. This option is discussed in more detail in Section 8.4.

oneside, twoside—formats for printing on one or both sides of each page. When oneside is used, the height of the page can vary slightly as \raggedbottom, a local declaration, is in force. When twoside is used, the default declaration is \flushbottom which makes all pages the same height by adding vertical space between paragraphs.

openbib—causes the second and following lines in the bibliography to be indented by the rigid length specified by \bibindent.

openright, openany—the former specifies that chapters must begin on right hand (odd-numbered) pages, creating blank pages if necessary; the latter allows chapters to begin on any page.

notitlepage, titlepage—latter causes title and abstract to appear on separate pages; former puts all onto one page.

onecolumn,twocolumn —the number of columns on a page for most of the document. If parts of the document are to be in one- and others in two-column format, the commands \onecolumn and \twocolumn[*text*] start a new page and begin producing one- and two-column output, respectively. The option *text* in \twocolumn will print the *text* across the top of the page. The *text* can contain sectioning commands.

The `\pagestyle` declaration remains in effect until it is changed by another such declaration (the declaration alters the style of the page being typeset when the declaration is "found" and all following pages); the declaration `\thispagestyle{`*`style`*`}` will affect only the page being typeset when the declaration is "found".

The standard style options are:

`plain` Header is empty, footer contains centred page number.

`empty` Header and footer are empty.

`headings` Footer is empty, Header contains information determined by the document class; typically chapter and/or section headings and page number (note any un-numbered section titles from a starred section-type command are **not** used).

`myheadings` Similar to `headings`, but the information displayed is user-specified in the commands:
`\markright{`*`right_heading`*`}` in a document with the `oneside` option in force;
`\markboth{`*`left_heading`*`}{`*`right_heading`*`}` in a document with the `twoside` option in force.

Page numbering is determined by the global declaration: `\pagenumbering{`*`num_style`*`}`
where *`num_style`* is one of `arabic`, `roman`, `Roman`, `alph` or `Alph` (for arabic, lower-case roman, upper-case roman, alphabetic or upper-case alphabetic, respectively).

The declaration redefines `\thepage` to be *`num_style`*`{page}`. (See Section 8.2 for more detail on counters and how they are represented.)

Several such declarations can appear in a document, each remaining in effect until the next one.

7.2.2 Article class

See Lamport pg 176 ff for a full discussion, including details about the style parameters.

The characteristics and default settings of an `article` class document are:

- It is set up to be printed on one side of the paper (`oneside` option is the default).
- The title page and abstract are printed as part of the first page of the document (`notitlepage` option is the default).
- If you wish to have a separate title page and the table of contents on the same page (so you need `notitilepage` in force) and to have the first page(s) un-numbered, you need to put a `\thispagestyle{empty}` command **after** the `\maketitle` command and on each successive un-numbered page. The page number counter will need adjustment, too.
- Has parts, sections, subsections, ... but **no** chapters.
- All sections etc. that have numbers (this includes parts, if used) are numbered in the same style across the whole document, apart from any following an `\appendix` command. What this means is that, if an `article` has 10 sections in 2 parts and an appendix with 2 sections, the 10 sections are numbered from 1 to 10 (whether there are parts or not) and the sections in the appendix are numbered alphabetically (A and B). Subsections (and subsubsections, if they are numbered) are numbered within sections (and subsections).
- Has the same style of page numbering throughout the document.
- By default has `\pagestyle{plain}` (no headings and numbering in the centre of the foot of the page).
- `\bibliography{`*`files`*`}` causes a new section headed "References" to be started (to change the title of this section, redefine `\refname`, see page 139).
- Typically, `article` is used for short(ish) documents.
- Equations, theorems, figures and tables are all numbered $1, 2, \ldots, n_i$, where n_i is the total number of that type of construct in the document, throughout the document (ie the default is **not** to number within sections).
- `\raggedbottom` is the default, so that pages may be of slightly different lengths. To change this, insert `\flushbottom` in the preamble, which causes vertical space to be added to some pages to get equal length pages. The `twoside` option also sets `\flushbottom` (but has other consequences too).

A document in this class will look something like:

```
\documentclass{article}
\usepackage{...}
\title{...\\...}        %\\ if long; spread over 2 lines
\author{...             % Author 1 - can be {name\\address \and ...}
    \and ... \and ...}  % Authors 2&3 - MUST have \and between the names.
\date{...}              % Can have \date{...\\Revised ...}.
                        % If omitted, get \today.
                        % Use \date{} to suppress date.

% any other preamble stuff

\begin{document}
\maketitle              % Authors printed across page if they all fit,
                        % else 1 per line
\begin{abstract}
  text                  % First line is indented; suppress with \noindent.
\end{abstract}
\section{...}           % Start body of article .
  Main document

% Bibliography: Either using BIBTEX:
\bibliographystyle{...} % plain|unsrt|alpha|abbr etc.
\bibliography{foo.bib}  % Name of .bib file.

% Or, making your own bibliography, but still using \cite:
\begin{thebibliography}{widest_label}
                         % Text that will typeset to be as wide as the
                         % widest of the labels in [label] options.
  \bibitem[label]{citekey} Bibliographic information, fully formatted
  \bibitem[label]{citekey} Bibliographic information, fully formatted
\end{thebibliography}
\end{document}
```

Note that `\title{...}`, `\author{...}`, `\date{...}` can each contain a `\thanks{...}` command, which makes a footnote (by default using a symbol, not a number) containing the `{...}` at the bottom of the page. Say for addresses, as in:

```
\author{Joe Bloggs\thanks{VUW, \ldots}
        \and John Doe\thanks{Elsewhere \ldots}}
```

To produce a document containing a bibliography:

- **using** BibTeX you need at least 4 passes:
  1. LaTeX to write the `\cite{`*citekey*`}`s into the `.aux` file;
  2. BibTeX to find the references and make a `.bbl` file containing formatted references in the given style, and write warning and error messages in the `.blg` (log) file;
  3. LaTeX to read the `.bbl` reference file;
  4. LaTeX to resolve all references.
- **using** `thebibliography`, you have effectively made your own `.bbl` file, so you get a final version of your document with the 2 LaTeX runs (3 and 4 above). This approach makes sense if you have a large, personal `.bib` file used for all your academic documents, and are writing a journal article to be published in LaTeX. Your best bet is to use your `.bib` file when writing the article, and once

it is finalised, to copy the contents of the `.bbl` file (the `thebibliography` environment) into the `.tex` file containing the article, so it can be submitted as a single file.

If you change the bibliography style, you again need all 4 steps (as you need to include the markup for the new style in the `.bbl` file).

If you use BibTeX, the order of the `.bib` entries is determined by the bibliography style.

If you use `thebibliography`, the order of the bibliography entries is determined by the order in which the *`citekey`*s appear on the `\bibitem` commands.

7.2.3 Book and Report classes and making it in bits

7.2.3.1 Book and Report classes

The characteristics and default settings of a `report` class document are:

- Set up to be printed on one side of the paper (`oneside` option is default).
- The title page and abstract will appear on two separate pages, the document itself starting on the next page (`titlepage` option is the default).
- Has `\chapter` as a sectioning command, as well as all the sectioning commands for an `article` class document. Chapters must be used to get sensible numbering of the sections, subsections, etc. (if no chapter is defined then the first section would be Section 0.1).
- Numbering of "sections" is in the same style throughout the document, except after an `\appendix` command (when the **chapters** are numbered `Alph`—A, B, etc.) and sections are numbered within chapters, subsections within sections ...
- Has `\pagestyle{plain}` as the default.
- `\bibliography{`*`file`*`}` causes a new chapter headed "Bibliography" to be started (to change this title, redefine `\bibname`—see page 139).
- Typically used for long(er) documents, but without great internal structure (less formal documents, perhaps).
- Equations, theorems, figures and tables are all numbered within chapters (the counters are set to 0 at each `\chapter`). The default numbering style is as in Theorem 3.1 or Table 6.4 for the first theorem in chapter 3 and fourth table in chapter 6.
- `\raggedbottom` is the default, so that pages may be of slightly different lengths. To change this, insert `\flushbottom` in the preamble, which causes vertical space to be added to some pages to get equal length pages. The `twoside` option also sets `\flushbottom` (but has other consequences too).

Characteristics and default settings of `book` class document are:

- Set up to be printed on both sides of the paper (`twoside` option is the default), with `openright` the default (all chapters start on an odd-numbered, right-hand-side page).
- The title page and abstract will appear on two separate pages, the document itself starting on the next page (`titlepage` option is the default).
- Has `\chapter` as a sectioning command, as well as all the sectioning commands for an `article` class document. Chapters must be used to get sensible numbering of the sections, subsections, etc. (if no chapter is defined then the first section would be Section 0.1).
- It is possible to have some chapters in the `\frontmatter`; some in the `\mainmatter`; some in the `\backmatter` (see page 116), with the page style, page numbering and chapter numbering being affected by the three commands.
- Has `\pagestyle{headings}` as default.

- `\appendix` causes the following chapters to be labelled `Alph` (A, B, ...).
- `\bibliography{`*file*`}` causes a new chapter headed "Bibliography" to be started (to change, redefine `\bibname`—see page 139).
- Typically used for long documents, with a "proper" book structure.
- Equations, theorems, etc. numbered as for `report`.
- `\flushbottom` is the default. To change this, insert `\raggedbottom` in the preamble.

7.2.3.2 Making it in bits

"Books" are typically long, and best worked on in short sections. There are two commands which include another document into the current document:

Table 7.2: To input or include?

`\input{`*nextfile*`}`	`\include{`*filei*`}`
Dumps contents of *nextfile*`.tex` at that point in the current document.	Starts a new page (issues a `\clearpage`) and typesets the contents of *filei*`.tex`; writes the usual information to *filei*`.aux` and this information will be available on future runs (on any of the parts of the document). It is possible to include only some of *file1*`.tex ...` *filen*`.tex` by specifying: `\includeonly{`*filelist*`}`, which causes only the files on *filelist* to be included on the current run (but the `.aux` files of all files mentioned in an `\include` statement will be accessed, and the information contained in them will be used as needed).
Good for:	
• short sections of text, used in more than one document; • something complex (like a LaTeX picture that was easier to design separately) in a situation where the number of files needed for a document need not be restricted; • your personalised command and environment list; • a template for a document—say one setting up the document class, packages, page dimensions, another `\input` for your clever commands, special headings for a tutorial or test, etc.	• separate chapters of a book, or sections of a very long article; • maintaining cross-references and building up the bibliography across a whole (long) document; • a document consisting of a few files containing relatively larger chunks of the document.
Disadvantages:	
• doesn't keep `.aux` files for each file, only for the "root" file that contains the outer level of `\input` commands; • if you work on some parts of the document and not others, the `\cite`s and `\ref`s to the other parts of the document not being `\input` at that time won't work as there is no `.aux` file for them); • can end up with many little files.	• new page started (sometimes this is a disadvantage, sometimes not); • files may end up a bit long (slow to typeset) if chapters are long. (Can get around that by using an `\input` of the second half, but that has other disadvantages.)

7.2.3.3 Subdivision of the bits in a book

LaTeX offers three commands to subdivide a book into its most typical constituent parts:

`\frontmatter`

What follows this command has:

- Pages numbered with Roman numerals.
- Chapters (as specified by `\chapter{...}`) unnumbered, but listed in the table of contents.
- Other sectioning commands numbered, so they should be given in the * form to suppress numbering—they will then not be in the table of contents.
- Pages without running headings (if that is the form chosen for the whole document).

"Frontmatter" typically consists of title pages; tables of contents, figures and tables; and Preface and/or Introduction and/or Abstract.

`\mainmatter`

What follows this command has:

- Arabic page numbers, starting from page 1 (the counter `page` is reset).
- Numbered chapters and other sections.
- Chapter and section headings in page headers (Chapter on left, section on right).

"Mainmatter" is exactly that—the body of the document.

`\backmatter`

This typically contains the Index and Bibliography and anything else (for instance, Glossary) and is typically at the back of a book. The effect of the command is:

- As for `\mainmatter`, except:
- `\chapter` command does not produce a numbered heading, but does produce a table of contents listing.

A typical root file (`mybook.tex`, say) for a book will look something like:

```
\documentclass[...]{book}
\usepackage{makeidx, ...} % Need makeidx to make index.
\title{...}               % May need to custom design the title pages if
\author{ ...}             % there are several of them.
\date{...}
% Any other preamble stuff

\makeindex                % Causes .idx file to be written.
\includeonly{...}         % While document is being created - work only on
                          % the specified files.

\begin{document}
\pagestyle{plain}         % If no headings in front.
\frontmatter
\include{frontguff}       % frontguff.tex contains \maketitle, commands
                          % for tables of contents, figures and tables:
                          % \tableofcontents \listoffigures \listoftables and
                          % \chapter{Preface} or \begin{abstract}
                          % ... \end{abstract} etc.
\mainmatter
\pagestyle{headings}      % For automatic headings from sectioning
                          % commands.
\include{Ch1}             % Include the chapters one at a time.
\include{Ch2}
  ....
\include{Chn}
```

```
\appendix                % Causes next chapters to be numbered Alph.
\include{App1}           % Which would start \chapter{Whatever ...}.
\include{App2}
...
\backmatter

% As for article, either bibliography using BIBTEX:
\bibliographystyle{...}   % See article/
\bibliography{foo.bib}    % Name of the data base file for bib.

% Or, making your own:
\begin{thebibliography}{widest_label}
                          % Text that will typeset to be as wide as the
                          % widest of the labels in [label] option.
  \bibitem[label]{citekey} Formatted entry
  ....
\end{thebibliography}

% Either index, using makeindex (the sensible option)
\printindex

% Or, "yukkybunny", by hand:
\begin{theindex}
\item aaa, pg#1, pg#2,
   \subitem aagh, pg#-pg#
   \subitem aaargh, pg#
\item aba, pg#, pg-pg
\indexspace              % Space between letters in index.
\item baa, \see{aaa}
 ....

\end{theindex}

\end{document}
```

Notes

- The only sensible way to make an index is automatically. The command `\makeindex` causes LaTeX to make an `.idx` file of index entries; *MakeIndex* runs on this list and produces the `.ind` file, containing the alphabetised `\begin{theindex} ... \end{theindex}`, much as in the do-it-yourself section above. A final LaTeX run will include the index at the point specified by `\printindex`.
- To make an index, you need to use the makeidx package.
- The title generated by `\maketitle` may not be adequate for a book – may also need several "title" pages of bumph.
- The abstract will appear on a separate page (see Table 7.1).
- Use `\includeonly` while developing the document. Once it is complete, comment out or remove this line, and typeset the whole document if possible (more than once, including a BibTeX and *MakeIndex* run to be safe).
- The default chapter numbering after the `\appendix` command is `Alph` (capital letters).
- For an article with a short bibliography that is not likely to be used again, it's probably easier to use `\begin{thebibliography} \Bibitem[..]{...} ... ... \end{thebibliography}`. For a book

(or article) with many references, or any set of references liable to be re-used, it's probably worth constructing one or more bibliography data bases (`*.bib`) that can be re-used. If you have a large, personal `.bib` file used for all your academic documents, and are writing a journal article to be published in LaTeX, your best bet is to use your `.bib` file when writing the article, and once it is finalised, to copy the contents of the `.bbl` file (the `thebibliography` environment) into the `.tex` file containing the article, so it can be submitted as a single file.

- Bibliography and index generation are discussed further later in Chapter 9 on page 150.

7.2.4 Slides

Two document classes are available for making slides: `\documentclass{slides}` (Lamport pg 80) and `\documentclass{seminar}` (documentation available on the Web, see Section 9.8).

7.2.4.1 `slides`

- The default font is sans serif, except for mathematical formulae (although a `\mathsf{...}` inserted in each formula will change that—it would probably be worth defining new math environments that incorporated turning `\mathsf` on and off).
- Each slide, as defined by `\begin{slide} ... \end{slide}` gets a new page number.

  Overflows (where the contents of a `slide` environment are too long for a single page) and pages started with `\newpage` or `\pagebreak` **do not** get a new page number.

  So if you want your pages numbered correctly, you have to be very careful to get at most one page worth of text per slide.
- It is possible to make overlays, using the color package (by making the text that is not printed white, and then reversing the process, and making what was black white and what was white, black).
- It is possible to print notes on a slide, or time instructions (see Lamport pg 83).
- It is possible to print only some slides using, say `\onlyslides{3,7-10,19-999}` which will print slide 3, slides 7–10 (inclusive) and from slide 19 to the end. **NOTE** the page numbers must be quoted in ascending order.

  It is probably easier to use `ghostview` to select a few pages for printing.
- It is possible to have text outside a `\begin{slide} ... \end{slide}` environment. Such text will be printed and typeset as usual, and
  - can include `\pagebreak`;
  - the page(s) created will not be numbered—so title pages may be made this way.

A typical document will have the form:

```
\documentclass{slides}
\usepackage{...}
% and other preamble stuff

\begin{document}
\Large          % If you find the default size too small, increase it.
\bf             % If you like really chunky writing - but maths
                % won't be bold.
Heading text    % Title page (un-numbered). Optional.
\begin{slide}
   Text etc     % First slide.
```

```
\end{slide}
%----------------- Helps to show the breaks between slides.
\begin{slide}
   Text etc      % Second slide.
\end{slide}
%-----------------
  ....
\end{document}
```

7.2.4.2 seminar

- Default font is Computer Modern.
- Each slide is defined by the `slide` environment, but if too long for a single page, the overflow will appear on the next page, which will be numbered in sequence.
- Overlays are possible, using `semlayer` and `semcolor` options.
- Notes are possible, and are more flexible than in `slides`.
- `\onlyslides` works—better. The numbers need not be in ascending order, and can include `\ref`, as in `\onlyslides{\ref{fig:1}-10}`, which will print from the page containing `\label{fig:1}` to page 10.
- There are many other features:

  ➤ Default orientation is `landscape`.

  ➢ To make `portrait` slides
  1. include the `portrait` option in the `\documentclass[portrait]{seminar}` command, and
  2. use a `slide*` environment for each slide that is to be portrait.

  ➢ If a document contains both `portrait` and `landscape` slides, these must be printed separately. Put `\landscapeonly` in preamble; typeset and print. Repeat with `\portraitonly` in preamble.

  ➤ The height and width of all slides are set by `\slidewidth` and `\slideheight`, with default values of 8.5 inches and 6.3 inches, respectively (if `landscape`, or vice versa if `portrait`). These values can be reset in the preamble.

  Changed dimensions of a single slide can be achieved in a `slide` environment, using an option:

  `\begin{slide*}[`*`newwidth,newheight`*`]`

  for a `portrait` slide (same, un-starred, for a `landscape` slide).

  Other slide length parameters are

  `\slideframewidth \slideframesep \semin \semcm`

  and all of these are set with `\setlength`. The last first 2 are self-explanatory, and the last 2 are discussed further later.

  ➤ Justification:

  ➢ Vertical—default is centred. The command `\centerslidesfalse` causes it to be flush top; `\centerslidestrue` resets it to be centred.

  ➢ Horizontal—default is ragged right, and it can be more or less so:

  `\raggedslides`—very ragged.
  `\raggedslides[0pt]`—flush right
  `\raggedslides[2em]`—maximum space between the end of the line and the right margin is `2em` (or any length you prefer), which gives a semi-ragged effect.

  ➤ `\newslide` within a slide environment will start a new slide.

➤ Page parameters (commands, so values reset with `\renewcommand`) are:

`\slideleftmargin \sliderightmargin \slidetopmargin`
`\slidebottommargin`.

All 4 have default values of 0.5 inches.

Headers and footers are inside the top and bottom margins, and the slide is centred between margins (if `\centerslidestrue`).

➤ `seminar` magnifies things (everything), including space measures such as `\vspace{1cm}` and graphics. This means that diagrams to be included should not be made especially large, or using large fonts, as the document class will do the magnification.

This magnification is done in magsteps; the default is 4, but can be changed by the command `\slidesmag{`*n*`}`, where $-5 \leq n \leq 9,\ n \in \mathbb{Z}$.

The lengths `\semin` and `\semcm` are equal to one inch and 1cm, respectively, in pre-magnified size. So

`\rule{1pt}{4\semcm}`

draws a (vertical) line 4cm long on the transparency (but wider than `1pt`), whereas

`\rule{1pt}{4cm}`

would draw a line longer than 4cm and wider than 1pt.

It will be safer to define values in terms of, say, `\textwidth` where possible.

➤ Other ways to change sizes:

➢ 11 pt and 12pt options work in the document class command, and serve to magnify the slides slightly; a rise of $\frac{1}{2}$ magstep per 1 pt increase (12pt is magstep 5).

➢ Use of `\Large` etc., will work as usual to make the text larger.

➢ `\ptsize{`*n*`}` will change the size of the fonts on, say, a single slide ($n \in \{8, 9, 10, 11, 12, 14, 17\}$).

➤ Slides can be framed (`plain`, a square frame, is the default). Frames can be changed using

`\slideframe[`*commands*`]{`*style*`}`

where *style* can be `plain` or `none`. Additional styles are available, if you use the packages:

➢ fancybox which gives frames styles `shadow, double, oval, Oval` (see also page 149 in these notes).

➢ semcolor which gives the styles `scplain, scdouble, scshadow`.

Length parameters `\slideframewidth` and `\slideframesep` have default values of 4pt and 0.4 inches, respectively.

The optional *commands* would be used to change various style or length parameters associated with the frame.

Further commands provided by the packages fancybox (GMS pg 278) and semcolor can be used to customise the frame.

It is possible to define new frame styles (but this is not discussed any further here).

➤ Counters—there are several options and points to bear in mind:

➢ The counter for slides is `slide`; the default of `\theslide` is `\arabic{slide}`, but this can be changed (using `\renewcommand`).

➢ `\label` and `\ref` can be used. `\pageref` works most reliably when each slide is in a `slide` environment (or was started by `\newslide`).

➢ Can reset counters at the end of each slide (this happens, by default to the counter `footnote`). Use `\slidereset{`*list*`}`, where *list* is the list of counters to be reset.
Additional counters to be reset can be specified later by
`\addtoslidereset{`*list*`}`.

➢ Footnote counter marks are made using `\theslidefootnote` which is `\alph{footnote}` by default, i.e. letters, not numbers or symbols. This can be reset using `\renewcommand`.

- Page styles `empty, plain, headings, myheadings` work as in an article. An additional style `align` puts + signs in the corners.

  `\slideheadfont \slidefootfont` will change the size of fonts in the header and footer (use `\renewcommand`); the default is `\scriptsize`.

A typical document will have the form:

```
\documentclass{seminar}             % If all slides landscape, & "10pt".
% \documentclass[portrait,12pt]{seminar} % If some portrait, & all slides "12pt".
\usepackage{fancybox,...}           % fancybox, if you want oval frames.
% And other preamble stuff.
\onlyslides{...}                    % If only some slides to be printed.
\landscapeonly       %or            % Landscape slides to be printed.
%\portraitonly                      % Portrait slides to be printed.

\begin{document}
\Large             % If you find the default size too small, increase it.
\bf                % If you like really chunky writing - but maths
                   % won't be bold.
\slideframe{Oval}                   % \slideframe{none} for unframed.
\begin{slide}
   Text etc                         % First slide - landscape.
\end{slide}
%----------------- Helps to show the breaks between slides.
\begin{slide*}
   Text etc                         % Second slide - portrait.
\end{slide*}
%-----------------
  ....
\end{document}
```

7.2.5 Letters

Typical document has the form:

```
\documentclass[..]{letter}
\usepackage{...}
\address{your..\\address..\\..}   % Usually in preamble, can be anywhere.
\signature{Your name\\Title}
\makelabels                       % Print mailing labels that may fit
                                  % sheet of sticky labels.
% Other preamble stuff

\begin{document}
\begin{letter}{Who to\\address..\\..}
                                  % Used at top of letter and for labels.
\opening{Dear Mr Bloggs}          % The greeting.
  Text                            % Body of letter.
\closing{Yours sincerely}         % Closing of your choice.

\end{letter}
%---------------------------       Next letter.

\begin{letter}{Next who to\\address..\\..}
                                  % Used at top of letter and for labels.
\opening{Dear Ms ... }            % The greeting.
  Text                            % Body of letter.
```

```
\closing{Yours faithfully}        % Closing of your choice.
\cc{First name//Second name// ...}% Optional CC: list.
\ps{P.S. Afterthought ...}        % Optional text below signing off.
\encl{List}                       % Optional list of enclosures.
\end{letter}
%--------------------------       Next letter.
....
\end{document}
```

All letters within a document have same address and signature (from `\address`, `\signature`).

By default, `\parindent` = 0, but space is added between paragraphs. No sectioning commands are available.

If you wish to add information in a list structured like the cc list, but have it called something else (in consequence you can have no actual cc list), use `\renewcommand{\ccname}{`*`newname`*`}`, which will cause the list to be headed *`newname`*.

Chapter 8

Fragility, Counters, Footnotes, Landscaping and Modifying Appearances

Many of the topics discussed in this chapter are covered in GMS pages 17–66. Most of these topics have to do with modifying the default appearance of a LaTeX document, and giving some understanding of how LaTeX sets about its task of typesetting a document. Some of the topics are left-over fundamental concepts that haven't fitted in anywhere else.

8.1 Fragility

An essential, if obscure, concept. See Lamport pg 167–168.

Bits of text are carried to other parts of the document if they are section headings (that go to the Table of Contents or page headings whether generated automatically or specified by `\markboth` or `\markright`), or captions (to the list of Figures/Tables). Some commands will "break" (i.e. not work properly, or give an error message that can be quite alarming) if they are in such text—these are **fragile** commands—other commands, the **robust** ones, won't. Fragile commands will also give problems if used in `\thanks` in a `\title` command; in the `letter` environment in the `letter` document class; or between `@{` and `}` in an `array` or `tabular` environment. Fragile commands do not break in `\footnotes`.

Fragile commands can be protected by a `\protect` command placed just before each fragile command name. It is, of course, better to use equivalent robust commands where these are available in all places where fragility might be an issue.

Table 8.1: Fragile and Robust Commands

Fragile	Robust
`\(` and `\)`	`$`
`\[` and `\]`	`$$` (But it won't work with `fleqn` option.)
	`\noindent` and `\indent`
`\footnote`	
`\footnotemark` and `\footnotetext`	
`\twocolumn` and `\onecolumn`	
`\item`	
`\usecounter`	`_` `^` in maths
	`\frac`
`\sqrt`	log-like operators
`\underline`	`\overline`

Continued on next page

Continued from previous page

Fragile	Robust
	`\left`*delim* and `\right`*delim*
	Maths type styles
`\newcommand` and `\renewcommand`	
`\providecommand`	
`\newenvironment` and `\renewenvironment`	
`\newtheorem`	
`\newcounter`	`\value{`*ctr*`}`[1]
`\setcounter` and `\addtocounter`	Numbering commands (`\arabic{`*ctr*`}` `\qquad` etc. and `\the`*ctr*)
`\caption`	
`\label`[2]	
`\vline` `\hline` `\cline`	`\multicolumn`
`\cite` and `\nocite`	
`\includeonly` and `\include`	`\input` (I think)
`\index` and `\glossary`	
`\linebreak` and `\nolinebreak`	`\-`
`\\` `\\*` `\newline`	`\hyphenation`
`\pagebreak` and `\nopagebreak`	
`\enlargethispage`	`\newpage` and `\clearpage`
`\cleardoublepage`	`\fill`
`\newlength`	`\stretch` and `\setlength`
	`\addtolength`
`\markboth` `\markright`	`\setto`*dim*
`\vspace` `\vspace*`	`\hspace` `\hspace*`
`\bigskip` `\medskip` `\smallskip`	`\par`
`\addvspace`	
`\vfill`	`\hfill`
`\makebox` `\framebox`	`\mbox` `\fbox`
`\newsavebox`	`\usebox`
`\savebox`	`\sbox`
`\parbox`	
`\rule`	
`\raisebox`	
All commands for `picture` environment ...	except `\thicklines` `\thinlines`
Text size declarations (eg `\Large`).	Text font and face declr. (eg `\em`) and commands (eg `\texttt`). `\symbol`
	All length commands are robust and must not be preceded by `\protect`.
	Nor should a length command be used in an argument of a `\setcounter` or `\addtocounter` command.

A related issue, but not quite one of fragility, is that `\verb` may not appear in the argument of any other command (i.e. not in a sectioning command, or footnote, or caption, or ...).

[1] `\value{`*ctr*`}` gives current value of *ctr*—will print it or assign it to another counter.
[2] But `\label` can be used in a caption or any of the sectioning commands.

8.2 Counters and cross-references

A counter is an integer-valued variable that is, typically, incremented to keep a record of how many of the things being counted there have been. Cross-references are generated using counter values. Some of the most obvious counters are `page`, `section` (etc.), `\caption` in a `figure` or `table` environment[3], `enumi` (etc.), and `footnote`, which are incremented at each new page, section, figure, table, item in the outermost enumerate environment and footnote, respectively.

Where a counter is associated with a command (like `\chapter`) or environment (like `table`) the counter is given the same name as the command (`chapter`) or environment (`table`).

8.2.1 Built-in counters

LaTeX has 23 built-in counters. The most obvious ones are listed first, and those that are not so obvious are explained briefly:

`chapter`	`enumi enumii`	`enumiii enumiv`	`equation`
`figure`	`footnote`	`mpfootnote`[4]	`page`
`paragraph`	`part`	`section`	`subsection`
`subsection`	`subsubsection`	`subparagraph`	`table`

These are counters that typically are used to generate references. Other counters, more concerned with controlling the appearance of the document, are:

`bottomnumber`	Maximum number of floats (figures or tables) that can occur at the bottom of each text page. Default value of 1.
`dbltopnumber`	If `twocolumn` option is used (only). Maximum number of 2-column-floats at the top of a page. Default of 2.
`tocdepth`	See page 140. Determine which levels of heading are in the Table of Contents; the default is to show all headings to the level of subsection, so the default value is 3 in an article and 2 in a book or report class document.
`topnumber`	Maximum number of floats at the top of a text page. Default of 2. If `twocolumn` option is used, this affects only 1-column floats.
`totalnumber`	Maximum number of floats per text page. Default of 3.
`secnumdepth`	See page 136. Determine which sectional units are numbered. It is advisable to have `secnumdepth` $\geq$ `tocdepth`.

8.2.2 How references work

Each `\label{`*`key`*`}` command causes a `\newlabel{`*`key`*`}{{`*`counter`*`}{`*`pg`*`}}` command to be written to the `.aux` file.

The *`counter`* is the representation of the counter of the current "section", equation, figure, table or item in a list-type of environment, so that it might be 2.4.1 for the first subsection in the fourth section in the second chapter in a book (or the first subsubsection in the fourth subsection in the second section in an article) or 10 for the tenth equation in an article or 6.10 for the tenth equation in the sixth chapter in a book.

The *`pg`* is the page number on which the label is defined.

This means that `\ref{`*`key`*`}` causes the (compound) representation of *`counter`* to appear in the text, and `\pageref{`*`key`*`}` causes the page number to appear in the text. So that the `aux` file entry (which came from, say, the first equation in the sixth section in an `article` which has a label `\label{eq:six.1}`): `\newlabel{eq:six.1}{{6.1}{37}}` can be referred to as

```
... in equation~\eqref{eq:six.1} on page~\pageref{eq:six.1} we have ...
```

which will appear as

[3] Remember that the `\label` must be after the `\caption` in figures and tables, as it is actually the captions that are numbered—the environment names are merely used to give two clearly identifiable counters.

[4] Counter for footnotes in a `minipage` environment

... in equation (6.1) on page 37 we have ...

The use of the ~, a non-breaking space, is advisable to prevent lonely numbers at the start of a new line (or as the sole entry in the last line of a paragraph).

How counters are represented is determined by `\thectr`, which is a command to print a representation of the value associated with the counter `ctr`. The actual representation of the counter is determined by one of `\arabic{ctr}`, `\roman{ctr}`, `\Roman{ctr}`, `\alph{ctr}`, `\Alph{ctr}`, or `\fnsymbol{ctr}`. The two alphabetic representations require that $ctr \leq 26$, and `\fnsymbol` can only be used in math mode and with $ctr \leq 9$. The default setting of, say `page` representation is (in the body of a book, or in an article)

```
\newcommand{\thepage}{\arabic{page}}
```

which can be changed by a `\renewcommand`.

The default representation of, say, a subsection in a book is:

```
\newcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

where `\thesection` and `\thechapter` would have been defined as

```
\newcommand{\thechapter}{\arabic{chapter}}
\newcommand{\thesection}{\thechapter.\arabic{section}}
```

It is useful to notice that while the actual representation of the number can be controlled (Arabic or Roman numeral, or letter; a compound incorporating related counters), the face, shape and family of the font cannot be modified when the reference is stored.

A package providing more flexible references is described in GMS page 41.

8.2.3 New counters

Use of `\newtheorem{thm}{Theorem}` will (automatically) cause a new counter `thm` to be created (and incremented as necessary and stored for reference using `\label`). But `\newtheorem{lemma}[thm]{Lemma}` will use the existing counter `thm` and not create a new counter.

Sometimes you need to create counters of your own (for a new list-type environment, say) and this is done with the global declaration

`\newcounter{`ctr_1`}[`ctr_2`]`

which makes the variable ctr_1 a counter, initialised to 0. By default `\thectr`$_1$ is defined to be `\arabic{`ctr_1`}`.

If the option ctr_2 (an already defined counter) is used, then every time ctr_2 is incremented (by `\stepcounter` or `\refstepcounter`), the new ctr_1 is reset (to 0).

8.2.4 Manipulating counters

The commands to manipulate (change the value of) counters are much more powerful if the package calc is used, as then more arithmetic on the value of the counter is possible (see GMS pg 468). However, the basic operations are:

`\setcounter {ctr}{val}` A command that globally sets the value of the counter. It is, for instance, used to manipulate the value of `footnote` when `\footnotemark` and `\footnotetext` are used (in a `tabular` or `minipage` or mathematical environment) and to "restart" an `enumerate` environment (say, after some left-margin-aligned text in the middle of "one" `enumerate` environment that actually consists of two). For instance:

```
\begin{enumerate}
  \item First item.
  \item Second item.
\end{enumerate}
Text flush with left margin...
 ... general comment.
\begin{enumerate}
\setcounter{enumi}{2}
  \item Third item.
\end{enumerate}
```

1. First item.
2. Second item.

Text flush with left margin—say a relatively general comment.

3. Third item.

A better way to do this is given below.

`\addtocounter {ctr}{val}` A command that globally adds *val* to *ctr*. It could be used, for instance, to give an odd- (or even-)valued numbered environment. Or in manipulating the value of `footnote`.

`\value {ctr}` A command to produce the value of *ctr*; useful in `\setcounter` to use the value of one counter to set that of another. For instance, a neater version of the interrupted `enumerate` is to have:

```
\newcounter{temp}
\begin{enumerate}
    \item First item.
    \item Second item.
\setcounter{temp}{\value{enumi}}
\end{enumerate}
Text flush with left margin
... general comment.
\begin{enumerate}
\setcounter{enumi}{\value{temp}}
    \item Third item.
\end{enumerate}
```

1. First item.
2. Second item.

Text flush with left margin—say a relatively general comment.

3. Third item.

`\stepcounter {ctr}` A command that globally increments *ctr* by one and resets all counters made subsidiary to *ctr* in a `\newcounter{`ctr_1`}[`ctr_2`]` declaration.

`\refstepcounter {ctr}` As `\stepcounter`, but in addition resets the current `\ref` value to be the text generated by `\the`*ctr* (in other words, any `\label` will write `\the`*ctr* for *ctr* in the `\newlabel` command in the `.aux` file).

8.3 Footnotes

`\footnote[`*num*`]{`*text*`}`

- *num* is a number, no matter how `\thefootnote` is defined (the actual mark on the paper can be alphabetic or a symbol), and specifies the counter value to be used for the footnote mark (in the text) and the corresponding footnote (at the bottom of the page).
- If *num* is not specified, then the `footnote` counter is incremented by one and used.
- The command can only be used in paragraph mode (not in any kind of box).
- Footnotes in any box (or in maths mode) should be made using:

  `\footnotemark[`*num*`]`

  inside the boxes or maths environment, wherever that footnote mark is to appear; if the optional *num* is absent, `footnote` is stepped and used. And

  `\footnotetext[`*num*`]{`*text*`}`

  outside the "difficult" environment; the *text* will appear at the bottom of the page, along with any other footnotes on that page, and will be numbered *num*. The counter `footnote` is not stepped (after `\footnotetext`).

  This means that

```
\fbox{test\footnotemark\ more text}
\footnotetext{To be at the bottom of the page.}
```

will work just fine; `footnote` is stepped when `\footnotemark` is specified, and the same value of the counter is used to make the footnote.

But if two or more footnotes are made in the same box (or other) environment, the value of `footnote` will need adjusting:

```
\fbox{First text\footnotemark\ more text\footnotemark}
```

will leave `footnote` = 2 (at least, 2 more than before the `\fbox`), so its value must be adjusted (using `\addtocounter` or `\stepcounter`) before specifying the text for the footnotes:

```
\addtocounter{footnote}{-1}
\footnotetext{For first footnote.}
\addtocounter{footnote}{1}    % or \stepcounter{footnote}
\footnotetext{For the second footnote.}
```

- There are two style parameters for footnotes, see Lamport pg 173:

  `\footnotesep`, a strut producing vertical height between footnotes

  `\footnoterule`, the line separating footnotes and text. It would appear that the definition of this parameter is more complex than just the length of the line.

- In `longtable`, any footnotes appear on the page on which the table ends. See for instance the table on page 123.

- Footnotes in a `minipage` made using `\footnote` will be numbered using a different counter (`mpfootnote`), and will appear at the bottom of the `minipage`. The default representation of the counter is lower case letters.

  Footnotes made using `\footnotemark` (inside `minipage`) and `\footnotetext` (outside `minipage`) will appear as usual, and be numbered in the overall footnote sequence.

  Inside this `minipage` we can insert a footnote[a] in one step, or in two, using `\footnotemark`[5] like this and again, if necessary[6].

  [a] Using `\footnote`

```
\begin{minipage}{0.35\textwidth}
  Inside this {\tt minipage} we can insert a
  footnote\footnote{Using
                {\tt $\backslash$footnote}}
  in one step, or in two, using
  \verb1\footnotemark1\footnotemark\
  like this and again, if
  necessary\footnotemark.
\end{minipage}
\addtocounter{footnote}{-1}
\footnotetext{First footnote text.}
\stepcounter{footnote}
\footnotetext{And even if not really necessary.}
```

- In `article` class, footnotes are numbered $1, \ldots, n$, across the whole document; in `book` and `report` classes they are numbered within chapters.

- For fine-tuning the appearance of footnotes, see GMS, page 72 ff.

- To print all notes at the end of a (say) chapter, as endnotes, requires the package `endnotes` which is discussed in GMS on page 74–75.

[5] First footnote text.
[6] And even if not really necessary.

8.4 Landscaping

Slides printed in landscape mode are easily made using the seminar package (see page 119), and `xdvi` and `dvips` work much as usual, without too much extra user assistance.

Unfortunately, the same is not true for "normal" documents produced using the `landscape` option in the `\documentclass` command, so this possibility is discussed further below.

There is a package, rotating, which rotates text (or boxes) in a variety of ways, and which can be used to produce a single landscape page containing a table or figure, and this too is discussed.

8.4.1 Whole landscape document

The procedure (which involves some user assistance) here is:

1. Use the `landscape` option in the `\documentclass` command. You may need to fine-tune the width and length of the page, but the default settings are now such that `\textwidth` > `\textheight`.
2. Once you have a `.dvi` file, it will be displayed correctly if you use `View` on the `Command` menu in `Emacs` in `LaTeX` mode. If you like to give the `xdvi` command in an `xterm` window, the command needs to be:

   ```
   xdvi -paper a4r -s 4 filename.dvi
   ```

   to specify that the paper, `a4r`, is landscape; the `-s` is for "shrink"—the default is 3, and the possible values 1, 2, 3, 4 correspond to the 100%, 50%, 33%, 25% shrink options on the edge of the `xdvi` window. If the `-s` option is omitted, you get the default size (33%), and can adjust the window, once open.
3. A further option specification is necessary in the unix `dvips` command. It must be

   ```
   dvips -t landscape filename.dvi
   ```

   A consequence of this seems to be that when you then view the file in `Ghostview` or `gv` the image is upside down. However, the `seascape` option on the orientation menu will turn it right-side-up again, and the printing, in any event, will not be affected.
4. If you print the document on Press and use the default duplexing option, you will need to ensure that you select the `-Z` option that you want — either `-Zshortbind` or `-Zlongbind`.

8.4.2 A landscape page or part thereof

Sometimes a figure or table in a document will be wider than it is high, wider than the page, and would be better printed in landscape orientation. This can be done using the package rotating. See GMS page 320.

Note that the effects may only be seen in the PostScript version of the typeset document.

1. In the preamble insert:

   ```
   \usepackage[dvips]{rotating}
   ```
2. The package provides 5 environments that rotate text or any other box (eg graphics). Where the angle of rotation is specified, it measures a counter-clockwise rotation in degrees. (It is, however, incorrectly stated in GMS that the rotation is *clockwise*.)

   (a) The `rotate` environment number of degrees given in the argument, but does not leave space for the result (you can use a strut, a `\rule` of 0 width or height, to create space) which can be useful if you're rotating long headings in a table (in a `tabular` or `array` environment), but probably not otherwise.

   ... preceding text following text of more than one line—note that no space is left for the rotated text ...

   ```
   \ldots preceding text
   \begin{rotate}
     {-35}$-35^\circ$ turn
   \end{rotate} following text
   \ldots \ldots
   ```

```
\ldots preceding text

\begin{tabular}[t]{|*3{r}|}
       \hline
 \rule{0em}{2.8em}% vertical
                  % placement
 \begin{rotate}{45}First
 \end{rotate} &
 \begin{rotate}{45}Second
 \end{rotate} &
 \begin{rotate}{45}Third
 \end{rotate}\\\hline
    a & b & c\\
    d & e & f\\ \hline
\end{tabular}

more text \ldots
```

... preceding text

First	Second	Third
a	b	c
d	e	f

more text ...

The column spacing can be improved using two lots of struts: one for the header line height (as above) and the other in each column, to increase the space between columns:

```
\ldots preceding text

\begin{tabular}[t]{|*3{r|}}
                    \hline
 \rule{0em}{2.8em}% vertical
                  % placement
 \begin{rotate}{45}First
 \end{rotate}
   \rule{2.7ex}{0pt}&
 \begin{rotate}{45}Second
 \end{rotate}
   \rule{2.7ex}{0pt}&
 \begin{rotate}{45}Third
 \end{rotate}
   \rule{2.7ex}{0pt}\\\hline
    a & b & c\\
    d & e & f\\ \hline
\end{tabular}

more text \ldots
```

... preceding text

First	Second	Third
a	b	c
d	e	f

more text ...

(b) The `turn` environment rotates text through the number of degrees given in the argument, and does leave space for the rotated box.

... preceding text −55° turn which is equivalent to −55° turn using the graphics package.

```
\ldots preceding text
\begin{turn}{-55}
  $-55^\circ$ turn
\end{turn} which is
equivalent to
\rotatebox{-55}{$-55^\circ$
turn} using the
\pk{graphics} package.
```

In a `tabular` environment, struts are not necessary, but the columns are fairly widely spaced, which is only a problem if the other column entries are narrower than the rotated parts.

... preceding text

Left	Centred	Right
a	b	c
d	e	f

more text ...

```
\ldots preceding text

\begin{tabular}[t]{|l|c|r|}
       \hline
 \begin{turn}{45}Left
 \end{turn} &
 \rotatebox{45}{Centred} &
 \begin{turn}{45}Right
 \end{turn}\\\hline
    a & b & c\\
    d & e & f\\ \hline
\end{tabular}

more text \ldots
```

Note the equivalence of `\rotatebox` and `turn`.

(c) The `sideways` environment is equivalent to a turn though 90°.

```
In {\tt sideways}, there is
the \begin{sideways}
     $90^\circ$ default
    \end{sideways} and there
 are no other options.
```

In sideways, there is the 90° default and there are no other options.

It is possible to rotate a table by putting the `tabular` environment inside a `sideways` environment:

... preceding text

Left	Centred	Right
a	b	c
d	e	f

more text ...

```
\ldots preceding text

\begin{sideways}
\begin{tabular}[t]{|lcr|}
       \hline
 \begin{turn}{45}Left
 \end{turn} &
 \rotatebox{45}{Centred} &
 \begin{turn}{45}Right
 \end{turn}\\\hline
    a & b & c\\
    d & e & f\\ \hline
\end{tabular}
\end{sideways}

more text \ldots
```

Note that there can be nested rotations, as in the example above.

(d) The `sidewaystable` environment will rotate both the table and its caption—see Table 7.1, page 111, which was constructed with the code:

```
\begin{sidewaystable}
\caption{Default options for the document classes\label{tab:def}}

  \begin{tabular}{|*7{l|}}\hline
    Option & General default & {\tt article} & {\tt book} &
    {\tt report} & {\tt letter} & {\tt slide}\\ \hline
.................
    \multicolumn{7}{|p{0.9\textheight}|}{{\tt onecolumn},
      {\tt twocolumn}---the number of columns on a page for most of the
      document. If parts ....
```

```
      will print the \opt{text} across the top of the page.
      The \opt{text} can contain sectioning commands.}\\ \hline
    \end{tabular}
  \end{sidewaystable}
```

(e) The `sidewaysfigure` environment will rotate both the figure and its caption, in the same way. (To rotate the figure or table only, not the caption, use `turn` or `\rotatebox` or `sideways` in a `figure` or `table` environment.)

Thus, for a `figure` or `table`, the options are:

1. The usual:

```
\begin{figure}[h!tbp]
  \begin{center}
    \leavevmode
    \includegraphics{foo.eps}
%or \epsfig{....}
    \caption{Normal}
    \label{fig:rotnorm}
  \end{center}
\end{figure}
```

2. A rotated figure or table, usual caption orientation:

```
\begin{table}[h!tbp]
  \begin{center}
    \leavevmode
    \begin{sideways}
      \begin{tabular}{....}
%could has well have been a figure
      \end{tabular}
    \end{sideways}
    \caption{Rotated table.}
    \label{tab:rottab}
  \end{center}
\end{table}
```

3. A `sidewaystable` or `sidewaysfigure`, in which both the caption and the figure/table are rotated.

8.5 Text typeset in two or more columns

Vanilla LaTeX offers two ways to toggle between one- and two-column formats within a document. The biggest disadvantages of using these methods are that

- a switch from the one layout to the other causes a `\clearpage` to be issued (that is, a page can only contain one type of layout),
- there can be at most two columns of text,
- if the text in the two columns does not fill the page, there is no attempt made to balance the contents of the two columns.

The package multicol offers much greater flexibility (in some ways), allowing up to 10 columns of text, and attempting to balance (or not, in which case there is some control as to the degree of the lack of balance in the right-most column) the text in the columns.

8.5.1 Vanilla LaTeX options

8.5.1.1 The `twocolumn` option

Specification of this option in the `\documentclass` command sets length parameters such as `\parindent` and list-type environment indents to values deemed suitable for two-column typesetting. So this is probably the most suitable way to set up a document all or most of which is to be typeset in two columns.

The declaration `\onecolumn` will cause a `\clearpage` to be issued and the next page to be typeset in a single column (but with the same two-column values of the length parameters).

A subsequent declaration `\twocolumn` will cause a `\clearpage` to be issued and the next page to be typeset in two columns.

When the `twocolumn` option is chosen, the command `\marginpar` (see Section 8.9.2, page 148) puts marginal notes in the margin nearest to the column containing the associated text, and this is not affected by whether one-sided or two-sided printing is in force.

A gap of length `\columnsep` (a rigid length parameter, default 10pt) is left between the columns of text, and a rule of width `\columnseprule` (default of 0pt) is placed in the gap. The width of a single column of text is `\columnwidth` (a rigid length parameter), and is calculated by LaTeX from the values of `\textwidth` and `\columnsep`. The value of `\columnwidth` should not be altered by the user. Any lengths within a column are probably best specified in terms of `\columnwidth`, as in `\begin{minipage}{0.9\columnwidth}`.

In two-column output both one-column and two-column floats (typically, `figure` and `table`) are available. The environments `figure*` and `table*` are one column wide (ie as wide as the page), and `figure` and `table` are a single column wide (ie are within a column on a two-column page). The starred form of the environments will by default tend to float to single-column pages of floats, although use of counters such as `dbltopnumber` (see Section 8.2) will give some control over float placement.

`\dblfigrule` is only available with the `twocolumn` document class option, and determines what is to appear between the float and text below it when a double-column float is to appear at the top of a page. The default is nothing, but re-defining this command can put a line, say. Whatever is inserted should not add to the vertical height of the page. The rule (or whatever) will appear after the last of the float(s) to appear on the page, even if the second and following floats at the top of the page are single-column.

`\dblfloatpagefraction` is the equivalent of `\floatpagefraction` (see page 107); the default is 0.5.

`\newpage` ends the current paragraph and current column. `\pagebreak` ends the current column (not page). `\nopagebreak` affects the current column (not page).

Footnotes are, by default, placed at the bottom of the two separate columns. The package ftnright causes all the footnotes for a page to be placed at the bottom of the right-hand column. It also implements (automagically) some fairly minor changes to the formatting of the footnotes. It is recommended that ftnright be the last package listed, to prevent any of its settings being overwritten by other packages.

8.5.1.2 The `\twocolumn` declaration

In a document that is mainly typeset one-column, with length parameters suitable for one-column formatting, it is possible to have some pages typeset in two columns by using the same `\twocolumn` and `\onecolumn` declarations (but **not** specifying the `twocolumn` option).

The declarations have exactly the same effect as if the `twocolumn` option is specified, apart from the preset values of the length parameters.

8.5.2 The multicol package

This package allows the use of different numbers of columns in different sections of the same page. The environment `multicols` is defined that:

- can create any number of columns (up to 10), which can fill several pages (or part of one page);
- when the environment ends, the columns on the last page will be balanced so that they are of nearly equal length;
- can be used inside other environments such as `figure` or `minipage` where it will produce a box containing the text distributed between the requested number of columns;
- can insert vertical rules of user-defined width between adjacent columns;
- can customise the formatting globally or for individual environments.

The `\begin` command for the environment (the environment is ended by `\end{multicols}`) has the form:

`\begin{multicols}{`*cols*`}[`*preface*`][`*skip*`]`

where *cols* is the (mandatory) argument specifying the number of columns required (up to 10), and the optional arguments are *preface* which is any text (such as sectioning command(s)) to be typeset in single-column format before the multi-columned text (this will be kept on the same page as the start of the multi-columned text, a new page being started if there is insufficient space) and *skip* which is a local value for the minimum amount of space required at the bottom of the page for the start of the multi-columned text (there is a default value for the amount of space—see below).

The formatting above was achieved using:

```
\columnseprule=0.1pt     %This causes ALL multicol environments to have a rule
                         %between the columns, until otherwise specified. Had
                         %the value been changed INSIDE a multicols environment,
                         %it would have applied to that environment only.
\begin{multicols}{3}\raggedright   %\raggedright can prevent excessive
                                   %hyphenation.
\raggedcolumns                     %In fact, in this case, this has little effect.
This package allows ... individual environments.
\end{itemize}
\end{multicols}

\begin{multicols}{2}[The \texttt{\bsl begin} command for the environment (the
  environment is ended by \texttt{\bsl end\{multicols\}}) has the form:]
                                   %The preface.

\verb1\begin{multicols}{1 ....  amount of space---see below).
\end{multicols}
```

The parameters controlling formatting and their associated default values (which should be changed before the start of the environment to which they are to apply) are:

`\premulticols` The minimum amount of free space at the bottom of the page if the multiple columns are to start on that page. Default value of 50pt. If *skip* is specified in one of the environments, that value is used. If the remaining space is less than the minimum specified, then a `\newpage` command is issued.

`\postmulticols` The minimum amount of space left on a page after the `multicols` environment for the next text (or whatever) to appear on the same page. Default of 20pt. If the remaining space is less than the minimum specified, then a `\newpage` command is issued.

`\multicolsep` The vertical (stretchy) space placed before and after a `multicols` environment. Default of `12pt plus 4pt minus 3pt`.

`\columnsep` The space left between adjacent columns. Default of 10pt. The width of the columns is calculated automatically from the number of columns, `\textwidth` and `\columnsep`.

`\columnseprule` The width of the vertical line between adjacent columns. Default of 0pt (no rule). A rule width greater than `\columnsep` will overprint the column text.

Formatting is also controlled by the declarations and counters:

`\flushcolumns` The default, which causes the environment to try to keep all columns the same length by stretching the available vertical space inside the columns.

`\raggedcolumns` The alternative to `\flushcolumns`; places all surplus space at the bottom of each column.

`unbalance` A counter with default value of 0, which measures the number of extra lines of text to be typeset in the left-most column (which adds space in the bottom of the right-most column). Reset (to 0) at the end of the environment.

`columnbadness` **and** `finalcolumnbadness` Counters used by the typesetting algorithm to decide on column breaks and formatting (the algorithm seeks optimal values of these counters among the possible solutions).

`tracingmulticols` Counter with default value of 0. If set > 0, it will cause a trace of the column setting algorithm, with higher values giving more tracing information.

`collectmore` Counter with default of 0, which controls how far ahead `multicols` looks to assemble material. It is possible that if a footnote appears on a different page to the text referring to it, an increase in `collectmore` will prevent this happening, or the situation may be more simply resolved by the strategic positioning of a `\pagebreak`.

The use of some of these formatting controls is illustrated in a modified repeat of the multicolumn text above:

```
\columnseprule=0.1pt     %This causes ALL multicol environments to have a rule
                         %between the columns, until otherwise specified. Had
                         %the value been changed INSIDE a multicols environment,
                         %it would have applied to that environment only.
\begin{multicols}{3}\raggedright   %\raggedright can prevent excessive
                                   %hyphenation.
This package allows       ...          individual environments.
\end{itemize}
\end{multicols}

\begin{multicols}{2}[The \texttt{\bsl begin} command for the environment (the
  environment is ended by \texttt{\bsl end\{multicols\}}) has the form:]
                                   %The preface.
\raggedcolumns \raggedright  \columnsep=20pt
\setcounter{unbalance}{4}          %Cause left column to have 4+ more lines than
                                   %the right column.
\columnseprule=0pt                 %No column rules in this environment.
\verb1\begin{multicols}{1 ....  amount of space---see below).
\end{multicols}
```

which gives the result:

This package allows the use of different numbers of columns in different sections of the same page. The environment `multicols` is defined that:

- can create any number of columns (up to 10), which can fill several pages (or part of one page);
- when the environment ends, the columns on the last page will be balanced so that they are of nearly equal length;
- can be used inside other environments such as `figure` or `minipage` where it will produce a box containing the text distributed between the requested number of columns;
- can insert vertical rules of user-defined width between every two columns;
- can customise the formatting globally or for individual environments.

The `\begin` command for the environment (the environment is ended by `\end{multicols}`) has the form:

`\begin{multicols}{`*`cols`*`}[`*`preface`*`][`*`skip`*`]`

where *`cols`* is the (mandatory) argument specifying the number of columns required (up to 10), and the optional arguments are *`preface`* which is any text (such as sectioning command(s)) to be typeset in single-column format before the multi-columned text (this will be kept on the same page as the start of the multi-columned text, a new page being started if there is insufficient space) and *`skip`* which is a local value for the minimum amount of space required at the bottom of the page for the start of the multi-columned text (there is a default value for the amount of space—see below).

Note that the starred version of `figure` and `table` (page-width) are supported in a `multicols` environment, but the unstarred (within a column) versions are not. And `\marginpars` are not supported either.

Footnotes are typeset (full width) at the bottom of the page.

Note also that the `twocolumn` option and `multicols` environment are not compatible with several packages, in particular those offering fancy placement of floats or special text formatting.

8.6 Sectioning commands

8.6.1 Default effects

- The size, face, and fonts used are determined by the document class.
- The "first layers" of the sections are numbered—see below.
- Table of Contents entries are generated for the numbered sections (and some front and back matter that is not numbered).
- Running headers are generated, depending on class and page style.

8.6.2 Managing long headings

Where headings are too long to fit across the top of the page in a header (for example, the title of this chapter), or to fit onto one line in the Table of Contents, there are two ways (actually exactly equivalent) of defining a shorter version of the section heading.

- `\section[`*`toc-entry`*`]{`*`full title`*`}`, where the *`toc-entry`* is shorter than the *`full title`*.
- `\section{`*`full title`*`}`
  `\sectionmark{`*`toc-entry`*`}`
  The two methods are equivalent as the *`toc-entry`* option in `\section` (or other sectioning command) is stored in `\sectionmark` (that should be `\`*`section`*`mark` as there are `\chaptermark`, etc. commands).

8.6.3 To have neither a Table of Contents entry, nor numbered headings

Use the starred version of the sectioning commands, like `\section*{...}`. The heading generated is unnumbered and no Table of Contents entry is made. The heading will be of the same size and font, with undesirable page breaks before and after the heading suppressed, as for the unstarred version.

8.6.4 Control of numbering and Table of Contents entry

The counter `secnumdepth` determines how many levels of heading are numbered.

- The default values are 3, 2, 2 in `article, book, report` class, respectively, which corresponds to numbering down to `\subsubsection` (3 in `article`) and `\subsection` (2 in `report, book`).
- To have no numbered headings at all, put `\setcounter{secnumdepth}{-2}` in the preamble.

The counter `tocdepth` determines how many levels of heading are in the Table of Contents. Default values are the same as for `secnumdepth` in the different classes, so that all numbered levels of heading appear in the Table of Contents.

The value of the counters can be changed with `\setcounter`.

8.6.5 Entering unnumbered "sections" in the Table of Contents

This is mainly useful in the `article` (or `report`) class where (apart from Part) the highest level of heading is `\section` (`\chapter` for `report`), and all sections (chapters) either are (if `secnumdepth` ≥ 1 for both classes) or are not (if `secnumdepth` < 1) numbered. But sections (or chapters for `report` class) at the front and back of the document, such as Preface (`\section{Preface}` or `\chapter{Preface}`) or Glossary or a single Appendix (which, if it follows the `\appendix` declaration will be numbered A)

would, ideally, not be numbered but would be listed in the Table of Contents (if there is more than one appendix, they will need to be numbered).
This can be achieved by, for instance,

```
\section*{Foreword}
\addcontentsline{toc}{section}{\numberline{}Foreword}
```

`\numberline{}` leaves a space where the section number would go. Alternately,
`\addcontentsline{toc}{section}{Foreword}`
would typeset "Foreword" flush left.
The same method can be used in `book` class documents to get the bibliography and index listed in the Table of Contents (Lamport and GMS imply it is automatic, but this does not seem to be true).
In this case the entries are on the lines of

```
\addcontentsline{toc}{chapter}{Bibliography}
\bibliography{file.bib}
\addcontentsline{toc}{chapter}{Index}
\printindex
```

8.6.6 Type of numbering

The command `\the`*`secname`* determines how the heading is numbered, with the default being something like

For a book or report

```
\newcommand{\thechapter}{\arabic{chapter}}
\newcommand{\thesection}{\thechapter.\arabic{section}}
\newcommand{\thesubsection}{\thesection.\arabic{subsection}}
```

For an article

```
\newcommand{\thesection}{\arabic{section}}
\newcommand{\thesubsection}{\thesection.\arabic{subsection}}
\newcommand{\thesubsubsection}{\thesubsection.\arabic{subsubsection}}
```

Which produces (book) headings like: Chapter 1 Chaptername
1.1 Sectionname
1.1.1 Subsectionname

The definitions can be changed using `\renewcommand`. The choices for representing counters are

`\arabic{`*`ctr`*`}`	1, 2, ...
`\alph{`*`ctr`*`}`	a, b, ...
`\Alph{`*`ctr`*`}`	A, B, ...
`\roman{`*`ctr`*`}`	i, ii, ...
`\Roman{`*`ctr`*`}`	I, II, ...

If you give one of these sectioning commands a `\label{sec:P}`, then the `\ref{sec:P}` reference will give the reference number in the format specified.

8.6.7 Fancy formatting in the numbering of the heading

If you want fancy formatting around the number in the heading, (say, stars and stripes or boxes), then you need to get into the TEX of it:

```
\makeatletter
\renewcommand{\@seccntformat}[1]{***{\csname the#1\endcsname}*** }
\makeatother
```

would make each section heading (all levels numbered) appear like:

8.6 Sectioning commands

But, if this section were referenced in the text, the reference would be as follows: in Section 8.6 ... (no stars).

```
\makeatletter
\renewcommand{\@seccntformat}[1]%
      {\fbox{\csname the#1\endcsname}\hspace{0.5em}}
\makeatother
```

would make each section heading (all levels numbered) appear like:

8.6 Sectioning commands

But, if this section were referenced in the text, the reference would be as follows: in Section 8.6 ... (no box).

These commands must appear in the preamble. They will affect all levels of sectioning commands.

Because the @s are used, and must have their TEX (and not LATEX) meaning, you need the `\makeatletter` first (to turn it on) and `\makeatother` after (to turn it off).

8.6.8 Format of whole heading (Note the different format for this heading.)

(GMS pg 25)

For the sectioning commands `\section`, ... , `\subparagraph`, the command `\@startsection` controls the name, level, indentation (of the heading) space before and after, and the style of the heading. It can be used to define a new kind of heading (using `\newcommand`), which requires some care (must read GMS pg 24), or to alter an existing command (using `\renewcommand`). Say we redefine `\subsection`.

```
\makeatletter
\renewcommand{\subsection}{\@startsection
  {subsection}%     % Name - MUST be specified.
  {2}%              % Level: part = -1, chapter=0, ..., subpara=5.
  {0mm}%            % Indent before no./heading; specified in mm, in,
                    % em, ex etc - fixed lengths.
  {-\baselineskip}  % Beforeskip, in addition to a \parskip.
                    % If <0, 1st para not indented (default).
  {0.5\baselineskip}% Afterskip, in addition to a \parskip.
                    % If <0, heading is run-in (in 1st line text).
                    % If >0, heading is displayed.
                    % Beforeskip and afterskip stretchy, so best defined in terms
                    % of \baselineskip. Usually beforeskip > afterskip.
  {\normalfont\large\itshape\centering}%
                    % Style, specifying face, font, size. Can also
                    % specify \centering, \raggedleft or \raggedright
                    % to position the heading in the line.
}
\makeatother
```

To illustrate such a command, the format of the subsection heading (which is 8.6.8 Format of a whole heading, page 138) has been changed—but note how the reference appears in the text.
(`which is \ref{sec:headformat} Format of a whole heading, page~\pageref{sec:headformat}`)

As was pointed out in Section 8.2 of these notes, markup of the counter cannot be stored in the `\newlabel` in the `.aux` file. If you wanted matching labels, they could be achieved by something on the lines of

```
\newcommand{\secref}[1]{Section~\textit{\ref{#1}}}
```

which would have the effect: ... which is Section *8.6.8* Format of a whole heading ...
(... `which is \secref{sec:headformat} Format of a whole heading ...` .

To change a sectioning command (heading), firstly, the actual definition of the change should be given outside any environments (such as list-type environments) which would limit the scope of the changes to be within that environment, and secondly, { ... } can be used to limit the effect to that section heading (all the following headings revert to the default).

It can be a good idea to look up the default definition of the command you want to change in the `.cls` or `.sty` file used for your document, and make your changes accordingly.

If you want to force a heading to have either all upper or all lower case letters, there are two commands that will do this: `\MakeUppercase` and `\MakeLowercase`. The effect of `\MakeUppercase{AaBbCc}` is AABBCC, and of `\MakeLowercase{AaBbCc}` is aabbcc.

To change the formatting of the `\part` or `\chapter` command, you need to use `\secdef`. See GMS pg 28.

8.6.9 Changing names

Say you want to have "Summary" appear instead of "Abstract", or "Addendum" instead of "Appendix". The commands:

```
\contentsname  \listfigurename   \listtablename  \bibname
\indexname     \chaptername      \refname        \appendixname
\partname      \abstractname
```

define the text to be inserted at the start of the Table of Contents, ... , Abstract (i.e. when the respective command is given or environment begun: `\tableofcontents`, ... , `\begin{abstract}`).

Note that "text" does not mean "word", so that if, for some reason, you wish to add a paragraph or two at the top of the index or whatever (and this can include sectioning commands) this can be done when defining the new name for the index.

`\figurename` and `\tablename` determine what is written at the start of a caption for a figure or table environment, respectively (Figure and Table, by default).

`\seename` defines the text to be inserted in an index entry of the form `\index{age|see{era}}`.

These names can be changed (in the preamble) like any other command, for instance as in:

```
\renewcommand{\abstractname}{Summary}
\renewcommand{\appendixname}{Addendum}
```

8.7 Fine-tuning a Table of Contents, etc.

See GMS pg 31—there are several options as to how these tables are presented. Some of the possibilities are discussed below:

8.7.1 Long numbers

One problem in a really long document (like a book) with a lot of chapters (more than 9 is typically where the problems start), each with many subsections (again, some running into double digits), is that the left alignment of the entries in the table of contents may not be optimal.

What is needed is an increase in the amount of space allocated to the numbers (of sections, subsections, etc.) in the table of contents when it is typeset. The command to format an entry in a table of contents file is:

`\@dottedtocline{`*`level`*`}{`*`indent`*`}{`*`numwidth`*`}{`*`text`*`}{`*`page`*`}`

The parameters in this command are:

level	The nesting level of the entry—typically, section is level 1, subsection is level 2, ...
indent	The total indentation from the left margin—the space between the left margin and the number (if given) at the start of the entry in the table.
numwidth	The width of the box that contains the number (if given) — entries with a higher value of *level* will have this space added to their *indent*, as will the second, and following lines of entries more than one line long. In other words, this is the width of the box used to preserve the vertical alignment of the *text* of entries of the same *level*. And this is the value that usually needs to be changed.
text	The actual text to appear in the entry.
page	The page number associated with *text*.

There are formatting parameters that control the width of the box in which the page number is typeset, the indentation of the right margin for entries that are more than one line long (used on all but the last line of the entry), and the distance between the dots between the text and the page number (see GMS pg. 33). But the defaults will probably be acceptable in most cases.

More likely to be of interest are *indent* and *numwidth*.

The default settings for each of the *level*s are determined in the `book.cls` (or whatever) document, and for `book` and `report` classes they are

```
\newcommand*\l@section{\@dottedtocline{1}{1.5em}{2.3em}}
\newcommand*\l@subsection{\@dottedtocline{2}{3.8em}{3.2em}}
\newcommand*\l@subsubsection{\@dottedtocline{3}{7.0em}{4.1em}}
\newcommand*\l@paragraph{\@dottedtocline{4}{10em}{5em}}
\newcommand*\l@subparagraph{\@dottedtocline{5}{12em}{6em}}
```

and for `article` class

```
\newcommand*\l@subsection{\@dottedtocline{2}{1.5em}{2.3em}}
\newcommand*\l@subsubsection{\@dottedtocline{3}{3.8em}{3.2em}}
\newcommand*\l@paragraph{\@dottedtocline{4}{7.0em}{4.1em}}
\newcommand*\l@subparagraph{\@dottedtocline{5}{10em}{5em}}
```

Note that the highest levels of entry (part or chapter for books and reports; section for articles) have a more complicated definition that includes extra space above the entry, and maybe prints that entry in bold and without dots. These entries seldom present the type of problem being addressed here, and so are not discussed further.

Note also that *indent* for a higher value of *level* is equal to *indent* + *numwidth* for the preceding level. So if *numwidth* (or *indent*) for one sectioning level is altered, the values of *indent* must be adjusted accordingly for all sections with a higher value of *level*.

The levels more likely to be of interest can be reset by giving a command along the lines of

```
\makeatletter
\renewcommand{\l@section}{\@dottedtocline{1}{1.5em}{3em}}
\makeatother
```

which would make the box allocated for the numbers of the sections `3em` instead of `2.3em` wide.

A certain amount of trial and error will be necessary to determine exactly what widths are needed in a particular document.

8.7.2 How much to show?

How much detail should be in each table of contents? Chapters only here, right down to sub-paragraphs there, maybe. The counter which controls how much detail is given is `tocdepth`, which has the default values of 2 in book and report classes (parts, chapters, sections and subsections) and 3 in article class (sections, subsections and subsubsections).

As a counter, its value can be altered using `\setcounter` as in

```
\setcounter{tocdepth}{4}
```

which would give paragraph entries (in any class of document).

8.7.3 Direct entry and new mousetraps

It is possible to edit a contents file directly and to make a table of things other than contents, figures or tables (say, examples) —discussed in GMS pg. 35.

8.8 List Structures

You can

- ➥ modify predefined lists across the whole document, by inserting the changes in the preamble;
- ➥ modify one or more features of a list environment by putting the changes inside `{...}`s within the body of the document;
- ➥ modify the appearance of one or more items by making changes within a list-type environment;
- ➥ define your own list-type environments.

This topic is discussed at length in GMS pg 56 ff; the `dinglist`-type environments are on pg 335.

8.8.1 `enumerate`

The appearance of the four levels of `enumerate` are determined by the following:

- There are 4 counters (*`ctr`*): `enumi, enumii, enumiii, enumiv`.
- The command `\the`*`ctr`* determines how the counters are represented, and the default definitions are:

  `\arabic{enumi}  \alph{enumii}  \roman{enumiii}  \Alph{enumiv}`

- The format of the label field is determined by the command `\label`*`ctr`*, and the default definitions are:

  `\theenumi.  (\theenumii)  \theenumiii.  \theenumiv.`

  Thus we'd get:

  1. The outer level.
  2. (a) The next level.
     (b) i. The third level.
         A. The fourth level
  3. Back to the first.

```
Thus we'd get:
\begin{enumerate}
 \item The outer level.
 \item
  \begin{enumerate}
    \item The next level.
    \item
     \begin{enumerate}
       \item The third level.
        \begin{enumerate}
          \item The fourth level
        \end{enumerate}
     \end{enumerate}
  \end{enumerate}
 \item Back to the first.
\end{enumerate}
```

- The prefix (when using `\ref`) is determined by the command `\p@`*`ctr`*, with the default definitions:

```
\p@enumi     {}      %ie nothing and no space
\p@enumii    \theenumi
\p@enumiii   \theenumi(\theenumii)
\p@enumiv    \p@enumiii\theenumiv
```

which would give results of \ref{*keystring*} like: 1, 2c, 3(d)ii, 4(a)iiiE.

All redefinitions of these commands should be enclosed in \makeatletter ... \makeatother (because of the @s). To redefine all third level numbering to be enclosed in (), the following would be inserted in the preamble:

```
\makeatletter
\renewcommand{\labelenumiii}{(\theenumiii)}
\makeatother
```

Changing both the type of representation (\the*ctr*) and the formatting around it (\label*ctr*) can be done in either 1 or 2 steps—so long as you don't intend to refer to the counter (with \ref{...}). For example, to change the first level to have representation (i), (ii), ... , and then, later, (a), (b) ... :

```
\makeatletter
\renewcommand{\theenumi}{\roman{enumi}}
\renewcommand{\labelenumi}{(\theenumi)}
\makeatother
Which has the visual effect:
\begin{enumerate}
  \item \label{enu:1} Here.
    \begin{enumerate}
      \item Next level.\label{enu:2}
    \end{enumerate}
\end{enumerate}
```

Which has the visual effect:

(i) Here.

(a) Next level.

and

```
\makeatletter
\renewcommand{\labelenumi}{(\alph{enumi})}
\makeatother
Which has the visual effect:
\begin{enumerate}
\item \label{enu:3} Here.
\end{enumerate}
```

Which has the visual effect:

(a) Here.

The question of representation is solved in \ref{enu:1} which gives i, \ref{enu:2} which gives (i)a, or \ref{enu:3} which gives 1.

So that the short form is fine to change the immediate representation, but not to both change the representation and provide similar-looking cross-references.

If the counters are to be used in cross-references, it would probably be better in such cases to redefine \p@enumii too:

```
\makeatletter
\renewcommand{\p@enumii}{(\theenumi)}
\makeatother
```

which gives cross-references: i (\ref{enu:1}), (i)a (\ref{enu:2}) and 1(\ref{enu:3}). Note that this last redefinition needs to be done **before** the start of the first enumerate environment it is to affect (ie before the label is written to the .aux file), and also that its effect can be restricted by use of {...} (to have a global effect, the redefinition would need to be in the preamble; to have an effect on all that follows it would not be in an environment nor in {...}).

- It is possible to redefine `\labelctr` within the `enumerate` environment—for instance to show which questions in a tutorial are to be handed in.

```
\begin{enumerate}
  \item An item that is just for
    the tutorial.
  \item And another.
\renewcommand{\labelenumi}{\theenumi.$^*$}
  \item First question to be handed in.
  \item Second question to be handed in.
\renewcommand{\labelenumi}{\theenumi.}
  \item Next (third) tutorial question.
  \end{enumerate}
```

1. An item that is just for the tutorial.
2. And another.
3.* First question to be handed in.
4.* Second question to be handed in.
5. Next (third) tutorial question.

- There is an enumerate package (GMS pg. 58) which provides flexibility of definition, as each separate `enumerate` environment has an option to define the labelling system, as well as to add text to appear in the numbering of each item—see Table 8.2.

Table 8.2: Use of enumerate package.

```
\begin{enumerate}[{A}-1.]
 \item These items are at ....

  \label{enu:A1}

  \begin{enumerate}[(i)]
    \item These are at the second
         level.
    \item As many as are necessary.
      \begin{enumerate}[Note {[}a{]}]
        \item It is very easy to add
           text to the numbering.
           \label{enu:Na}
        \item But every extra package
           can slow down typsetting.
      \end{enumerate}
    \item  In the inner level, as
      for the outer level, in order
      to get square brackets in the
      numbering, it is necessary to
      specify \verb4[Note {[}a{]}]4.
  \end{enumerate}
    \item Back to outer level.
\end{enumerate}
```

A-1. These items are at the outer level. Note that to get the label A- on each item it is necessary to specify `[{A}-1.]` as an A alone would indicate `Alph` representation of the counter.

(i) These are at the second level.

(ii) As many as are necessary.

Note [a] It is very easy to add text to the numbering.

Note [b] But every extra package can slow down typesetting.

(iii) In the inner level, as for the outer level, in order to get square brackets in the numbering, it is necessary to specify `[Note {[}a{]}]`.

A-2. Back to outer level.

If, however, you add text to the item numbers, any reference will give the counter representation only, not the text, as in `\ref{enu:A1}` which gives 1, and `\ref{enu:Na}`, which gives 1(ii)a. To get representations that include the text, it is necessary to add the text by hand, as in `Note A-\ref{enu:Na}`, which gives Note A-1(ii)a.

8.8.2 itemize

The appearance of the four levels of `itemize` are determined by the following:

- The four counters are `itemi, ... , itemiv`.
- The label fields are determined by `\labelctr`, and the default commands are:
  `\m@th\bullet  \bfseries---  \m@th\ast  \m@th\cdot`

giving representations •, –, ∗, ·.
The `\m@th` ensures there's no extra space around the mathematical character (as there would be around `$\bullet$`).

- Changes, as for `enumerate`, can be global (if made in the preamble), or for one or more list environments (if in the body of the document), or even a single item in a single list environment.
- Good hunting grounds for new (alternative) characters are the mathematical symbols (Tables 4.1–4.7 in these notes) and ZapfDingbats (GMS pg 336), which require use of the pifont package (add it to the `\usepackage` list).
  e.g. `\renewcommand{\labelitemi}{\ding{46}}` which uses ✎ for item labels, or `\ding{228}` which uses ➤, or `\ding{164}` which uses ❤.
- The pifont package provides the `dinglist` and `dingautolist` environments. The argument in the `dinglist` environment specifies the number of the character to be used in the list, and the argument of the `dingautolist` environment specifies the starting point of one of the series of encircled numbers: 172 (①), 182 (❶), 192 (①), 202 (❶)—but note that such lists cannot contain more than 10 items and be correctly numbered.

An example of a `dinglist` is on pg 141.

```
\begin{dinglist}{229} \label{eg:ding}
\item modify ..
\item modify one or more ...
\item modify the appearance ...
\item define your own list-type environments.
\end{dinglist}
```

```
\begin{dingautolist}{192}
  \item Here is an example of the
    list.
  \item References work ``as they
    should''.
  \item So an example of a
    reference to one of these items
     is not given here.
\end{dingautolist}
```

① Here is an example of the list.

② References work "as they should".

③ So an example of a reference to one of these items is not given here.

8.8.3 description

By the nature of the environment, the only possible formatting changes are to the way the label is typeset—this is done by the `\descriptionlabel` command, which has one parameter, the label text:

```
\renewcommand{\descriptionlabel}[1]{new format{#1}}
```

where `new format` might be `\textit` or `\normalfont` or `\textsf` or whatever. Default is `\textbf`.
The face, series and family of individual `description` labels can be changed by including the appropriate command(s) in the label, eg

```
\item[\textsf{\textit{Sans serif italics}}]
```

8.8.4 New list environments

"List" environments include `quote, quotation, verse, center, flushleft` and `flushright`, as well as the three discussed above, so it's possible to generate fairly "unlisty" effects using one of these environments. The basic definition is:

```
\begin{list}      % Give name of (new) list.
 {default label}  % Text used at each \item if optional argument not specified.
 {decls}          % Sets up parameters of vertical & horizon. lengths using \setlength.
 item-list        % The actual list of \items.
\end{list}        % Close environment.
```

and would be given as part of a `\newenvironment` command, with a `\newcounter` defined first, if necessary.

The vertical space parameters all define rubber lengths:

`\partopsep`	Extra space added when the environment starts a new paragraph (i.e. there was a blank line above the start of the list environment), and between the last item and the next paragraph (if it's a new paragraph—i.e. there is a blank line before the following text).
`\topsep`	Space between the previous paragraph and the first item, and the last item and the next paragraph (whether the list environment is regarded as being a separate paragraph, or part of the preceding or succeeding text).
`\parskip`	The space between paragraphs; is added to `\topsep` where that is used.
`\itemsep`	The space between items (to which `\parsep` is added).
`\parsep`	The space between paragraphs within items.

The horizontal lengths (fixed lengths) associated with list environments are:

`\leftmargin`	Space between the left margin of the enclosing environment and the left margin of the list. It must be nonnegative. Its value depends on the list level.
`\rightmargin`	Much as for `\leftmargin`, but its value is usually `0pt`
`\listparindent`	Extra indentation at the start of each paragraph, except for those started with `\item`. Can be < 0, but usually is `0pt`.
`\itemindent`	Extra indentation on the first line of each item, usually `0pt`.
`\labelwidth`	Nominal width of the box containing the label.
`\labelsep`	Space between the end of the label box and the text of the first item. Default value of `0.5em`.

I found that `enumerate` and `itemize` have too much white space in `slides` and `seminar` class documents, and defined the following environments that have less vertical space (they work equally well where less space is desirable in other classes of document). Of course the same effect can be achieved by resetting the length of `\itemsep` to 0ex or any other suitably small value within (before the first `\item`) any predefined list environment.

```
\newcounter{clnum}
 \newenvironment{closenum}
{\begin{list}{\arabic{clnum}. }{\usecounter{clnum}%
        \setlength{\topsep}{0ex}%
  \setlength{\itemsep}{0ex}}}{\end{list}}

\newcounter{clnuma}
 \newenvironment{closenuma}
{\begin{list}{(\alph{clnuma})}{\usecounter{clnuma}%
        \setlength{\topsep}{0ex}%
  \setlength{\itemsep}{0ex}}}{\end{list}}

\newcounter{clnumr}
 \newenvironment{closenumr}
{\begin{list}{(\roman{clnumr})}{\usecounter{clnumr}%
        \setlength{\topsep}{0ex}%
  \setlength{\itemsep}{0ex}}}{\end{list}}

\makeatletter
\newenvironment{closebul}{%
  \begin{list}{$\m@th\bullet$}%
  {\setlength{\topsep}{0ex}%
  \setlength{\itemsep}{0ex}%
  \setlength{\parsep}{.1ex}%
  \setlength{\parskip}{.2ex}%
  \setlength{\partopsep}{0ex}}}
{\end{list}}
\makeatother
```

The last of these environments has the most stringent control of space, as most of the length parameters are specified. The other environments can have equivalent lines added to achieve a similar result.

GMS pg 60 ff, shows examples defining a succession of neater description-type environments in which the vertical alignment is made to suit the length of the labels.

Last among the list-type environments is one developed for the STAT 131 class-notes. The requirement was for a named and numbered (with reference capabilities) environment that had ruled lines and indentation to make it clearly separate from the surrounding text.

Adding formatting such as the rules and indentation is not obviously possible for theorem-like environments (that otherwise could give any font and reference-capable numbers), but is possible in a list-type environment, although the TEX commands after `\setcounter{exa}` are necessary to ensure that a `\label` in an `exa` picks up the correct counter value for the `aux` file.

The other issue that must be addressed is the suppression of unsuitable page breaks above or below the two rules—the current version is good, but not perfect in this respect!

```
\makeatletter
\newcounter{exa}[chapter]
\renewcommand{\theexa}{\thechapter.\arabic{exa}}
\newenvironment{exa}{\sffamily\footnotesize
  \begin{list}{}{
  \setlength{\leftmargin}{0.03\linewidth}
  \setlength{\rightmargin}{\leftmargin}}
  \item[]\rule{\linewidth}{.3pt}\nopagebreak%
              \stepcounter{exa}\edef\@currentlabel{\p@exa\theexa}%
              {\em Example \theexa}\quad}
    {\nopagebreak\unskip\par\nopagebreak\rule{\linewidth}{.3pt}\end{list}}
\makeatother
```

Example 8.1 An example of a numbered example environment that is distinguishable from the surrounding text by its indentation, ruled lines and font.

```
\begin{exa}   \label{exa:one}
  An example of a numbered example
  environment that is distinguishable from
  the surrounding text by its indentation,
  ruled lines and font.
\end{exa}
That was Example~\ref{exa:one} (and only).
```

That was Example 8.1 (and only).

8.9 Odds and Ends

8.9.1 Simulating Typed Text

`\verb`, `verbatim` and `{\tt...}` or `\texttt{...}` are quite adequate most times, but if you feel that they're not enough, or need to insert a lot of "verbatim" text with more complex formatting, read GMS pg 66 ff.

`\verb` or `\verb*`, `verbatim` or `verbatim*` cannot appear in the argument of any other LATEX command (such as `\section`, `\caption`, `\footnote`, etc.) (but they can appear in another environment).

- `\verb` (command) is used for a short string of letters, numbers, spaces and special characters (short enough to fit within a line). The command is followed by a numerical or special character which does not appear in the string of text, but does appear at the end to show where the string of text terminates:

  `\verb`*char text char* as in `\verb1$x+y$1` or `\verb+\input+`
  (but `\verb1$x_1 + y_1$1` gives $x_ + y_1 1$ and an error message in the log).

  The entire `\verb` command (to the closing character) must be within a single line of the input file.

- the `verbatim` environment is used for several lines of text:

  ```
  \begin{verbatim}
  Text of space, letters, numbers, special characters and carriage returns
  \end{verbatim}
  ```

This environment was used for most sections of LaTeX code shown in this document.

- `\verb*` and `verbatim*` have the extra feature that spaces are shown as in `$x␣+␣y$` (`\verb*1$x + y$1`).

- `{\tt ...}` and `\texttt{...}`, the declaration and command, respectively, affect the appearance of letters and numbers and characters other than those with special uses (eg `\` and `{ }`), so they are good for short sections of text that contain no characters with a special use. The input and output can flow across several lines (as determined by the text editor, in the case of the former, or LaTeX, in the case of the latter) which can sometimes be an advantage (and sometimes is not).

- The alltt package (include the argument `alltt` in a `\usepackage` command) defines the `alltt` environment which acts like a `verbatim` environment, except that `\ { }` have their usual meaning. This means that other commands and environments can appear inside an `alltt` environment.

```
It's possible to change the     appear-
ance of text.
Or to input a file
This represents the file of commands

of LaTeX commands.

Maths can appear (with some care)
using \(...\) and \[...\]
and commands for special characters,
eg \sp for a superscript and
\sb for a subscript:
\(x\sp2 + y\sb3\)
x^2 + y_3.
```

```
\begin{alltt}
It's possible to change the
{\bf appearance} of text.
Or to input a file
\input{verb}
of \LaTeX\ commands.

Maths can appear (with some care)
using \verb1\(...\)1 and \verb1\[...\]1
and commands for special characters,
eg \verb1\sp1 for a superscript and
\verb1\sb1 for a subsscript:
\verb1\(x\sp2 + y\sb3\)1
\(x\sp2 + y\sb3\).
\end{alltt}
```

- Making a backslash and other special characters. Two options to make a backslash in `{\tt ...}` or `\texttt{...}` or `alltt` environment are:

  - `\texttt{\(\backslash\)}` (or `\texttt{$\backslash$}`) which produces the desired character in a math environment: \.
  - Define a new command, such as `\newcommand{\bsl}{\symbol{'134}}` which works as in: `\texttt{\bsl begin}` which gives `\begin`. This method is used extensively in this document.

  In general, `\symbol{num}` produces a character in the current font with the number *num*. This number can be given as a decimal, octal (preceded by ') or hexadecimal (preceded by ") number, so that `\texttt{\symbol{92}}` gives \, `\texttt{\symbol{'134}}` gives \ and `\texttt{\symbol{"5C}}` gives \ (as 92 is represented as 134 in the octal number system and 5C in the hexadecimal number system). Another special font that was mentioned on page 141 was the ZapfDingbats provided by the pifont package.

  To see the numbers of the fonts you can:

  - Use a LaTeX program along the lines of

```
\documentclass{article}
\usepackage{ifthen,multicol}
\setlength{\textheight}{25cm}
\setlength{\textwidth}{16cm}
\hoffset=-2cm
\voffset=-2cm

\pagestyle{empty}
\linespread{1.2}
\begin{document}
  \newcounter{s}
```

```
% To see the ZapfDingbats substitute:
%\usepackage{ifthen,pifont,multicol}

% First 32 font #s not used in \ding
```

```
  \setcounter{s}{0}                  %  \setcounter{s}{33}
\begin{multicols}{4}[\centering Symbols using texttt]
       %\begin{multicols}{6}[\centering\Large\textbf{Pifont ZapfDingbats}]
         % The limits to the scope of the declarations seems to need
         % adjustment here, so reset (else rest of doc is Large and centered):
       % \normalsize\raggedright

  \whiledo{\value{s}<178}{\thes \texttt{\qquad\symbol{'\thes}}
       % \whiledo{\value{s}<255}{\thes\qquad\ding{\thes}

 \stepcounter{s}}
\end{multicols}
\end{document}
```

Use `\texttt` `\textrm` `\textsf` in the un-commented out version of the above will, in turn, list the characters for each of the font numbers for the three families (they are not the same). To see the ZapfDingbats, substitute all the bits that are commented out in the above (apart from the actual explanatory comments).

– Use `nfssfont.tex` in `/usr/local/share/texmf/tex/latex/base/`. If you give the command

  `latex /usr/local/share/texmf/tex/latex/base/nfssfont.tex`

  the `dvi` and `log` files will be put in the current directory.

  What happens is:

  * You are asked to supply the name of the font (cmr10 for 10-pt Computer Modern Roman, cmtt12 for 12-pt Computer Modern Typewriter, etc). For a full list of installed fonts see `/usr/local/share/texmf/fonts/`.
  * Type `\help` to see all the options. Probably you just want `\table`, which prints the table of font characters with the octal and hexadecimal font numbers in the borders.
  * Type `\init` to change to another font.
  * Type `\bye` to finish.

8.9.2 Marginal Notes

These are discussed in GMS pg 74, and in Lamport pg 59–60 and pg 200 ff.

There is a caveat that they're not handled efficiently, and if there are too many, and too many figures and tables, LaTeX may run out of space.

A marginal note, a kind of float (along with a figure and a table), is made with the command

`\marginpar[`*`lefttext`*`]{`*`righttext`*`}`

where the mandatory argument *`righttext`* gives the text to appear as a marginal note.

This text is typeset in a parbox (and so is never broken across pages), its first line is aligned with the text in the body of the document that surrounded the command. If the `\marginpar` command is placed between paragraphs the note is usually level with the last line of the preceding paragraph (not the first line of the next).

If two-column formatting is in force, the marginal note is put in the nearest margin.

If one-sided printing is in effect, all notes are put in the right-hand margin.

If two-sided printing is in effect, all notes are in the outside margin, and the optional argument *`lefttext`* can be used to specify text to be used if the note is on a left (even?) page or a right (odd?) page.

The declarations `\reversemarginpar` and `\normalmarginpar` can be used in a one-column format page to put marginal notes in the opposite margin from the default one (the former declaration) or in the default one (the latter declaration).

If the margin is narrow and the words in the note are long, the first word will not hyphenate automatically unless it is preceded by `\hspace{0pt}`.

These "problems" of alignment between paragraphs and hyphenation can be overcome by the definition of:

```
\newcommand{\marginlabel}[1]
  {\mbox{}\marginpar{\raggedleft\hspace{0pt}#1}}
```

which also typesets the note ragged left. The `\mbox{}` addresses the alignment problem.

A `\marginpar` command cannot be given inside a `minipage` environment (and probably not in any other parbox).

The position and appearance of marginal notes are controlled by the style parameters:

`\marginparwidth` The width of the parbox containing the note.

`\marginparsep` The horizontal space between the outer margin and a marginal note.

`\marginparpush` The minimum vertical space allowed between two successive marginal notes (the second is pushed down to avoid an overlap).

Final formatting, after all other changes have been made, may be necessary as some marginal notes may be printed below the bottom of the page, or some may overlap.

8.9.3 Fancy headings

The package fancyheadings provides an easy way to customise the headers and footers of a document, and is discussed in GMS pg 96.

Note that graphics inserted in a header or footer (say, the company logo) using `\includegraphics` will be inserted most efficiently using `\newsavebox` and `\sbox`, see Section 6.2.2.5, page 94.

8.9.4 Ornaments

If you like boxed environments, and in particular, those that look after themselves with minimal assistance from you, you may find the packages discussed on GMS pg 277 ff useful. These include

1. boxedminipage package which provides an environment by the same name. The box will enclose the `minipage` and its footnotes, and the box's characteristics can be altered using `\fboxrule` and `\fboxsep`.

   In all other respects it is specified as any `minipage` would be.

2. fancybox, which is used in the `seminar` class of slides. It provides four variants of `\fbox`:
   - `\shadowbox` with line thickness defined by `\fboxrule` and the width of the shadow by `\shadowsize` which has a default of 4pt.
   - `\doublebox` which has inner and outer frames of width `0.75\fboxrule` and `1.5\fboxrule` respectively; the frames are a distance `1.5\fboxrule plus 0.5pt` apart.
   - `\ovalbox` which has width of the frame defined by the `\thinlines` command, and default value of `\cornersize{0.5}`.
   - `\Ovalbox` which is similar to `\ovalbox`, but the thickness of the lines is controlled by the `\thicklines` command.

   Further boxed environments provided by fancybox (see GMS pg 279) are:

   - an `Sbox` environment, that is similar to `\sbox`;
   - boxed equivalents of `center`, `flushleft`, `flushright` (`Bcenter`, `Bflushleft`, `Bflushright` resp.);
   - boxed list environments (`Bitemize`, `Benumerate`, `Bdescription`); and
   - `Beqnarray`, `Beqnarray*` to contain aligned equations in a box; commands for framing a whole page and for framing verbatim texts in a box.

Chapter 9

Version Control, Bibliographies and Indexes

Lastly we look at version control, cross-references, making an index and bibliography, "programming" in LaTeX, and getting information, packages and updates off the Web.

9.1 Version Control

If one core document is to be used with several minor "version" changes, it is possible to define environments containing each of the "versions" and to determine, with two commands, which "version" is typeset.

1. Need the package version (in `\usepackage` command).
2. In the preamble (at least before the beginning of the document) insert:

   `\includeversion{`*`tagname1`*`}`—say *`tagname1`* of `trimi` for stuff for the first trimester of a course repeated in the first and second trimester

   `\excludeversion{`*`tagname2`*`}`—say *`tagname2`* of `trimii` for the stuff for the second trimester

   which is fairly self-explanatory.
3. Enclose all material to be included in *`tagname1`* version in

   `\begin{`*`tagname1`*`} ... \end{`*`tagname1`*`}`,

   and enclose all material to be included in *`tagname2`* version in

   `\begin{`*`tagname2`*`} ... \end{`*`tagname2`*`}`.

9.2 Bibliography

To make a bibliography:

1. Specify any special package(s) used in `\usepackage` (like harvard).
2. Construct a `.bib` file (or `thebibliography` environment) and keep an accessible list of the *`citekey`*s used.
3. Insert `\cite` (or variations) throughout the document (where needed).
4. (a) Insert a `\bibliographystyle` (see Section 9.2.2) and `\bibliography{`*`bibfile-list`*`}` command in the document at the point the bibliography is to appear. If there are two or more files in the list, their names are separated by commas. It is assumed that the files have a `.bib` extension, so it is not necessary to give the extension (`\bibliography{mine}` will find the file `mine.bib`). OR

(b) Insert (or `\input`) the `thebibliography` environment at the point at which the bibliography is to appear. This makes sense for a journal article: if you have a one or more very complete bibliographies, you use the bibliographies when writing and editing the article, but submit a single-file version of the article with `thebibliography` copied from the `.bbl` file.

5. Run LaTeX (to write the results of the `\cites` to `.aux` file(s)); BibTeX (to read these and write a `.bbl` file); LaTeX (to read the `.bbl` file); LaTeX (to resolve all references).

6. Remember that if you change the style, or need to edit the `.bib` file, you need all four runs listed above to get an updated, typeset version. A diagrammatic representation of the process is in Figure 9.1.

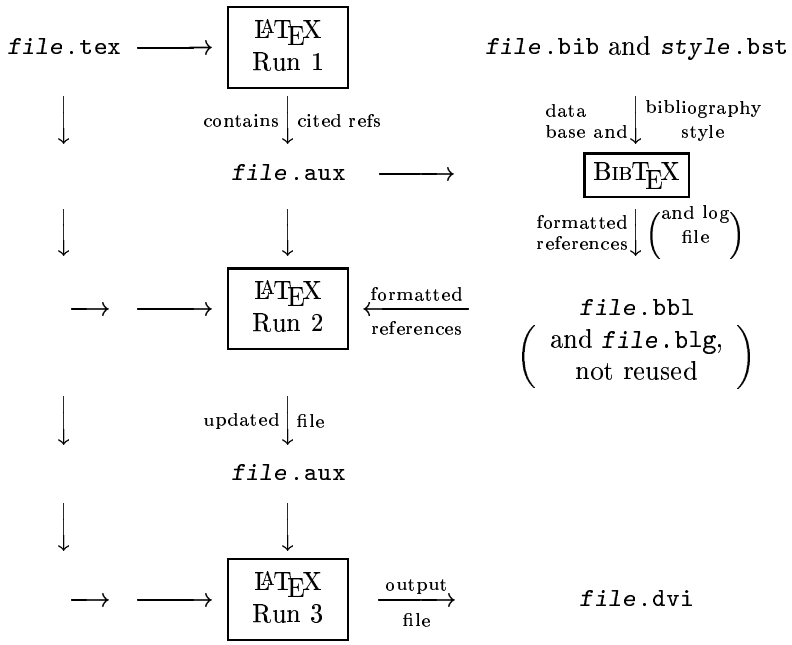


Figure 9.1: Steps to generate a bibliography in a document.

Making and using a bibliography is discussed in Lamport, pg 70 ff, 109 ff, 156–164, and in GMS pg 371 ff, where there are several examples of the effects of the different bibliography styles.

The harvard style is on Stats and OR machines, and on-line documentation is available (see Section 9.8). The style of the bibliography is fairly standard (if that can ever be said about bibliographies) and it offers great flexibility as to how citations are made. Before using it to write a journal article, it is advisable to check whether the journal requires a more standard style of bibliography, or has its own style files that should be used.

9.2.1 The `.bib` file

This file contains the list of all the publications in your bibliography — this can be far more than you quote in any one article.

Note that % is not a comment character in a `.bib` file. For each type of publication (discussed shortly) there are required, optional and ignored "fields" (like "author", "year" etc.) to be filled in. A comment can be given as the argument for any of the ignored fields for that particular publication type (see Table 9.1)—the "comment" text will be visible in the `.bib` file, but will not appear anywhere else.

If you use `emacs` or `bibcard` to generate the file, a lot of the choices are made clear for you, which is very helpful. What sort of choices?

Each publication should be able to be classified as one of: `article, book, booklet, conference, inbook, incollection, inproceedings, manual, mastersthesis, misc, phdthesis,`

`proceedings`, `techreport`, or `unpublished`.

For each type of publication, there are a number of required fields to be filled in. If you don't have this information for one or more of these fields, you may need to classify your publication as something else. In addition, there are a number of optional fields that you may or may not want to complete, and there may be some fields that are irrelevant for that type of publication and are ignored. See Table 9.1. The information on the publication (author, title, date, etc.) is formatted according to the rules in the bibliography style used that would apply for that type of publication.

Each publication must have a unique *key* that is used in the `\cite{`*key*`}` command. Good options for conventions for the *key* are the last name of the first author, plus date if they have several publications—for instance, lamport, lamport:86, lamport:86b, ... Or the first initial(s) plus dates, if necessary—for instance L for Lamport, GMS for three authors, Le for Lehman, ... Or whatever other system makes sense to you. Numbers alone can be confusing if the bibliography style chosen uses numbers to cite publications—this would give rise to two sets of numbers for one set of publications. The *key* can contain letters, numbers and special characters.

Note that the case of letters used in the entry-type specification (`@book`, `@article`, etc.) and field names (`article =`, `title =`, etc.) is ignored, so that BibTeX doesn't distinguish between

```
@book{L,
  author = {Lamport, ...}
.... }
@BOOK{L,
  Author = {Lamport, ...}
.... }
```

However, LaTeX will look for two different bibliography entries for `\cite{L}` and `\cite{l}`.

Some of the entries for the bibliography of this document are:

```
@Book{diller,
  author =       {Diller, Antoni},
  ALTeditor =    {},
  title =        {\LaTeX\ Line by Line},
  publisher =    {John Wiley \& Sons},
  year =         {1999},
  OPTkey =       {},
  OPTvolume =    {},
  OPTnumber =    {},
  OPTseries =    {},
  OPTaddress =   {},
  edition =      {2},
  OPTmonth =     {},
  OPTnote =      {},
  OPTannote =    {}
}

@Manual{graphics2,
  title =        {Using imported graphics in \LaTeXe},
  OPTkey =       {},
  author =       {Reckdahl, Keith},
  OPTorganization = {},
  OPTaddress =   {},
  OPTedition =   {},
  month =        {December},
  year =         {1997},
  OPTnote =      {},
  OPTannote =    {}
}
```

The `.bib` file was made using Emacs, and using the drop-down menu (Entry-Types). This method of

Table 9.1: Fields and publications in a `.bib` file.

	@article	@book	@booklet	@conference	@inbook	@incollection	@inproceedings	@manual	@mastersthesis	@misc	@phdthesis	@proceedings	@techreport	@unpublished
address		O	O	O	O	O	O	O	O		O	O	O	
annote														
author	R	E	O	R	E	R	R	O	R	O	R		R	R
booktitle				R		R	R							
chapter					A	O								
crossref	O	O	O	O	O	O	O	O	O	O	O	O	O	O
edition		O			O	O		O						
editor		E		O	E	O	O					O		
howpublished			O							O				
institution													R	
journal	R													
key	O	O	O	O	O	O	O	O	O	O	O	O	O	O
month	O	O	O	O	O	O	O	O	O	O	O	O	O	O
note	O	O	O	O	O	O	O	O	O	O	O	O	O	R
number	O	V		V	V	V	V					V	O	
organization				O			O	O				O		
pages	O			O	A	O	O							
publisher		R		O	R	R	O					O		
school									R		R			
series		O		O	O	O	O					O		
title	R	R	R	R	R	R	R	R	R	O	R	R	R	R
type					O				O		O		O	
volume	O	V		V	V	V	V					V		
year	R	R	O	R	R	R	R	O	R	O	R	R	R	O

A In `@inbook` only: can include either `chapter` or `pages` field or both. If neither are present a warning is issued, but the output is not affected.

E In `@book` and `@inbook` either author or editor fields can be entered. If both are present, then a warning message is issued and `editor` is ignored. If neither of these fields is present, nor the `key` field, then a warning is issued that BibTeX cannot sort this entry, but it will appear in the bibliography.

O The field is optional, will be used if present, and causes no problems if not present.

R The field is required (mandatory) and, if omitted, will cause an error message to be produced. The formatting of the entry may not be correct.

V Either `volume` or `number` fields can be used for these publications (but not both). If both are used, there is a warning message and `number` is ignored. If just `volume` is present or both are absent, all is well. If `number` is present and the `series` field is absent, there is a warning message but the output is not affected.

An empty cell in the table indicates that the field, if present, will be ignored. (These are the ones to use for comments.)

starting an entry causes all possible entries for each type of publication to be given, so that, for instance, when you specify that a new article or collection in a book is to be used, you get:

```
@Article{,                      @InBook{,
  author =       {},               ALTauthor =    {},
  title =        {},               ALTeditor =    {},
  journal =      {},               title =        {},
  year =         {},               chapter =      {},
  OPTkey =       {},               publisher =    {},
  OPTvolume =    {},               year =         {},
  OPTnumber =    {},               OPTkey =       {},
  OPTpages =     {},               OPTvolume =    {},
  OPTmonth =     {},               OPTnumber =    {},
  OPTnote =      {},               OPTseries =    {},
  OPTannote =    {}                OPTtype =      {},
}                                  OPTaddress =   {},
                                   OPTedition =   {},
                                   OPTmonth =     {},
                                   OPTpages =     {},
                                   OPTnote =      {},
                                   OPTannote =    {}
                                 }
```

The ALT in front of "author" and "editor" in `@InBook` indicates that either one of these two (or both, see Table 9.1) should be supplied. The OPT in front of all the optional fields shows that this is what they are (optional). The ALT and OPT act almost like comments, in that any field starting with these characters is not recognised, and so is ignored. The characters (ALT or OPT, as the case may be) that are not needed (where the field is to be used) can be deleted by hand, or a whole entry (bibliographic reference) can be "cleaned" (on the BibTex-Edit menu, Operating on Current Entry submenu), which operation will remove the ALT and OPTs from those fields for which the argument has been supplied, and delete all lines where the field arguments have not been supplied. The "cleaned" version of the entries given above is:

```
@Book{diller,
  author =       {Diller, Antoni},
  title =        {\LaTeX\ Line by Line},
  publisher =    {John Wiley \& Sons},
  year =         {1999},
  edition =      {2},
}

@Manual{graphics2,
  title =        {Using imported graphics in \LaTeXe},
  author =       {Reckdahl, Keith},
  month =        {December},
  year =         {1997},
}
```

On a different sub-menu is the option to clean a single line of ALT or OPT.

The edit menu in `emacs` on a `.bib` file offers many powerful operations—browse and see!

If we now have the overall structure of a `.bib` file, what exactly do we need to know about the field arguments? BibTeX formats the bibliographic entries automatically—how does the user influence this process when the default version is not appropriate? These issues are addressed next.

9.2.1.1 Author's name

BibTeX will

- Assume in "Name1 Name2 Name3" that Name3 is the last name.
- Assume in "Name1, Name2 Name3" that Name1 is the last name.
- Assume in "Name1 name2 Name3" that Name3 is the last name (e.g. James de la Harpe; James is taken to be the first name, de la to be the "von" and Harpe to be the last name, and the position on the alphabeticised list is determined by "Harpe". But "Name1 {name2 Name3}" has name2 Name3 as the last name, so that {de la Harpe} would be in a position determined by "de la Harpe".
- Expect, if Name1 Name2 is last name, that the name is specified "Name1 Name2, Name3".
- Expect American "Jr"s to be specified as in "Smith, Jr., Adam".
- Recognise special characters if they are in {...}, as in "Kurt Gödel" (`Kurt G{\"{o}}del`).
- Recognise names of multiple authors if they are separated by `and`: "Author1, A. and Author2 Two and Author3, Bee". For a very long string of names, specify the first few, and end the string with `and others`, which appears "et al." in the text.

Non-standard oddities, such as alternate spellings of the author's name, are best given as a `Note = { .... }` (one of the optional fields).

9.2.1.2 Publication Titles

BibTeX, not you, decides if these are sentence case or not (typically, they are for books, and are not for articles). If a word must be capitalised, use { ... }, as in:

```
The complete {M}arkov chain ...
The complete {Markov} chain ...
The {New Zealand} version ...
```

9.2.1.3 Abbreviations

You may make several references to articles appearing in the same journal, or to books with the same publisher, in which case it's useful to define a shorthand version of the name. Some abbreviations are provided (e.g. months: jan, feb, etc.).

- Put an `@string` command in the `.bib` file (at the top is best), e.g.

  ```
  @string{JNZS = "Journal of New Zealand Statistics"}
  ```

  then in any `.bib` entry, this journal can be specified either as `Journal = JNZS` or as `journal = "Journal of New Zealand Statistics"`, but not `journal = "NZJS"`.

- If you have many abbreviations, or many bibliographies that reference the same journals, you can create a `.bib` file of abbreviations that is specified first in your list of bibliographies as in:

  ```
  \bibliography{abbrev,other}
  ```

- Two strings, or a string and other text can be concatenated, using #, as in:

  ```
  @string{IEV= "IEEE Proceedings on Vehicular Technology"}

  journal = IEV # 1986
  ```

- Abbreviations must start with a letter and may not contain a space, nor any of the characters: `" # ' ( ) , = { }`

9.2.2 Bibliographic Styles

This is specified in the `\bibliographystyle{`*style*`}` command. The standard LaTeX styles are:

`plain`	Numeric labels, list sorted alphabetically.
`unsrt`	As above, but printed in order of citation.
`alpha`	As `plain`, but the labels used are the author's name and year of publication.
`abbrv`	As `plain`, but first names, month and journal names are abbreviated (automatically)
`acm`	For journals of the Association of Computing Machinery; the author names are given in small caps, numbers are used as labels.
`apalike`	For journals of the American Psychology Association. Need to specify the apalike package. The bibliography entries are listed alphabetically, last name first, with hanging indent and no label.

There are many more styles available; some of those installed on our system are:

`natbib, abbrvnat, amsalpha, amsplain, asa, astin, ieeetr, siam, esub2acm, IEEE, jasa, nucenamed, plainnat, unsrtnat, prsty` and the `harvard` styles: `agsm, apsr, dcu, jmr, jphysicsB, kluwer, nederlands`.

Differences can be subtle, but the style is easy to change:

- change the package (in `\usepackage`) if necessary;
- change the *`style`* in the `\bibliographystyle` command.
- LaTeX, BibTeX, LaTeX, LaTeX.

Then you can see the effects and decide which you prefer. Of course, the earlier you do this, so the shorter the document, the quicker the repeated test will go . . . .

9.2.3 Packages

There are several packages to satisfy particular needs over and above the style of the bibliography. For example, chapterbib (GMS pg 385) will produce multiple bibliographies in one document (say one for each chapter), which is on our system, or bibunits (GMS pg 386) which does more or less the same (but is not on our system).

9.2.4 "Hacking"—low and dirty

GMS pg 407 ff discusses technical details of style files, and possible modifications to them.

9.2.5 Citing

A citation is "made" in the text by inserting

`\cite[`*`text`*`]{`*`key`*`}`

where *`key`* corresponds to an entry in the `.bib` file (or `thebibliography`).

The format of the citation will depend on the bibliography style in use, but *`text`* will be an additional note (say, pages 3–10) to be inserted in the citation (which may then look something like: [1, pages 3–10]).

9.2.5.1 Invisible citation

`\nocite{`*`key`*`}`

will cause the reference corresponding to *`key`* to be included in the list of references, although it is never cited in the document.

`\nocite{*}`

causes all entries in the BibTeX data base to be included, whether they are cited or not.

9.2.6 Better mousetrap—harvard family

Need to use the package harvard (in `\usepackage`).

The references appear, sorted by last name, with a hanging indent. The date appears in brackets, following the name. Citations are by last name and date (not numbers).

There are six bibliography styles (at least) available: `agsm`; `dcu` which is based on the conventions in use in the Design Computing Unit, Dept. of Architectural and Design Science, University of Sydney; `jmr` for the Journal of Management Research; `jphysicsB` for the Journal of Physics B; `kluwer` for Kluwer Academic Publishers; and `nederlands` which conforms to Dutch conventions.

There are two citation styles invoked by `\citationstyle{agsm}` and `\citationstyle{dcu}` respectively, with `agsm` the default. Any `\citationstyle` command should appear before the first citation.

In `agsm` style, `\cite` works as before, and causes a (Lastname date) to appear in the text. `\cite[`*text*`]{`*key*`}` results in a (Lastname date, *text*) in the text.

In addition to `\cite`, the following options are also provided for citation:

Citation	Result
`\citeasnoun{`*key*`}`	Lastname (date)
`\citeasnoun[`*text*`]{`*key*`}`	Lastname (date, *text*)
`\possessivecite{`*key*`}`	Lastname's (date)
`\citeaffixed{`*key1,key2*`}{`*text*`}`	(*text* Name1 date1, Name2 date2)
`\citeaffixed[`*text1*`]{`*key*`}{`*text2*`}`	(*text2* Name date, *text1*)
`\citename{`*key*`}`	Lastname
`\citeyear{`*key*`}`	(year)
`\citeyear*{`*key*`}`	year

The style `dcu` differs in the punctuation it inserts. There will be a , between name and date, and a ; between two citations, as in (Name1, date1; Name2, date2).

Note that where there are multiple authors, the first reference causes a full listing (of last names); subsequent references will result in "Lastname1, et al." This is the `\citationmode{default}`.

`\citationmode{full}` causes all citations to be full.
`\citationmode{abbr}` causes all citations to be abbreviated (using "et al.").

The same effect is achieved using `full, abbr` or `default` options in the `\usepackage{harvard}` command (although `default` need not be specified).

This behaviour can be overridden for all individual citations involving a name:

`\cite*{`*key*`}` (or `\citeasnoun*` or ...) will force a full list of authors;
`\cite**{`*key*`}` (or `\citeasnoun**` or ...) will force an abbreviated list of authors.

There are further options to control the shape of the brackets used, and the conjunction (either "and" or "&") between the last pair of names.

It is also possible to do it "by hand" using `thebibliography`, where each item must be specified as a `\harvarditem` (most sensibly this would be in the final edit of a journal article where it is undesirable to submit the long `.bib` file(s) used when writing the article).

9.3 An Index

To make an index:

1. Use the package makeidx.
2. Put `\makeindex` in the preamble of the document.
3. Put `\printindex` at the point where the index is to appear (usually, but not necessarily, at the end of the document). If the index is to be called something other than "Index", redefine `\indexname` (see page 139).
4. Add `\index{...}` commands (or user-made variations) throughout the text to identify index entries and specify their formatting.
5. Run LaTeX (to generate the `.idx` file), *MakeIndex* (to create the alphabetised `.idx` and log `.ilg` files—in `xterm` type `makeindex` *`filename`*`.idx`[1] or use `Index` on the `Command` menu in `Emacs`), LaTeX (to typeset the document, including the index file).

[1] The `.idx` extension is usually not necessary. However, in the event that there is a file or subdirectory with a name exactly *`filename`* (no extension), the index will not be made unless the extension is specified, as the program uses the wrong file when trying to make the index.

A diagrammatic representation of the process is in Figure 9.2.

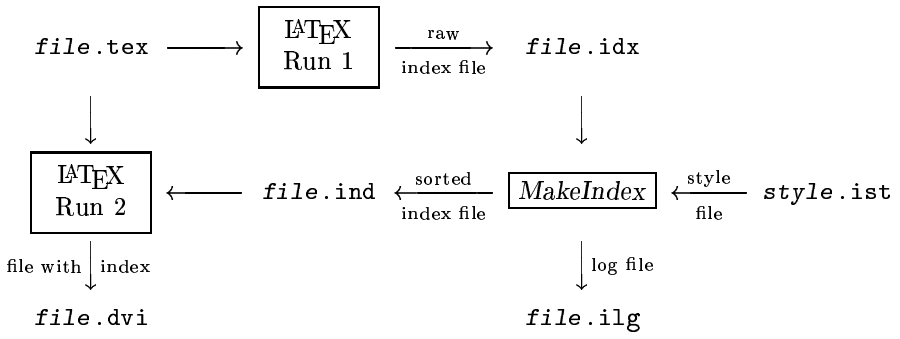


Figure 9.2: Steps to generate an index in a document.

9.3.1 Considerations for index entries

1. Leading and following spaces are ignored in the printed (final) version, but are used when sorting—so avoid them. This means that `\index{ space}` will come above (and separate from) `\index{space}`. Double spaces and new lines are also taken into account by *MakeIndex* when sorting (but not printing). For example, the following two entries will result in two separate entries in the index:

```
\index{sample
   mean}
\index{sample mean}
```

   The case of the entry also affects the sort order, so that `\index{Mean}` appears before `\index{mean}`. It is easiest to use lower case letters for all entries other than proper nouns.

   The following should give you an idea of how to format index entries, and the appearance of the entries generated, although I'm not absolutely sure of the order of the formatted entries:

Index entry in document	Page(s) of entry
`\index{mean}`	20, 5, 45
`\index{mean!sample}`	19, 5
`\index{mean!sample@\xb}`	19, 20
`\index{mean@$\mu$}`	5, 20
`\index{mean!sample@\Xb}`	20, 45
`\index{mean@\textbf{mean}}`	19
`\index{mean@\texttt{mean}}`	46
`\index{mean@$\mu$!sample@\xb}`	40
`\index{sample mean|see{mean, sample}`	19

   The `!` indicates a sub-entry, and the `@` indicates that the entry is sorted as if it consisted of what appears before the `@`, but what follows the `@` is what appears in the index.

Entry in Index	Comments
mean, 5, 20, 45	The basic entry.
$\overline{X}$, 20, 45	Sub-entry "sample", to root entry "mean".
sample, 5, 19	Sub-entry "sample", to root entry "mean".
$\overline{x}$, 19, 20	Sub-entry "sample", to root entry "mean".
μ, 5, 20	μ would be sorted as if it is "mean".
$\overline{x}$, 40	This entry is here, not under "sample".
mean, 19	Note that entries are sorted by the
`mean`, 46	representation of the entry, so appear as separate entries.
sample mean, *see* mean, sample	Cross-reference

The text inserted in the cross-reference (as in the last entry above) can be changed by redefining the `\seename` command—see page 139.

2. It's also possible to format the page numbers—say bold for a primary reference, or italics for figures—and to reference a range of pages. The `|` (used in `|see`) will encapsulate the page number, as is shown below:

Index entry in document	Action	Entry in Index
`\index{mean|textbf}`	Gives a bold page number.	mean, **20**
`\index{mean|(textit}`	Define the start of an italicised page range.	
`\index{mean|)}`	Define the end of an italicised page range.	mean, *20–44*

Note that the encapsulating process does not require a `\` at the start of the command name.

3. Special characters, for *MakeIndex*, are `! " @ |`. To print one of these characters, precede it by `"` (Shift + '). In the examples below, assume that all index entries were made on page 35.

Index entry in document	Action	Entry in Index
`\index{bar@\texttt{"|}}`	`|` in place of "bar", but sorted as if "bar".	`|`, 35
`\index{quote ("")}`	Sorted by "quote"	quote ("), 35
`\index{at@\texttt{"@} sign}`	`@` in place of "at", but sorted as if "at".	`@` sign, 35

4. When sorting the page numbers (within an index entry) it's assumed that the document page numbers will be in order:
   1. `roman`
   2. `arabic`
   3. `alph` or `Alph`.

   See GMS page 359 to change this order.

5. Entries are sorted in the order:
   1. symbol
   2. number
   3. alphabetic.

9.3.2 Bells, whistles and consistency

1. The showidx package will print all the `.idx` entries as margin notes—this will make it easier to compare what you did on different pages of the document, and how the entries compare to the final index.

2. User commands that drop the original word and an index entry into the text are easy to define, and will ensure consistency. For instance, for one-word entries (other than when the the word is capitalised at the beginning of a sentence, which will cause double-entry problems in the index):

```
\newcommand{\Index}[1]{#1\index{#1}}
```

   For formatted entries, which are to be sorted in their unformatted version, something like:

```
\newcommand{\Indextt}[1]{\texttt{#1}\index{#1@\texttt{#1}}}
```

   and to do the same for LaTeX commands, with the backslash inserted automatically,

```
\newcommand{\bs}{\symbol{'134}}    % print backslash
\newcommand{\Com}[1]{\texttt{\bs#1}\index{#1@\texttt{\bs#1}}}
```

For automatic double entries (for instance, all the LaTeX environments are listed both under their name and under environment in the index of this document):

```
\newcommand{\Eindex}[1]{\texttt{#1}\index{#1@\texttt{#1} environment}%
\index{environment!#1@\texttt{#1}}}
```

For further, deeper and more technical discussion on generating an index, see GMS page 357.

9.4 "Programming"

There are two packages, calc and ifthen, that provide the commands for some simple programming.

`calc` (GMS pg 468) re-implements the commands `\setcounter`, `\addtocounter`, `\setlength` and `\addtolength` so that they can accept integer and length expressions, rather than simple numbers and lengths. The expressions can include binary operators (`+ - * /`).

This means that calculation on length parameters and counters can be expressed fairly succinctly.

`ifthen` (Lamport pg 195, GMS pg 470) allows logical tests and different outcomes depending on the result, including a "do while"-type command (see page 147 of these notes).

9.5 `latex2html`

This is an application that converts a LaTeX document to a set of (usually) fairly reasonable HTML documents, for display on the Web. To effect the translation, give the command (in `xterm` window):

`latex2html` *`filename`*`.tex` *`filename`*`.html`

where the filenames can be different, if you wish. The HTML documents are structured, with a table of contents giving access to the sections and/or subsections which are in separate documents, and many connectors forwards and backwards through the whole document.

This is a powerful (and slow) translator and it is worth reading the man pages (`man latex2html`↩ in an xterm window). Features that can cause problems (or unhappiness with the result) are:

- Each section, by default, is a separate document; the sectioning level at which sub-documents are made needs to be reset to a level that will generate moderate-sized documents.
- Complex displayed formulae are made into small graphics files. The text in the graphics does not necessarily match the surrounding text (the font size may be bigger) and the document is slow to load.
- Does not recognise some environments (eg `tabbing`).
- Prefers a diet of vanilla LaTeX.

An alternate translator is `tth` (see `http://hutchinson.belmont.ma.us/tth/`) which has the features:

- much faster than `latex2html`;
- makes one big (can be **very** big) document, certainly the earlier version did;
- builds formulae using tables rather than using images;
- prefers a diet of vanilla LaTeX.

9.6 `detex` and Counting Words

This will remove commands, extraneous braces, and environments, converting a LaTeX document to a text document. It works fairly well on a mainly text document, but can leave a mystifying residue as it leaves options (in [...]) as part of the text.

Give the command (in `xterm` window):

`detex -l` *`filename`*`.tex > ` *`filename`*

where the filenames can be different, if you wish.

An approximate word count can be obtained by

`detex` *`filename`* `| wc`

9.7 Ctan Website

A comprehensive list of ctan sites is given at

ftp://ftp.cdrom.com/pub/tex/ctan/CTAN.sites

with the nearest being

ftp://ctan.unsw.edu.au/tex-archive

A centralised list of LaTeX–and TeX–related websites is at

http://www.yahoo.com/Computers_and_Internet/Desktop_Publishing/TeX/

Happy hunting.

9.8 Local documentation

Documentation of many of the packages and main installations is in

/usr/local/share/texmf/doc/

in the form of .tex files and .dvi files.
There is further documentation available from the nearest ctan site:

ftp://ctan.unsw.edu.au/tex-archive/info/

Bibliography

[1] American Mathematical Society. *$\mathcal{AMS}$-LaTeX Version 1.2 User's Guide*, October 1994.

[2] D.P. Carlisle. *Packages in the 'graphics' bundle*, January 1999.

[3] Antoni Diller. *LaTeX Line by Line.* John Wiley & Sons, 2 edition, 1999.

[4] Michel Goossens, Frank Mittelback, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley, 1994.

[5] Leslie Lamport. *LaTeX: a document preparation system.* Addison-Wesley, 2 edition, 1985.

[6] Keith Reckdahl. *Using imported graphics in LaTeX 2$_\varepsilon$*, December 1997.

[7] Timothy Van Zandt. *`seminar.sty` a LaTeX style for slides and notes: User's guide*, April 1993.

[8] Peter Williams and Thorsten Schnier. *The Harvard Family of Bibliography Styles*, June 1994.

Index