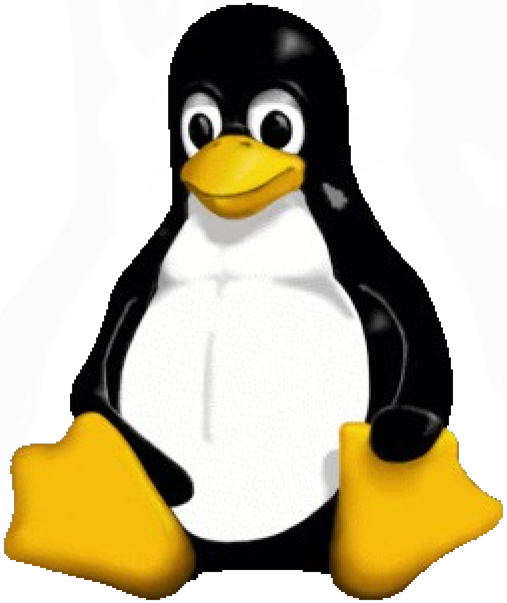
Introduction to Linux

(2011 Course)



Tom Latham
Ben Morgan

THE UNIVERSITY OF WARWICK

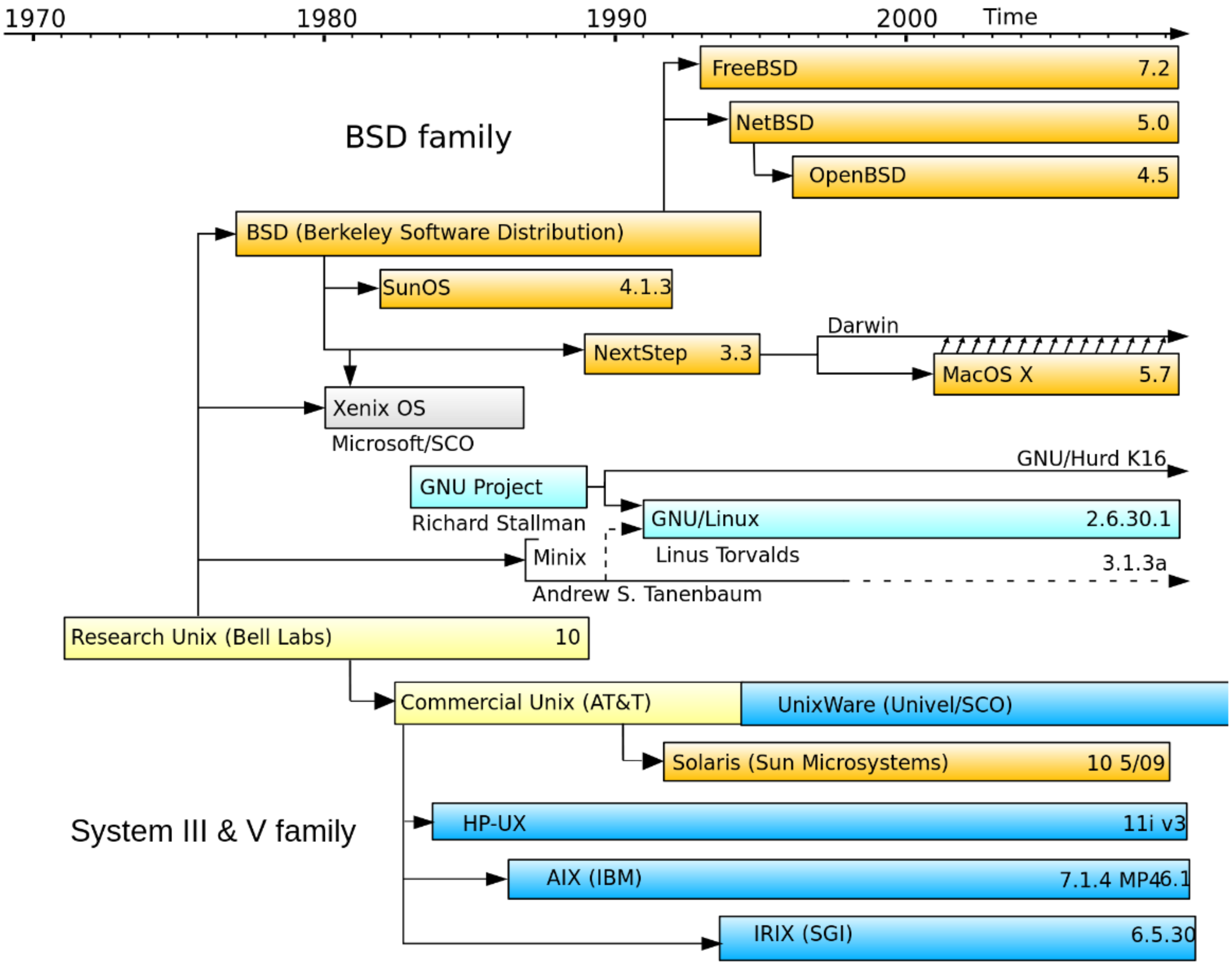
Introduction

- These slides are designed to introduce you to UNIX based operating systems, in particular Linux
- We want to give you enough info here to allow you to concentrate on the C++ in the hands-on sessions rather than worrying about how to copy files or change directory
- There are exercises provided throughout the booklet and there are several more detailed ones at the end that you should go through ***before*** the first hands-on session
- There will be a couple of hours at the first hand-on session to go through anything from these slides that isn't clear and to do a few extra exercises
- We'll start with basic usage including an introduction to the shell and how to navigate the file system
- Later on, we'll see how to view and edit files
- We've also provided extra material in further booklets on the web that will go through various very useful commands, more advanced shell usage and finally shell scripting – please dip into these when you can
- It's important to gain a good working knowledge so that you can become more productive

What is Linux?

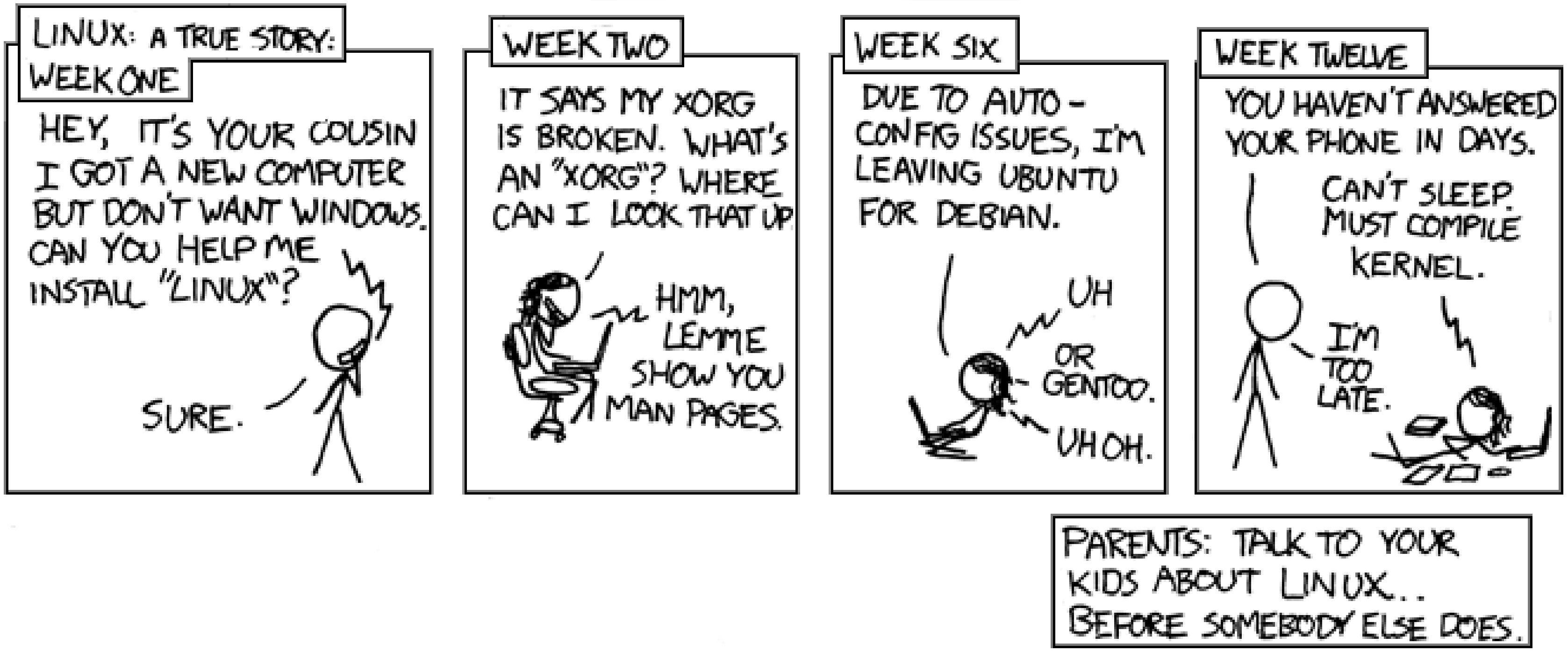
- From **http://en.wikipedia.org/wiki/Linux**
  - *'Linux (also known as GNU/Linux is a Unix-like computer operating system. It is one of the most prominent examples of open source development and free software; unlike proprietary operating systems such as Microsoft Windows or Mac OS X, its underlying source code is available for anyone to use, modify, and redistribute freely.'*
  - *'Initially, Linux was primarily developed and used by individual enthusiasts on personal computers. Since then, Linux has gained the support of major corporations such as IBM, Sun Microsystems, Hewlett-Packard, and Novell, Inc. for use in servers and is gaining popularity in the personal computer market. It is used in systems ranging from supercomputers to mobile phones.'*
- *Large use within scientific community.*
- *As Linux is Unix-like, you'll be at home on other systems such as Solaris, BSD/OS X and the various Unices.*
- *Be mindful of the subtle differences though!*

The Evolution of Linux



Linux Distributions

- Linux typically comes as a 'distribution', consisting of:
  - The Linux kernel
  - GNU software (e.g. utilities, editors, compilers)
  - Desktop/Window manager (i.e. GUI)
  - Software management system (rpm, deb etc.)
- Currently ~300 different (active) distributions!


LINUX: A TRUE STORY:
WEEK ONE
HEY, IT'S YOUR COUSIN I GOT A NEW COMPUTER BUT DON'T WANT WINDOWS. CAN YOU HELP ME INSTALL "LINUX"?
SURE.
WEEK TWO
IT SAYS MY XORG IS BROKEN. WHAT'S AN "XORG"? WHERE CAN I LOOK THAT UP
HMM, LEMME SHOW YOU MAN PAGES.
WEEK SIX
DUE TO AUTO-CONFIG ISSUES, I'M LEAVING UBUNTU FOR DEBIAN.
UH
OR GENTOO.
UH OH.
WEEK TWELVE
YOU HAVEN'T ANSWERED YOUR PHONE IN DAYS.
CAN'T SLEEP. MUST COMPILE KERNEL.
I'M TOO LATE.
PARENTS: TALK TO YOUR KIDS ABOUT LINUX.. BEFORE SOMEBODY ELSE DOES.

Cartoon courtesy of xkcd.com

How to Choose?

- In HEP you're most likely to end up using
  - Scientific Linux
  - RedHat Enterprise Linux
  - Fedora
- Differences between distros that could affect you
  - Choice of software
  - Arrangement of system files
- However, this course is distribution neutral so it can be applied to (almost) any Linux system.
- **Further information:**
  - **http://en.wikipedia.org/wiki/Linux_distribution**
  - **http://distrowatch.com**

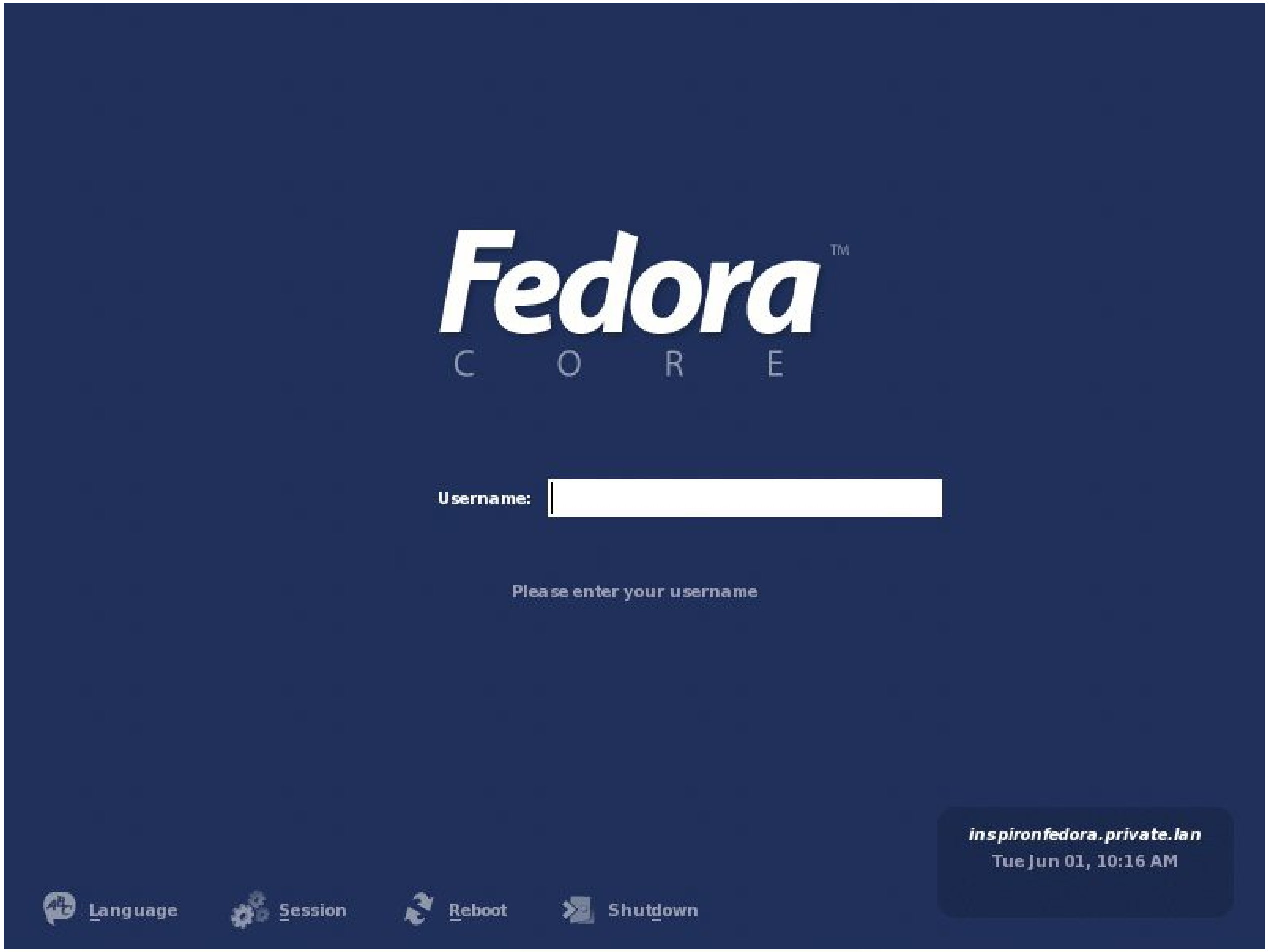
User Accounts

- Linux is a multi-user system
  - You need a user account to access workstations and other remote systems.
  - Hopefully you should all have an account on your university systems.

- Security of your login is **VERY, VERY IMPORTANT!**
  - Passwords should be at least 8 characters long, mixture of letters (upper and lower case), numbers and symbols, **NO DICTIONARY WORDS!**
  - **Never, ever share your login with anyone.**
  - If you leave your workstation unattended, lock the screen so no one else can access your session.

Graphical Login

- Using your account details, you can now login...
- Generally, this is through a graphical screen, e.g.



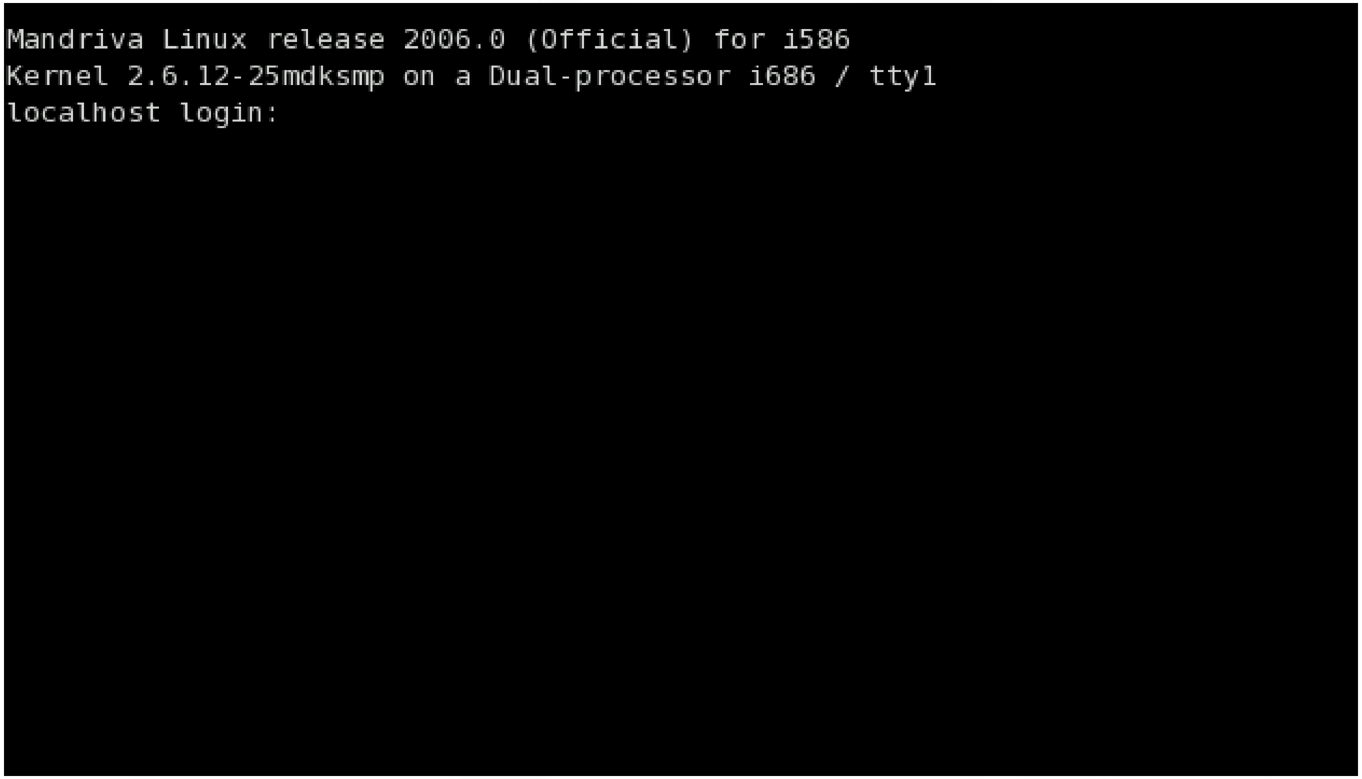
Username:

Please enter your username

inspironfedora.private.lan
Tue Jun 01, 10:16 AM

Terminal Login

- There are also terminal logins, accessed via `Ctrl+Alt+F1`, `Ctrl+Alt+F2` etc.



```
Mandriva Linux release 2006.0 (Official) for i586
Kernel 2.6.12-25mdksmp on a Dual-processor i686 / tty1
localhost login:
```

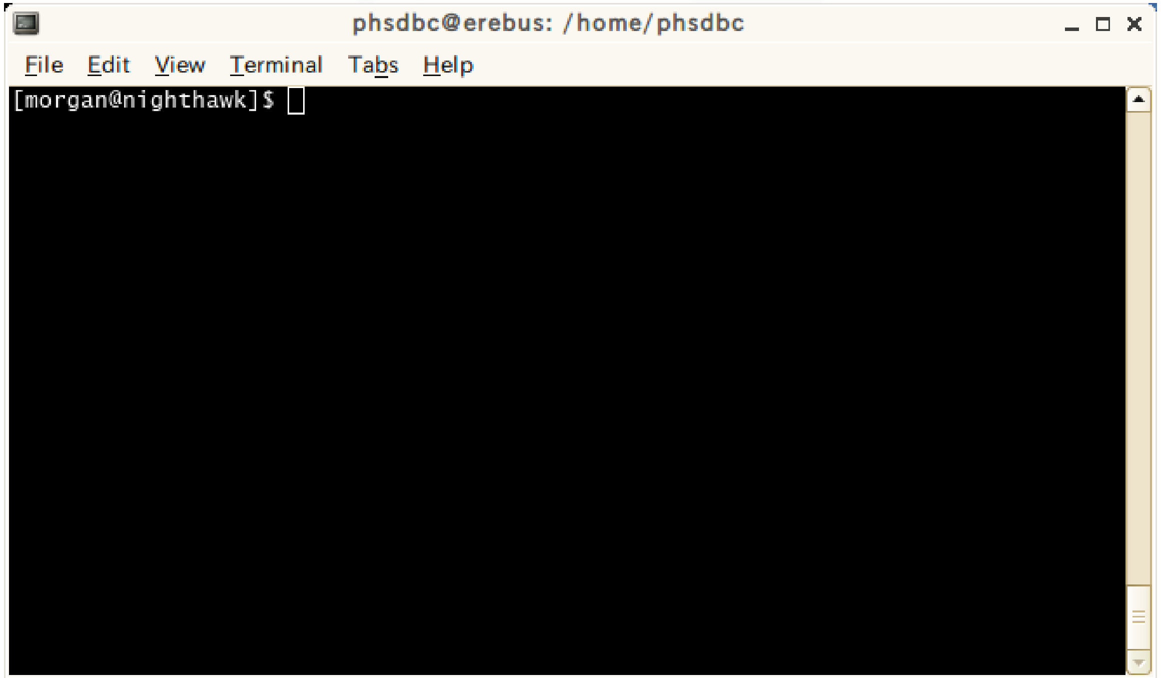
- Useful for quick logins, and may be the only login available on old systems(!).
- To get back to graphical login, do `Ctrl+Alt+F7`

The Linux Desktop GUI

- Login through the graphical interface, the GUI starts up...
- Most Linux distributions generally provide two main GUIs, KDE and GNOME, selectable from the login screen
    - Very much a personal choice.
    - Try both and see which you prefer.
    - Some systems may only have one or the other installed
- Both present a Windows-like desktop, so transition should be fairly easy.
- Common applications are available through the menus or panel buttons, e.g.
    - Web browsers (Mozilla, Firefox etc.).
    - GUI email readers (Kmail, Thunderbird etc.).
    - PDF and PostScript viewers (Acroread, gv etc.).

The Terminal

- Whilst the GUI is useful, the terminal is where the real power lies and where you'll do most work.
- It looks (something) like this:



```
[morgan@nighthawk]$
```

- Terminals can be opened from a button on the panel or a menu item – have a look round...
- NB, it is essential to get used to using the terminal for ALL tasks including file management

What's a terminal good for?

- When started, a terminal window contains a process called a 'shell'
  - Essentially a program that knows how to find other programs and run them.
- Programs can be started from the terminal by typing in a command

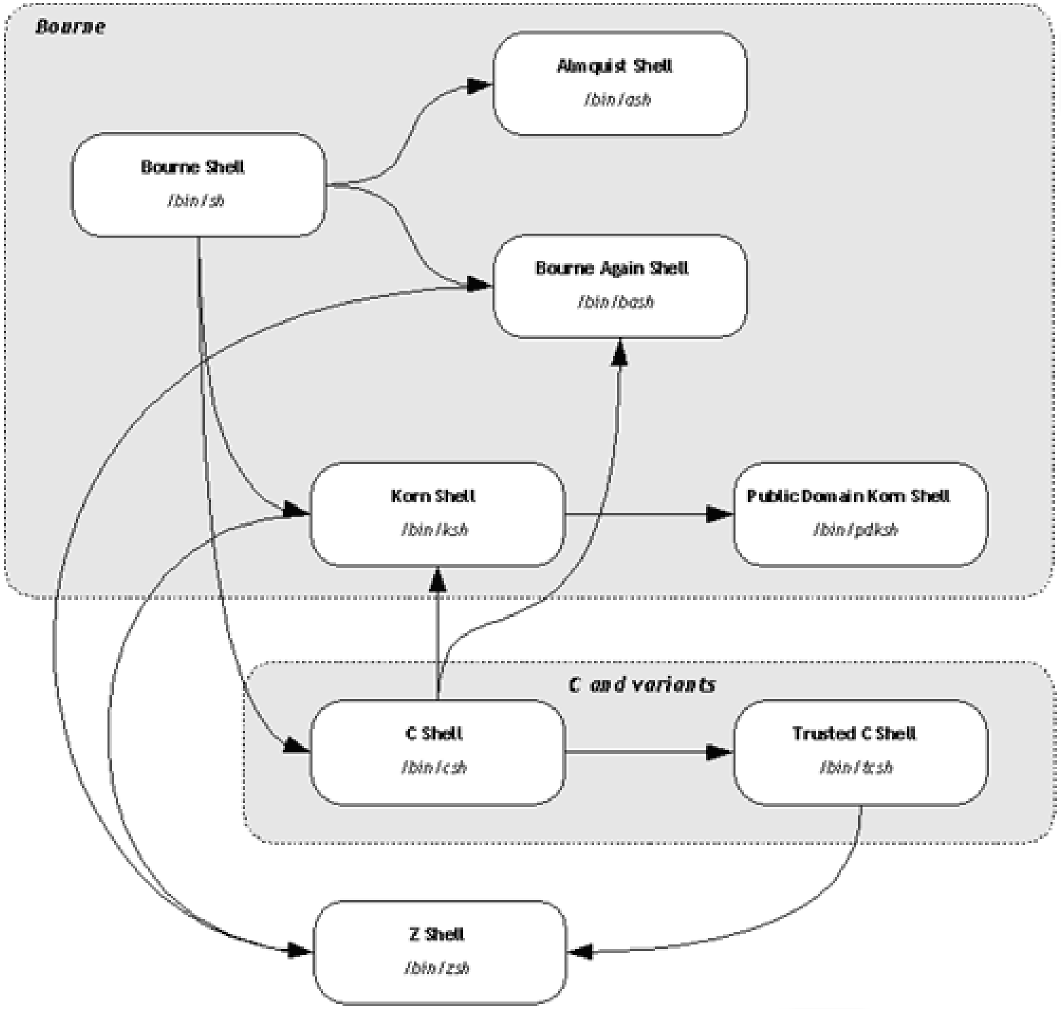
```
progname options arguments
```

- For instance, open a terminal and try

```
ls -la ~/.mozilla
```

- which lists the contents of your browser config directory.
- You'll learn about `ls` and its options later.

Shell Flavours



- Several different shells available.
- `bash` and `tcsh` are the most common of the two main families.
- Unfortunately, the different families have different syntax and behaviour for certain operations.
- Will try to be as shell-neutral as possible here but will point out where there are differences.

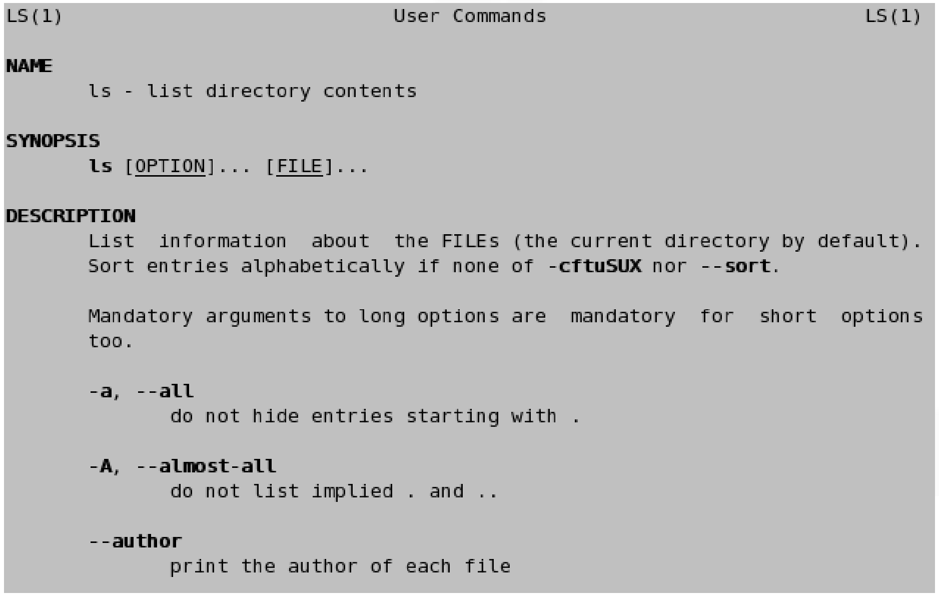
- ***Shell choice is personal – so experiment – and argue about the relative merits of each with other users!***

Manual Pages

- To obtain more information on a command you can view its manual page
- These are accessed using the `man` command
- The following command will display the manual page for the `ls` command that we've just met

```
man ls
```

- The following shows a small portion of the page:



```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a, --all
              do not hide entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              print the author of each file
```

- It's vitally important to get used to reading man pages.
- They will be one of your main sources of help and information.

Finding Commands

- Often you'll have need of a command to do a specific task, but you don't know the exact command name.
- You can use the apropos command to search the man page names and descriptions.
- For example, we want a command to list the contents of a directory:

```
apropos “list directory”
```

- this will return a list of (possibly) relevant commands and their descriptions.
- Exercise: a common task in HEP is to connect to a remote machine at CERN or SLAC etc. – use apropos to find potential commands to do this....
- Also Google is very useful if apropos comes up short

The Filesystem

- The Linux filesystem (fs) is arranged rather differently from that of Windows
- There are no drive letters (C: etc) but instead everything is “mounted” under a single “root” directory – /
- Instead of the “My Documents” folder you have a “home” directory, which will be the working directory when you open a terminal
- The main parts of a typical Linux fs include:
    - /home – where users' home directories can be found
    - /usr – where most programs are installed
    - /etc – where the system configuration files are
- Much more information at:

  http://tldp.org/LDP/Linux-Filesystem-Hierarchy/html/index.html

Navigating the Filesystem

- To find where you are within the fs you can use the command `pwd`
- To change directory you use the command `cd`
  - To navigate to your home directory you can do:

    ```
    cd   or   cd ~/
    ```

  - Target directory is within current directory:

    ```
    cd target   or   cd ./target
    ```

  - Target directory is parent of current directory:

    ```
    cd ..
    ```

  - Target directory is at arbitrary location in fs:

    ```
    cd /path/to/target
    ```

  - Target directory is another user's home directory:

    ```
    cd ~username/
    ```

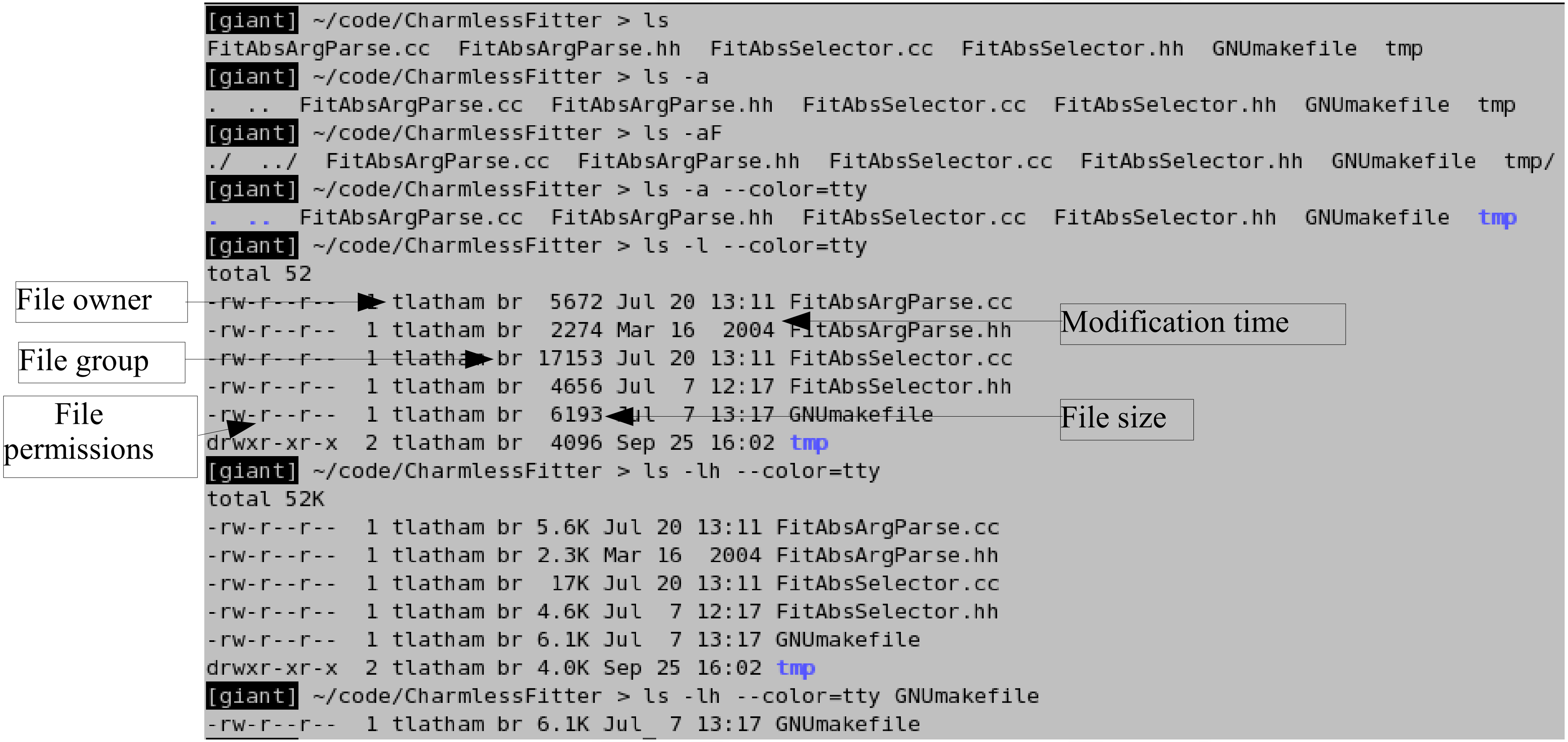
  - Target directory is the previous working directory:

    ```
    cd -
    ```

Listing Directories/Files

- To find out the contents of a directory or to get information on a particular file you use `ls`

- There are many options for this command, some of which are illustrated below:



```
[giant] ~/code/CharmlessFitter > ls
FitAbsArgParse.cc  FitAbsArgParse.hh  FitAbsSelector.cc  FitAbsSelector.hh  GNUmakefile  tmp
[giant] ~/code/CharmlessFitter > ls -a
.  ..  FitAbsArgParse.cc  FitAbsArgParse.hh  FitAbsSelector.cc  FitAbsSelector.hh  GNUmakefile  tmp
[giant] ~/code/CharmlessFitter > ls -aF
./  ../  FitAbsArgParse.cc  FitAbsArgParse.hh  FitAbsSelector.cc  FitAbsSelector.hh  GNUmakefile  tmp/
[giant] ~/code/CharmlessFitter > ls -a --color=tty
.  ..  FitAbsArgParse.cc  FitAbsArgParse.hh  FitAbsSelector.cc  FitAbsSelector.hh  GNUmakefile  tmp
[giant] ~/code/CharmlessFitter > ls -l --color=tty
total 52
-rw-r--r--   1 tlatham br   5672 Jul 20 13:11 FitAbsArgParse.cc
-rw-r--r--   1 tlatham br   2274 Mar 16  2004 FitAbsArgParse.hh
-rw-r--r--   1 tlatham br  17153 Jul 20 13:11 FitAbsSelector.cc
-rw-r--r--   1 tlatham br   4656 Jul  7 12:17 FitAbsSelector.hh
-rw-r--r--   1 tlatham br   6193 Jul  7 13:17 GNUmakefile
drwxr-xr-x   2 tlatham br   4096 Sep 25 16:02 tmp
[giant] ~/code/CharmlessFitter > ls -lh --color=tty
total 52K
-rw-r--r--   1 tlatham br 5.6K Jul 20 13:11 FitAbsArgParse.cc
-rw-r--r--   1 tlatham br 2.3K Mar 16  2004 FitAbsArgParse.hh
-rw-r--r--   1 tlatham br  17K Jul 20 13:11 FitAbsSelector.cc
-rw-r--r--   1 tlatham br 4.6K Jul  7 12:17 FitAbsSelector.hh
-rw-r--r--   1 tlatham br 6.1K Jul  7 13:17 GNUmakefile
drwxr-xr-x   2 tlatham br 4.0K Sep 25 16:02 tmp
[giant] ~/code/CharmlessFitter > ls -lh --color=tty GNUmakefile
-rw-r--r--   1 tlatham br 6.1K Jul  7 13:17 GNUmakefile
```

Finding Directories/Files

- To locate files or directories within a given part of the file system you can use the `find` program
- This is actually a very powerful program but we'll just look at the most basic options here
- To find a file with a particular string in its name you can do:

```
find basedir -name '*string*'
```

  - where basedir is the directory within which you want to recursively search, e.g. use . for the current dir

- You can also use the `locate` command to search for files:

```
locate string
```

- This command uses a database that is *usually* updated every night on most Linux systems

Directory/File Manipulation

- To create a directory:

```
mkdir mynewdir
```

- To remove an empty directory:

```
rmdir myolddir
```

- To delete a file.

```
rm myoldfile
```

- To move a file:

```
mv myoldfile mynewfile
```

- To copy a file:

```
cp myfile1 myfile2
```

- Recursively delete a directory and its contents:

```
rm -r myolddir
```

Exercise:

Look up some of the options these commands have in their manual pages.

WARNING:

rm is **exceptionally** powerful, files are deleted **permanently**.

Network File System

- NFS allows a computer to access files over a network as easily as if they were on its local disks.
- An NFS server holds the actual disks and exports them over the network
- The clients can mount the exports into their filesystem
- Users do not need to know the files are not local
- File permissions are determined by user ID
- Therefore user ID's must be the same on the NFS server and the clients
- Server decides which client machines are allowed to connect

Andrew File System

- AFS is another distributed file system
- Some advantages over NFS in terms of security and scalability
- Authentication to an AFS “cell” (e.g. cern.ch) is done by password using the `klog` command, which acquires a “token”
- So no restriction on which machines are permitted to connect
- However, AFS is not part of the standard filesystem tools in Linux so need to have client software and kernel modules installed
- Some labs place users' home directories in AFS, e.g. CERN
- AFS is usually mounted in /afs on most systems
- e.g. to access SLAC cell go to

```
/afs/slac.stanford.edu
```

wget

- Command line program to download items from the web
- Supports http, https and ftp protocols
- Simplest usage to retrieve a single file:

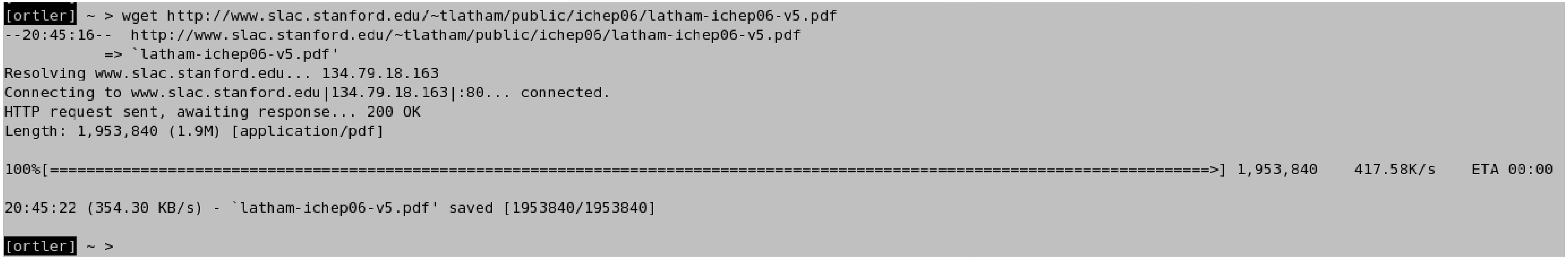
```
wget URL
```

- To give the file a different local name do:

```
wget URL -O local_name
```

- If you have a text file with a list of URLs you want to download you do:

```
wget -i file
```



```
[ortler] ~ > wget http://www.slac.stanford.edu/~tlatham/public/ichep06/latham-ichep06-v5.pdf
--20:45:16--  http://www.slac.stanford.edu/~tlatham/public/ichep06/latham-ichep06-v5.pdf
           => `latham-ichep06-v5.pdf'
Resolving www.slac.stanford.edu... 134.79.18.163
Connecting to www.slac.stanford.edu|134.79.18.163|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1,953,840 (1.9M) [application/pdf]

100%[==============================================================================================================================>] 1,953,840    417.58K/s    ETA 00:00

20:45:22 (354.30 KB/s) - `latham-ichep06-v5.pdf' saved [1953840/1953840]

[ortler] ~ >
```

gzip

- gzip is a compression utility
  - Reduces file size
  - Multiple levels of compression – trade-off between speed vs. size reduction
- Common usage:

```
gzip myplot.eps
```

- myplot.eps becomes myplot.eps.gz and has much reduced file size
- To uncompress do:

```
gunzip myplot.eps.gz
```

- To change level of compression do (#=1-9):

```
gzip -# myplot.eps
```

tar

- tar is an archive utility
  - Allows multiple files and even large directory structures to be archived into a single file
- Common usage:

```
tar -cf archive.tar mydirectory
```

- Interaction with gzip allows creation of compressed archives:

```
tar -zcf archive.tar.gz mydirectory
```

- Compressed archives sometimes have the file extension .tgz rather than .tar.gz
- NB source code for the C++ course will be distributed as (gzipped) tar archives so need to get used to using this tool!!

tar (cont.)

- NB with all below commands add 'z' to the options if the archive is compressed
- To list the contents of an archive do:

```
tar -tf archive.tar
```

- To extract an entire archive do:

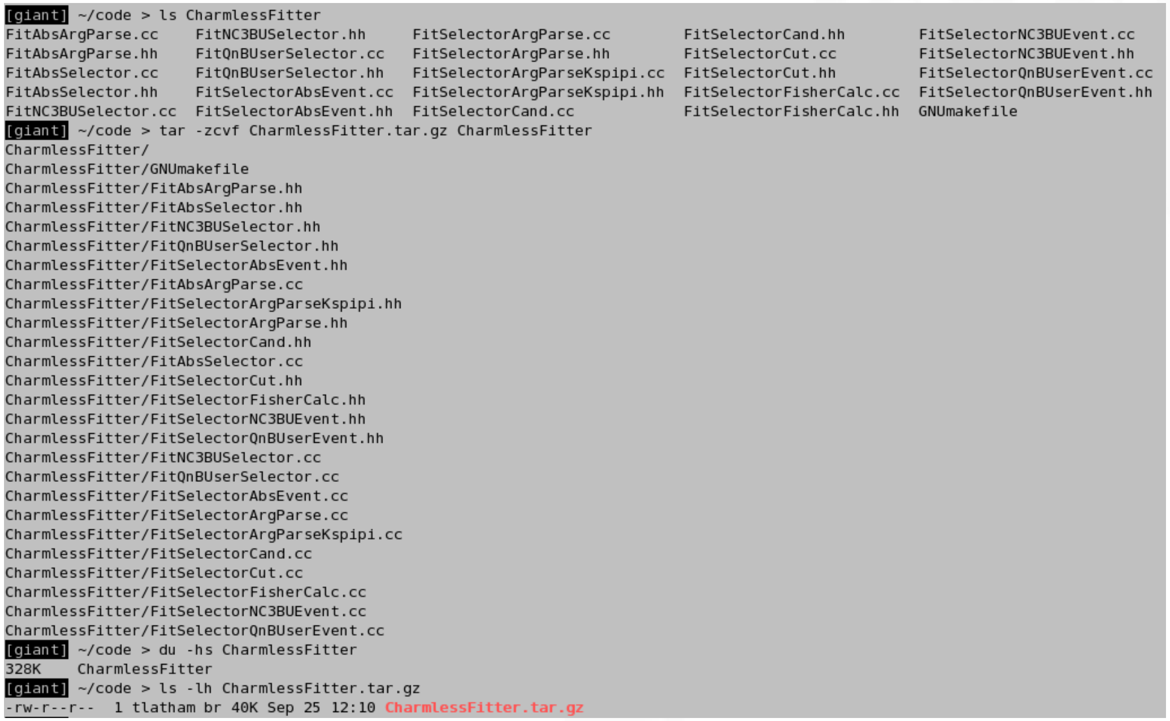
```
tar -xf archive.tar
```

- To extract a specfic file from an archive do:

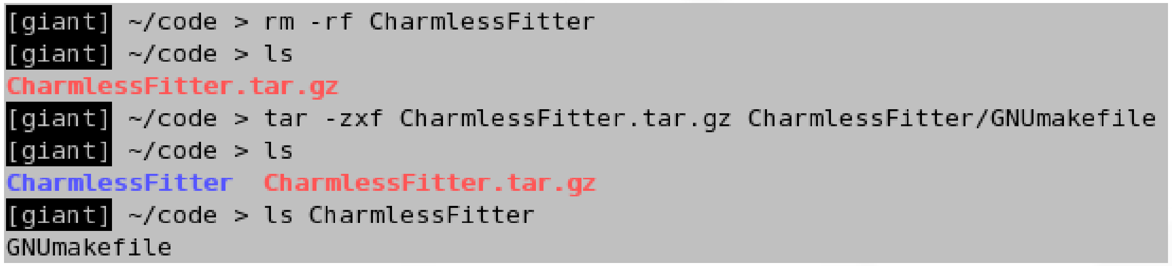
```
tar -xf archive.tar filename
```

  - where the filename must match that given by listing the archive's contents
- The option 'v' makes the output verbose, e.g. it lists the files as it archives/extracts them

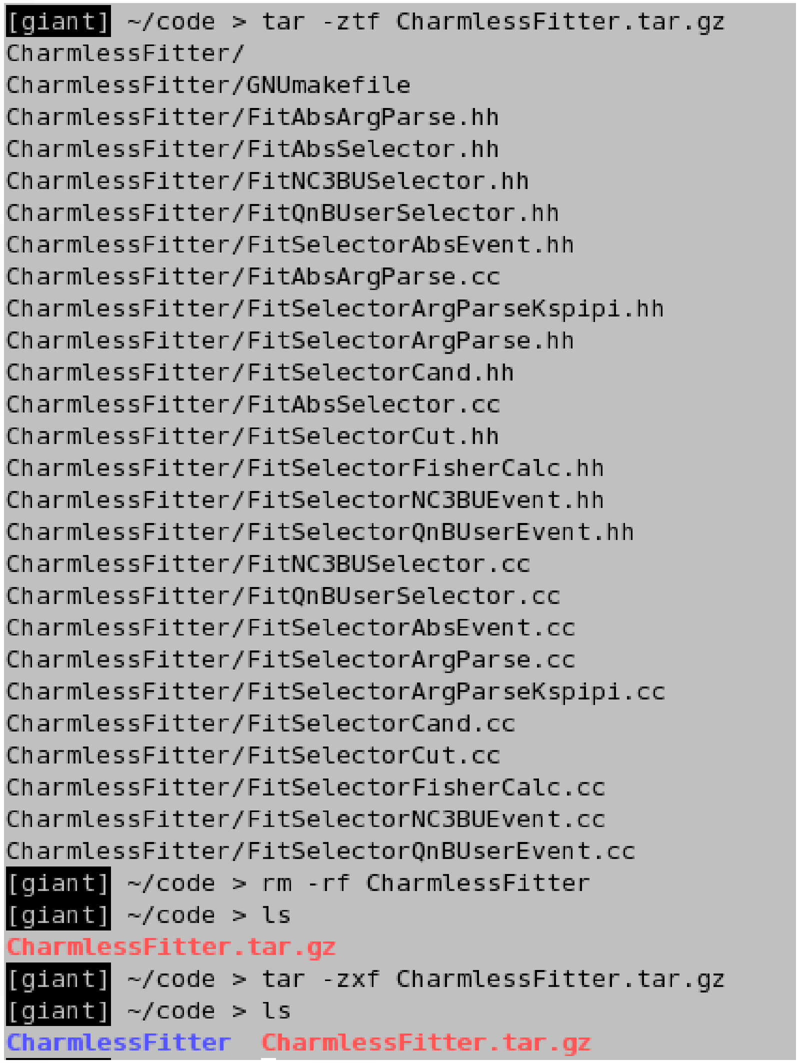
tar and gzip in use



```
[giant] ~/code > ls CharmlessFitter
FitAbsArgParse.cc    FitNC3BUSelector.hh     FitSelectorArgParse.cc        FitSelectorCand.hh         FitSelectorNC3BUEvent.cc
FitAbsArgParse.hh    FitQnBUserSelector.cc   FitSelectorArgParse.hh        FitSelectorCut.cc          FitSelectorNC3BUEvent.hh
FitAbsSelector.cc    FitQnBUserSelector.hh   FitSelectorArgParseKspipi.cc  FitSelectorCut.hh          FitSelectorQnBUserEvent.cc
FitAbsSelector.hh    FitSelectorAbsEvent.cc  FitSelectorArgParseKspipi.hh  FitSelectorFisherCalc.cc   FitSelectorQnBUserEvent.hh
FitNC3BUSelector.cc  FitSelectorAbsEvent.hh  FitSelectorCand.cc            FitSelectorFisherCalc.hh   GNUmakefile
[giant] ~/code > tar -zcvf CharmlessFitter.tar.gz CharmlessFitter
CharmlessFitter/
CharmlessFitter/GNUmakefile
CharmlessFitter/FitAbsArgParse.hh
CharmlessFitter/FitAbsSelector.hh
CharmlessFitter/FitNC3BUSelector.hh
CharmlessFitter/FitQnBUserSelector.hh
CharmlessFitter/FitSelectorAbsEvent.hh
CharmlessFitter/FitAbsArgParse.cc
CharmlessFitter/FitSelectorArgParseKspipi.hh
CharmlessFitter/FitSelectorArgParse.hh
CharmlessFitter/FitSelectorCand.hh
CharmlessFitter/FitAbsSelector.cc
CharmlessFitter/FitSelectorCut.hh
CharmlessFitter/FitSelectorFisherCalc.hh
CharmlessFitter/FitSelectorNC3BUEvent.hh
CharmlessFitter/FitSelectorQnBUserEvent.hh
CharmlessFitter/FitNC3BUSelector.cc
CharmlessFitter/FitQnBUserSelector.cc
CharmlessFitter/FitSelectorAbsEvent.cc
CharmlessFitter/FitSelectorArgParse.cc
CharmlessFitter/FitSelectorArgParseKspipi.cc
CharmlessFitter/FitSelectorCand.cc
CharmlessFitter/FitSelectorCut.cc
CharmlessFitter/FitSelectorFisherCalc.cc
CharmlessFitter/FitSelectorNC3BUEvent.cc
CharmlessFitter/FitSelectorQnBUserEvent.cc
[giant] ~/code > du -hs CharmlessFitter
328K    CharmlessFitter
[giant] ~/code > ls -lh CharmlessFitter.tar.gz
-rw-r--r--  1 tlatham br 40K Sep 25 12:10 CharmlessFitter.tar.gz
```



```
[giant] ~/code > rm -rf CharmlessFitter
[giant] ~/code > ls
CharmlessFitter.tar.gz
[giant] ~/code > tar -zxf CharmlessFitter.tar.gz CharmlessFitter/GNUmakefile
[giant] ~/code > ls
CharmlessFitter  CharmlessFitter.tar.gz
[giant] ~/code > ls CharmlessFitter
GNUmakefile
```



```
[giant] ~/code > tar -ztf CharmlessFitter.tar.gz
CharmlessFitter/
CharmlessFitter/GNUmakefile
CharmlessFitter/FitAbsArgParse.hh
CharmlessFitter/FitAbsSelector.hh
CharmlessFitter/FitNC3BUSelector.hh
CharmlessFitter/FitQnBUserSelector.hh
CharmlessFitter/FitSelectorAbsEvent.hh
CharmlessFitter/FitAbsArgParse.cc
CharmlessFitter/FitSelectorArgParseKspipi.hh
CharmlessFitter/FitSelectorArgParse.hh
CharmlessFitter/FitSelectorCand.hh
CharmlessFitter/FitAbsSelector.cc
CharmlessFitter/FitSelectorCut.hh
CharmlessFitter/FitSelectorFisherCalc.hh
CharmlessFitter/FitSelectorNC3BUEvent.hh
CharmlessFitter/FitSelectorQnBUserEvent.hh
CharmlessFitter/FitNC3BUSelector.cc
CharmlessFitter/FitQnBUserSelector.cc
CharmlessFitter/FitSelectorAbsEvent.cc
CharmlessFitter/FitSelectorArgParse.cc
CharmlessFitter/FitSelectorArgParseKspipi.cc
CharmlessFitter/FitSelectorCand.cc
CharmlessFitter/FitSelectorCut.cc
CharmlessFitter/FitSelectorFisherCalc.cc
CharmlessFitter/FitSelectorNC3BUEvent.cc
CharmlessFitter/FitSelectorQnBUserEvent.cc
[giant] ~/code > rm -rf CharmlessFitter
[giant] ~/code > ls
CharmlessFitter.tar.gz
[giant] ~/code > tar -zxf CharmlessFitter.tar.gz
[giant] ~/code > ls
CharmlessFitter  CharmlessFitter.tar.gz
```

Exercise

- A quick exercise in using `wget`, `tar` and file system navigation
- Please download the following file to your home directory using `wget`:

  http://www2.warwick.ac.uk/fac/sci/physics/staff/research/tlatham/teaching/computing2011/linux/exercises.tar.gz

- Firstly list the contents of the archive
- Next extract “directory1” from the archive
- Navigate around this directory and its sub-directories and try listing, copying, moving and deleting files
- Also try using `find` to locate certain files
- Now extract “directory2” from the archive, move all its sub-directories into “directory1” and delete empty “directory2”
- Finally perform a recursive delete on “directory1”

Useful Commands

- We've already seen some Linux commands relating to filesystem operations
- We're now going to look at a range of commands to help with viewing and processing text and text files.
- To help in learning these commands, we've supplied 2 basic text files in directory3 of the archive: `particles_a.dat` and `particles_b.dat`
- So please extract these two files from the archive
- They are hypothetical data files containing event number, particle name, momenta, and raw data source file name:

```
Event   Name    p_x     p_y     p_z     datasource
1001    e-      1.0     1.2     3.1     run00001.dat
...
```

less – viewing text files

- To view (i.e. read but not edit) text files you can use `less`
- `less` is an improved version of an earlier program called `more` (computer scientists' idea of a joke)
- Allows scrolling both forwards and backwards through file as well as basic searching
- Up and down arrow keys (or “j” and “k”) scroll through file line by line
- “Ctrl+f” and “Ctrl+b” go through page by page
- Type a number then “G” to go directly to a line no.
- Typing “/” allows you to type a search string
- Typing “?” does the same but search is backward
- To quit, type “q”
- Practice using `less` by viewing `particles_a.dat` and `particles_b.dat`

cat

- less is a basic text viewer, but cat is simpler still
- It just concatenates the contents of one or more files and outputs it to standard output.

```
[me@here ~]$ cat particles_a.dat particles_b.dat
...contents of particles_a.dat...
...contents of particles_b.dat
```

- Not exactly exciting, but is good for quickly viewing a short file
- Its real power comes later when we look at linking commands
- Typically cat is used to pipe (see later) the contents of a file to another command for processing

head/tail

- `head(tail)` prints the first(last) n lines of files:

```
[me@here ~]$ head -n 2 particles_a.dat particles_b.dat
...first 2 lines of particles_a.dat...
...first 2 lines of particles_b.dat
```

- tail is more useful as it provides the options:

```
-f, --follow, output appended data as file grows

-s –sleep=S, used with -f, sleep for S seconds
  between iterations
```

- This is handy for monitoring files that are updated regularly.
- ***Try this:*** Open two terminals. In one create mon.txt and do

```
[me@here ~]$ tail -f -s 5 mon.txt
```

- In the other terminal, keep doing

```
[me@here ~]$ echo “muon” >> mon.txt
```

grep

- `grep` is used to search for patterns in files and print lines matching/not matching the pattern.
- ***Try this:*** Say we want to find all electron entries in particles_a

```
[me@here ~]$ grep “e-” particles_a.dat
```

- ***Try this:*** We can find all lines that DON'T list an electron with the -v option

```
[me@here ~]$ grep -v “e-” particles_a.dat
```

- ***Try this:*** Pattern matches can also be based on regular expressions, e.g.

```
[me@here ~]$ grep e[+-] particles_a.dat
```

- This finds all electrons and positrons.
- We don't look at 'regexps' in this course, but there's plenty of documentation out there to help you.

diff

- `diff` is used to compare files line by line and present any differences found.
- Useful for creating file 'patches' so that whole file doesn't have to be redistributed when you make a small change.
- ***Try this:*** Use the `-q` option to simply check for differing files

```
[me@here ~]$ diff -q particles_a.dat particles_b.dat
```

- ***Try this:*** Use `-y` (output in two columns) and `--suppress-common-lines` so we just see the differing lines

```
[me@here ~]$ diff -y --suppress-common-lines \
  particles_a.dat particles_b.dat
```

- ***Try this:*** Use `-u` to output a unified diff (standard for patches)

```
[me@here ~]$ diff -u particles_a.dat particles_b.dat
```

cut

- `cut` removes sections from each line of a file and outputs the removed sections as required.
- Most useful options are

```
-d, --delimiter=DELIM : Use DELIM as the thing
  separating fields in the line (default is TAB).

-f, --fields=LIST : Use LIST as a comma separated
  list of output fields

--output-delimiter=DELIM : Use DELIM as the thing
  separating output fields
```

- ***Try the following:***

```
[me@here ~]$ cut -f 1,2 --output-delimiter “, ” \
  particles_a.dat
```

- ***Try this:*** Can you print the particle name followed by the event id ***in that order***?

Chaining Commands

- Whilst the commands we've looked at are useful on their own, they become even more useful when chained together
- Linux enables this chaining through I/O redirection
  - We'll go into this a lot more in the extra booklets
- The output of a command can be redirected to a file:

```
[me@here ~]$ grep “e-” particles_a.dat > elec.dat
```

- Or it can be fed into another command:

```
[me@here ~]$ grep “e-” particles_a.dat | cut -f 1
```

- The first example writes all the lines with electrons into a file called elec.dat
- The second finds all the lines with electrons in and then prints only the first field

xargs

- `xargs` allows you to use the output of one command as the command line arguments of another, e.g.

```
[me@here ~]$ ls *.dat | grep particles | xargs diff -q
```

- The above command lists all the names of all files in the current directory with the extension .dat and then filters that list to only those names that contain the string “particles” and finally passes them as the arguments to `diff`
- NB that we can provide other specific options to the executed command, e.g. the `-q` here
- There are, as always, various options available for the `xargs` command – see the man pages for details

sed

- `sed` (lit. Stream Editor) takes a stream of input from a file or stdin and performs operations on it, outputting the result.
- Most often used to match and replace text

```
[me@here ~]$ sed 's/oldtext/newtext/' file.txt
```

- Here the string `oldtext` is replaced where found with the string `newtext` in the output stream.
- ***Try this:*** Say we want to rename all pi+ in particles_a.dat to pion

```
[me@here ~]$ sed 's/pi+/pion/' particles_a.dat
```

- `sed` can perform much more advanced operations than this, we'll give details in a couple of slides.

awk

- `awk` (from surnames of its creators Aho, Weinberger, Kernighan) is actually a programming language.
- The `awk` command inteprets input to the `awk` language
- Naturally, it's quite complicated – but very useful for some tasks.
- ***Try this:*** Saw before that `cut` could not swap order of output fields, but this can be done using `awk`:

```
[me@here ~]$ awk '{print $2,$1}' particles_a.dat
```

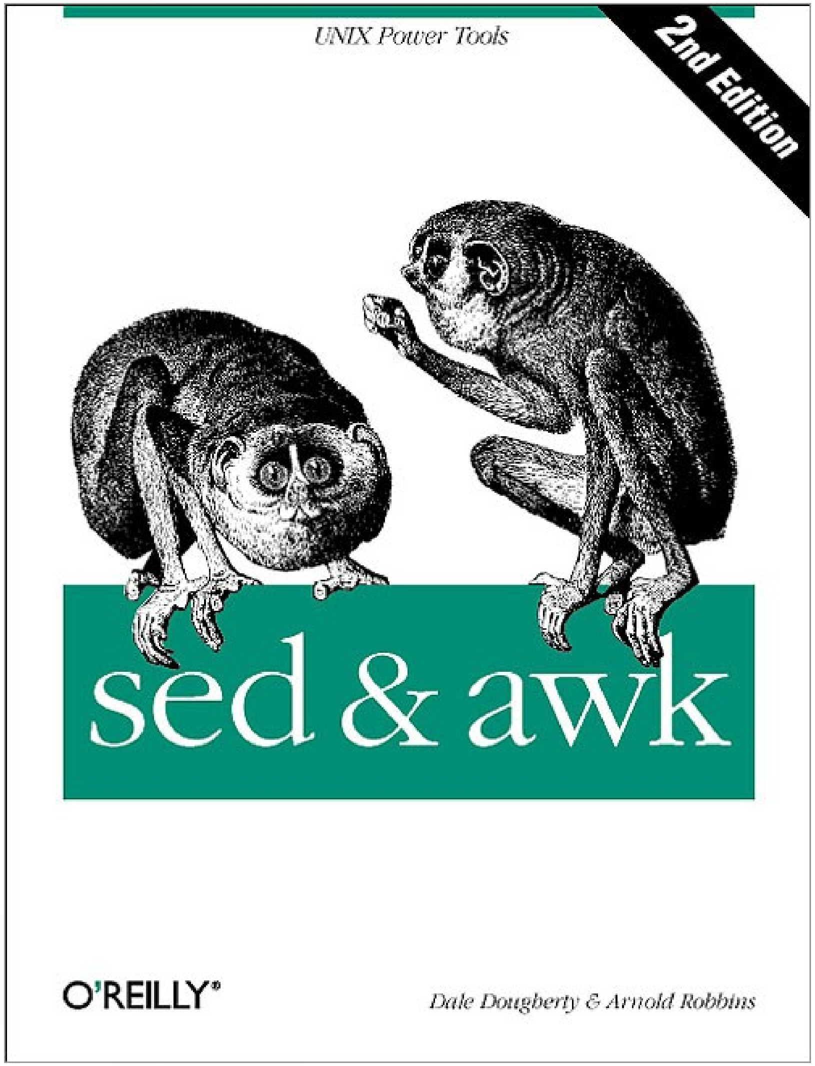
- The quoted portion contains an `awk` script.
- As with `sed`, much more advanced operations are possible

More on sed and awk

- Such is the depth of `sed` and `awk` that there's an entire book devoted to them if you want to investigate further.

- As with most Linux/Unix information, there're tons of helpful guides just a Google search away.

- Whilst `sed` and `awk` are useful, if you find yourself writing long commands in them, you may well be better off using Perl or Python instead.



Text Editors

- So we've seen how to view the contents of a file with `less`.
- Since we'll soon be moving on to C++ programming we'll want to edit them as well
  - Write C++/Java/Python/Shell script source files
  - Write reports
  - Edit system files
- Linux provides a wide range of ***text editors:***

`vim` `emacs`

`kate/gedit`

`nedit` `pico`

And many others...

vim & emacs

- To start vim:

```
vim <filename>
```

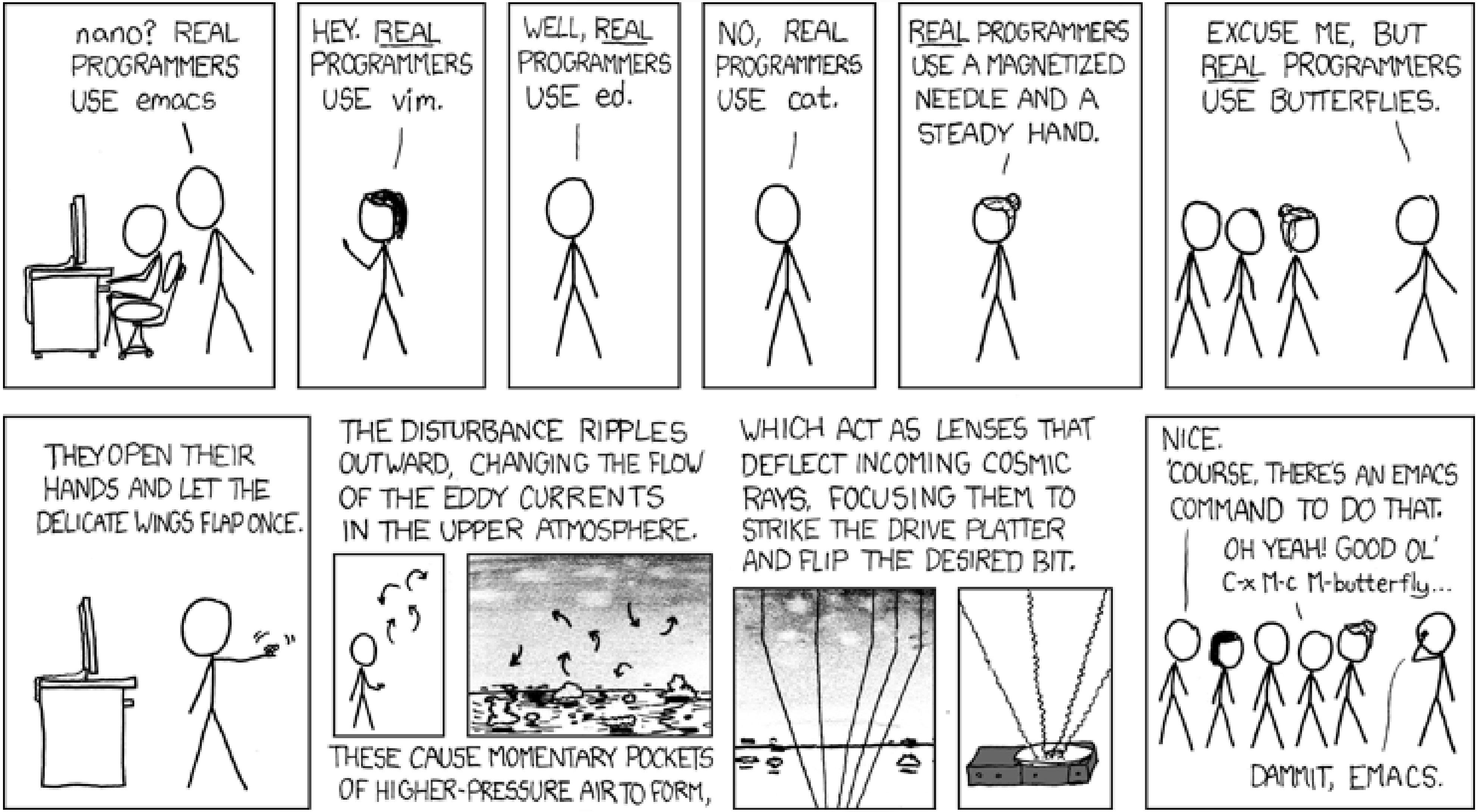
- Runs in terminal, but GUI interface may be available (e.g. `gvim`)
- Very useful for system and remote work.
- Cleaner than emacs, but steeper learning curve.

- To start emacs:

```
emacs <filename>
```

- May also be in menu.
- Good for desktop work.
- Runs in GUI or terminal.
- Extremely configurable, but that can lead to confusion...

- Whilst arguing about which editor is best is a common pastime, the best editor is the one that enables you to be most productive.
- ***It's therefore important that you try several editors and find the one that suits you best.***


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

Cartoon courtesy of xkcd.com

Other Information Sources

- Our extra material booklet gives much more info:

  http://www2.warwick.ac.uk/fac/sci/physics/staff/research/tlatham/teaching/computing2011/linux/

- We've seen that man pages provide help with the use of commands.
- Many other sources of more detailed info.
- Websites:
  - http://www.tldp.org (Linux Documentation Project)
  - http://www.linux.org
  - Many, many others through Google and Wikipedia.
- Books:
  - http://www.oreilly.com - the famous 'animal books'.
  - **HIGHLY** recommended – always worth starting with the O'Reilly text on the subject of interest.

Exercises

- Exercise 1:
  - Find all electrons in particles_a.dat
  - Sort on p_z
  - Get rid of the file extension on the data source file name
  - Print out the data source file name, event number and p_z (in that order)
- Exercise 2:
  - If you're happy with the above, can you find other ways of doing the same thing?
- Exercise 3:
  - Sort all muons, firstly by p_x and then by charge
  - Print out the p_x, p_y and event number (in that order) into a new file called selected-muons.dat