Linux Training Materials Project

GBdirect Limited
27 Park Drive
Bradford, BD9 4DS
West Yorkshire
tel: +44 (0)1274 772277
linux@gbdirect.co.uk

with contributions from
Nick Urbanik
nicku@nicku.org

4 October 2013

Contents

Module 1

Overview

Objectives

Having completed this module, you will have an overview of a Linux system, including its:

- Underlying philosophy
- System layering — kernel vs. applications
- Core services
- Multiuser and timesharing facilities
- File System
- Network Services
- Desktop and X windowing system

1.1 Generic Features of Unix

- Component-based systems
- Very popular with technically skilled
- Not ‘solution’ oriented
- Building blocks not the building
- Highly network-aware
- Robust, powerful, reliable

1.2 Linux — The Kernel of a System

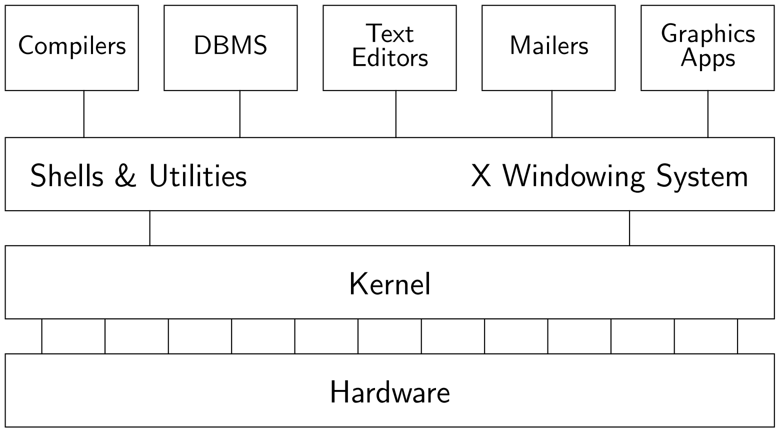


Figure 1.1: kernel-layering

- What is *called* Linux is actually a collection of components from many sources
  - freely copiable, under 'open source' licences
- Linux is, strictly, just the *kernel* which provides:
  - A common interface between *user process* and hardware
  - Minimal functions to user applications, i.e. system calls
  - Scheduling

1.3 Fundamental Characteristics of Linux

- Multi-tasking
- Multi-user access
- Multi-processor
- Architecture independence
- POSIX 1003.1 plus basic System V and BSD
- Protected memory mode
- Multiple filesystem types
- Comprehensive networking (TCP/IP and others)
- Multiple executable formats (MS-DOS, iBCS UNIX, SCO, etc)

1.4 Multiuser Multitasking and Time-sharing

- Designed as a multi-user system
  - Each user's shells, apps and commands are separate processes
  - Number of simultaneous users limited only by:
    - CPU speed and available memory
    - Min. response times required by users/apps
- Multi-tasking:
  - Many jobs can be under way at the same time
  - Jobs truly *simultaneous* on multi-cpu
- Time-sharing:
  A single cpu is shared by all processes
  - Processes exec briefly, passing cpu to others
  - *Process switches* occur in miliseconds or less
  - Kernel gives process a sense of total control

1.5 Protected memory mode

- Uses the processor's protection mechanisms
- Prevent access to memory already allocated to kernel or other processes
- Bad programs can't crash the system
  - Theoretically

1.6 Multiple Filesystem Types

- Native FS is ext3 (Third Extended File System)
  - File names up to 255 chars
  - More secure than conventional UNIX
- Others include:
  - MS-DOS (FAT16), VFAT, FAT32
  - ISO9660 (CD-ROM)
  - HPFS (OS/2)
  - NTFS (Windows NT)
  - reiserfs, XFS, other journalling file systems for Linux,
  - UPS, SysV and other proprietory UNIX
  - NFS (Unix network file system)
  - SMB / CIFS (MS Windows file sharing)

1.7 The Many Faces of a GNU/Linux System

- The user may see up to five aspects of Linux:
  - the *filesystem*
  - *processes*
  - the *shell*
  - the X *windowing system*
  - *Inter-Process Communication* (IPC)
- The system is very highly configurable
- Different users may experience totally different views of the same system
- Multiple simultaneous users are normal
  - Linux is designed from the ground up as a *multi-user system*, NOT a 'personal' system

1.8 The Filesystem

- The filesystem contains all data in the system
- A name in the filesystem can refer to:
  - a *data file*, which can be:
    - a *plain file*
    - a *directory*
  - a *device* (disk, tape etc.)
  - internal memory
  - OS information (the *proc* system)
- Directories are groups of files
  - Grouped in hierarchical *trees*
- Files are fully specified with their *pathname*
- An original Unix structure; copied by most OSs

1.9 Filenames

- Maximum length depends on filesystem type
  - Most allow up to 255 characters
- Can use almost any character in a filename, but avoid ambiguity by sticking to:
  - (A-Z) Uppercase letters
  - (a-z) Lowercase letters
  - (0-9) Numbers
  - (.) Full-stop
  - (,) Comma
  - (_) Underscore
  - (-) Hyphen
- Should convey meaningful info about contents
- Type longer filenames using completion for:
  - Filenames
  - Pathnames
  - Commands

1.10 Filename Extensions and File Types

- Filenames *don't* determine other attributes of file,

  i.e. do not, *automatically*, cause command interpreters to treat them in a particular way
- However:
  - Extensions can enable meaningful naming and automatic file manipulation
  - C compilers and some other programs *do* depend on specific file extensions to carry out particular tasks
- Common conventions for extensions:

Filename	Meaning of Extension
program.c	C programming source file
program.o	Object code
program.sh	Shell executable
letter.txt	Text file of a letter
letter.ps	Postscript version of same letter file
letter.ps.gz	`gzip` compressed version of same
letter.tar.bz2	`tar` archive of same compressed by `bzip2`
letter.tgz	`tar` archive of same compressed by `gzip`
letter.tar.gz	Another, more common, way of naming `*.tgz`
letter.Z	Same file compressed with outdated `compress` utility

Table 1.1: Common conventions for filename extensions

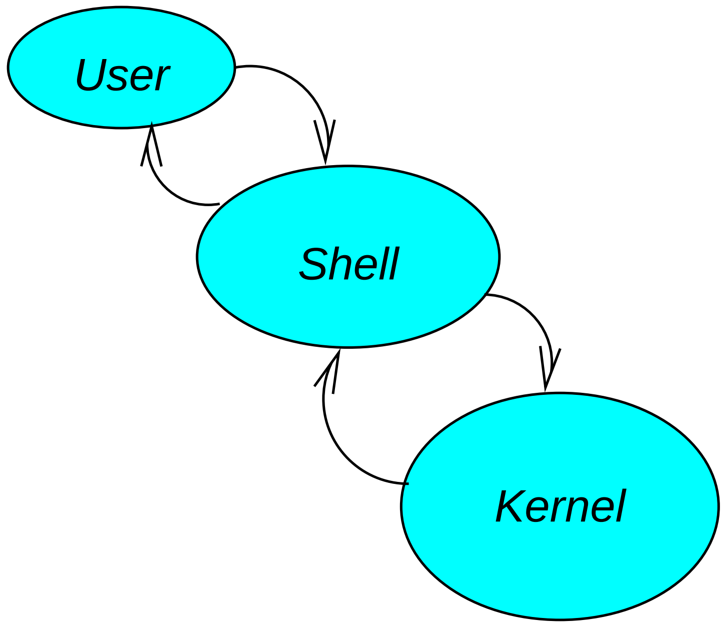
1.11 Hidden Filenames

- Filenames beginning with a full-stop are *hidden*
- Typically used:
  - To hide personal configuration files
  - To avoid cluttering dirs with rarely used files
- Every dir contains 2 special hidden files:

.	The current directory file
..	The parent directory file

1.12 The Shell (`bash`)

- A *shell* is a program that you interact with



- Can be any program, but is normally a *command interpreter*
- A command interpreter is usually started when you log in (but this is just one way)
- The 'standard' Linux command interpreter is a `Bourne` *shell* look-alike called `bash` *
- The command line syntax provided by `bash` enables manipulation of files & processes
- The command-line frightens beginners but is the preferred home of the skilled

*`Bash` has more functions than true `Bourne` shells; incorporating most of the innovations added by the `C` and `Korn` shells. `Bash` functions and flags differ between implementations of UNIX and Linux. The version of `bash` in current Linux releases tends to be the most fully functional `Bourne` shell around.

1.13 Key Features of the Bash Shell

- Command history
- Command aliasing
- Shell scripting
- Filename completion
- Command completion
- Command line editing (`emacs` and `vi` styles)
- Job control
- Key Bindings
- Directory stacking
- Tilde directory notation
- Help function, e.g.

```
$ help history
history: history [n] [ [-awrn] [filename]]
    Display the history list with line numbers.  Lines listed with
    with a '*' have been modified.  Argument of N says to list only
    the last N lines.  Argument '-w' means to write out the current
    history file;
```

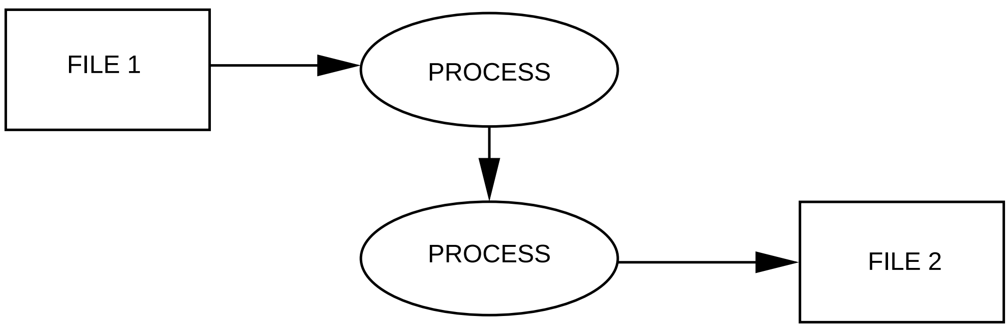

1.14 Interacting with a Linux 'Terminal'

- Linux can support any number of 'terminal' types
  - nowadays, monitor/keyboard combinations
  - previously, dumb terminals
  - occasionally, printers (debugging servers)
- Most will use the *console* or a windowed terminal, but if not:
  - Linux usually keeps a database of terminal capabilities in `/etc/termcap` *
  - If your terminal type is not recorded in `/etc/termcap`, you'll have problems running certain programs e.g.
    - cursor driven apps (`top`, `linuxconf`, `vi` etc)
  - The *environmental variable* `TERM` tells programs what terminal type you are using

*AT&T flavours of UNIX use `/usr/lib/terminfo` to store the same information and Linux can, if necessary.

1.15 Software Tools: The UNIX Philosophy

- True UNIX-like systems treat programs as *tools*
  - Each tool should:
    - Do just one thing well
    - Be generic (untied to specific applications)
  - For new jobs, build new tools
  - (Re-)combine, don't complicate old tools
- Linux can do this because it has:
  - two simple *objects*:
    - the file
    - the process
  - simple methods of *connecting*:
    - processes to files
    - processes to processes



1.16 Tasks/Processes

- A *program* is an execut*able* object, stored in a file
- A *process* is an execut*ing* object, i.e. *
  - an *instance* of a program currently being run
- Existing processes can '*fork*' to create other processes
  - the only way to make new processes
- A user may run multiple copies of same program
- Multiple users may run single/multiple copies
- System tracks *ownership* and *permission*

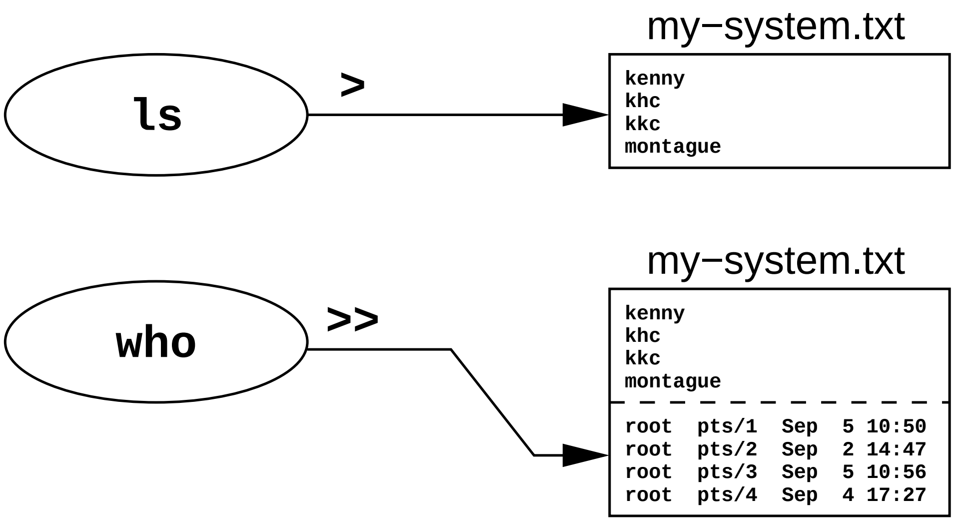
*Processes are often called *tasks*, as in 'multi-tasking'

1.17 Process Communication

- Processes may need to co-operate by
  - sharing files
  - signalling events
  - direct transfer of data
  - pipelines (data streams)
  - synchronising with each other
- Linux provides facilities for:
  - signals
  - shared memory
  - pipes, both named and unnamed
  - semaphores
  - and others
- Processes may use network connections for communication, permitting *client-server* model
  - Common for shared services like printing

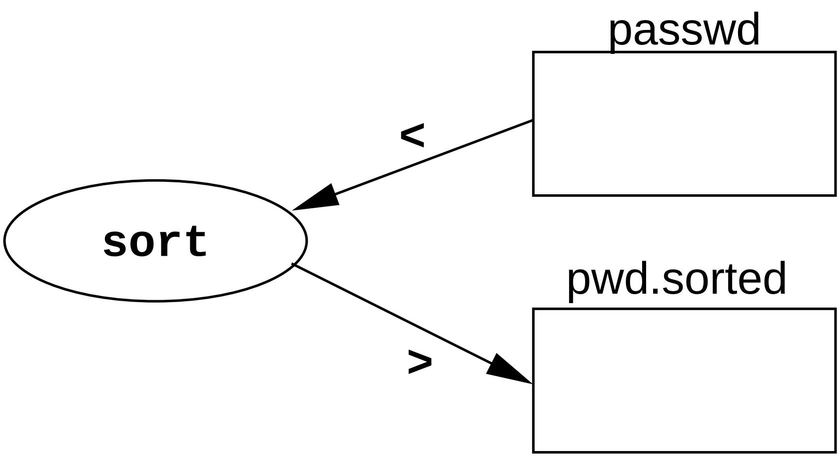
1.18 Re-directing I/O to and from Files

- Most processes will take input from the keyboard and output to the screen
- Both input and output streams can be *re-directed* to/from files
- Output to a file (creating or overwriting):
  `$ ls > my-system.txt`
- Appending output to a file: `$ who >> my-system.txt`



1.19 Re-directing I/O to and from Files (continued)

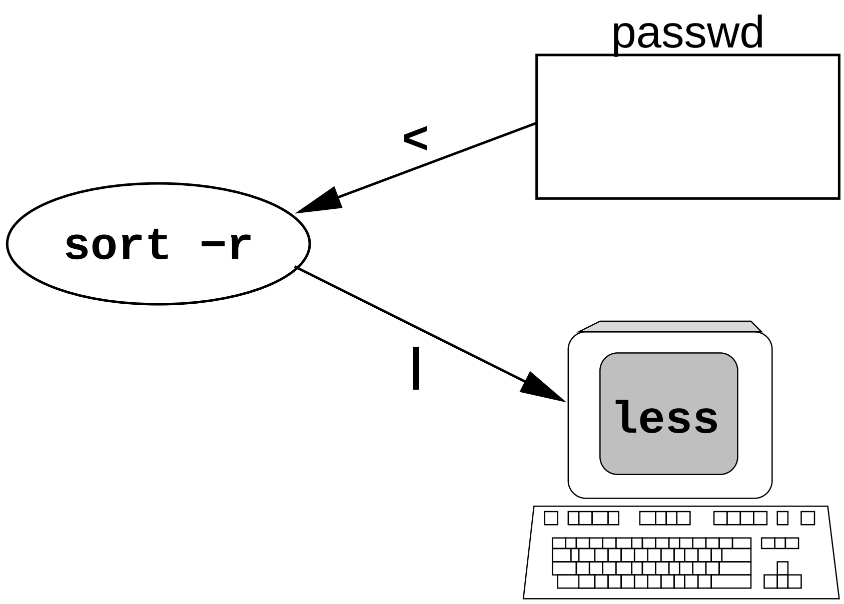
- Take input from one file, output to another:
  `$ sort < /etc/passwd > pwd.sorted`



1.20 Pipes & Tools

- Linux tools act as filters:
  - taking data from input streams, modifying it, sending it elsewhere
  - expecting data to come from other tools
  - producing output which *any* other tool can process, e.g. ASCII text
- One tool's output is connected to another's input:
  - *Indirectly*, via a file created by the first tool
  - *Directly*, via a *pipe* or *pipeline*
- For example, to page through a reverse-sorted version of your password file on screen:

```
$ sort -r < /etc/passwd | less
```



1.21 Linux as a Programming Environment

- *Hierarchical Filestore*
- Extensive set of *powerful tools*
  - for software production, admin and support
- A *common system interface*
  - only one set of procedures to learn
- Processes interface with *anonymous files*
  - programs output to files or devices identically
- *Modular architecture* provides for a completely customised OS, e.g.
  - An OS dedicated solely to graphics rendering
  - A general-purpose system on one floppy
- *Flexible user interface* allows for uniquely customised programming environments

1.22 Networking

- Linux is a network operating system.
- The Internet network protocols (TCP/IP) are implemented in the kernel
- Although other media are supported (e.g. radio, infra-red), links are usually across:
  - Ethernet
  - Serial Line (Point-to-point)
- Proprietory file/print serving protocols supported:
  - Appletalk
  - DECNET
  - IPX / Novell Netware
  - SMB / CIFS (MS Windows/NT)

1.23 TCP/IP

- A suite of Internet-standard protocols and apps for managing data transfers
- Depicted as a 'stack'
  - hardware and transport control protocols at the bottom
  - user applications (e.g. browsers) at the top
- Client-server apps provide facilities for:
  - Remote login
  - File transfer
  - Resource sharing (e.g. expensive peripherals)
  - Remote command execution
  - Email (internet/intranet/extranet)
  - Web browsing

1.24 Documentation

- Copious, but fragmented and/or duplicated

Programmer's Manual `/usr/man`	The classic '*man pages*', first stop for skilled users, worth learning
`info` pages	hypertext browsable texts, often identical or updated versions of man pages
`/usr/share/doc/`*`program-name`*	ascii/html docs installed with the named program
Howtos	Tutorials on Linux-related topics, available on-line if installed (usually in `/usr/share/doc`)
www	Recently-released programs are usually documented on authorised web sites, many (including older tools) are documented by third-party sites

Table 1.2: Sources of Linux Documentation

- Linux man pages divided into sections:

  1. User Commands
  2. *System calls*
  3. Subroutines (inc library routines)
  4. Devices (inc network interfaces)
  5. File Formats
  6. Games
  7. Miscellaneous
  8. System Administration

- The `apropos` command word searches the description line in man pages. Thus:

```
$ apropos printer
```

  will find man pages relating to printers, e.g.

```
lp (4)   - line printer devices
lpd (8)  - line printer spooler daemon
lprm (1) - remove jobs from the line printer spooling queue
```

1.25 Using the *man pages* (On-Line Manual)

- Use `man` to see man pages on a named command, e.g

```
$ man date
```

- The result should be something like:

```
DATE(1)                          FSF                          DATE(1)

NAME
       date - print or set the system date and time

SYNOPSIS
       date [OPTION]... [+FORMAT]
       date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

- `DATE(1)` Shows page is in manual section 1
- To view a page from a certain section use:

  `$ man -S` *`section-number command-name`*

- Square brackets surround optional arguments

```
ls  [-abcdfgiklmnpqrstuxABCFGLNQRSUX1]
```

1.26 Overview Exercises

What should I focus on?

At the end of this practical session, as a minimum, you should be able to:

- within a `man` page, be able to:
  - go to the beginning of the `man` page with one keystroke
  - similarly, go to the end of the `man` page
  - search for a word within the `man` page
  - search again for the same word using one keystroke.
- Find the `man` page that you need:
  - quickly find a man page with a description that matches a keyword
  - exhaustively search all man pages to find all that contain a keyword.
- You should know how to use the `df` command to tell how much space is left in your root (`/`) partition
- Have a basic understanding of what a pipe (`|`) is useful for.
- You should know how to use the `wc` command, and clearly understand that it has nothing much to do with the wash room.

The exercises

1. *Changing password*

(a) Set yourself a new password using the `passwd` command. Run the command by typing `passwd`, followed by a `<RETURN>` .

2. *Navigating Man Pages*

(a) Type `man man` to open the man page which details how to use the `man` command

(b) Press the `h` (help) key, which opens a “Summary of Less Commands”, including all the keystrokes you need to navigate a man page

(c) Make sure you can quit this page (by typing `q`) and quit the man page (by typing `q` again). When you get back to the shell prompt, repeat the first 2 steps to open the “Summary of Less Commands” from the `man` man page.

(d) Use the “Summary of Less Commands” to make sure you know how to do the following bits of navigation inside a man page:

i. Move to the top and bottom of the man page
ii. Move up and down one screen of text
iii. Move up and down one line of text
iv. Search forward for a *pattern* (e.g. a word)
v. Search backwards for a *pattern*

vi. Repeat a forward *pattern* search using one key

vii. Repeat a backward *pattern* search using one key

viii. Move to a specific line number. For this, read the text at the top of "Summary of Less Commands", particularly:

> Commands marked with $*$ may be preceded by a number, N. Notes in parentheses indicate the behavior if N is given.

The phrase *preceded by a number* N means that the number N may come first, before the command.
Next, have a look at what is under the heading "Jumping". Note that you can check where you are by pressing "=", which shows the line numbers at the first and last lines in the terminal window.
Try this on the `man` page for ethereal, which has more than 40,000 lines. See if you can go to line 10,000.

(e) With a partner, test each other on how well you can navigate the `man` man page, e.g. set each other target locations or words to go to.

3. *Invoking the Right Man Pages*

Note that some time after you start your lab session, a program automatically builds a file `/var/cache/man/whatis` which contains the one summary line from each of the thousands of `man` pages on your computer. There are two programs, `apropos`, mentioned in section 1.24 on page 25. There is another program, `whatis`, that matches a program name in the same "whatis" database. The `man` command has options that call these programs. Each has their own `man` page.

(a) Using the `man` man page, find the command string you need to use to get the following:

i. A list of man pages whose description lines contain details about the 'whatis' database

ii. A list of man pages containing the string 'cdrom' *

iii. A list of man pages from a specific section (e.g. 1) of the manual, whose description lines contain 'print'

(b) Practice using these flags to find and view man pages which deal with computer keywords your partner sets for you (and vice versa), e.g.

i. bitmap formats like jpg, gif, xpm, bmp

ii. communications concepts like modem, serial, telnet, pcmcia, ppp

iii. filesystems like NFS, ext3, ext2, FAT, vfat, msdos, samba

4. *Finding Out About Your System and Users*

(a) Type the following commands. Identify what each of them tells you about your system.

i. `$ whoami`

ii. `$ who am i`

iii. `$ users`

iv. `$ who`

v. `$ w`

vi. `$ date`

vii. `$ cal 10 2003`

viii. `$ cal 9 1752` †

ix. `$ df`

*Actually running this sort of search can take a long time, given that many systems contain over 10,000 man pages, some of which are very long.

†You should notice something very strange about the output from this string. The `cal` utility is perfectly functional, so what's wrong?

x. `$ which man`
xi. `$ type man`
xii. `$ whereis less`
xiii. `$ help cd`
xiv. `$ time sleep 2`

(b) Use the appropriate man page, to check that you have interpreted the screen output correctly

5. *Creating New Files*

(a) Try creating a new *empty* file in your home directory using the `touch` command, e.g.
`$ touch` *filename*

(b) Get the file details on *filename* using this command:
`$ ls -l` *filename*

(c) Wait 1 minute, then repeat the previous two steps, i.e.
`$ touch` *filename*
`$ ls -l` *filename*

i. Which of the file details have changed?
ii. What does this tell you about the purpose of `touch`? Check the man page if you are unsure.

(d) Creating new files using re-direction

i. Create a new file containing the output from the `df` command, using re-direction, e.g.
`$ df > diskspace.txt`
ii. Ask a partner to create new files, with appropriate filenames, containing output from the commands used in the questions on “Finding Out About Your System and Users”.

6. *Appending information to files*

(a) With a partner, choose several of the system information commands whose outputs may have changed since you completed the previous question. Practice appending the updated information to the file which contains the earlier output.

(b) Create a file containing output from `w`, then append the output from `date` to it, i.e. time-stamp the output data.

7. *Using Simple Pipes*

(a) Pipe the output from `who` through the `sort` command to reverse its order.

(b) Sort your `/etc/passwd` file alphabetically and send the output to a new file (`passwd.sorted`).

(c) Find out what `wc` does from its man page, then use it at the end of a pipe to analyse the output from other utlities.

(d) Repeat the last step, limiting `wc` to counting words only

1.27 Overview Solutions

Module 2

Basic Shell

Objectives

On completion of this module, you should be able to understand and use the Linux shell to create and combine tools.

Topics covered include:

- An overview of the command line
- The software tools model
- File names and types
- Shell programming
- Command scripts
- Job control
- I/O — pipes and redirection

2.1 Introduction

- The standard command line interpreter under Linux is `bash` (`/bin/bash` or `/bin/sh`)
- An enhanced version of the classic Bourne shell
- Shares most features of other shells (`C`, `Korn`, etc) and has some more advanced features
  - ‘Plumbing’ — transparent redirection and pipes
  - Background processes
  - Process suspension, resumption, termination
  - Filename completion and wildcard generation
  - History

2.2 Getting around the command line

- You can use the cursor keys to move around and edit the current line*
- By default, `bash` uses `emacs`-like keystrokes for navigation and editing. Here are 4 examples:

Keystroke	Action
`^a`	Move to the beginning of the line
`^e`	Move to the end of the line
`^k`	Delete to the end of the line
`^w`	Delete the previous 'word'

- To choose `emacs` or `vi`-like keystrokes:

```
$ set -o emacs
$ set -o vi
```

- `bash` man page gives details of all keystrokes

*This may not work on badly-configured systems

2.3 History

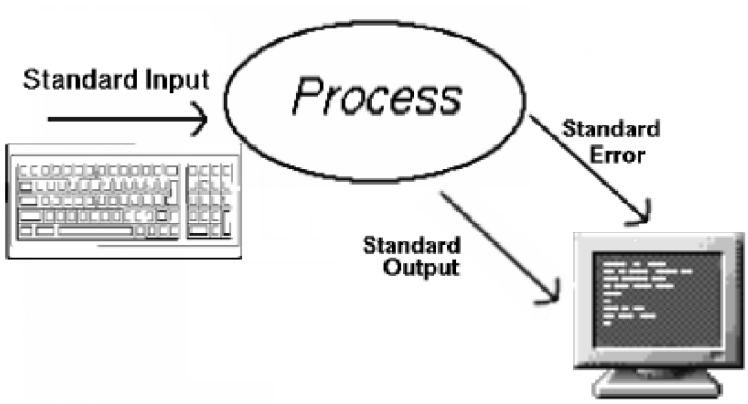
- Bash remembers used commands (in a 'history')
- Old commands are retrievable in different ways
- Repeat the previous command by typing `!!`
- Execute the *n*th previous command by typing `!-`*n*
- Typing `!string` repeats the last command beginning with `string`
- To view your history command by command, use the `up` and `down` cursor keys
- View your history at any time by typing `history`
- History is a *very* useful feature, if used well
    - Incrementally searchable using `CTRL-R`

2.4 Plumbing

- Processes typically start with three files open:

Name	Descriptor
Standard input	0
Standard output	1
Standard error	2

- Later we see how to refer to their *file descriptors*
- These are normally connected to the keyboard and your command-line terminal



2.5 Plumbing (continued)

- Data can be redirected by the shell
  - Transparently to the process concerned
  - Any or all streams can be redirected
  - You can redirect to/from a file or to/from another process
- Redirection to a process is known as ‘piping’

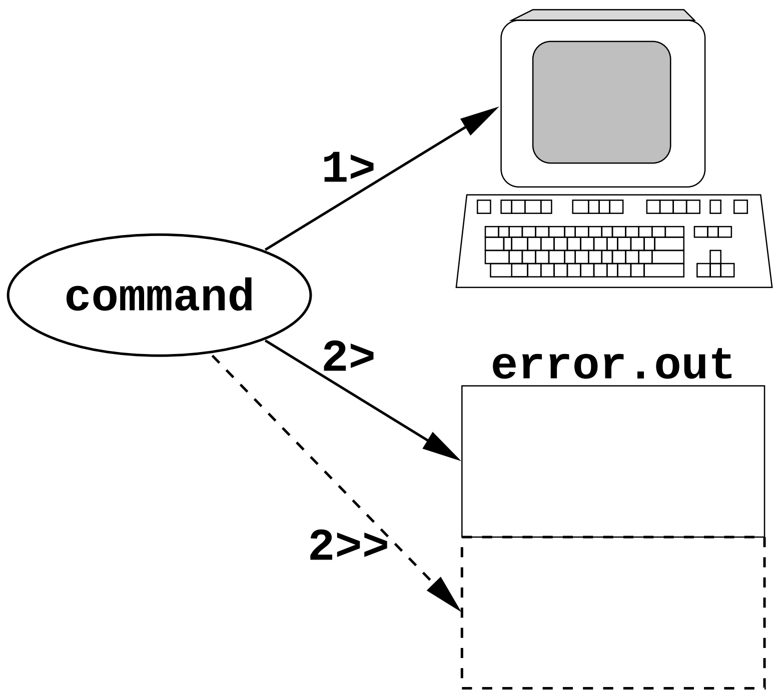
2.6 Output Redirection

- Redirection of output is done using '>'
- For example:

  ```
  $ command > output
  ```

  Creates the file `output` (or overwrites it if it already exists) and places the standard output from `command` into it
- We can append to a file rather than overwriting it by using >>
- > and >> are actually shorthands for `1>` and `1>>`
- Error output can be redirected using `2>` or `2>>`

  ```
  $ command 2> error.out
  ```

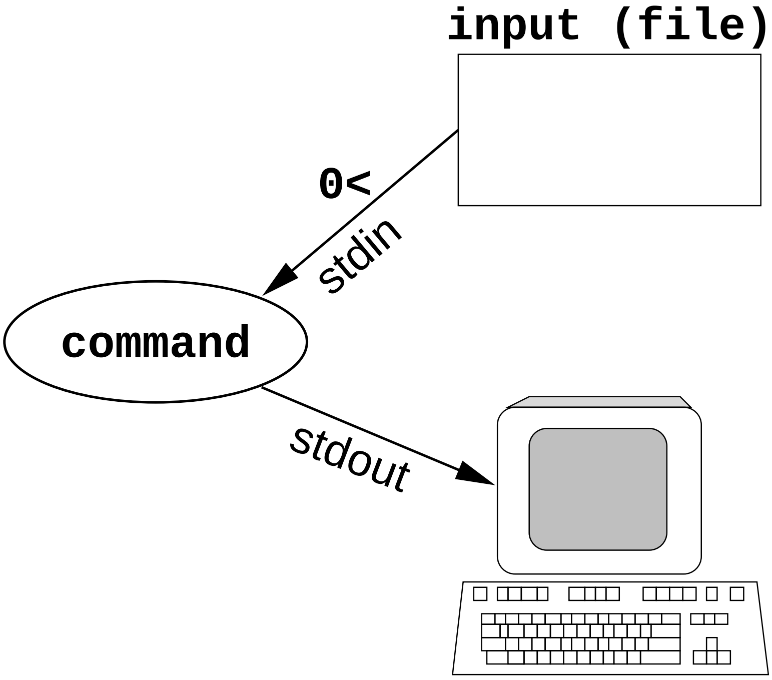


2.7 Input Redirection

- < redirects standard input from a file, e.g.

  ```
  $ command < input
  ```

- `command` will now take the contents of the file `input` as its input
- This could also be written as `0<`
- Consistent with > and `1>`



2.8 Combining Redirection

- Redirect more than one descriptor by giving more than one redirection, e.g.

  ```
  $ command 1> output 2> error
  ```

- Group redirections using the `>&` operator, e.g.

  ```
  $ command 1> output 2>&1
  ```

  - Output to the file called `output` (`> output`)
  - Send errors to the same place as the standard output (`2>&1`)

- The order of these is *very* important
- The redirections are evaluated left-to-right, e.g. the following differs from the previous example

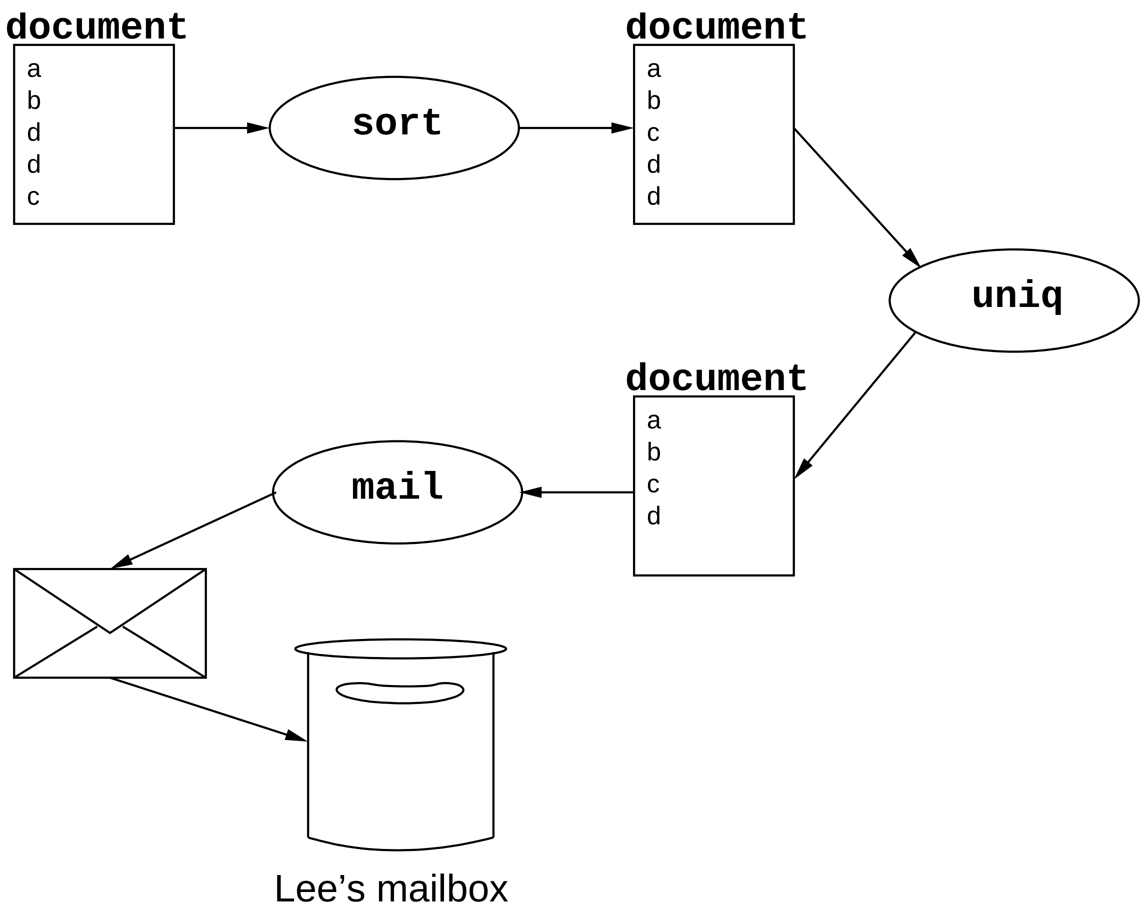
  ```
  $ command 2>&1 > output
  ```

  - It sends error to the normal output and normal output to the file called `output`

2.9 Pipelines

- You can output to another process with '`|`'
  - Known as the *pipe* symbol
- A *pipe* connects the output of one process to the input of another
- The data waiting to be transferred is buffered
- The processes run concurrently
- Linux ensures that the processes keep in step
- For example:

```
$ sort document | uniq | mail lee
```



2.10 Background Processes

- Most commands run to completion before you get your shell prompt back
- A ‘background’ process continues while you get your prompt back immediately
- To launch a process in the background place `&` at the end of the line, e.g.

  ```
  $ sort /var/log/maillog > output &
  ```

- Unless you use redirection (plumbing), output and error continue to appear on your terminal
- Input is disconnected, so typing goes to the shell, not to the background process
- If a process needs user input, and can’t take it from a file, it is ’stopped’
  - It won’t resume until brought to the foreground to receive input
- You should normally start background processes with their output and error redirected to a file, e.g.

  ```
  $ sort big_file > output 2> error_output &
  ```


2.11 Background Processes (continued)

- Running processes can be put in the background
  - Suspend the process by typing `^Z` in the terminal that the process is running in
  - Put the process in the background using `bg`
- Bring a process back to the foreground using `fg`
- `fg` and `bg` operate on the most recent process by default
  - Change to a process, by job number or name
- `jobs` displays current shell processes:

```
$ jobs
[1]+  Stopped (tty output)    top
$ fg %1
```


2.12 Background Processes and `nohup`

- Sometimes it is necessary to start a process and leave it running when you log out
- If your shell is killed, any background processes will also be lost
- `nohup` gets round this by detaching the process from the terminal
- Always redirect output and error with `nohup`, e.g.

```
$ nohup sort bigfile > out 2>&1 err.out
```

- If you don't redirect them then they will end up in `./nohup.out` and `./nohup.err`

2.13 Command Grouping and Sub-shells

- `bash` can execute multiple commands on a line
- Sequential commands are separated by ';'

```
$ sort data ; mail lee < sorted_data
```

- It's possible to launch a sub-shell to execute a command or group of commands
  - Put commands in parentheses, e.g.

    ```
    $ (command1 ; command2)
    ```

- Can also put a subshell in background, e.g.

```
$ (command1 ; command2) &
```


2.14 Process Management

- `ps` (process status) prints info about a users' processes:

```
PID   TTY STAT TIME COMMAND
22074 p0  S    0:02 Eterm -t trans
22075 p0  S    2:13 emacs -bg black
22081 p0  S    0:00 asclock
22590 p5  R    0:00 ps
```

- `jobs` only prints info about processes belonging to the current shell
- `wait` postpones shell until process is finished
  - Usually given a process id as an argument
  - If no argument is given it waits until all the shell's processes have terminated
- `kill` is used to send *signals* to processes
  - Can terminate background processes
- Some processes use signals to trigger tasks, e.g. log rotation, re-reading config files, etc

2.15 Signals

- `kill` can be given a signal name or number
- There are a variety of signals:

SIGHUP	1	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Interrupt from keyboard
SIGQUIT	3	Quit from keyboard
SIGKILL	9	Kill signal
SIGTERM	15	Termination signal
SIGUSR1	30 10 16	User-defined signal 1
SIGUSR2	31 12 17	User-defined signal 2

2.16 Signals (continued)

- Unless specified, `kill` sends a SIGTERM which causes most processes to terminate
- If a process is unresponsive, it can be forcibly killed by sending it SIGKILL

```
$ kill -9 1512
1512: Terminated
```

  or

```
$ kill -KILL 1512
1512: Terminated
```

- Can only signal your own processes
  - Superuser can signal all

2.17 Background Processes: `top`

- `top` displays the processes running on a machine
- Results can be sorted in various ways
- Options:
  - See `man top` for full details, including command-line options
  - Inside `top` use `h` for help on interactive options
- Typical output:

```
  PID USER    PRI  NI  SIZE  RSS SHARE STAT LIB %CPU %MEM  TIME COMMAND
22594 user     10   0   736  736   556 R      0  8.2  0.5  0:00 top
    1 root      0   0   144   96    76 S      0  0.0  0.0  0:03 init
    2 root      0   0     0    0     0 SW     0  0.0  0.0  0:19 kflushd
    3 root    -12 -12     0    0     0 SW<    0  0.0  0.0  2:42 kswapd
  486 root      5   5  3160 2356   804 S N    0  0.0  1.8  0:03 mysqld
  869 root      0   0    68   12    12 S      0  0.0  0.0  0:00 mingetty
  838 www       0   0 11468 6280   488 S      0  0.0  4.9  4:14 squid
   48 root      0   0   100   80    48 S      0  0.0  0.0  0:01 kerneld
  230 root      0   0   384  372   272 S      0  0.0  0.2  0:12 syslogd
  239 root      0   0   164  120    72 S      0  0.0  0.0  0:00 klogd
  250 daemon    0   0   164  132    88 S      0  0.0  0.1  0:00 atd
  261 root      0   0   192  160   112 S      0  0.0  0.1  0:01 crond
  272 bin       0   0   244  224   168 S      0  0.0  0.1  0:00 portmap
  283 root      0   0   572  296   248 S      0  0.0  0.2  0:17 snmpd
  295 root      1   0   136   88    60 S      0  0.0  0.0  0:00 inetd
  306 root      0   0   516  488   224 S      0  0.0  0.3  0:06 named
  317 root      0   0   124   56    48 S      0  0.0  0.0  0:00 lpd
```

- N.B. `top` is not available on all unices

2.18 Filename Generation

- Some characters are ‘special’ to the shell

Chars	Meaning
`*`	Matches any string, including the null string
`?`	Matches any single character
`[...]`	Matches any one of the enclosed characters. A pair of characters separated by a minus sign denotes a range. Any character lexically between those two characters, inclusive, is matched. If the first character following the ‘`[`’ is a ‘`!`’ or a ‘`^`’ then any character not enclosed is matched. A ‘`-`’ or ‘`]`’ may be matched by including it as the first or last character in the set.

Table 2.1: Special characters under `bash`

- Special characters can be used to match filenames, e.g. to show files beginning with `f`

```
$ echo f*
```

- To show files starting with ‘`f`’, followed by a vowel:

```
$ echo f[aeiou]*
```


2.19 Quoting Mechanisms

- Sometimes it's necessary to ignore a character's special meaning
- Use a backslash (\) to quote a special character, e.g. to list a file called `f*`

```
$ ls f
```

- To quote a longer string, enclose it in quotes:

`'`	disable all interpretation
`"`	disable filename generation and blank space interpretation

2.20 Shell built-in commands

- Some commands must be built in to the shell, because they can't be executed independently
  - `cd`, if executed independently would change its own directory, *not* that of your shell
  - `umask`, would change its umask, *not* the shell's
  - `logout`
  - `history`
- Other commands are built in for speed e.g.
  - `pwd`
  - `echo`

2.21 Basic Shell Exercises

What should I focus on?

You should:

- clearly understand what redirection is, and what a pipe is
- understand that it is the *shell* that expands the filename generation symbols (`*`, `?` and `[...]`), and not `echo`, `ls`,...
- understand the differences between these three ways of quoting: `'...'`, `"..."`, and quoting a single character with '\'
- understand how to put a process into the background, bring it to the foreground, and how to interrupt it (*not* with `Control-z`!)
- understand the difference between using `Control-c` and `Control-z`
- understand that subshells have their own working directories.

The exercises

1. *Redirection*

(a) Try typing the following commands exactly as they appear here. Note that the `cat` command copies input files (or standard input) to standard output. If standard input comes from the keyboard, the keystroke Control-d means "end of file" (like Control-z in Windows). Refer to sections 2.6 on page 37 to § 2.9 on page 40 first. Also refer to section 2.13 on page 44 before doing the last exercise.

```
$ cat
# Then type some words, followed by Control-d
$ cat > newfile 2> newfile.error
$ car > newfile 2> newfile.error
$ car
$ Space Shuttle
$ echo
$ echo foo
$ ech
$ ech foo
$ cat > newfile 2>&1
$ car > newfile 2>&1
$ cat < newfile
$ echo foo | cat > newfile 2>&1
$ ech foo | cat > newfile 2>&1
$ echo foo | car > newfile 2>&1
$ (ech foo | cat) > newfile 2>&1
```

After each step, examine the contents of `newfile` and `newfile.error` by typing:

```
$ cat newfile
$ cat newfile.error
```

Make sure you understand what happens in each case, ask the tutor if you are not sure.

2. *Filename expansion and Quoting*

(a) Do the following in the `/bin` directory; refer to section 2.18 on page 49 first.

i. List all filenames with exactly three characters.

ii. List all filenames with exactly three characters in which the second character is a vowel.

iii. List all filenames with a, b, c, or d as the last character.

iv. Construct a command to print the number of filenames consisting of exactly three characters. (You may combine commands together in a pipeline. You may find the wc utility useful here; check `man wc` for more information.)

v. Construct a command to print the total number of files with exactly two, three or four characters in their name. (Again, you may find the wc utility useful.)

(b) Compare the effect of the following commands. Refer to section 2.19 on page 50 first. Then search for “quoting” in the **bash** man page.

```
$ echo $HOME
$ echo "$HOME"
$ echo '$HOME'
$ echo *
$ echo "*"
$ echo '*'
$ echo $HOME/bs*
$ echo "$HOME/bs*"
$ echo '$HOME/bs*'
```

(c) Change back to your home directory and try to create a file with the name `*`. Was this a sensible thing to do? How would you delete it? (Be *very* careful!)

(d) Create a file called `--file`. Try to remove this. Use the `rm man` page to help you.

3. *Background processes and `nohup`*

Refer to section § 2.10 on page 41 to § 2.12 on page 43 first.

(a) Start the command `sort /dev/random` in the background in your current shell

(b) Bring it back to the foreground and terminate it by typing Control-c

(c) Start it again, and once more so that you have two copies running in the background

(d) Bring them to the foreground and terminate them in the order you started them

(e) Start the same command in the background, and terminate it using `kill`

4. *Grouped commands*

Compare the following command sequences, and make sure you understand the differences. Refer to section § 2.13 on page 44 first.

(a)
```
$ cd /tmp
$ cd /usr; ls
$ pwd
```

(b)
```
$ cd /tmp
$ (cd /usr; ls)
$ pwd
```

(c) `$ sleep 5 & sleep 5`

(d) `$ (sleep 5; sleep 5) &`

Check you can use your history to get at and repeat any of the commands you have typed.

2.22 Basic Shell Solutions

Module 3

Basic Tools

Objectives

At the end of this section, you will be able to:

- Use the most frequently used Linux tools to:
  - Find files
  - Get information about commands
  - View file contents
  - Get information about files
  - Operate on file contents
  - Do simple text manipulation
  - Schedule jobs
- Combine tools to solve problems
- Understand and use the Linux printing subsystem

3.1 Introduction

The basic Linux command-line utilities dealt with here, are:

Finding files:	`find` `locate`
Getting info about commands:	`man`

Viewing file contents:	`cat`
	`less`
	`head/tail`

Getting information about a file:	`ls`
	`file`
	`wc`
	`diff`
	`cmp`

Operating on file contents:	`grep`
	`sort`
	`uniq`
	`split/csplit`
	`cut`
	`paste`
	`tar`
	`gzip/bzip2`

Simple text manipulation:	`tr` `expr`
Scheduling jobs:	`at` `crontab`

Table 3.1: Basic Linux utilities

3.2 Using Tools

- Typical Linux systems contain over 1000 command-line tools
- Tools are combined (via pipes and redirection) to solve specific problems
- Most tools have a standard syntax:

  $ command [*options*] [*files ...*]

- Some arguments must be quoted
- Standard input often read if no filename given
- Most tools can take several filename arguments
- Desktop/windowing environments may provide graphical wrappers to some tools
- Serious Linux administrators and users know the key command-lines well
- The terms ‘command’ and ‘tool’ are used interchangeably here

3.3 The On-Line Manual (`man`)

- Most commands have an associated man page
- Accessed by typing:

  `$ man` *`command`*`[s]`
- Brings up a page of information usually detailing:
  - command name, section number, description
  - syntax
  - options
  - version information
  - location of configuration files
  - other related commands
  - examples of usage
  - known bugs (if any . . .)

3.4 Finding Files the Long Way (`find`)

- `find` searches the filesystem in real time;
  - Makes disks work hard
- Can find files by name, type, size, dates, e.g
  - To find all files ending with `.jpg` under the current directory:
    `find .  -name "*.jpg"`
  - To find all filenames ending in `.jpg` *and* modified in the last 8 days below `/etc`
    `find /etc -name "*.jpg" -mtime -8`
- Tests can be combined with `-o` and negated with `!`, for example:
  - To find all filenames *not* ending in `.jpg` *or* files which were modified in the last 8 days under `/etc`
    `find /etc \!  -name "*.jpg" -o -mtime -8`
- Can execute commands on the files it finds. The name of the file found is placed in {} *

```
find . -name "*.gif" -exec ls -l {} \;
```

*This is not a resource friendly way of doing things. `xargs` may be better

3.5 Find examples

- First, an example which is wrong:

  ```
  $ find /usr/bin/c* # WRONG!
  ```

  This is *wrong* because the shell expands the `c*` to all the files before `find` even begins.

- Here is the right way to find files beginning with "`c`" under the `/usr/bin` directory:

  ```
  $ find /usr/bin -name c\*
  ```

  Note that we need to quote the "`*`", because we want `find` to have the star, we don't want the shell to turn it into a list of files starting with "`c`" in the current directory.

- Find man pages that are bigger than 10 k:

  ```
  $ find /usr/share/man -size +10k
  ```

- Find man pages that are bigger than 10 k but smaller than 15 k:

  ```
  $ find /usr/share/man -size +10k -size -15k
  ```

- Find man pages that are smaller than 10 k OR bigger than 15 k:

  ```
  $ find /usr/share/man -size -10k -o -size +15k
  ```


We use the logical OR operator “-o”

- Find man pages that are smaller than 10 k but bigger than 15 k and which are ordinary files (not directories or symbolic links):

```
$ find /usr/share/man -type f \( -size -10k -o -size +15k \)
```

We need to use parentheses to group the OR operation first, otherwise it will test for files that are smaller than 10 k and which are normal files, OR for *any* files that are bigger than 15 k. The parentheses must be quoted or the shell will try to start a shubshell.

3.6 Locate Files (`locate`)

- `locate` searches a periodically-updated database of the filesystem(s)
  - Not available on all systems
  - Very fast, but DB needs regular updating, e.g.

    ```
    $ updatedb
    ```
  - Usually updated nightly automatically
- Won't be there on a fresh install
- Won't show files created since last database update
- Given the command '`locate` *`string`*', `locate` will show all files containing *string* in their full pathname, e.g.

  ```
  $ locate logrotate
  /usr/man/man8/logrotate.8
  /usr/sbin/logrotate
  /etc/logrotate.d
  /etc/logrotate.d/cron
  /etc/logrotate.d/syslog
  /etc/logrotate.d/linuxconf
  /etc/logrotate.d/ftpd
  /etc/logrotate.d/samba
  /etc/cron.daily/logrotate
  /etc/logrotate.conf
  ```


3.7 View and Concatenate Files (`cat`)

- Displays and/or joins (con$\boxed{\text{cat}}$enates) files
- Sends the content of named file(s) to standard output
- If no filename is given, it reads from standard input and writes to standard output
- Given more than one filename, it displays each file's contents sequentially, i.e, joins them
- Example:

```
$ cat file1 file2 file3 > all_files
```

3.8 View Large Files & Output (`less`)

- `less` displays the contents of file(s) in a controlled way on stdout
  - Usually, one page at a time
  - Like UNIX/DOS command `more`, on steroids
- You can search for patterns in the file
- It allows you to move quickly to *any* point (backwards or forwards)
- Similar usage to `more`, `vi`, and `lynx`:

Action	Keystokes
Top of page	`g < ESC-<`
Bottom of page	`G > ESC->`
Forward one screen	`f ^F ^V SPACE`
Backward one screen	`b ^B ESC-v`
Up one line	`y ^Y k ^K ^P`
Down one line	`e ^E j ^N RETURN`
pattern Search forward	`/`*pattern*
pattern Search backward	`?`*pattern*
Repeat *pattern* Search forward	`n`
Repeat *pattern* Search backward	`N`
Move to *n*th line	*n*`g`
!command	Execute the shell command with $SHELL
—Xcommand	Pipe file between current pos & mark X to shell command
v	Edit the current file with $VISUAL or $EDITOR

Table 3.2: Commands within `less`

3.9 Viewing Parts of Files (`head` and `tail`)

- `head` displays the first few lines of a file
- `tail` displays the last few lines of a file
- You can specify how many lines are displayed
  - To display only the first 4 lines:
    `$ head -4` *`filename`*
- `tail -f` often used to monitor growing files, e.g.,

```
$ sudo tail -f /var/log/messages
```

3.10 Listing File Information (`ls`)

- Without any options, `ls` lists files in the current directory
- By default all files starting with `.` (dot) aren't shown
- The most common options to `ls` include:

Flag	Option
`-l`	*Long* (detailed) listing of file info, including: size, ownership, permissions and type
`-a`	Show *all* files, including hidden ones
`-F`	Highlight directories and executables with / and `@` respectively
`-R`	*Recursively* list subdirectories
`-t`	Sort list by last modification *time*
`-u`	Sort list by last access time (with `-t`)
`-X`	Sort list by file e*X*tension
`-r`	Reverse order of listing
`-d`	Show directory information not directory contents

Table 3.3: Common options to `ls`

- For example:

```
$ ls -lrt
```

show files in reverse order based on their modification time

3.11 File Classification (`file`)

- `file` displays the type of data contained in named file(s)
- It works by:
  1. opening each named file
  2. reading the beginning of the file
  3. comparing what it finds there with the patterns in the "magic" file `/usr/share/magic`
  4. reports the matching file type found in the magic file.
- Results not always correct
- Classifications include: executable, archive, C program, ASCII text, JPEG image ...
- You can use it like this:

```
$ file /usr/bin/*
```

3.12 Count Words, Lines, Characters (`wc`)

- `wc` displays the number of lines, words,* and characters in a file

Flag	Option
`-l`	Only displays the number of lines
`-w`	Only displays the number of 'words'
`-c`	Only displays the number of characters

Table 3.4: Options to the `wc` command

*A 'word', in this context, is a character string surrounded by SPACEs, TABs, NEWLINEs, or a combination of them.

3.13 Differences Between Files (`diff`)

- `diff` displays the difference between two *text* files, line-by-line
- Output from `diff` can be confusing
- For example, given the files *text1* and *text2*:

  text1:
  This is a temprary test
  to check the diff
  utility

  text2:
  This is a temporary test
  to check the diff
  utility.

1. A simple line-by-line comparison:

```
$ diff text1 text2
1c1
< This is a temprary test
---
> This is a temporary test
3c3
< utility
---
> utility.
```

2. Using the context output format (-c):

```
$ diff -c text1 text2
*** text1       Mon Apr 19 14:46:25 1999
--- text2       Mon Apr 19 14:46:05 1999
***************
*** 1,3 ****
! This is a temprary test
  to check the diff
! utility
--- 1,3 ----
! This is a temporary test
  to check the diff
! utility.
```

3. Using the unified output format (-u): (Most common)

```
$ diff -u text1 text2
--- text1       Mon Apr 19 14:46:25 1999
+++ text2       Mon Apr 19 14:46:05 1999
@@ -1,3 +1,3 @@
-This is a temprary test
+This is a temporary test
 to check the diff
-utility
+utility.
```


3.14 Compare Binary Files (`cmp`)

- Displays differences between 2 *binary* files
- Locates the byte and line number of the first difference
- Can show *all* differences if required, e.g.

  `cmp -l` *`file1 file2`*
- `cmp -s` suppresses output and returns exit status
  - `0` if the files are identical
  - `1` if the files differ
  - `2` if an error has occurred
- Often used in shell scripts

3.15 Regular Expression Searches (grep)

- Search for regular expressions in file(s)

  i.e "g lobally find r egular e xpressions and p rint"

- Usage:
  `grep [options] search-pattern files`
- Reads standard input if no filenames are given
- Matching lines are printed to standard output
- Popular options:

Flag	Option
-i	Ignore case
-l	List only filenames containing the expression
-v	Reverse sense of test, i.e. find non-matching lines
-w	Word search, i.e. match whole word
-E	Extended regular expression search (more complex patterns), egrep similar
-F	Fixed string pattern search, same as fgrep

Table 3.5: Popular grep options

3.16 grep examples

- Search for a user in the password file

```
$ grep lee /etc/passwd
lee:x:500:500:Lee Willis:/home/lee:/bin/bash
```

- Search for events in a log*

```
# grep connect /var/log/secure
Mar  8 14:42:53 rafters in.telnetd[579]: connect from 192.168.0.129
Mar 13 10:27:40 rafters in.telnetd[724]: connect from 192.168.0.139
Mar 13 16:54:24 rafters in.telnetd[2135]: connect from 192.168.0.139
Mar 20 10:54:38 rafters in.telnetd[816]: connect from 127.0.0.1
```

- Search for a function in source code

```
$ grep "int main" *.cpp
checkrefint.cpp:int mainbit();
checkrefint.cpp:int mainbit(){
do_maint.cpp:int maint(Form &Myform){
expire.cpp:int main(void){
maint_login.cpp:int maint_login(Form &) {
```

*You need to be logged in as root to see inside /var/log/secure.

3.17 Sort and Merge Files (sort)

- sort and/or merge files
- Acts as a filter without file arguments
- Sorts entire lines lexically, by default
- Alternative sort orders:

Flag	Option
-n	Numerical order
-r	Reverse order

Table 3.6: Alternative sort orders

- Other popular options:

Flag	Option
-b	Blanks (TAB, SPACE) ignored
-f	Fold lowercase to upper before sorting
-i	Ignore non-printable characters
-m	Merge files, without checking if sorted
-t*x*	Set field delimiter in file as *x*
-u	Unique, outputs repeat lines once only
-k *POS1[,POS2]*	Specify a field in each line as a sorting key, starting at *POS1* and ending at *POS2* (or NEWLINE). Fields and character positions are numbered starting at 1

Table 3.7: Popular sort options

3.18 `sort` Examples

Consider `/etc/passwd` which typically contains lines in the following format:

`username:password:UID:GID:Realname:Homedirectory:shell`

- To sort by username:
  `$ sort -t:  -f -k1,1 /etc/passwd`
- To sort numerically by user ID:
  `$ sort -t:  -n -k3,3 /etc/passwd`
- To sort by real name within group ID:
  `$ sort -t:  -k4,4n -k5,5f /etc/passwd`

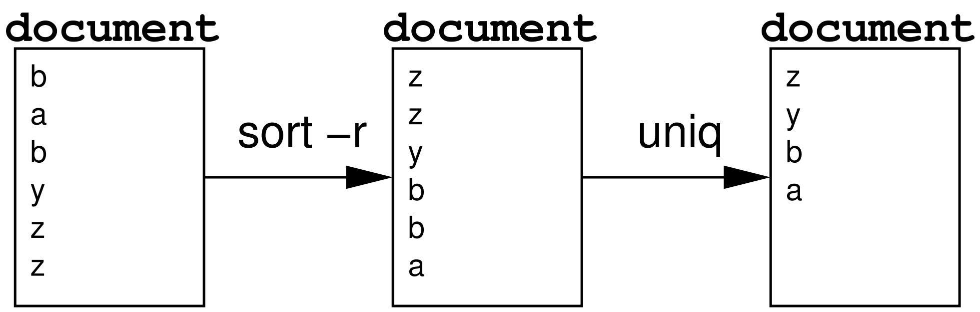
3.19 Display Unique Lines (`uniq`)

- Removes all but one of successively repeated lines
- Acts on standard input, often piped from `sort`
- Most popular options:

Flag	Option
`-c`	Put a count of how many lines were duplicate in front of each line
`-d`	Duplicated lines only are displayed
`-u`	Unique lines only are displayed
`-`*n*	Ignore the first *n* fields
`+`*n*	Ignore the first *n* characters
`-w`	Specify the number of chars to compare

Table 3.8: Popular `uniq` options

- Example:



3.20 Split Files (`split`)

- Split a file into pieces
- Outputs sections to new files or standard output
- Creates files named `prefixaa`, `prefixab`, `prefixac` ...
- Main `split` options:

Flag	Option
`-l`*n*	Put *n* lines of the input file into each output file
`-b`*n*	Put *n* bytes of the input file into each output file
`-C`*n*	Put as many complete lines of the input file as is possible into the output file, up to *n* bytes

Table 3.9: Main `split` options

3.21 Splitting Files by Context (`csplit`)

- Splits file into sections determined by context (patterns or regular extressions)
- Syntax:

```
csplit [-f prefix] [-b suffix] [-n digits] filename pattern...
```

- Main `csplit` arguments:

Argument	Instruction
/regexp/[offset]	Split the file at occurrence of *regexp*. The line after the optional offset ('+' or '-' followed by a number) begins next bit of input
{*repeat-count*}	Repeat the previous pattern split *n* times. Substitute an asterisk for *n* to repeat until the input is exausted
-f*string*	Use *string* as prefix of output filename
-b*string*	Use *string* as suffix of output filename
-n*n*	Use output filenames *n* digits long

Table 3.10: Main `csplit` arguments

3.22 Dividing files into columns: `cut`

- Often need to separate one or more columns from input.
  - `cut` provides a simple way;
  - `awk` provides another way (more flexible; see later).
  - Use `paste` to join columns together
- Syntax: `cut` ⟨*options*⟩ [⟨*files*⟩]
- Cuts out the selected columns or fields from one or more ⟨*files*⟩.
- Can specify a range of values with a hyphen, and use a comma to separate values, e.g., `1-10,15,20`, or `50-`
- Options:

  `-b`, `--bytes` ⟨*list*⟩ Specify ⟨*list*⟩ of positions; only bytes in those positions will be printed.

  `-c`, `--characters` ⟨*list*⟩ Cut the column positions given in ⟨*list*⟩

  `-d`, `--delimiter` ⟨*c*⟩ Use `-f` to specify field delimiter as character ⟨*c*⟩ (default is tab); special characters must be quoted.

  `-f`, `--fields` ⟨*list*⟩ Cut the fields given in ⟨*list*⟩

- Examples:
  - Extract usernames and real names from `/etc/passwd`:

    ```
    cut -d: -f1,5 /etc/passwd
    ```

  - Find out who is logged in, but only login names:


```
who | cut -d" " -f1
```

- Cut characters from the fourth column of `file` and paste them back as the first column in the same file:

```
cut -c4 file | paste - file
```


3.23 Compression Utilities: `bzip2` and `gzip`

- Linux has many compression utilities; `bzip2` and `gzip` dominate thanks to integration with `tar`, and excellent compression
- Takes files or standard input and compress them to file(s) or standard output
- Uses *lossless compression*, so safe on any file
- Key `bzip2` and `gzip` options:

Flag	Option
`-r`	Recursive compression of subdirectories
`-d`	Decompress (same as `gunzip`)
`-`*[1-9]*	Fast or best compression, where 1 is fastest and 9 is most intense compression

Table 3.11: Key `bzip2` and `gzip` options

- `bzip2` has significantly better compression ratios and is now widely used for distributing software.
- A set of files put into an archive with `tar` and `bzip2` gives *much* better compression than can be achieved in zip files or rar archives.

3.24 Store and Retrieve Archives (tar)

- Originally designed to make tape archives
- Takes a group of files and creates one big file containing their contents *and* details
- Widely used for:
  - Compression with gzip or bzip2 *
  - Maintaining Linux file details (permissions, dates, ownership etc) on other filesystems
  - Bundling file trees for distribution (called a *tarball*)
- Key *tar* options:

Flag	Option
-t	List files in an archive
-x	Extract the contents from an archive
-c	Create a new archive
-r	Append files to an existing archive
-v	Verbose: list file names, tell more of what's happening
-j	Create/Open bzip2 compressed tar file(s)
-z	Create/Open gzip compressed tar file(s)
-f	Filename of the file or device to hold the archive
-p	preserve permissions when extracting files

Table 3.12: Key tar options

- To create a gzip compressed archive of /etc:

```
$ tar -cvzf /home/username/filename.tar.gz /etc
```

*The gzip or bzip2 compression options are not available on all UNIX versions of tar.

- To extract the files into the current directory:

```
$ tar -xvzf filename.tar.bz2
```

- To create a `bzip2` compressed archive of `/etc`:

```
$ tar -cvjf /home/username/filename.tar.gz /etc
```

- To extract the files into the current directory:

```
$ tar -xvjf filename.tar.bz2
```

3.25 Translating Characters (tr)

- Translate characters in standard input into different characters in output
- Syntax:

  `tr [options] [`*string1* `[`*string2*`]]`
- Characters in *string1* are replaced by the corresponding character in *string2*

```
$ tr 'abc' '123'
abcdad
123d1d
```

- Character position in both strings matters
- Both strings should be the same length *

*If string1 is shorter than string2, the extra characters at the end of string2 are ignored. If string1 is longer, GNU tr follows BSD in padding string2 to the length of string1, by repeating the last character. With the -t option it follows AT&T by truncating string1 to the length of string2.

3.26 Examples of `tr` Usage

- To replace all vowels in the input with spaces

```
$ tr 'aeiou' '     '
```

- Using character ranges to translate all lower case letters into their upper case equivalents

```
$ tr 'a-z' 'A-Z'
```

- Using the asterisk `*` to replace all 10 digits with the same letter 'n'. In other words, `[n*10]` is the same as `'nnnnnnnnnn'`

```
$ tr '0-9' '[n*10]'
```

- Use the `-c` option (complement) to replace all characters in *string1* which *don't* belong in a range

```
$ tr -cs 'a-zA-Z0-9' '[\n*]'
```

  N.B. This puts every word on a line by itself, by converting all non-alphanumeric characters to newlines, then 'squeezing' repeated newlines (with `-s`) into a single newline.

  The pattern `[\n*]` in the second set means repeat the newline `\n` as many times as there are characters in the first set. See `man tr`.

3.27 Execute programs at specified times (at)

- `at` executes a shell script at a specified time
- Syntax:

  $ at [*options*] *time* [*date*] +*increment*
- `stdin` is scanned for commands to execute, e.g.

```
at 2300
at> fetchmail
<CTRL>+D
```

- Some more examples of how to specify time and date:

```
at 11pm
at 1am
at 5am tomorrow
at 08:00 Sep 12
at now
at now + 4 hours
at now + 1 day
at now + 2 months
at now + 3 years
at 5:30pm Thursday
at teatime
```

- Commands are executed in the current environment at the given time
- `stdout` and `stderr` are sent as mail

3.28 Options and commands related to `at`

- `at` belongs to a family of utilities for managing time-specified commands

Command	Purpose
`atq`	Display list of queued `at` commands
`atrm`	Remove queued `at` commands
`batch`	Schedule jobs at low CPU loading

- All of these can be run as `at` options:

Option	Purpose
`-l`	Display list of queued `at` commands
`-d`	Remove queued `at` commands
`-b`	Schedule jobs at low CPU loading
`-f`*`file`*	Specify script file in command-line
`-m`	Send mail after running `at`, whatever the `stdout` or `stderr`

- The use of `at` is controlled by `/etc/at.allow` and `/etc/at.deny` *

*UNIX NOTE: On System V these live in `/usr/lib/cron/`

3.29 Running commands regularly (crontab)

- crontab lets you submit job lists at regular times using the crond daemon
- 2 syntax formats:

```
$ crontab filename
$ crontab [-u username] [options]
```

- Options, etc:

Command	Purpose
crontab *myfile*	Install contents of *myfile* (stdin if no file specified) in appropriate directory
crontab -r	Remove the crontab for the current user
crontab -l	List (on stdout) current user's crontab. (might be useful for editing a cron table)
crontab -e	Run a text editor on your crontab file

Table 3.13: crontab usage

- Examples of Crontab Entries

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
# LW poll client1  for mail
0 10,12,16,18 * * 1-5 root /www/bin/client1_poll
LW 16/09/1998 - Hylafax cron stuff ...
0 0 * * *  root /usr/local/sbin/faxqclean
10 0 * * * fax /usr/local/sbin/faxcron -info 7
[0-59/10] 9-18 * * 1-5 root /usr/local/bin/domail
```


3.30 Evaluate expressions (`expr`)

- `expr` is used to evaluate expressions, like `bash` arithmetic expressions, e.g., `echo $((1 + 2))`
- Takes arguments and operators on the command line
- Prints the result
- Also returns true or false depending on the result; can be tested with shell—see example with the `while` loop below.
- Watch out for special meaning to shell of characters like ∗ and < or >

```
$ expr 1 + 2
3
$ expr 6 \* 7 #must escape the '*'
42
$ expr 5 + 10 / 2
10
$ expr \( 5 + 10 \) / 2
7
$ i=10
$ i=`expr $i + 1`
$ echo $i
11
```

`expr` prints 1 for true comparisons, 0 for false.

- Has some string manipulation facilities too:

```
$ expr index abcdefg d
4
$ expr substr abcdefghijk 3 3
cde
```


- Sometimes used in shell scripts for looping; `expr` returns false if the expression is null or 0, and true otherwise:

```
i=0
while expr $i \< 20 >/dev/null
do
    echo $i
    i=$(expr $i + 1)
done
```

- See `info expr` for details.

3.31 Linux Printing

- Two main printing sytems:
  - Common Unix Printing System (CUPS) `http://www.cups.org/`
  - LPRng `http://www.lprng.com/`
- Both currently provided on Red Hat Linux 8.0 and 9
- Only CUPS provides on Fedora Core 1+.
  - Can select either system with `redhat-switch-printer` command
- Expect CUPS to be more prevalent in future

3.32 LPRng and CUPS

- Completely network-oriented
- Any printer can be made available to any client (machine and application)
- All print jobs are sent to a queue
- Queues can be viewed, edited, maintained from anywhere
  - Subject to permission
- Formatted files can be sent straight to queues
- Both CUPS and LPRng both share common printing commands

3.33 Main Printing Tools

- `lpr` sends job to the queue for a named printer
- `lpq` returns info about jobs in a queue
- `lprm` removes unwanted jobs from a queue
- `lpc` enables system administrator to control the operation of the printing system
  - see `man lpc` for details
- Desktop environments may offer “drag ‘n’ drop”, visual facilities, etc

3.34 Using `lpr`

- Syntax:

```
lpr [options] file ...
```

- Main Options:

Flag	Options
`-P`	Name of the printer to send the job to
-n	Print *n* copies of the document
`-m`	Send mail on completion

Table 3.14: Main `lpr` options

- Example:

```
$ lpr -Pdjrmt -2 filetypes.txt
```

3.35 Using `lpq`

- Syntax:

  `lpq [options]`

- Options:

Flag	Options
`-P`	Name of the printer/queue to interrogate
`-l`	Get info on each file within a job

Table 3.15: `lpq` options

- Example:

```
$ lpq -Pdjrmt -l
```


3.36 Using `lprm`

- Syntax:

  `lprm [options]`

- Options:

Flag	Options
`-P`	Remove jobs from named printer/queue
`-`	Remove all jobs belonging to yourself
user	Remove all jobs belonging to *user*
n	Remove job number *n*

Table 3.16: `lprm` options

- Example:

  `$ lprm -Pdjrmt -davef`

3.37 Printing Information

- This is a bare introduction
- See chapter 8 of the *Red Hat Getting Started Guide* (for Red hat 9), and
- chapter 27 of the *Red Hat Customization Guide*
- All these are available on the documentation CDROM, or can be downloaded from `http://www.redhat.com/docs/manuals/linux/`.

3.38 Basic Tools Exercises

In these exercises, you will combine simple tools together using *pipes*. Do this one step at a time, and check the results as you go.

1. *Find and Locate Files*
   Use first `find` *then* `locate` to do each of the following. First read § 3.4 on page 59 to § 3.6 on page 62, especially § 3.5 on page 60. Also, please read § 3.25 on page 84 and § 3.26 on page 85 before doing part 1c.
   (a) Display all the filenames under `/usr/sbin`.
   (b) Display all the filenames under `/usr/sbin` begining with a lowercase '`c`'.
   (c) Repeat the previous question, but *translate* the output to uppercase.
   (d) Display all the files under `/usr/sbin` which are over 5k in size in uppercase.
2. *Display Parts of Files*
   Note that the file `mime.types` lists the standard names used for different file types in both http and email. Before sending any data, a web server sends a header which includes the *mime type*. That is how the web browser knows what to do with the data (i.e., how to display it or play it). Similarly, email containing attachments puts a mime type header before each attachment. That is how the email client knows what to do with the attachment (such as to automatically execute it and infect the machine with a worm).
   (a) Display the first 10 lines of the file `/etc/mime.types`
   (b) Display the last 10 lines of `/etc/mime.types`
   (c) Display the first 25 lines of `/etc/mime.types`
   (d) Display `/etc/mime.types` one screen at a time
   (e) While viewing `/etc/mime.types` page-by-page, search for 'html'
3. *Classify, Count and Compare Files*
   (a) Find out what file types you have in the following directories and below. Please read § 3.11 on page 67. Please also see the example on the bottom of § 3.4 on page 59.
      i. `/etc`
      ii. `/usr/bin`
   (b) Repeat the previous question, but this time:
      i. Re-direct `/etc` listing to new file `filetypes.txt`
      ii. Append the listing for `/usr/bin` to `filetypes.txt`. Please refer to § 2.6 on page 37.
   (c) Build a tool (i.e. write a command) to find out how many files are in the `/usr/bin` directory. Hint: consider using `wc`.
   (d) Create two new files from listings of 2 user's home directories, then find the differences between them.
   To do this, you could:
   - Find a neighbour you can work with, and open a terminal window on their computer
   - In that terminal window, type:
     `$ su - 012345678`
     (assuming your user ID is 012345678). Of course, you need to enter your password.

- Make a list of files from your home directory into `/tmp`, making sure that the file names do not have `/home/012345678` in front of them, or otherwise your files will all seem to have different names!
- Compare your file list with that generated by your neighbour, who should also put their file list into `/tmp`.

(e) Use `diff` and `patch` to make the two files created in the last question identical. (Check `man patch`)

4. *`grep`*

Use the `filetypes.txt` file that you created before, and do the following. Please read § 3.15 on page 72 and § 3.16 on page 73 first.

(a) List all the lines that contain ‘directory’

(b) List all the lines that don’t contain ‘directory’.

(c) Find out how many files *are* directories, then find out how many aren’t.

(d) Why does the following give an error message (try redirecting the output to `/dev/null` so you can see the error).

```
$ grep ASCII text filetypes.txt
```

(e) Find out how many `English text` files are listed in the `filetypes.txt` file.

5. *Sorting*

(a) Sort the `filetypes.txt` file into reverse alphabetical order on the first field. Please see § 3.17 on page 74 and § 3.18 on page 75 first.

You may notice that capital and lowercase letters are sorted independently, e.g. ‘`A`’ comes before ‘`a`’.

(b) Repeat the first sorting exercise but ignoring case differences

(c) Sort the `filetypes.txt` files into alphabetical order on the second field (the file type).

3.39 Basic Tools Solutions

Module 4

More Tools

Objectives

Having completed this module you should be able use the following tools appropriately:

- `top`
- `ps`
- `find`
- `vmstat`
- `free`
- `ldd`
- `uptime`
- `xargs`
- `cpio`
- `tar`
- `gzip`

4.1 Introduction

Tools covered in this module have these functions:

Command	Function
`top`	display top CPU processes
`ps`	display process status
`find`	find files in a directory hierarchy
`vmstat`	display virtual memory statistics
`free`	display free and used memory
`ldd`	display shared library dependencies
`uptime`	display system uptime
`xargs`	build and exec commands from stdin
`cpio`	copy files to and from archives
`tar`	create and extract `*.tar` archive files
`gzip`	create and extract `*.gz` archive files

Table 4.1: More tools and their functions

4.2 Displaying System Processes (`top`)

- Displays ongoing processor activity in real time
- Shows processes for *all* users, unlike `ps`
- Has several modes:
  - Secure Mode, disables potentially dangerous interactive commands
  - Cummulative Mode, shows time for a process *and* its dead children
  - No-idle Mode, ignores idle or zombie processes
- Typical output may be:

```
 12:37:53  up 11 days,  2:48, 13 users,  load average: 1.71, 1.35, 0.76
143 processes: 137 sleeping, 3 running, 3 zombie, 0 stopped
CPU states:  cpu    user    nice  system    irq  softirq  iowait    idle
           total   12.1%    0.0%    4.6%   0.0%     0.0%    0.0%   83.1%
Mem:   255616k av,  251200k used,    4416k free,       0k shrd,   10148k buff
        17452k active,            206676k inactive
Swap: 1024120k av,  397956k used,  626164k free                   81720k cached

  PID USER      PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME CPU COMMAND
 4621 nicku      18   0 10164 9.9M   688 R     8.3  3.9   1:22   0 rsync
 5519 nicku      17   0  1248 1248   848 R     3.7  0.4   0:00   0 top
 1466 root       15   0 19552 3172  2636 S     2.7  1.2   1:03   0 spamd
    1 root       16   0   492  464   440 S     0.0  0.1   0:14   0 init
    2 root       15   0     0    0     0 SW    0.0  0.0   0:01   0 keventd
    3 root       15   0     0    0     0 SW    0.0  0.0   0:00   0 kapmd
    4 root       34  19     0    0     0 SWN   0.0  0.0   0:00   0 ksoftirqd/0
    6 root       25   0     0    0     0 SW    0.0  0.0   0:00   0 bdflush
    5 root       15   0     0    0     0 SW    0.0  0.0   3:35   0 kswapd
    7 root       15   0     0    0     0 SW    0.0  0.0   1:06   0 kupdated
    8 root       21   0     0    0     0 SW    0.0  0.0   0:00   0 mdrecoveryd
   13 root       15   0     0    0     0 SW    0.0  0.0   1:13   0 kjournald
   63 root       15   0     0    0     0 SW    0.0  0.0   0:00   0 khubd
  786 root       15   0     0    0     0 SW    0.0  0.0   0:02   0 kreiserfsd
 1097 root       15   0   584  564   524 S     0.0  0.2   0:33   0 syslogd
 1101 root       16   0   440  380   380 S     0.0  0.1   0:00   0 klogd
 1110 rpc        16   0   604  580   580 S     0.0  0.2   0:00   0 portmap
 1129 rpcuser    18   0   644  568   568 S     0.0  0.2   0:00   0 rpc.statd
 1182 root       15   0   496  444   444 S     0.0  0.1   0:00   0 apmd
 1234 root       16   0   580  516   496 S     0.0  0.2   0:00   0 automount
 1271 root       15   0   832  612   612 S     0.0  0.2   0:05   0 sshd
```

4.3 Options and Interactive Commands for top

Significant command-line options for top include:

Option	Function
-d	delay between screen updates (seconds)
-q	Refresh without any delay.
-S	Specifies cumulative mode
-s	Secure mode
-i	Non-idle mode
-c	Show full *command line* instead of command name

Table 4.2: Command line options for top

The key interactive commands are:

Command	Function
<SPACE>	Update display
fF	add and remove fields
oO	Change order of displayed fields
h or ?	Help on interactive commands
S	Toggle cumulative mode
i	Toggle display of idle proceses
c	Toggle display of command name/line
l	Toggle display of load average
m	Toggle display of memory information
t	Toggle display of summary information
k	Kill a task (with any signal)
r	Renice a task
P	Sort by CPU usage
M	Sort by resident memory usage
T	Sort by time / cumulative time
u	Show only a specific user
n or #	Set the number of process to show
W	Write configuration file /.toprc
q	Quit

Table 4.3: Interctive commands for top

4.4 Reporting process status (`ps`)

- `ps` prints info about a user's processes:

```
PID   TTY STAT TIME COMMAND
22074 p0  S    0:02 Eterm -t trans
22075 p0  S    2:13 emacs -bg black
22081 p0  S    0:00 asclock
22590 p5  R    0:00 ps
```

- Unlike `jobs` which only prints info about processes belonging to the current shell
- Various display formats:
  - Long format (`l`), e.g.

```
$ ps l
 FLAGS   UID   PID  PPID PRI  NI  SIZE  RSS  WCHAN   STA TTY TIME COMMAND
     0   512 19665 19653   0   0  7000 5616  12d63d  S   p2  0:19 /usr/bin/emacs
100000   512 23656 23652   0   0  1076  548  1897da  S   p5  0:00 /usr/bin/less -is
```

  - User format (`u`), e.g.

```
$ ps u
$ ps u
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
nicku    15317  0.0  0.4  5940 1224 pts/26   S    Nov20   0:00 bash
nicku     4897  0.0  0.3  4260  828 pts/26   R    09:02   0:00 ps u
```

  - Jobs format (`j`), e.g.

```
$ ps j
 PPID   PID  PGID   SID TTY      TPGID STAT   UID   TIME COMMAND
 2374 15317 15317 15317 pts/26    4980 S     1000   0:00 bash
15317  4980  4980 15317 pts/26    4980 R     1000   0:00 ps j
```

  - Virtual Memory format (`v`), e.g.

```
$ ps v
  PID TTY      STAT   TIME  MAJFL   TRS   DRS  RSS %MEM COMMAND
15317 pts/26   S      0:00    968   562  5377 1312  0.5 bash
 4987 pts/26   R      0:00    173    62  4033  828  0.3 ps v
```

4.5 Options for Reporting process status (ps)

More options to ps:

- Show 'family tree' (f) for processes, e.g.

```
$ ps f
  PID TTY STAT TIME COMMAND
19652  p2 S    0:00 /bin/login -h oakleigh gbdirect.co.uk -p
19653  p2 S    0:00  _ -bash
19664  p2 S    3:02      _ /home/lee/bin/emacs -f gnus-no-server
19665  p2 S    0:23      _ /usr/bin/emacs
19668  p2 S    3:41      _ /usr/lib/netscape/netscape-communicator
19681  p2 S    0:00          _ (dns helper)
```

  Try pstree also.

- Other important options:

Option	Effect
a	Show **a**ll processes
w	Show a **w**ider display (normally just 80 characters wide)
www	Show an even **w**ider display
wwww	Show an even **w**ider display

- Show environment after command line:

```
$ ps e
 PID TTY STAT TIME COMMAND
19653 p2 S    0:00 -bash DISPLAY=oakleigh:0.0 TERM=xterm HOME=/home/davef PATH=
```

4.6 Sorting output of ps

- Sort `ps` results by specific fields:

Option	Function	Option	Function
`Ou`	user	`Oc`	simple executable name
`OU`	uid	`Op`	PID
`OP`	parent PID	`Ot`	tty
`Oo`	session ID	`Ok`	user time
`OK`	system time	`Oj`	cumulative user time
`OJ`	cumulative system stime	`Oy`	priority
`OT`	start_time	`Or`	resident set size
`Ov`	VM size	`Os`	size
`OC`	full command line	`OS`	share

Table 4.4: CLI options for sorting `ps` results

- In `man ps`, search for SORT KEYS.
- See `http://mail.gnome.org/archives/gnome-list/1999-April/msg02337.html`, where Miguel de Icaza explains the memory usage options of `ps`

  Summary:

  size is the total size of a process, including what is in swap. Not so useful.

  ***resident set size* (rss)** is the total amount of RAM memory actually used by a process, i.e., not in swap, but includes memory shared with other processes

  share the amount of memory used by a process that is shared with other processes.

- So actual RAM used by a process is rss − share

4.7 Flavours of `ps` Options

- There are three *flavours* of options for `ps`:
  - BSD options (that we use here), with no dash
  - Unix98 options that must be preceded with a dash
  - GNU long options that must be preceded with two dashes
- It is important to understand that there are BSD options and Unix98 options that have the same letter but which do completely different things, e.g., the BSD option `ef`

```
$ ps ef
  PID TTY      STAT   TIME COMMAND
 6744 pts/3    S      0:00 bash SSH_AGENT_PID=6649
  HOSTNAME=nicksbox.tyict.vtc.e27436 pts/3    R      0:00 ps ef
  SSH_AGENT_PID=6455 HOSTNAME=nicksbox.tyict.vtc.
```

  is completely different from the Unix98 option:

```
$ ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 Dec15 ?        00:00:04 init [5]
root         2     1  0 Dec15 ?        00:00:00 [keventd]
root         3     1  0 Dec15 ?        00:00:00 [kapmd]
root         4     1  0 Dec15 ?        00:00:00 [ksoftirqd/0]
root         6     1  0 Dec15 ?        00:00:00 [bdflush]
...
nicku    27369  8764  0 07:03 pts/4    00:00:00 /bin/sh /usr/bin/xdvi masterfile
nicku    27375 27369  0 07:03 pts/4    00:00:00 xdvi.bin -name xdvi masterfile
nicku    27438  6744  0 07:12 pts/3    00:00:00 ps -ef
```

- You can mix these options together, e.g., to get a wider display of `ps -ef`:

```
$ ps -ef www
UID        PID  PPID  C STIME TTY      STAT   TIME CMD
root         1     0  0 Dec15 ?        S      0:04 init [5]
root         2     1  0 Dec15 ?        SW     0:00 [keventd]
...
```

- See `man ps`

4.8 Examples using ps

- To determine information about the yum process:

```
$ ps auxwww | grep yum
root      6440  2.2  2.8 14832 7284 pts/29   S    09:19   0:00 /usr/bin/python /usr/bin/yu
nicku     6542  0.0  0.1   968  332 pts/11   S    09:20   0:00 grep yum
```

- Hmm, how do we get rid of the line showing the "grep yum" process? One way is to exclude it using grep -v:

```
$ ps auxwww | grep yum | grep -v grep
root      6440  4.7  5.0 20684 12960 pts/29  S    09:19   0:05 /usr/bin/python /usr/bin/yu
```

- But we can do that more simply with:

```
$ ps auxwww | grep [y]um
root      6440  2.6  5.0 20684 12960 pts/29  S    09:19   0:05 /usr/bin/python /usr/bin/yu
```


4.9 Finding Files using specified criteria (`find`)

- `find` searches your filesystem for files matching certain criteria, as we saw in section $ 3.4 on page 59.
- Can match on name, owner, size, modification/access time, name
  - and many others
- Can execute commands on files it finds
- Commonly used to archive sets of files, or clear out old files

4.10 Criteria used in `find` expressions

- Basic syntax:

  `find` ⟨*base directory*⟩ . . . ⟨*search criteria*⟩

`Test`	What It Does
`-name` ⟨*string*⟩	Filename matches string (Shell metacharacters included)
`-mtime -/+`⟨*days*⟩	Modification time matches ⟨*days*⟩
`-user` ⟨*UID/username*⟩	Owner matches ⟨*UID*⟩ or ⟨*username*⟩
`-size -/+`⟨*size*⟩	Size of the file matches ⟨*size*⟩
`-perm -/+`⟨*mode*⟩	Permissions of the file match ⟨*mode*⟩
`-type` ⟨*t*⟩	File is of type ⟨*t*⟩ (`f` — normal file, `x` — -executable file, `d` —directory,. . . . See man page for full details)

- The values to match are very flexible, e.g. to find all files below `/home/lee` that were last modified less than 24 hours ago:

```
$ find /home/lee -mtime -1
```

- Find all files below current directory greater than 1000k in size and with permissions `664` (`-rw-rw-r---`)

```
$ find . -size +1000k -perm 664 -exec ls -l {} \;
```

4.11 Examples of using (`find`)

- Find all files ending in `.jpg` under current dir:

  ```
  $ find . -name "*.jpg"
  ```

- Find filenames ending in `.jpg` *and* modified in the last 8 days below `/etc`:

  ```
  $ find /etc -name "*.jpg" -mtime -8
  ```

- Combine tests with `-o` and negate with `!`, e.g:
  - To find all filenames *not* ending in `.jpg` *or* modified in the last 8 days:

    ```
    $ find . \! -name "*.jpg" -o -mtime -8
    ```

- Execute commands on files found. For example, to find then gzip-compress tar files:

  ```
  $ find . -name "*.tar" -exec gzip {} \;
  ```

  i.e. found filenames substitute for {} above *

- N.B. `find` searches the filesystem in real time; making disks work hard

*Piping the `find` results to `xargs` (section 4.19) is a more efficent approach

4.12 Reporting virtual memory statistics (`vmstat`)

- `vmstat` is used to identify system bottlenecks
- Reports on processes, memory, paging, block IO, interrupts (*traps*), and cpu activity
- SYNTAX:

  `vmstat [-n] [-V] [`*delay* `[`*count*`]]`

- If no `delay` specified gives averages since last reboot
  - Otherwise updates every *delay* seconds
  - Shows averages since last report
- *count* is the number of updates to give
  - If no `count` is specified, just keeps on running
- Option `-n` causes header display only once

4.13 Output from `vmstat`

- The headings in the output have very short names:

Section Head	Field	Description
`Procs`	`r`	no. of runnable processes
	`b`	no. of processes sleeping
	`w`	no. of processes swapped out but otherwise runnable
`Memory`	`swpd`	virtual memory used (kb)
	`free`	idle memory (kb)
	`buff`	memory used as buffers (kb)
`Swap`	`si`	memory swapped in from disk (kb/s)
	`so`	memory swapped out to disk (kb/s)
`IO`	`bi`	Blocks sent to a block device (blocks/s)
	`bo`	Blocks received from a block device (blocks/s)
`System`	`in`	interrupts per second, inc the clock
	`cs`	context switches per second
`CPU`	`us`	user time (as % of total CPU time)
	`sy`	system time (as % of total CPU time)
	`id`	idle time (as % of total CPU time)

Table 4.5: Field descriptions for `vmstat` output

4.14 free

- Another tool to examine memory status
- Displays in kilobytes by default

`-k`	Output in kilobytes
`-m`	Output in megabytes
`-b`	Output in bytes
`-s x`	Poll every x seconds
`-t`	Display a ‘total’ line

- Simpler alternative to `vmstat`
- May be useful if `vmstat` not available
- Quick check on overall memory usage

4.15 ldd

- Modern operating systems support *shared libraries*
  - In Windows, they are called DLLs — **d**ynamic **l**ink **l**ibraries
- Link the program with dynamically loaded shared libraries
  - instead of including a copy of the same executable library code in all applications
  - Makes the application *much* smaller
  - But the application will not run correctly unless the correct version of each shared library is installed
    - In Windows, this is sometimes called *DLL hell* (Do a google search for "dll hell" or see `http://www.desaware.com/Articles/DllHellL3.htm`
    - Linux shared libraries have the version number in their file name, solving the problem in a simple way.
- `ldd` prints out dependencies on shared libraries for a given executable

```
$ ldd /bin/bash
        libtermcap.so.2 => /lib/libtermcap.so.2 (0x40003000)
        libc.so.6 => /lib/libc.so.6 (0x40006000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x00000000)
```

- `/bin/bash` requires three shared libraries to run correctly
- `ldd` is a very useful debugging tool
- `ldd` is also needed for setting up restricted (`chroot`'d) environments

4.16 uptime

- `uptime` prints out basic information about performace of system

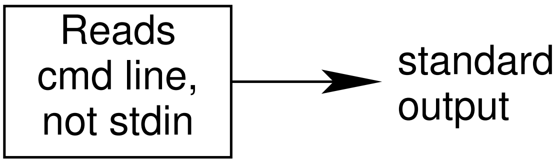
```
$ uptime
 4:26pm  up 165 days, 23:23,  9 users,  load average: 0.23, 0.13, 0.10
```

- Shows:
  - Current Time
  - Time since last reboot (Days, Hours:Minutes)
  - Number of logged-in connections
  - Load average past minute, past 5 minutes and past fifteen minutes
    - *Load average* is the average number of processes that are ready to run, but no CPU is available.
    - A load average of 0.5 is light loading; a load average of 3 is a fairly heavy work load; a load average of 10 will make people complain about the system being “slow”; a load average of 100 may result in a five minute wait between pressing a key on the keyboard and seeing the result.
- Same information given in `top` — see section 4.2 on page 103.

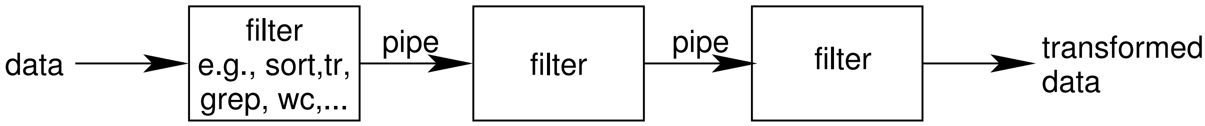
4.17 `xargs` — Filters

- Most tools we examine are *filters*; they read standard input and, when no filename is specified, write to standard output:

- However, some, such as `ls`, `file`, `echo`, `find`, `rm`, `cp`, `mv` read the command line or file system, and do *not* read standard input:



- Sometimes we still want to combine these tools into a pipeline:



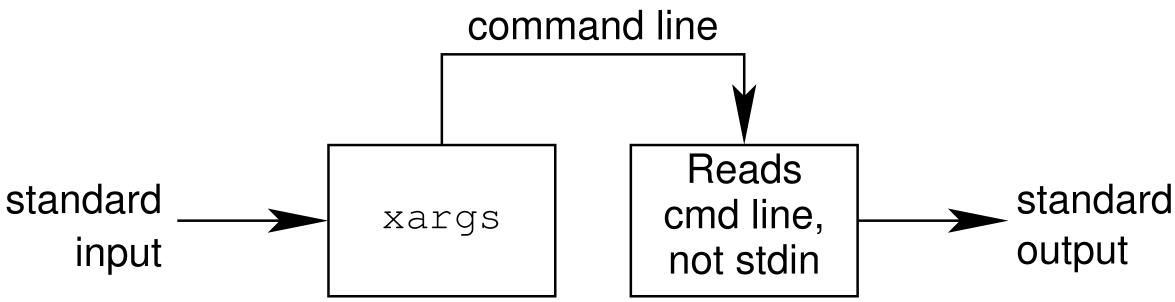
- For example, how can we pipe data from the `find` command into the `file` command?

- This doesn't work:

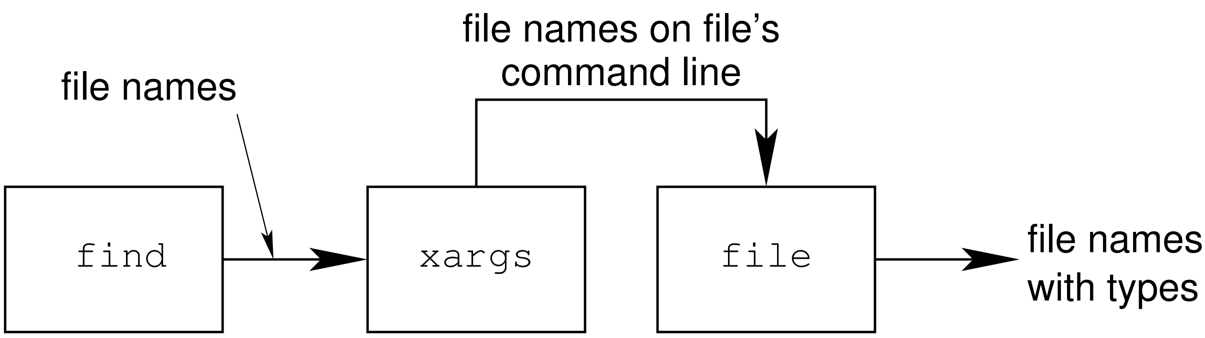
```
$ find /etc | file
Usage: file [-bcikLnNsvz] [-f namefile] [-F separator] [-m magicfiles] file...
       file -C -m magicfiles
Try ‘file --help’ for more information.
```


4.18 `xargs` — an Adapter

- The `xargs` command works as an adapter to read standard input, execute another command by passing the intput data to the command line of the second command:



- We can use `xargs` as an adapter between `find` and `file` to allow them to be part of en efficient pipeline:



- We can write it like this:

```
$ find /etc | xargs file
```

4.19 `xargs`

- Constructs and executes command-lines from information given on standard input
- Commonly used in conjunction with find
- Syntax:

```
| xargs command
```

- N.B. Data is normally piped to `xargs`
- Example: Delete all files named `core` not modified in the past 8 days

```
$ find / -name "core" -mtime +8 | xargs rm -f
```

- This is more effient than

```
$ find / -name "core" -mtime +8 -exec rm -f {} \;
```

as it spawns fewer processes, therefore easing the load on the machine

4.20 Options to `xargs`

`-p`	Interactive mode
`-t`	Verbose mode
`-n num`	Limit arguments used to command

- Verbose mode will print out the commands it executes
- Interactive mode prints out the command-line and awaits confirmation before executing
- `-n` tells `xargs` to use at most `num` arguments to the command you are running with `xargs`
- Example:

```
$ find / -name "core" -mtime +8 | xargs -p -n 2 rm -f
rm -f /home/lee/core /home/davef/core ?...y
rm -f /usr/local/bin/core /root/core ?...y
$
```

4.21 Positioning filenames with `xargs`

- By default `xargs` places the filenames at the *end* of the command you give
- If they need to be somewhere else you can use `--replace`
- Put {} at the point that you want the filenames inserted
- Example:

```
$ find /var/log -mtime -0.5 | xargs --replace cp {} /tmp
```

- Disadvantage: execute new command process for each file, e.g.,
  - Here, when `xargs` executes `echo` just once, i.e., there is one `echo` process

    ```
    $ ls
    file1  file2  file3
    $ ls | xargs echo
    file1 file2 file3
    ```

  - But when we use `--replace`, `xargs` executes `echo` three times, once for each file, i.e., we have three separate `echo` processes:

    ```
    $ ls | xargs --replace echo {}
    file1
    file2
    file3
    ```


4.22 cpio

- Similar to `tar`
- Creates archives of files
- Operates in *copy-in* or *copy-out* mode
  - Copy-out mode writes archives
  - Copy-in extracts from them
- Takes filelist on standard input
  - Not given on the command line
  - Can use `find` to feed it filenames

`-i`	"Copy-in mode" — extract from archive
`-o`	"Copy-out mode" — create an archive
`-A`	Append to an archive
`-F` ⟨*file*⟩	Use ⟨*file*⟩ instead of standard input/output
`-H` ⟨*format*⟩	Use archive format ⟨*format*⟩

- Example: make a `cpio` archive of `/etc`:

```
$ find /etc | cpio -o -F /tmp/etc.cpio
```

- There are two main reasons for sometimes using `cpio` instead of `tar`:
  - `cpio` will not include the contents of a directory if you list the directory name as a file. This is good when you want to include some but not all files from that directory.
  - The RPM Package Manager (`rpm`) provides the `rpm2cpio` command that allows you to extract individual files from an RPM package.

4.23 `gzip`

- Compresses data taken from `stdin` or a file
- *'Lossless'* compression
  - Safe on any file
- Various compression levels, 1-9

  1 — Fast, less compression

  9 — Slower, more compression
- By default takes `filename` and turns it into `filename.gz`

```
$ ls foo*
foobar
$ gzip -9 foobar
$ ls foo*
foobar.gz
```

  - Replaces original file
- Can write compressed data to `stdout`

```
$ ls foo*
foobar
$ gzip -9 -c foobar > foobar.gz
$ ls foo*
foobar     foobar.gz
```

  - Leaves original file intact

4.24 Unzipping

- To unzip a file there are two methods
  - `gunzip`
  - `gzip -d`
- Really one file linked to two names
- Uncompress all files in the current directory

```
$ gzip -d *.gz
```

- `gzip` uses Lempel-Ziv coding
- Can also unpack files created with `zip`, `compress` and `pack`

4.25 tar

- tar creates archives of files
- Used for transferring files between machines
  - Or from place to place
- Options:

-x	Extract from an archive
-c	Create archive
-t	List files in an archive
-v	Be verbose
-j	Compress/decompress archive with bzip2
-z	Compress/decompress archive with gzip
-d	Find differences between archive and the filesystem
-f	Specifies a file name for the archive. Can be a device, such as tape device /dev/nst0

- Can use with or without the dash

4.26 `tar` Examples

- `tar` all files ending in `.tex` into a gzipped * archive called `alltex.tar.gz`

  ```
  $ tar cvzf alltex.tar.gz *.tex
  ```

- Check which files were included

  ```
  $ tar tvzf alltex.tar.gz
  ```

- Extract them again

  ```
  $ tar xvzf alltex.tar.gz
  ```

  Same again, but with `bzip2 compression`:

  ```
  $ tar cvjf alltex.tar.bz2 *.tex
  ```

- Check which files were included

  ```
  $ tar tvjf alltex.tar.bz2
  ```

- Extract them again

  ```
  $ tar xvjf alltex.tar.bz2
  ```

*Integrated gzipping is available with GNU `tar`, but may not be available on some proprietary versions of `tar`.

4.27 Raw devices and `tar`

- Originally designed to talk to magnetic tapes
  - My SCSI tape device is `/dev/nst0`
  - See kernel documentation in `/usr/share/doc/kernel-doc-2.4.22/devices.txt`
- Can also write to other raw devices
- Useful to maximize use of the space on the floppy
  - Fit 1.44Mb of data on a floppy
  - Don't need any space for filesystem information

```
$ tar cvj filelist > /dev/fd0
```

or simply

```
$ tar cvjf /dev/fd0 filelist
```

or simply

- Extract it again

```
$ tar xvj < /dev/fd0
```

or

```
$ tar xvjf /dev/fd0
```

- A useful option with floppy disks or writable CDROMs is `-M` pro**M**pt for disk change when full

4.28 Exercises

1. Use `top` to show the processes running on your machine. See section 4.2 on page 103.
2. Make `top` sort the list by memory usage. Press (h), and also see section 4.3 on page 104
3. Try killing a process, a good example would be your top process itself! Note that you can only kill your own processes unless you are `root`.
4. Use `top` to re-nice a process so that it gets more, or less CPU time. You can only lower the priority of your own processes. Only `root` can raise the priority of a process. Nice value of -20 is highest priority. Nice value of 19 gives lowest priority. See `man nice`.
5. Find a full list of every process on your machine and their full command name using `ps`. See the table in section 4.5 on page 106 and the examples in section 4.8 on page 109
6. Get the same view but tell `ps` to display the output in 'family tree' mode. Use the options `axf` to see the family relationships. See section 4.5 on page 106.
7. Request that `ps` sort it's output by system time used. See section 4.6 on page 107.
8. Set `vmstat` running in a spare terminal updating every 5 seconds. See section 4.12 on page 113.
9. Set up `free` in another window doing the same thing. See section 4.14 on page 115.
10. Use `ldd` (see section 4.15 on page 116) to find out what shared libraries some common applications use:
    - **(a)** `/sbin/halt`
    - **(b)** `/usr/bin/gnome-session`
    - **(c)** `/usr/bin/display`
    - **(d)** `/bin/bash`
    - **(e)** `/sbin/nash`
11. See sections 4.17 on page 118 to 4.21 for more about `xargs`. See section 4.19 on page 120 for examples of using `xargs` with `find`. Practice using `xargs` by finding sets of files and performing simple (Non-destructive!) operations on them, e.g.,
    - **(a)** Find all files in the file system modified in the last 24 hours (see §111) and make copies of them in a directory called `backup` in your home directory. See section 4.21 on page 122. (You may want to redirect standard error to the null device, `/dev/null`, both for `find` and for `cp`.) Do this two ways:
        - **i.** Using the `--replace` option with `xargs`, described in section 4.21 on page 122, and
        - **ii.** Using the `--target-directory` option to `cp`. See `man cp`.
        - **iii.** Which of the two methods is most efficient? See section 4.21 on page 122.
    - **(b)** Find all files over 5000k and make copies of them in the `backup` directory
    - **(c)** Find all files ending in `.txt` in your home directory and below, and compress them using `bzip2`
12. Use `cpio` and `tar` to create an archive of the files you've copied in to the `backup` directory. See section 4.22 on page 123 and section 4.25 on page 126 to section 4.27.
13. Write an archive of `/etc` to a floppy as a raw archive. See section 4.27 on the preceding page.

4.29 Solutions

Module 5

Basic Filesystem

Objectives

- After completing this section, you will be able to:
  - Understand a typical Linux filesystem
  - Navigate the file hierarchy
  - Manipulate files and directories
  - Handle access control
  - Deal with 'special files' and *links*

5.1 Filesystem Overview

- Linux uses *ext3*, *reiserfs* and *ext2* as its *native* filesystems
  - Also supports many other types
- All data stored on a Linux system is a file
- Ext2/3 file names can be 1 to 255 characters long
  - Only */* and *nul* are disallowed
- Non-native filesystems have different features
- Ext2/3 sees only two basic types of files:
  - *directories*
  - *files*
- Other specialised types exist (FIFOs, and ‘special files’), these are covered later

5.2 Files

- Linux imposes no structure on files
- All files are accessible at the byte level
- Individual files used to have a maximum size of around 2GB (*in an ext2 filesystem with an older kernel, 2.2.x*)
- Modern kernels (2.4.x) have a maximum file size of about 8TB
- They have a minimum size of 0 bytes
- Files can be extended after creation
- Filename extensions such as *.exe* and *.bat* are unnecessary
- Executable files are simply marked as such using file permissions (*see later*)

5.3 Directories

- *Directories* are files that list other files
  - Can be normal files or directories
  - Enables a hierarchy to be built
- Each directory entry consists of two parts: a file name and an *inode number*
  (An inode is roughly a *pointer to a file*, see below)

Filename	Inode number
`.`	512
`..`	500
`bin`	17324
`basic_linux.tex`	24567

- The topmost directory is always called `/`
  - Called the fileystem 'root'
- Directory information can only be changed by Linux itself
  - Ensures a proper structure is maintained

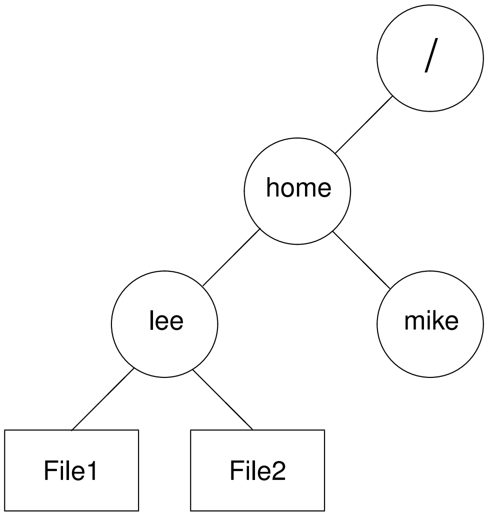
5.4 Directory Hierarchy

- By the *Linux Filesystem Hierarchy Standard* `http://www.pathname.com/fhs/`, many directories have specialised rôles

`/bin`	essential executable commands, available when no other filesystems are mounted
`/sbin`	essential system administrator commands, needed for repairing filesystems, when no other filesystems are mounted
`/boot`	static files of the boot loader
`/etc`	system configuration files
`/lib`	essential shared libraries and kernel modules
`/dev`	device files which control peripheral devices
`/tmp`	temporary files
`/mnt`	to mount external devices
`/var`	variable data files, e.g. logs, status and lock files, spooling files—let `/usr` be mounted read only
`/proc`	system information, dynamically generated by kernel
`/usr`	shareable, read-only data
`/usr/bin`	most user commands
`/usr/sbin`	further system administration commands
`/usr/lib`	further libraries for programming and shared libraries used by software packages
`/usr/share`	architecture independent data, including man pages and documentation

- User-installed programs typically go under the `/usr/local` hierachy
- `/mnt` is a convenience to place all ‘mounted’ devices under one place

5.5 Pathnames



- Files can be referred to by *relative* or *absolute* pathnames
- Absolute pathnames begin with `/`

```
/usr/sbin/httpd
/usr/local/bin/safe-mysqld
```

- The *absolute* pathname refers to one file only
- A *relative pathname* does not begin with `/` and describes the path from the current directory to find a file, e.g.

```
sbin/httpd
bin/safe-mysqld
```


5.6 Current Directory, Home Directory

- Every process has a `current directory`
- When you log in your shell's current directory is your *home directory*
  - Typically `/home/username`
  - Superuser usually has `/root` for a home directory
- `pwd` tells you the current directory

∼ the shell expands this to name of your home directory

- The "∼" character is called "tilde"

∼⟨*user*⟩ the shell expands this to the name of the home directory of the user with user name ⟨*user*⟩

The `cd` Command

- `cd` changes your current directory
  - Typing `cd` ⟨*path*⟩ changes your current directory to ⟨*path*⟩
- `path` can be *absolute* or *relative*
- Without arguments `cd` changes to your home directory

5.7 Dot (.) and DotDot(..)

- Directories always contain two entries “.” and “. .”

.	Current directory
. .	Parent directory

- Used for relative pathnames and navigation
- Example:

`$ cd ..`	Change to the parent directory
`$ mv file ..`	Move `file` to the parent directory
`$ du .`	Display space used by files in current directory and below
`$ du ..`	Display space used by files in parent directory and below
`$ ./a.out`	Execute the file `a.out` in the current directory

- This last row above shows execution of the particular file that is in the current directory
  - If we had simply typed `a.out` then our `PATH` environment variable would be used to search for the file
  - It may execute another `a.out` instead of the one we *actually* want

5.8 Moving and Copying Files

- The `mv` command is used to move files:

  `mv` [⟨*options*⟩] ⟨*source file*⟩ ⟨*dest file*⟩
  `mv` [⟨*options*⟩] ⟨*source file*⟩ . . . ⟨*target directory*⟩

  e.g.

  ```
  $ mv oldname newname
  $ mv somefile ..
  ```

- The `cp` command is used to copy files:

  `cp` [⟨*options*⟩] ⟨*source file*⟩ ⟨*dest file*⟩
  `cp` [⟨*options*⟩] ⟨*source file*⟩ . . . ⟨*target directory*⟩

  e.g.

  ```
  $ cp -a thisfile newfile
  $ cp file1 file2 file3 /tmp
  ```

  - A *very important option* is `-a`, which preserves permissions and symbolic links (see slide § 5.15 on page 146 for more about symbolic links).
  - Can even use `cp -ax` to clone a hard disk partition.

- See `man cp` and `man mv`

5.9 Removing Files

- Files are removed using the `rm` command:

  `rm [`⟨*options*⟩`]` ⟨*file*⟩`...`

  e.g.

```
$ rm -i thisfile thatfile
rm: remove 'thisfile'?  y
rm: remove 'thatfile'?  y
$
```

- Most useful options are:

`rm -f`	Force removal and say nothing if cannot remove
`rm -r`	Recursively delete files

  e.g.

```
$ rm -rf somedir
```

  will delete all files and subdirectories in directory `somedir` and below

- See `man rm`
- Removing a file is *not* considered an operation on the file
  - It is an operation on the directory
  - Filenames are merely *links* (Explained below)

5.10 Operations on Directories

`mkdir`	Create a new directory
`rmdir`	Remove a directory
`ls`	List the contents of a directory

- These commands can take many arguments
- `mkdir` can be told to create the whole pathname of directories if they don't exist, e.g.

  ```
  $ mkdir -p ~/new1/test/directory
  ```

  Will create the directories `~/new1` and `~/new1/test` as well as `~/new1/test/directory` if they don't already exist
- `ls` arguments control what information is shown and how it's sorted
  - Some are explained later, consult `man ls` for full details. Also see figure 5.1 on page 148 for meaning of output of `ls -l`.

5.11 Inodes

- Each file is represented by an inode*
- An *inode* contains information about:
  - File type (ordinary, directory, FIFO, block device etc.)
  - Owner ID (user the file belongs to)
  - Size (in bytes)
  - Access, creation, and modification times
  - Group ID (group the file belongs to)
  - File permissions
  - Mapping of the file contents (data sectors)
- Inode layout and location varies with filesystem type

*The term *inode* was invented by Dennis Ritchie of AT&T. He admits to forgetting why he chose that name.

5.12 Inodes: `ls -i` and `stat`

- `ls -i` displays inode numbers of entries, e.g.

```
$ ls -i
200808 Ext2fs-0.1-14.html   542726 include
200795 Ext2fs-0.1-2.html    188447 info
200797 Ext2fs-0.1-4.html    333831 ldap
200802 Ext2fs-0.1-8.html    329729 man
200803 Ext2fs-0.1-9.html    278533 misc
200793 Ext2fs-0.1.html      428042 nsmail
204802 systemprogramming
```

- `stat` prints the inode contents for files inc. permissions, size, links, access times etc.

```
$ stat /home/lee
File: "/home/lee"
Size: 4096         Filetype: Directory
Mode: (0755/drwxr-xr-x) Uid:(504/lee) Gid:(502/lee)
Device:  3,0   Inode: 200705    Links: 30
Access: Thu May 27 10:54:55 1999(00000.00:01:43)
Modify: Thu May 27 10:44:41 1999(00000.00:11:57)
Change: Thu May 27 10:44:41 1999(00000.00:11:57)
```


5.13 Links

- More than one filename can refer to an inode
  - These file names are *links* to the file
- `ln` creates links to files
  - Creates *hard links* by default
  - `ln -s` creates *symbolic* or *soft links*
- Erasing a file just removes its directory entry
  - The file is only lost when all entries for it have been removed
- Important: A filename is *not* the file
  - The inode *is* the file
  - All names are simply links (references) to the inode
  - A bit like a Windows' 'shortcut'

5.14 Hard links

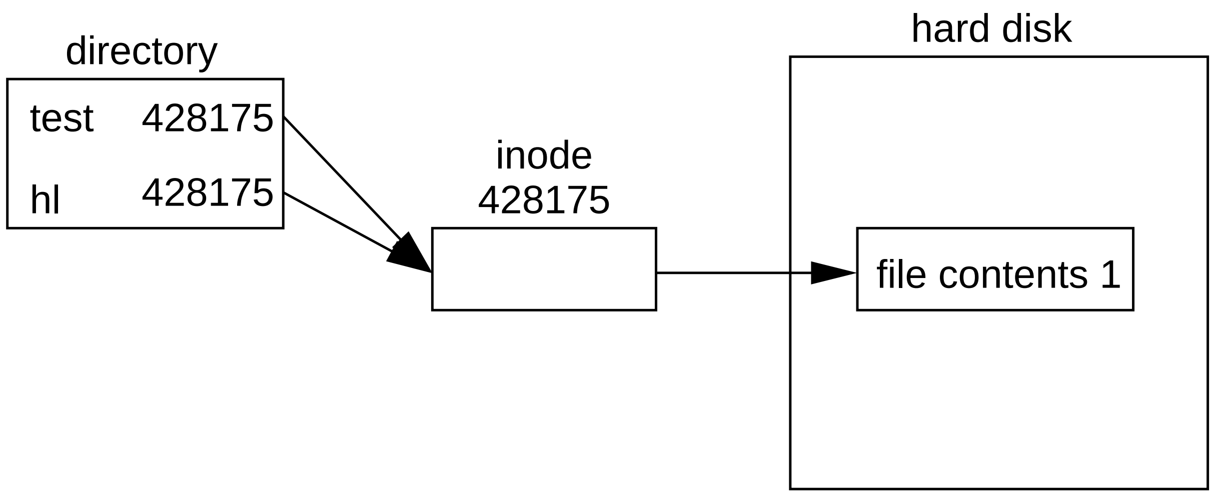
- A *hard link* is merely a directory entry with the relevant *inode* number
- Consider the following
  - Start with:

    ```
    $ ls -li
    428175 -rw-rw-r-- ... 4 May 26 13:18 test
    ```

  - We create a *hard link*:

    ```
    $ ln test hl
    $ ls -li
    428175 -rw-rw-r-- ... 4 May 26 13:18 hl
    428175 -rw-rw-r-- ... 4 May 26 13:18 test
    ```

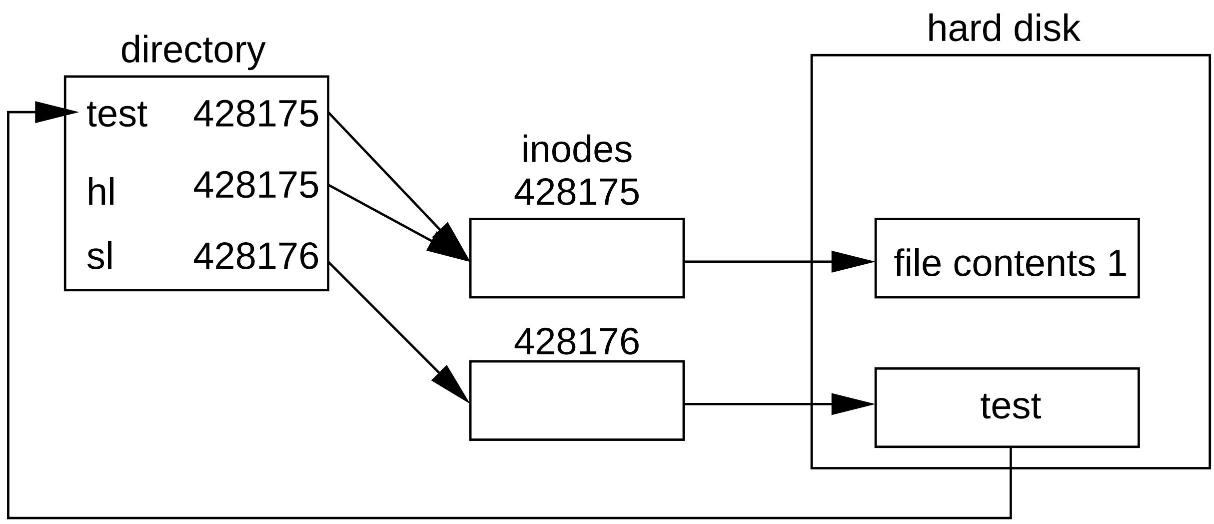
- N.B. *Hard links* cannot cross filesystems
- *Inode* numbers are filesystem specific



5.15 Symbolic Links (Soft Links)

- *Symbolic* (or *Soft*) *links* store the pathname of the linked file
- This means they can cross filesystems.
- Adding a *symbolic link*:

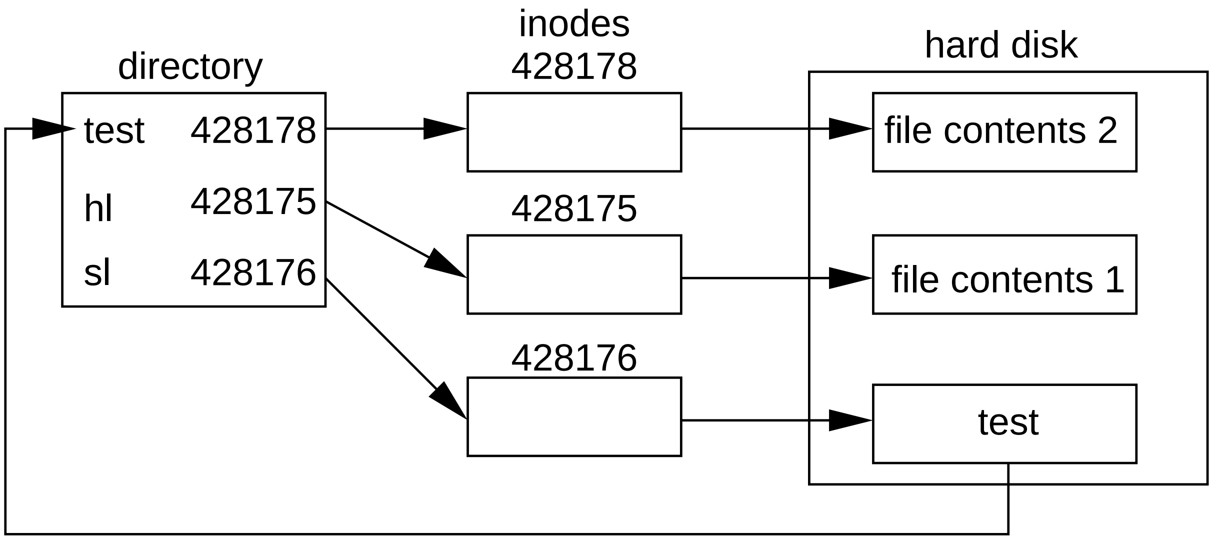
```
$ ln -s test sl
$ ls -li
428175 -rw-rw-r-- ... 4 May 26 13:18 hl
428176 lrwxrwxrwx ... 4 May 26 13:19 sl -> test
428175 -rw-rw-r-- ... 4 May 26 13:18 test
```



5.16 Symbolic (or Soft) Links (continued)

- If we replace the `test` file with another then the symbolic link still works, but the hard one still points to the old file!

```
$ mv ../test2 test
$ ls -li
428175 -rw-rw-r-- ...  5 May 26 13:45 hl
428176 lrwxrwxrwx ...  4 May 26 13:19 sl -> test
428178 -rw-rw-r-- ... 15 May 26 13:47 test
```



5.17 File Ownership, Users and Groups

- Every user has a unique user ID and a unique primary group
- When a user starts a process, the process is owned by the user ID and primary group of the user
- Every file has a user that owns the file, and a group that owns the file.
- When a process creates a file, the file is owned by the same user and group as the process
- E.g., if user `nicku` with primary group `staff` creates a file, then the file will be owned by user `nicku` and group `staff`.
- Figure 5.1 shows how `ls -l` shows file ownership.

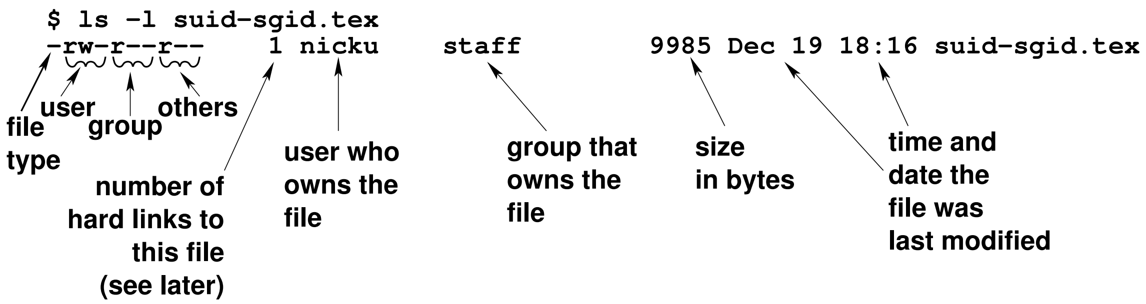


Figure 5.1: The output of `ls -l`: what each field is.

5.18 Access Control, Users and Groups

- File access can be limited to specific users
- *Super* user(s) can override access control
- Access control is set by user and group ID
- Each user has a user-id (*UID*) and one or more group-ids *GIDs*)
- Processes have an associated UID and GID
  - Inherited from the user who created the process
- They can however can be changed:
  - Processes are known as set-user ID (SUID) if they set their own user ID equal to the user that owns the executable file,
  - or set-group ID (SGID) if they set their own group ID equal to the group owner of the file
- There are three special modes that we will examine in the next module: “set user ID”, “set group ID” and the “restriction deletion flag”. You can read about them in `info`:

```
$ info '(coreutils)Mode Structure'
$ info '(coreutils)Numeric Modes'
```

5.19 Categories of Access Control

- There are three categories of access, and three categories of user:

Category of Access		Set of Users	
read	`r`	**u**ser	`u`
write	`w`	**g**roup	`g`
e**x**ecute	`x`	**o**thers	`o`

Permission	For Files	For Directories
`read`	permission to open the file for reading	permission to list the file names
`write`	permission to open the file for writing	permission to create and delete files in the directory
`execute`	permission to execute the file; scripts also need read permission.	permission to change into the directory

- There are three sets of access control, one for each of the three sets of users:

user: the user that owns the file

group: members of the group that owns the file, except for the user that owns the file

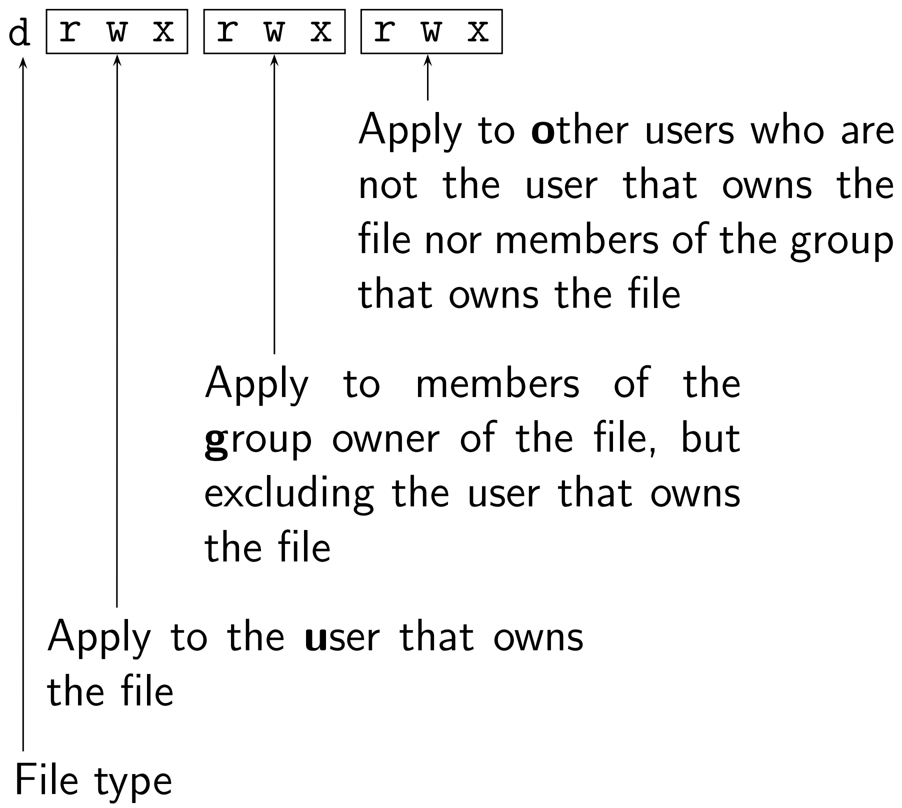
others: all users who are not the owner of the file, and who are not members of the group that owns the file.

5.20 Access Control — Example

- `ls -l` shows the access permissions, e.g.

```
$ ls -l
-rw-rw-r-- 1 www    www    x_windows.tex
lrwxrwxrwx 1 lee    lee    img -> ../linux/img/
-rw-rw-r-- 1 lee    lee    test.log
```

- There are three sets of permissions, each of which applies to one set of users
- To find permissions that apply to a user, determine what set of users the user belongs to, and then those permissions apply to that user.



5.21 Examples of minimum file permission requirements

- This table shows examples of minimum file permission requirements
- It is based on table 2-2 on page 37 of the book *Essential System Administration* by Æleen Frisch, O'Reilly 2002.
- I recommend this excellent book.

Command	minimum access required	
	on the file	on the directory
`cd /var/project`	no file	`--x`
`ls /var/project`	`---`	`r--`
`ls -l /var/project`	`---`	`r-x`
`cat /var/project/user1.txt`	`r--`	`--x`
`echo "hello" >> /var/project/user1.txt`	`-w-`	`--x`
`/var/project/binary-program`	`--x`	`--x`
`/var/project/script-program`	`r-x`	`--x`
`rm /var/project/user1.txt`	`---`	`-wx`

5.22 Changing Access Permission: `chmod`

- Only the owner of a file (or the super-user) may alter its access permissions
- `chmod` (change mode) changes access permissions
  - Works in two ways, symbolically or numerically
  - Symbolically is easier to remember (for most)

5.23 `chmod` symbolically

- Select who you want to change permissions for `(u=user, g=group, o=others, a=all)`
- Decide whether you want to `grant a permission(+), remove(-), or set(=)` it
- Take the permission that you want to change `(r=read, w=write, x=execute)`
- Example:

  ```
  $ chmod gu+w filename
  ```

  Adds write permission for user and group
- You can make several changes by separating the settings with commas, e.g.

  ```
  $ chmod a-w,gu=rw filename
  ```

  Removes write permission for all, then grants it for the user and group

5.24 `chmod` numerically

- Once you know this it is often quicker
- An octal digit represents each permission type

4	read permission
2	write permission
1	execute permission

- Add up the permission numbers you want for each user group (owner, group, all) and supply these to `chmod`
- Example:

```
$ chmod 755 filename
```

grants all permissions to the owner (4+2+1), and read and execute (4+1) to group and all others

  - `chmod 755 filename` is equivalent to `chmod u=rwx,go=rx filename`

5.25 Special Permissions: SUID, SGID

- When the SUID permission applies to an *executable file*, then, when it is executed, the file will execute with the UID of the owner of the file, instead of the UID of the person executing the program.
- When the SGID permission applies to an *executable file*, then, when it is executed, the file will execute with the GID of the group owner of the file, instead of the GID of the person executing the program.
- When the SGID permission applies to a directory, then all files created there have the same group owner as the directory, instead of the group owner of the process that created the file, as would be the case otherwise.
- When the *restricted deletion flag* ("sticky bit") is enabled on a directory, then only the owner of a file (and `root`) may delete or modify the file. Normally, *if other permissions allow it*, anyone may delete or modify a file that they do not own.

5.26 `chmod`: Symbolic Permissions

- To apply the `SUID` permission to `file`:

  ```
  $ chmod u+s file
  ```

- To apply the `SGID` permission to `file`:

  ```
  $ chmod g+s file
  ```

- The *restricted deletion flag* ("sticky bit") permission applies only to the "others" part of the file permissions, but it behaves as I described above. To apply the "restricted deletion flag" permission to `directory`:

  ```
  $ chmod o+t directory
  ```

5.27 `chmod`: SUID, SGID

- The permissions for SUID and SGID are specified numerically like this:

numeric mode	changes:
2000	set group ID
4000	set user ID
1000	restricted deletion flag (“Sticky bit”)

- Example:

```
# make the program executable file /tmp/ash SUID:
$ sudo chmod 4755 /tmp/ash
# make the program executable file /tmp/ash both SUID and SGID:
$ sudo chmod 6755 /tmp/ash
# remove the SUID and SGID permissions from /tmp/ash:
$ sudo chmod 755 /tmp/ash
```

- The permission for SUID and SGID is specified symbolically with ‘`s`’, and can be added or removed from user or group permissions.

- Example:

```
# make the program executable file /tmp/ash SUID:
$ sudo chmod u+s /tmp/ash
# make the program executable file /tmp/ash both SUID and SGID:
$ sudo chmod ug+s /tmp/ash
# remove the SUID and SGID permissions from /tmp/ash:
$ sudo chmod ug-s /tmp/ash
```


5.28 Set Group ID Directory

- If the “set group id” (SGID) permission is set on a directory, then:
  - if a user makes a change to a file or creates a file in that directory, the file will have group owner the same as the directory.
  - If a user creates a directory in that directory, it too will have the Set Group ID bit set. As with the file, it will have a group owner the same as the group owner of its parent SGID directory.

5.29 Set Group ID Directory — Example

- Let's see the result of creating a file in `/var/project` both before and after adding this permission:

```
$ ls -ld /var/project
drwxrwx---    2 root     admin        4096 Jan  2 13:48 project
$ touch /var/project/test1
$ ls -l /var/project
-rw-rw-r--    1 nicku    nicku           0 Jan  2 13:53 test1
```

  Now we add the set group ID bit to the directory permissions, and see the effect:

```
$ sudo chmod g+s /var/project
$ ls -ld /var/project
drwxrws---    2 root     admin        4096 Jan  2 13:48 project
$ touch /var/project/test2
$ ls -l /var/project
-rw-rw-r--    1 nicku    nicku           0 Jan  2 13:53 test1
-rw-rw-r--    1 nicku    admin           0 Jan  2 13:54 test2
```

  - Note that the user `nicku` has primary group `nicku`, and is also a member of the `admin` group.
- when the directory is not SGID, files I create have my primary group ID.
- when the directory *is* SGID, files I create have the group ID of the directory. This allows others in the group to read, write and change the files created by any group member.

5.30 Restricted Deletion Flag (“Sticky Bit”) on Directories

- When a directory has the *restricted deletion flag* (“sticky bit”) set:
  - Files cannot be overwritten or deleted by any user except the user that created the file
  - Normally, *if other permissions on the directory allow*, anyone may delete a file they do not own.
- Use in `/tmp` to protect users’ files from being altered by other users

```
$ ls -ld /tmp
drwxrwxrwt  397 root     root        12288 Feb  5 13:07 /tmp
         ↑
```

- Notice that instead of “`x`” for others, there is a “`t`”
- Anyone can create a new file in `/tmp`, but only the `root` user can overwrite or delete files they do not own.
- Useful for many applications besides protecting files in the `/tmp` directory.
- Add to a directory with this command:

  `$ sudo chmod o+t` ⟨*directory*⟩

5.31 umask

- Files begin with a default access setting; files with `-rw-rw-rw-`, or 0666, and directories with `drwxrwxrwx` (0777)
  - Specified by a user's *umask* setting
- This only works numerically
- Unlike `chmod`, specified permissions are turned *off*
  - umask specifies permissions which are *absent*
- With a umask setting of 000 files are created with permissions `rw-rw-rw-` (666)
- Default umask (on Red Hat systems) is 002 which means files are typically created `rw-rw-r--` (664) * e.g.,

```
$ umask 002
$ touch foo
$ ls -l foo
-rw-rw-r-- 1 lee lee 0 Feb 10 17:17 foo
$ umask 222
$ touch foo2
$ ls -l foo2
-r--r--r-- 1 lee lee 0 Feb 10 17:17 foo2
```

- The default access setting is bitwise ANDed with the complement of the umask. For example, with a umask of 027, and a default setting of 666, we have:

	octal		binary		AND		octal
default	0666		110 110 110		110 110 110 &		
umask	~0027	→	~000 010 111	→	111 101 000		
					110 100 000	→	0640 → `rw-r-----`

*This is the case on Redhat systems where users typically belong to a group of their own; other distributions will probably use a default umask of 022.

5.32 Special Files — /dev

- Files under /dev typically represent devices attached to your computer
- Programs can open and close them and read from and write to them — as with regular files
- Kernel code handles exactly how these work
- Two types
  - Block — Disk drives, RAID devices, SCSI devices, CDROMS
  - Character — Printers, modems, tape drives, mice, sound devices, USB devices, etc.

5.33 Special Files — `/proc`

- The section of the filesystem called `/proc` doesn't contain *real* files
- It contains system status information
- It is built dynamically by the Linux kernel
- For example:

Location	Information
/proc/⟨*number*⟩	On specific running processes. See man proc for details
/proc/meminfo	How much memory is in your system and how much is being used
/proc/cpuinfo	What CPU(s) you are currently using
/proc/filesystems	Filesystems your kernel supports
/proc/mounts	The definitive list of what is actually mounted on your computer
/proc/uptime	The uptime of the system
/proc/kcore	An image of your physical memory
/proc/net	Network status of your machine
/proc/pci	PCI devices found at initialization
/proc/sys	Details on kernel variables, e.g. ◦ Maximum number of files we can open (fs/file-max) ◦ Number of files currently open (fs/file-nr) ◦ Whether IP is forwarded between network interfaces (net/ipv4/ip_forward) Set the values "permanently" in /etc/sysctl.conf. See man sysctl.conf and man sysctl

Table 5.1: System Information from /proc

5.34 Filesystem Structure and `/etc/fstab`

Multi-Volume Filesystems

- The filesystem can be held on several devices
- Large disks can be divided into partitions
  - This creates several *logical* devices
- A basic Linux system must be present on `/`
- Other parts of the fs may be mounted at any time
- The main ones are mounted at boot time
- This is controlled by the important `/etc/fstab` file which says which file system is mounted where:
  - while the computer starts up, and
  - when you type `mount` ⟨*mount point*⟩ or `mount /dev/`⟨*device name*⟩

5.35 `/etc/fstab` — Example

```
# Logical Volume     Mount Point         FS type  Options                 Dump   Check
#                                                                                order
/dev/hda1            /                   ext3     defaults                1      1
/dev/hda5            /home               ext3     defaults                1      2
/dev/hda7            /tmp                ext3     defaults                1      2
/dev/hda6            /usr                ext3     defaults                1      2
/dev/hda8            swap                swap     defaults                0      0
/dev/fd0             /mnt/floppy         ext3     noauto                  0      0
/dev/cdrom           /mnt/cdrom          iso9660  noauto,ro               0      0
\\kashmir\c          /mnt/kashmir        smbfs    guest                   0      0
landlord:/var/admin  /var/admin          nfs      defaults                0      0
landlord:/home/lee   /home/lee/LANDLORD  nfs      defaults                0      0
/dev/hda5            /mnt/winas          ntfs     default,ro,umask=0000   0      0
```

- See `man fstab`
- The first field is a device
- The second field is a directory, the *mount point*, that must exist;
- The third field is the type of the file system (see `man mount` and also `cat /proc/filesystems`)
- The fourth field is options to `mount`. See `man mount`
- The second last field says whether the program `dump` should backup this filesystem
- The last field is:
  - 0 if the program `fsck` (file system check) should check the file system on boot,
  - 1 if it should be checked first (should be just the root filesystem "/"),
  - 2 if it should be checked later in boot sequence.
- Currently (2004) Red Hat are not putting the NTFS module into the 2.4.x kernel—get the module RPM from `http://linux-ntfs.sourceforge.net/rpm/fedora1.html`

5.36 Mounting Additional Volumes

- To mount a filesystem use `mount`, e.g.

  ```
  $ mount /dev/cdrom /mnt/cdrom
  ```

- Mounts the filesystem `/dev/cdrom` in the directory `/mnt/cdrom`
- `cd /mnt/cdrom` changes directory to the root of the CDROM's filesystem
- To unmount use `umount name` where *name* is either the filesystem name or the mount point:

  ```
  $ umount /dev/cdrom
  $ umount /mnt/cdrom
  ```

- *Note* — A filesystem can only be unmounted when it is no longer in use. *'In use'* includes:
  - Having any file on that filesystem open
  - Having a shell in a directory on that filesystem
- Use the `lsof` ("list open files") program to identify and kill processes that prevent unmounting a file.

5.37 Mounting shared filesystems

- *NFS* filesystems can be mounted with

  `$ mount -t nfs` ⟨*hostname*⟩`:`⟨*path*⟩ ⟨*mount-point*⟩

  ⟨*mount-point*⟩ is a directory which must already exist. ⟨*path*⟩ is a directory on the NFS server that it *exports* to the client.

- Example:

  ```
  $ mount -t nfs landlord:/backup /mnt/backup
  ```

- The `mount` program is smart enough to determine that the type of the filesystem is `nfs`, so you do not need to specify it:

  ```
  $ mount ictlab:/var/ftp/pub /mnt/nfs
  ```

- Share files from MS-Windows machines using *Samba*
- This is a free implementation of the Windows file-sharing protocols, e.g.

  ```
  $ mount -t smbfs '\\ntbox\c' /mnt/ntbox
  ```

- *N.B.* Linux does not use the 'drive letter' concept at all
  - *Drives* and *shares* integrate seamlessly into the filename tree

5.38 Summary

- The primary Linux filesystems are Ext3, Reiserfs and Ext2
- It has a tree-like hierarchy of directories
- Directories merely contain pointers to files (*inodes*)
- *inodes* contain all the information about a file
- Can have multiple links to the same file
- Read/Write access is controlled per file
- Creation/Deletion of files is controlled by permissions of the directory
- Several filesystems can be mounted to create the directory hierarchy

5.39 Filesystem Exercises

1. *Basic navigation*

   (a) Log in and use `pwd` to discover what the full path of your home directory is.

   (b) Change directory to `/bin` and then `/tmp`. Use `pwd` to check you got there each time.

   (c) When in `/tmp` type `cd ..` and use `pwd` to find out where you end up. See section § 5.7 on page 138

   (d) What is the parent directory of the root of the filesystem? Why is this so?

   (e) Move back to your home directory. Think of three ways you can do this. See section § 5.6 on page 137.

2. *Directories*

   (a) Start in your home directory and create a directory called `new`

   (b) Change to the `new` directory and create a directory called `newer`

   (c) Go to your home directory. Now create a directory under `newer` called `newerstill` There are two ways to do this what are they? (Hint: You don't *have* to change directories to solve this.)

   (d) Remove all the directories that you've just created, there are several ways to do this. See section § 5.10 on page 141.

   (e) Create the same directory structure with one command. See section § 5.10 on page 141.

3. *Links*

   You may wish to see the handout: `http://nicku.org/ossi/lab/sym-link/sym-link.pdf` for more details about links.

   (a) Create a file called `test` in your home directory (Typing `echo abc > test` should do this). Now create a hard link to `test` called `h_test` and a symbolic link to test called `s_test`

   (b) Find out the inode number of the files. Check you understand why they are what they are.

   (c) Remove the original file called `test`. Can you still get at the contents of the original file?

   (d) What happens if you try `cat s_test`? Make sure you understand the distinction between `h_test`, and `s_test`

   (e) Try to make a *hard link* to your home directory. Why does this fail?

   (f) From your home directory, try to make a hard link from a file in the `/boot` directory to your home directory. In other words, the link should be in your home directory, and when you examine it, you should see the contents of the file in the `/boot` directory. Does it work?

   (g) From your home directory, try to make a *symbolic* link from a file in the `/boot` directory to your home directory. The link should be in your home directory. Does it work? How do you know whether it works properly or not? (*Hint*: look at the `-L` option to `ls`).

   (h) Change to the `/boot` directory. Make a symbolic link from a file in that directory to your home directory. The link should be in your home directory. Make sure that your result is correct. How to check?

   (i) Make a symbolic link from a file in the `/boot` directory to your home directory that is a *relative symbolic link*. A relative symbolic link is one where the *target* does not start with a `/`. Here is an example, in my home directory:

```
$ ls -l kernel.h
lrwxrwxrwx    1 nicku    nicku          19 Feb  4 08:22 kernel.h -> ../../boot/kernel.
```

The target file is the one shown to the right of the arrow "`->`". The link is absolute if it starts with a slash "`/`"; it is relative if it does not start with a slash.

4. `/proc` — *See section 5.33 on page 164*

(a) Use the files in `/proc` to find out how much memory your system has and what processor it is running on.

(b) Find out what PCI devices are attached to your machine.

(c) Find out what environment variables are set for your currently running shell using the information in `/proc`. *Hint* you can get the process-id of your shell using $$

(d) A router does *IP forwarding*; it allows IP traffic entering one network interface to travel though the machine to another network interface, even if the destination IP address is not on the router itself. Whether or not your machine is doing IP forwarding is stored in the file `/proc/sys/net/ipv4/ip_forward`. You can `cat` this file: a value of 1 means that IP forwarding is turned on. Find out whether or not your machine will forward IP.

Note: the file `/etc/sysctl.conf` will change the values of files under `/proc/sys` while the computer is starting up. See man `sysctl.conf` and `man sysctl`.

(e) Find out how many files are currently open on your system. Hint: do

```
$ cat /proc/sys/fs/file-nr
2812    70      52416
$ man proc
```

and then search the manual page for the string "`file-nr`".

5. */etc/fstab*

(a) Add an entry to your `/etc/fstab` file to automatically mount the NFS filesystem `/var/ftp/pub` from `ictlab` on a directory `/mnt/nfs` whenever the computer starts up. See section 5.35 on page 167. See also section 5.37 on page 169.

5.40 Filesystem Solutions

Module 6

Finding Documentation

Objectives

After completing this section, you will be able to:

- Identify sources of information to answer your particular questions
- Be familiar with the main sources of documentation on your machine:
  - man pages
  - info
  - documentation in the `/usr/share/doc` directory.
  - Ultimate documentation: the source code
- Make some of that documentation accessable through a web browser
- Identify all the documentation that is part of any software package
- Be able to locate documentation from the Linux Documentation Project, including *HOWTOs* and *Guides*
- Know how to find information from other people through the Internet, news groups and mailing lists

6.1 Documentation everywhere?

- Linux has a *huge* amount of documentation
- Much of it is installed on your hard disk
- For large distributions, there may be an additional few many megabytes of documentation available (e.g., the documentation CDROM with Red Hat Linux)
- The only problem is: “how do I find the documentation that I need to answer my questions?”

6.2 Where is the documentation on my computer?

- manual pages (read using `man` command)
- info (read using `info` command or `emacs`)
- documentation in the `/usr/share/doc` directory.
- documentation in the `/usr/src/Linux/Documentation` directory.
- The ultimate documentation: the source code

6.3 Some main sources of information from the Internet

- The Linux Documentation Project: `http://tldp.org/` provides these documents in many formats:
  - HOWTOs about a large number of topics
  - Guides: free online books
- Search using Google: `http://www.google.com/`
- Groups in `http://www.google.com/`, and usenet
- Red Hat (and other distributions) provide plenty of online information. These URLs are current 30 Oct 2003: `http://www.redhat.com/docs/`, `http://www.redhat.com/docs/manuals/linux/`. After you have read the *Getting Started Guide*, I particularly recommend the *Red Hat Reference Guide*, and the *Red Hat Customisation Guide*.

6.4 Mailing Lists

- A *mailing list* is a discussion group over email
  - Mostly have a narrow technical focus
  - a great way to get help from an expert on the subject.
- You *subscribe* to a mailing list by:
  - Send email to the list server, or fill in a simple web form
  - The list server sends you an email
  - You reply to that email
- Any subscribed user can send an email to the list
  - All subscribed users will receive that email
  - Set one mail folder for each list
  - Set your email software to filter all list email into the right folder, to keep list email separate from your personal email.
- Examples:
  - Red Hat and Fedora lists: `http://www.redhat.com/mailman/listinfo`
  - the Apache lists: `http://httpd.apache.org/lists.html`
  - the Samba lists: `http://us1.samba.org/samba/archives.html`, plus many others (I subscribe to about 40!)

6.5 Asking Questions on a Mailing List

- Before sending questions to a mailing list, read Eric Raymond's *How To Ask Questions The Smart Way*: `http://www.catb.org/~esr/faqs/smart-questions.html`

6.6 Online Magazines

- Some online magazines include:
  - Alan Cox's Portaloo: `http://www.linux.org.uk/cgi-bin/portaloo` This site is surprisingly useful; it collects the headlines from about 25 web sites with technical news, so that you can review the headlines from all the sites, and click on any headline that looks interesting. This takes you directly to the story. It saves me a lot of time.
  - Apache Week: `http://www.apacheweek.com/`
  - LWN.net (used to be Linux Weekly News): `http://lwn.net/`
  - NewsForge: `http://www.newsforge.com/`
  - Linux Magazine: `http://www.linux-mag.com/`
  - Linux Today: `http://linuxtoday.com/`

6.7 Info

- Many GNU tools are documented properly using the *Texinfo* system.
- *Texinfo* provides two forms of documentation from one source:
  - online hyperlinked *info* pages
  - Nicely formated printed documentation
- Can read *info* pages using the `info` or `pinfo` commands, or (my choice): `emacs`

6.8 Using the `info` command

- To get information about *command*, type:

  `$ info` *`command`*

- For example, to get information about `chmod`, type:

  ```
  $ info chmod
  ```

- The page has “cross references” (like hyperlinks) and menu items.
- You can press the `tab` key to select the next cross reference, and `Enter` to go there
- Go to the **n**ext page with the `n` key
- Go to the **p**revious page with the `p` key
- You can go **u**p, and visit the **l**ast page you just visited (like the back button on your browser)
- Get detailed **h**elp with the `h` key

6.9 Using `emacs` to read `info` pages

- While running `emacs`, you can enter *info* by pressing `C-h i` (That's `Control-h`, then press the key `i`).
- You will see a large number of links. You can search for the right topic by:
  - pressing `C-s` and typing the name you are looking for, or
  - go straight to the menu item by pressing m then the name of the menu item.
- press the middle mouse button while the mouse is over a cross reference or menu item
  - If your mouse only has two buttons, press both buttons at once.
  - If that doesn't work, enable "Emulate 3 buttons" in the `mouseconfig` program, and restart X.
- There are navigation buttons on the top of `emacs`

6.10 Using `rpm` to identify all documentation for a software package

- Suppose:
  - you want to find the information for the Apache web server, `httpd`
  - You type `man httpd`; there is no result.
  - What should you do next?
- Use the RPM package manager.
- The RPM package manager (RPM) is a set of programs to manage installed software.
- RPM maintains a database containing very detailed information about all installed software
- RPM can:
  - list all files in a software package,
  - list all documentation files in the package,
  - list all configuration files in the package,
  - determine if a file has changed since the original installation,
  - verify that a software package is correctly installed,
  - tell you what RPM package any file comes from,
  - . . . and countless other things.

6.11 A quick guide to `rpm`

- Here is a brief list of `rpm` ***q***uery commands. I have used the `httpd` package as an example.

command	effect
`rpm -qa | less`	list all installed software packages
`rpm -q httpd`	show the version of the httpd package, if it is installed
`rpm -qa | grep httpd`	show all installed packages that have *httpd* in their name
`rpm -ql httpd`	***l***ist all files in the httpd package
`rpm -qd httpd`	list all ***d***ocumentation files in the httpd package
`rpm -qc httpd`	list all ***c***onfiguration files in the httpd package
`rpm -qi httpd`	display ***i***nformation about the package
`rpm -V httpd`	***v***erify that the `httpd` package is correctly installed
`rpm -qf /etc/passwd`	determine which package the `/etc/passwd` ***f***ile belongs to

- Further information about `rpm`:
  - `man rpm`
  - The chapter on `rpm` in the *Red Hat Linux Customization Guide*
  - The book *Maximum RPM*, available in html on the Red Hat documentation CDROM.
  - It is worth learning how to build RPM packages.

6.12 A quick guide to `dpkg` (on Debian Linux)

- The Debian Linux distribution is the easiest to upgrade (but not so easy to install), thanks to the wonderful `apt-get` command. For you reference, here is a table comparing some `rpm` and `dpkg` commands:

command	effect
`dpkg --list | less`	list all installed software packages
`dpkg -l httpd`	show the version of the httpd package, if it is installed
`dpkg --list | grep httpd`	show all installed packages that have *httpd* in their name
`dpkg --listfiles httpd`	list all files in the httpd package
`dpkg --print-avail apache`	display information about the package
`dpkg -S /etc/passwd`	determine which package the `/etc/passwd` file belongs to

6.13 Browsing Documentation Via Your Web Server

- You may configure the web server, `apache`, to serve the documentation online
- Two steps may be necessary to achieve this:
  - create a file `/etc/httpd/conf.d/doc.conf` containing:

    ```
    Alias /doc/ /usr/share/doc/
    <Directory /usr/share/doc>
        Options Indexes FollowSymLinks
    </Directory>
    ```

  - enable your web server service:

    ```
    $ sudo /sbin/chkconfig httpd on
    $ sudo /sbin/service httpd start
    ```

- Now you should be able to browse the documentation in `/usr/share/doc` from the URL:

  http://⟨*ip-address*⟩/doc/

  where ⟨*ip-address*⟩ is the IP address or DNS name of your computer.

6.14 The exercises

The aim of these exercises is for you to learn how to find helpful information, not to just sit here reading it now! The information you need to do each of these exercises is provided for you in this chapter.

1. The Linux Documentation Project provides a number of *guides*; these are full length, online books. Locate:
   (a) a book about network administration
   (b) A book about shell programming using `bash`
2. Locate the documentation CDROM image on our server for Red Hat 9, and determine how to burn a copy of it.
3. Go to the location for Red Hat mailing lists, select the `shrike` list, and browse the most recent threaded archive of discussion about Red Hat 9. Subscribe to this list, and set up filtering on your mail client, so that the mailing list information is kept in a separate mailbox from your normal email. See section § 6.4 on page 179.
4. Install a copy of the Red Hat documentation onto your computer from the network filesytem containing the documentation RPMs, and install them. The last step is to read them! Here is how:
   (a) At a prompt, type:
   ```
   $ cd /home/nfs/redhat-9/doc/RedHat/RPMS
   $ sudo rpm -Uhv *.rpm
   ```
   (b) Now view this documentation in your web browser at the location: `file:///usr/share/doc/` and look for the Red Hat "Getting Started Guide" under `rhl-gsg-en-9`. Click on the file `index.html`. I particularly recommend this book for you to get started with Linux.
   (c) The book *Maximum RPM* under `maximum-rpm-1.0` explains much about how RPM works.
5. Visit Alan Cox's "Portaloo" (see section 6.6 on page 181)
   (a) add all the channels by clicking on "Add Everything"
   (b) place a bookmark to this site in your browser
   (c) Note that each rectangle contains current links to headlines on a technical news web site.
6. Visit the slashdot web site at `http://slashdot.org/`
   (a) Click on a "Read more ..." link
   (b) Change your "Threshold" to 3. This means that you only see postings that have a score of 3, 4 or 5.
   (c) See whether the discussion is sensible or not!
   (d) Slashdot uses a *moderation* system, a voting system, where silly posts are "modded down", (given a lower score), while useful posts are "modded up" (are given a higher score). This is essential, because anyone can post, and there are enough silly people in the world to make Slashdot unusable without such a moderation system.
7. View the on-disk documentation for the `bash` shell:
   (a) In the `man` page
   (b) using the `info bash` command
   (c) using the `pinfo bash` command

(d) using info within emacs

8. Determine the location of the documentation for the `bash` software package using `rpm`.

9. Determine what software package the file `/usr/lib/libc.so` belongs to, and find its documentation.

6.15 Documentation: Solutions

Module 7

Administering User Accounts and Permissions with `sudo`

Objectives

- After completing this section, you will be able to:
  - Create user accounts that include the user's full name, using the industry standard program `useradd`
  - Delete user accounts with `userdel`
  - Create and delete groups
  - Add users to, and remove users from groups
  - Set permissions on files and directories that enable and restrict access to users and groups of users.
  - Use `sudo` to run administration programs without becoming root

7.1 System Administration without always being SuperUser

- Working as the SuperUser (root) all the time is a disaster waiting to happen.
- Example: if you run `netscape` as root, and download a malicious Java or JavaScript program, it can execute with SuperUser privileges and destroy the server you are working on, and affect your career badly!
- Trust me: be root as little as possible. Never run X as root. If you are the system administrator, step one = make your own non-root account. Always log in as yourself, not root.
- So how to get work done?
- Solution: `sudo`
- Refer to the notes about configuring and using `sudo`

7.2 Setting your PATH

- When you type a program name, the shell searches a list of directories for it.
- The list of directories is stored in an *environment variable* `PATH`.
- Programs for system administration are kept in directories `/sbin` and `/usr/sbin`. (sbin = **s**ystem **bin**ary).
- Normally these are not put on your path.
- As a system administrator, it is good for you to put these directories onto your path, by editing *your* login script (*not* root's!) `~/.bash_profile` and adding the line:

```
PATH=$PATH:/sbin:/usr/sbin
```

- After saving your log in script, *source* your log in script:

```
$ source ~/.bash_profile
```

- The alternative is to type the full path name for each system administrator command:

```
$ sudo /usr/sbin/useradd -c "Noris Lurka" nlurka
```

- . . . otherwise you will see something like this:

```
$ sudo useradd user1
sudo: useradd: command not found
```

7.3 Linux is a Multiuser System

- Linux is a *multiuser* system
- This is different from what you have experienced with Microsoft operating systems[*].
- Many users can log into the computer at the same time (usually over the network using secure shell—see later)
- All can run programs (graphical and in text mode) interactively on the one computer
- Each user is protected by the operating system from accidental (and some malicious) damage from other users

[*]Although the terminal server is changing NT into a multiuser system.

7.4 User account overview

- Each user belongs to at least one *group*
- Each user has a *user id* and one *group id* for each group they belong to
- These ids are integer numbers
- All account information is stored in the following files:
  - `/etc/passwd` maps user name to user id, main group id, full name, home directory, and default shell (usually `bash` for normal users)
  - `/etc/shadow` maps user name to password, holds user password aging policies
  - `/etc/group` maps group names to group ids
  - `/etc/gshadow` which holds group passwords

7.5 `password` file

- It is a text file
- You looked at this file in sections 1.19 and 1.20, and in question 7b on page 29. It is also discussed in some detail in section 17.1 on page 454.
- If deleted, no one can log into the system!
- It is used when you type `ls -l` to show user names rather than user ids.
- The 7 fields are separated by colons '`:`'
- The manual page is in section 5 (section 5 is about file formats); read it with

  ```
  $ man 5 passwd
  ```

- In the old days, second field was the password, but not now.

7.6 Example passwd file

Here is part of a passwd file:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
adm:x:3:4:adm:/var/adm:
lp:x:4:7:lp:/var/spool/lpd:
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:
news:x:9:13:news:/var/spool/news:
uucp:x:10:14:uucp:/var/spool/uucp:
operator:x:11:0:operator:/root:
games:x:12:100:games:/usr/games:
gopher:x:13:30:gopher:/usr/lib/gopher-data:
ftp:x:14:50:FTP User:/home/ftp:
nobody:x:99:99:Nobody:/:
xfs:x:100:102:X Font Server:/etc/X11/fs:/bin/false
gdm:x:42:42::/home/gdm:/bin/bash
postgres:x:101:233:PostgreSQL Server:/var/lib/pgsql:/bin/bash
squid:x:102:234::/var/spool/squid:/dev/null
nicku:x:500:500:Nick Urbanik,C440,24368576:/home/nicku:/bin/bash
```


7.7 group

- Maps group names to group ids
- Maps users to additional *secondary* or *supplementary* groups (besides their *primary group*)
- The *primary group* is the group specified by the fourth field in the /etc/passwd file.
- Read the manual page with

```
$ man 5 group
```

- Here is part of a group file:

```
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
tty:x:5:
disk:x:6:root
lp:x:7:daemon,lp
mem:x:8:
utmp:x:101:
xfs:x:102:
floppy:x:19:
console:x:103:
gdm:x:42:
pppusers:x:230:
popusers:x:231:
postgres:x:233:
slocate:x:21:
squid:x:234:
nicku:x:500:pam
```


7.8 shadow file

- Current Linux distributions default to using the shadow file to hold passwords
- If it doesn't, consider changing to a better distribution!
- Holds the following information:
  - Login name
  - Encrypted password
  - Days since Jan 1, 1970 that password was last changed
  - Days before password may be changed
  - Days after which password must be changed
  - Days before password is to expire that user is warned
  - Days after password expires that account is disabled
  - Days since Jan 1, 1970 that account is disabled
- Must have permissions like this:

```
$ ls -l /etc/shadow
-rw-------    1 root     root         1114 Oct 30 12:50 /etc/shadow
```

- Here is a line from an /etc/shadow file:

```
root:$1$CBBmBWbw$OOHOLNS.gH1KXsMX6ACC2.:11175:0:99999:7:::134539268
```

7.9 logging in

- Many programs for a user to login; examples shown on next slide
- A group may also have a password (stored in the `/etc/gshadow` file)
- If a group has a password, and you know that password, you can become a member of that group with the command `newgrp`
- You can change or add a group password with the `gpasswd` command.

7.10 logging in—Pluggable Authentication Modules (PAM)

- Many programs ask for a password. Examples:
  - `login`
  - `slogin`
  - `telnet`
  - `ftp`
  - `samba`
  - `su`
  - `gdm` (the Gnome login to X)
  - `xscreensaver`
  - `sudo`
  - ...and many others (see `/etc/pam.d/*`)
- The login programs are *very* important for security; one small mistake, and your system can be cracked.
- To avoid replicating the code in every application, one library handles login for all these programs. Called Pluggable Authentication Modules library (PAM)
- Allows total customisation of login, including replacement or supplementing of password file by other systems, such as:
  - one-time passwords
  - smart cards
  - LDAP servers
  - biometric systems
  - ...

7.11 Adding User Accounts with `useradd`

- Many programs exist for adding users
- GUI programs:
  - `linuxconf` and
  - `userconf`
- one program is standard across many other Unix systems: `useradd`
- To add the user Chan Hei-man with the user name heiman, and unrestricted password age:

```
$ sudo useradd -c "Chan Hei-man" heiman
```

- To set Chan Hei-man's initial password:

```
$ sudo passwd heiman
Changing password for user heiman
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully
```

- Password is not shown as you type it, of course.

7.12 What happens when you create a user account?

- When you create an account with `useradd`, the program does a number of things:
  - The program identifies the highest used user ID number, and adds one to this value as the new user ID number
  - It does the same with the group numbers
  - It creates a home directory
  - It copies the default login scripts and other setup files from `/etc/skel`
  - It changes the ownership of this home directory to the new user and group ID numbers
  - It locks the password and group files, updates them with the new user and group entries

7.13 Local accounts and LDAP accounts

- You have configured your computer to use LDAP authentication
  - All the user account information is stored on an LDAP server
    - LDAP = Lightweight Directory Access Protocol
  - All the information normally stored in the `passwd` and `group` files is stored in the LDAP server
  - LDAP also provides information used by the *automounter*, a program running on the client that automatically mounts your home directory by NFS when you log in
- The automounter manages the directory `/home` all access to `/home` is controlled by the automounter
- You cannot create local directories in `/home`
- . . . so you need to tell `useradd` to use another directory when creating local accounts

7.14 Configuring `useradd` to create local accounts

- First create a new directory where your new local users will have their home directories:

  ```
  $ sudo mkdir /home2
  ```

- Next tell `useradd` where new home directories should be based:

  ```
  $ sudo useradd -D -b /home2
  ```

  - Note that `-D` sets the new defaults for future use of `useradd`
  - The option `-b /home2` selects `/home2` as the **b**ase of all future home directories created using `useradd`

7.15 Creating a group

- The industry standard for creating a new group is `groupadd`.
- To create the group `admin`, type:

```
$ sudo groupadd admin
```


7.16 Adding a user to a secondary group

- The command `gpasswd` can be used to add a user to a *secondary* or *supplementary* group, and also to remove a user from a group.
- A *secondary group* is a group that a user belongs to, which is not their primary group.
- You can add `heiman` to the `admin` group more simply with:

  ```
  $ sudo gpasswd -a heiman admin
  ```

- Note that this will *not* affect any other group memberships `heiman` already has.
- Type:

  ```
  $ man gpasswd
  ```

  for more information

7.17 What groups does this user belong to?

- To find out what groups you belong to, type:

```
$ groups
```

- To find out what groups a user `user` belongs to, type:

```
$ groups user
```

- The `id` command shows information about your groups and user id:

```
$ id
uid=2270(nicku) gid=2270(nicku) groups=2270(nicku),14171(staff)
```

- The `id` command is very useful, and has a number of options. See `man id`

7.18 Effective group ID and `newgrp`

- When a person logs in, any files they create are normally owned by their primary group.
- A user can change their *effective GID* to another group that they already belong to, using the `newgrp` command
- The effective GID is shown on the left when you type `groups`.
- When you create a file, the group owner is the same as your effective GID.

7.19 Directory for a Group Project

- As system administrator, you may need to provide a directory that a group of people can use for a project, so that all can read and write that directory, but access by others is restricted.
- Specifications:
  - directory name is `/var/project`
  - members of the group `admin` are allowed read and write access to `/var/project`
  - other users cannot read or write or change into the directory.
- Change the ownership of the directory so that it is owned by the user `root` and has group owner `admin`:

```
$ sudo chown root:admin /var/project
$ ls -ld /var/project
drwxr-xr-x    2 root     admin        4096 Dec 21 12:27 /var/project
```

- Change the permissions of the directory so that the user root and the group owners have read, write and execute permission, and other users have no access:

```
$ sudo chmod ug=rwx,o= /var/project
# or sudo chmod 770 /var/project
$ ls -ld /var/project
drwxrwx---    2 root     admin        4096 Dec 21 12:27 /var/project
```

- See section 5.18 on page 149 for more about file permissions and `chmod`.

7.20 File permissions for directories

- It is easy enough to understand the read, write and execute permission when applied to an ordinary file;
- What about for directories?
- The *execute* permission is the permission to use the `cd` command to change into the directory.
- The *read* permission is the right to list the contents of the directory with `ls`
- The *write* permission is the write to change entries in the directory
- What is a *directory entry*? For each file in the directory, there are only two items:
  - The file name
  - The inode number

  So the *write* permission is the right to change either of these.
- This *write* permission includes:
  - The right to create, delete or rename a file
  - The right to change the inode number

7.21 Examples of minimum file permission requirements

- This table shows examples of minimum file permission requirements
- It is based on table 2-2 on page 37 of the book *Essential System Administration* by Æleen Frisch, O'Reilly 2002.
- Buy this excellent book! SysAdmins can't work well without it!

Command	minimum access required	
	on the file	on the directory
`cd /var/project`	no file	`--x`
`ls /var/project`	`---`	`r--`
`ls -l /var/project`	`---`	`r-x`
`cat /var/project/user1.txt`	`r--`	`--x`
`echo "hello" >> /var/project/user1.txt`	`-w-`	`--x`
`/var/project/binary-program`	`--x`	`--x`
`/var/project/script-program`	`r-x`	`--x`
`rm /var/project/user1.txt`	`---`	`-wx`

7.22 Set Group ID Directory

- There are three "special permissions" that can apply to a file. You can read about them if you type:

```
$ info chmod
$ info "(coreutils)File permissions"
```

  See section 6.7 on page 182 for more about `info`.

- Here we look at the "set group id" (SGID) permission for directories.
- If this permission is set on a directory, then:
  - if a user makes a change to a file or creates a file in that directory, the file will have group owner the same as the directory.
  - If a user creates a directory in that directory, it too will have the Set Group ID bit set. As with the file, it will have a group owner the same as the group owner of its parent SGID directory.

7.23 Set Group ID Directory — Example

- Let's see the result of creating a file in `/var/project` both before and after adding this permission:

```
$ ls -ld /var/project
drwxrwx---    2 root     admin        4096 Jan  2 13:48 project
$ touch /var/project/test1
$ ls -l /var/project
-rw-rw-r--    1 nicku    nicku           0 Jan  2 13:53 test1
```

  Now we add the set group ID bit to the directory permissions, and see the effect:

```
$ sudo chmod g+s /var/project
$ ls -ld /var/project
drwxrws---    2 root     admin        4096 Jan  2 13:48 project
$ touch /var/project/test2
$ ls -l /var/project
-rw-rw-r--    1 nicku    nicku           0 Jan  2 13:53 test1
-rw-rw-r--    1 nicku    admin           0 Jan  2 13:54 test2
```

- when the directory is not SGID, files I create have my primary group ID.
- when the directory *is* SGID, files I create have the group ID of the directory. This allows others in the group to read, write and change the files created by any group member.

7.24 User Management Exercises

1. Configure your `PATH` as described in section 7.2 on page 195. You should already have done that.
2. Configure `sudo` as discussed in the handout on `sudo`. Again, you should already have done that.
3. Configure the new default base for local home directories, as discussed in section 7.14 on page 207
4. *Create four local user accounts, and create two additional local groups*, using `sudo useradd` and `sudo groupadd`. Set their passwords using `sudo passwd` *`username`*.
5. Add two users to one of the groups (make this their *secondary* group). Similarly, add the remaining 2 users to the other group (make it their secondary group). Do *not* change the users' effective group ID using the `newgrp` command in these exercises: simply leave their effective group ID as their primary group.
6. Create a directory `/var/project`, using `sudo` of course.
7. Set the permissions and ownership as shown in 7.19 on page 212:

   ```
   $ ls -ld /var/project
   drwxrwx---    2 root     admin        4096 Jan  2 13:48 project
   ```

   Note: instead of `admin`, type the name of one of the groups you made in step 4.
8. Log in as one of the users who belongs to the same group as the one who owns the directory `/var/project` by typing:

   `$ su - username`
9. As the user that you logged in as in step 8, Create a file in the directory `/var/project`.
   - **(a)** What are the permissions on that file?
   - **(b)** Which user owns the file?
   - **(c)** Which group owns the file?
10. In another window, log in as the *other* user who is *also* in the *same* group that owns the directory `/var/project`. Edit the *same file* as you created in step 9 and save it. Are you successful? Why or why not?
11. In another window, log in as the *other* user who is *not* in the group that owns the directory `/var/project`. Edit the *same file* as you created in step 9 and save it. Are you successful? Why or why not?
12. Now set the permissions on the directory to be SGID as shown in 7.22 on page 215. Create a second file as the first user, in the group that owns the directory. What are the permissions and ownership of the second file?
13. Can you edit and save the second file as the second user, who is also in the same group that owns the directory? Why or why not?
14. Can you edit and save the second file as a user who is also *not* in the same group that owns the directory? Why or why not? What access rights do you have as a user from a group that does not own the directory?
15. Additional exercise: marked by your supervisor on a scale of 0 to 4. You have thirty minutes to complete this exercise:

    Requirements:

- you are to create three user accounts and one directory.
- two users should have read and write access to the directory, and if one user saves a file, the other must be able to edit and save the file.

  So if these two users are `user1` and `user2`, then if `user1` creates a file in the directory, then `user1` and `user2` should have read `and` write access to that file, but not other users should have write access.
- No other user (including the third you created) should have write access to the directory; they should, however, be able to change into the directory and list the content of the directory, and should be able to read any files created in that directory by the first two users.
- No manual intervention is required by any of the users. It should all just work.

7.25 User Management Solutions

Module 8

Managing Users–quotas

Objectives

On completion of this module you should be able to:

- Understand key Unix user management issues
- Use key Unix user management tools [*]

[*]Many user management issues and tools are dealt with in other GBdirect modules, e.g. admin of users, passwords, groups, permissions, etc is covered in the Key Configuration Files module and the Filesystem modules. This module only covers points *not* dealt with there.

8.1 Checking `/etc/passwd` and `/etc/shadow` with `pwck`

- `pwck` verifies the integrity of password files
- It checks:
  - Number of fields
  - Uniqueness of user names
  - Validity of user and group identifiers
  - Validity of primary groups
  - Validity of home directories
  - Validity of login shells
- Typically, offers to delete error-ridden or duplicate lines
- Checks both `/etc/passwd` and `/etc/shadow` by default
- Can be run in read-only mode (`-r`)

8.2 Checking `/etc/group` with `grpck`

- `grpck` checks `/etc/group` for:
  - Correct number of fields
  - Uniqueness of group names
  - To ensure all group members are defined in `/etc/passwd`
- Typically prompts to delete error-ridden or duplicate lines
- Can be run in read-only mode (`-r`)

8.3 Managing User Connections: `login`, `/etc/securetty`, `/etc/usertty`

- `login` configures user connections
  - Most notably by setting what's available at login time
  - Part of the Shadow software suite
  - Works closely with `/etc/login.defs`
  - See `man login`
- `/etc/securetty` restricts the terminals that system administrators can use
  - One device name per line (no `/dev` prefix)
  - Consulted by `login`
- `/etc/usertty` restricts general user access
  - According to user ID and/or time of login
  - One rule per line, 3 colon separated fields
    1. Terminal (no `dev` prefix), * means all
    2. Users, * means all
    3. Lists of allowed times
       - **Times** in HHMM format, e.g. 0800-1732
       - **Days** Su Mo Tu We Th Fr Sa Wk (mon-fri) Al (all days, default)

8.4 Limiting User Resources with `ulimit`

- `ulimit` prevents users exhausting system resources
  - By limiting access to those resources
- Built-in to `bash` and `Ksh` shells
- Two types of restriction:
  - Soft limits, set default resource usage when a process is created, variable
  - Hard limits, set upper threshold soft limits can't exceed
- `ulimit -a` displays current soft limits
- `ulimit -Ha` displays hard limits
- There are about a dozen more options
  - See `man ulimit` for details

8.5 Managing Disk Use with Quotas

- Limit the amount of fs storage a user can consume
- Available on most Unix systems
  - As optional patch to Linux kernel *
  - Not on SCO UNIX
- Two distinct types:
  - Hard limits can *never* be exceeded
  - Soft Limits allow temporary excesses, e.g.
    - For a period of time
    - For a number logins
- Typically applied to `/home` filesystems
- *NOT* `/tmp` or `/`
- Main Commands:

Command	Description
`quotaon, quotaoff`	turn file system quotas on and off
`quota`	display disk usage and limits
`repquota`	summarize quotas for a file system
`quotacheck`	scan a file system for disk usages
`edquota`	edit user quotas
`quotactl`	manipulate disk quotas

*Must be made available either by choosing to enable quotas at installation time, or by compiling the option into a new kernel. Kernel compilation is the subject of another GBdirect training module entirely.

8.6 Setting up Quotas on a Filesystem

- Set quotas on a filesystem in the 4th field of `/etc/fstab` *

```
# device   directory   type  options
/dev/hda1  /           ext2  defaults
/dev/hda2  none        swap  sw
/dev/hda3  /usr        ext2  defaults
/dev/hdb1  /usr/users  ext2  defaults,usrquota,grpquota
/dev/hdb2  /usr/src    ext2  defaults,usrquota
none       /proc       proc  defaults
```

- Create quota records `quota.user` and `quota.group` at the root of relevant filestems: †

```
$ cd filesystem
$ su
Password:
$ touch /partition/quota.user
$ touch /partition/quota.group
$ chmod 600 /partition/quota.user
$ chmod 600 /partition/quota.group
$ halt -r now
```

*BSD introduced quotas and configured them in this way. See `man fstab` for details. Most actively developing Unixes e.g Linux, FreeBSD, NetBSD, etc follow the same pattern. System V format fs config files simply change `rw` to `rq`. AIX puts "`quota = userquota,groupquota`" in `/etc/filesystems`.

†Record files `quota.user` and `quota.group`, should be owned by root, and `read-writeable` by root alone.

8.7 Specifying Quotas for Users and Groups

- Set Users' quotas using `edquota`
- Invoking it with a user or group name creates a tmp file containing hard and soft limits for them

  `$ edquota` *`username(s)`*

- Then opens the file with the editor specified in `$EDITOR` environment variable
- Each line describes one filesystem *

```
Quotas for group www:
  /dev/hda4: blocks in use: 5349, limits (soft = 8000, hard = 10000)
  inodes in use: 1745, limits (soft = 3000, hard = 4000)
```

- If saved before exit, the editor auto-writes details to the quota records
- Can be used just on the command line, e.g.

  `$ edquota -p davef lee julie`

  Sets Julie and Lee's quotas to match Dave's

- Options:

Command	Meaning
`edquota -u`	Set individual user quotas
`edquota -g`	Set group quotas
`edquota -t`	Set grace period in days, hours, minutes or seconds
`edquota -p` *`username`*	Set others' quotas equal to match *username's*

*Formats vary between Unixes, the example above is from Linux.

8.8 Checking and Reporting on Quotas

- Use `quotaon` to activate quota system and enable quota checking

`quotaon` *`filesystem`*	Enable quota system on specified fs
`quotaon -a`	Enable on all filesystems

- `quotaoff` does the obvious
- `quotacheck` looks for consistency
  - Within quota records
  - Between records and current disk usage

`quotacheck` *`filesystem`*	Check consistency on specified fs
`quotacheck -a`	Check on all filesystems

- `quotaon -a` and `quotacheck -a` should run at boot time, i.e. be in system init scripts
- `repquota` reports the current quotas for specified filesystems
  - Can take multiple filesystems as arguments
  - Can report on all filesystems (`-a`)
- `quota` gives ordinary users basic info on current quota status

8.9 Managing Users–quotas: Exercises

1. *Checking Password and Group Files*
   (a) Use `pwck` and `grpck` to check their respective user detail files, twice.
      i. Begin in read-only mode.
      ii. Switch to write-mode if you find errors,
   (b) If you found no errors, assume super-user status and introduce some. Then try these commands in write-mode.
   N.B. Don't mess with the root user's details
2. *User Connections*
   (a) Create a new fictitious user on your system
   (b) Edit `/etc/usertty` to prevent their access at particular times and terminals
   (c) Attempt to login as the fictitious user at a "banned" terminal/time.
3. *Quotas*
   (a) Set quotas for a fictitious new user on their `/etc/home` directory
   (b) Use the quota checking tools to test your set up
   (c) Try to figure out and implement a practical test to see that it is really working on a live system.

Module 9

Introduction to Editing With `vi`

Objectives

In this section, you will learn how to:

- Use the `vi` editor to view, create and edit files
  - the `vi` screen layout
  - move round in files
  - replace, insert and change text
  - search files

9.1 Text editors under Linux

- There are a number of text-editors available
- `vi` is on virtually every Linux distribution
- Also comes with 99% of Unix systems
- Everyone should have a basic understanding
- `vi` is like Linux
  - Has some very complex and powerful functions that can make your life easier
  - However, you don't *have* to know everything; you get by knowing the basics
  - Shares key bindings with many utilities
- We'll just cover the basics here, `vi` is too big to cover everything!

9.2 vi and your terminal

- vi is fundamentally text-based
  - Graphical adaptations *are* available (gvim)
- Needs to know your terminal's capabilities
  - May not function if your terminal is misconfigured
  - Check your TERM environment variable
  - Terminal capabilities are listed in /etc/termcap
  - Generally not an issue except with Windows 98 telnet client. When use windows telnet, always type export TERM=vt100 before starting vi and other such programs, or they will not work properly.

9.3 `vi` screen layout

- Lines containing simply a ~ show that you are past the end of the file and there is nothing here.
- The terminal's bottom line is the *status line*
    - Shows status messages
    - Where you type some commands
      (The 'ed'/'ex' command set, explained later)

```
This is a test document
Some lines of text here

One, two, three
four, five, six
~
~
~
~
~
~
"test" 5 lines, 77 characters written
```

9.4 Opening files with vi

- Launch `vi` by typing its name on command line
- With no arguments `vi` starts with an un-named and empty buffer
- `vi filename` opens a specific file
- If you don't have write permission on a file the status line will tell you :

  ```
  "/etc/aliases" [readonly] 152 lines, 3215 characters
  ```

- If there is no such file status line will say something like :

  ```
  "some_filename" [New File]
  ```

9.5 vi Modes

- Unlike many editors vi does not always insert what you type into the file
- Has several 'modes'
  - Only one is responsible for inserting text into the current file
- vi has 3 modes: *

command mode	Moving the cursor, searching and manipulating existing text
insert mode	Entering new text
':' ('ed') mode	File manipulation, advanced searching and substitution

- vi starts in command mode
- Return to command mode at any time by hitting <ESC>

*Some people refer to "ex" instead of "ed". They are the same thing

9.6 Saving, changing file and quitting

- When you open a file, a copy of it is opened into memory
- Any changes you make apply to this copy *only*
- File on disk only changes if you explicitly say so
- To save (or *write*) a file you must be in command-mode, then type `:w`
- Can save your file under a new name, e.g.

```
:w newfilename
```

- To quit `vi` type `:q`
- `vi` normally prompts you if you have unsaved work
- To quit without saving your work type `:q!`
- `ZZ` will save your work and then quit

9.7 Moving around in command mode

- Many ways to move around a document
- You must be in command mode for the following :
  - On ‘friendly’ terminals you can use arrow keys
  - Arrow keys are sometimes unavailable on some terminals so `vi` has some alternatives

	k	
h		l
	j	

- Although ‘awkward’ at first, these make your life easier
  - Always work, regardless of system type
  - Fingers stay on the ‘home’ keys

9.8 Numeric Prefixes

- Key concept: ‘numeric prefixes’ or ‘multipliers’
  - Vastly improves the usefulness of many commands
- To supply a prefix simply type the number before the command
  - `vi` will then perform the command the specified number of times.
- Note: In subsequent examples a small box indicates the position of the cursor
- Starting with

```
The quick brown
fox jumped over
the lazy dog
```

and pressing `2l` will result in

```
The quick brown
fox jumped over
the lazy dog
```

9.9 Further Movement

- `vi` also allows movements by units other than characters.
- Moving by pages :

Key	Result
`^f`	Forward one screenful
`^b`	Back one screenful
`^u`	Forward half a screenful
`^d`	Back half a screenful

- Moving by ‘words’ :

Key	Result
`w`	Go to beginning of next word
`e`	Go to end of next word
`b`	Go to start of previous word

- For these commands punctuation is not counted as part of a word
- The commands `W`, `E` and `B` act the same but *do* include punctuation in words
- NOTE: Case is important to `vi` commands, `b` and `B` are different commands!
  - The upper and lower case versions of commands are *usually* related

9.10 Further Movement — Example

- From

  This, he $\boxed{\texttt{s}}$aid, is an example

Key	Result
w	This, he said$\boxed{\texttt{,}}$ is an example
W	This, he said, $\boxed{\texttt{i}}$s an example
e	This, he sai$\boxed{\texttt{d}}$, is an example
E	This, he said$\boxed{\texttt{,}}$ is an example
b	This, $\boxed{\texttt{h}}$e said, is an example
B	This, $\boxed{\texttt{h}}$e said, is an example

- As with virtually all commands these may be given a numeric prefix
- From the original start-point :

Key	Result
2w	This, he said, $\boxed{\texttt{i}}$s an example
2W	This, he said, is $\boxed{\texttt{a}}$n example
2e	This, he said$\boxed{\texttt{,}}$ is an example
2E	This, he said, i$\boxed{\texttt{s}}$ an example
2b	This$\boxed{\texttt{,}}$ he said, is an example
2B	$\boxed{\texttt{T}}$his, he said, is an example

- It's not *necessary* to know these, but they make life a lot easier when you get used to them!

9.11 Movement by lines

- What if we want to get to the *beginning* of the next line*?
- Commands to move to line start/end:

Key	Result
$	Move to the end of the current line
0	Move to start of current line
^	Move to first character of line

- Moving to start of a previous or subsequent line

Key	Result
+	Move to beginning of the next line
–	Move to beginning of the previous line

*A 'line' is the set of characters contained between newline characters, not necessarily what appears on one line in your terminal

9.12 Movement by lines — Examples

- In each case here, we start from:

```
This, he said,
is a most
interesting example
```

Key	Result
+ <RET>	This, he said, is a most interesting example
-	This, he said, is a most interesting example
0 or ^	This, he said, is a most interesting example
$	This, he said, is a most interesting example

9.13 Inserting text

- You probably want more from a text editor than the ability to move a cursor!
- At the bare minimum you need to be able to insert text into a file
- Don't worry, `vi` does this with ease
- As with everything else, though, there's more than one way
- Again, while this may seem confusing, you only *need* to know the bare minimum
- *But*, the more you know, the easier your life becomes!

9.14 i command

- The i command inserts text before the cursor
- This places vi into 'insert' mode
- Anything you type now is treated as text to insert into the file rather than as a command
- You leave insert mode by typing <ESC>
- This is insertion at its simplest!
- To insert text *after* the cursor we use the a (append) command
- Also :

Key	Result
A	Append at the end of the line
I	Insert at the beginning of the line
o	Create blank line below cursor for insertion
O	Create blank line above cursor for insertion

- If your cursor keys work then you may move around the line while in insert mode
- You can delete characters from the current insertion using backspace

9.15 Multiple Insertion

- Insertion commands can take numeric prefixes
- The result may be surprising!
- Consider the following sequence of keypresses (from command mode) in an empty document

```
5i 
```

- The result will be :

```
     
```

9.16 Deleting Text

- `vi` has a vast array of commands for deleting text
- The ‘odd-one-out’ is `x` which deletes the character under cursor
- The rest of the deletion commands are based-around the easy to remember `d` command
- `d` on its own does nothing
- You have to tell it how much to delete
- The amount to delete is given by the keys you used when studying movement

  Example:

Key	Result
`dw`	Delete to the beginning of the next ‘word’
`3dw`	Delete 3 ‘words’
`de`	Delete to the end of the ‘word’
`db`	Delete everything before cursor to the beginning of the ‘word’
`d$`	Delete to the end of the line
`d0`	Delete to the beginning of the line

- Two more special cases :

Key	Result
`dd`	Delete the entire line
`D`	Delete to the end of the line

9.17 Changing Text

- Now we know everything we need to know to delete text, insert new text and save changes
- `vi` however likes to give us choices!
- If we find a word that is wrong, we can delete it and insert the replacement
- We're *actually* 'changing' the word
- `vi` has a family of commands for just this, all starting with `c`
- Similar to deletion, i.e. you can use `cw` to change a word, `c$` to change to the end of the line, or `3cw` to change three words
- What actually happens is that the designated amount is deleted and you are placed in insert mode

Key	Result
`cw`	Change a word
`3cw`	Change 3 words
`c$`	Change to the end of the line
`c0`	Change to the beginning of the line

9.18 Copy and Paste

- We’re still missing the ability to copy a piece of text and paste it somewhere else
- `vi` does support this, but it calls it ‘yanking’ and putting
- All ‘yanking’ commands are prefixed with a `y` and follow the same rules as before, i.e. `yw`, `y$`, `3yw`
- `yy` and `Y` yank a whole line and the rest of a line, respectively
- Paste text using `p` or `P`
  - `p` pastes text *after* the cursor
  - Uppercase `P` pastes it *before*
- Deleted text is also considered to be yanked
  - `xp` will transpose two characters

9.19 Finding your place

- You can search through a file using `/`
- You will get a `/` as a prompt on your status line
- To search for the string `exam` type

  ```
  /exam
  ```

  and press `<RETURN>`
- If `vi` found your search string it will move the screen to a relevant place and highlight it
- `n` will skip to the next occurence; `N` to the previous
- Search backwards by using `?` instead of `/`

9.20 Miscellaneous Commands

- `vi` has a number of commands that don't really fit anywhere else
- `~` toggles the case of character under cursor
- `.` repeats the last action
- `u` undoes the last action
  - Linux `vi` supports multilevel undo
  - Standard `vi` does *not*
- `J` Join the current and following line

9.21 Search and replace

- `vi` can also replace the words it finds
- Basic form is:

  `s/searchfor/replacewith/modifier`

- By default it only changes one occurence per line, and only checks the current line
  - If we tag the `g` modifier on the end it will replace all matches on the current line
- If we use a range* we can search and replace a specified part of a document, e.g.
  - To search and replace from lines 10 to 15 inclusive:

    `:10,15 s/foo/bar/g`

  - To search and replace on the whole document

    `:1,$ s/foo/bar/g`

*Not explained here, this is an advanced topic

9.22 Regular Expressions

- Sometimes it's desireable to search for a word 'fuzzily'
- You may know the start of a word, or the end
  - Or both, but not the bit in the middle!
- *Regular expressions* can come in useful here
- Can be used in normal searches or 'search and replace' commands

9.23 Regular Expression Conventions

- Lots of things in Linux use regular expressions
  - Not all ‘exactly’ the same
  - 95% similar though
- Defines certain *special characters*

Character	Result
`.`	Match any character
`[a-z]`	Match any character in the range a to z
`*`	Match the preceeding character zero or more times
`^`	Match the beginning of a line
`$`	Match the end of a line
`\<`	Match the beginning of a word
`\>`	Match the end of a word

- Strictly speaking * can apply to more than one character
  - We won’t cover that here

9.24 Regular Expression Examples

- Suppose we want to find all words ending in `ent`
- We could read the entire document to check for `ent` by hand
  - Takes far too long
  - We'd probably still miss some
  - Easier to get the computer to do it
- We could do a search using `/ent<RET>`
  - Unfortunately that would also match words beginning with `ent` or with `ent` in the middle
- `/ent\>` will jump to the next word that ends with `ent`

9.25 Regular Expression Replacement

- We can also use regular expressions in the search section of search and replace commands, e.g.

  ```
  s/\<foo/bar/g
  ```

  will replace all occurences of `foo` at the beginning of a word with `bar`

9.26 Help

- `vi` on Linux has very extensive online help.
- There is an interactive tutorial too; try it out.

9.27 vi Exercises

1. *Recognizing vi*
   - **(a)** Start up vi with no filename to see what it looks like
   - **(b)** Exit vi and then start it again with the file /etc/passwd
   - **(c)** What can you tell about the file from this screen?
2. *Getting used to vi*
   - **(a)** Start vi with the the file /etc/passwd again
   - **(b)** Practise the basic movement commands on the file
   - **(c)** Check you can use both the cursors and hjkl to move around
   - **(d)** Check the other movement commands work as expected
3. *Creating with vi*
   - **(a)** Start vi with the filename vi_test. This should be a new file
   - **(b)** Insert your name into the file and then save it and leave vi
   - **(c)** Open the file again and check it still contains your name
   - **(d)** Next add some more names to the file, one on each line
   - **(e)** Go to a name roughly half way down your list. Check you can insert a name on the line above, and on the line below
   - **(f)** Check you can append to the end of lines and insert at the beginning of lines
4. *Movement and Multipliers*
   - **(a)** Check you can move through your file using combinations of the movement keys and numeric prefixes.
     For example
     - **i.** Move 3 lines down at a time
     - **ii.** Move 2 words along
     - **iii.** Move to the beginning of the second line below your cursor
5. *Deleting with vi*
   - **(a)** Try deleting various entities (Words, lines, characters) from your file
   - **(b)** Check that these work with the numeric prefixes
   - **(c)** You should be able to achieve all of the following
     - **i.** Delete a word
     - **ii.** Delete to the end of the line
     - **iii.** Delete to the beginning of the line
     - **iv.** Delete the whole line
     - **v.** Delete 2 lines at once
     - **vi.** Delete 2 words at once (Either including or excluding punctuation)
6. *Changes with vi*
   - **(a)** Repeat the exercises given for delete but do changes instead of deletions

7. *Yanking and Pasting*

   (a) Copy the first line of your file and paste it so that it becomes the last line

   (b) Paste it back at the top of the file

   (c) Place the cursor at the very beginning of the file and try the following keystrokes

      i. `yyjyyp`

      ii. `2yyp`

   (d) What was the difference and can you suggest why this may be?

   (e) Check that text deleted can be pasted back

8. *Miscellaneous*

   (a) Place the cursor at the beginning of the file and try the following command sequence:
   `yyp...`
   Explain the result

   (b) Place the cursor over a letter on the middle of a word. What happens when you type `xp`?

   (c) Join all the lines of your file into one long line. Check that the movement commands regarding lines work on *actual* lines rather than the lines as seen on your screen

9.28 `vi` Solutions

1. *Recognizing vi*
   - **(a)** Check you understand where the status line is, and what the ~ characters mean
   - **(b)** `:q` should exit `vi`. If you want to make sure you're in command mode press `<ESC>` first. `vi /etc/passwd` will start `vi` with `/etc/passwd` opened
   - **(c)** `vi` should tell you that this file is read only. This is because you don't have sufficient permissions to change the file. `vi` should also tell you how many lines and characters are in the file.
2. *Getting used to vi*
   - **(a)** `vi /etc/passwd`
   - **(b)** You should be fairly comfortable with the various navigation methods such as moving left, right, up and down, to the end or beginning of the line and moving up and down by intervals of pages and half pages.
3. *Creating with vi*
   - **(a)** `vi vi_test`. The status line should tell you that it is a new file and each line on the main screen should begin with a ~ indicating lack of content
   - **(b)** To insert my_name simply type:
     `imy_name<ESC>`
     There are several ways to save and exit:
     - **i.** `:w` followed by `:q`
     - **ii.** `:wq`
     - **iii.** `ZZ`
   - **(c)** `vi vi_test` and check that the text you entered is there. If not try again.
   - **(d)** There are several ways to do this:
     - **i.** When inserting using `i` you may type RETURN to insert a newline character. it is possible therefore to start with the cursor at the beginning of the file and type: `iname1<RET>name2<RET>name3<RET>` and so on
     - **ii.** Typing `o` or `O` will open a new line for insertion
   - **(e)** You should check that you understand which of `o` and `O` inserts above, and which below the current line
   - **(f)** Appending to the end of a line can be done using either:
     - **i.** `$atext_to_append<ESC>`
     - **ii.** `Atext_to_append<ESC>`

     Inserting at the beginning can be done using any of:
     - **i.** `^itext_to_insert<ESC>`
     - **ii.** `0itext_to_insert<ESC>`
     - **iii.** `Itext_to_insert<ESC>`
4. *Movement and Multipliers*
   - **(a)** You should practice moving around using the movement characters with the numerical prefixes
     - **i.** `3j`, `3+`, or `3<RET>`

   ii. `3e`, or `2w`
   iii. `2+`, or `2<RET>`

5. *Deleting with vi*
   (a) You should make sure that the various deleting methods work as you expected. If they surprise you, try to work out how they *do* work.
   (b) Again check you understand the various possibilities.
   (c) The following represent only possible solutions:
      i. `dw`
      ii. `d$`
      iii. `d0`
      iv. `dd`
      v. `2dd`
      vi. `2dw`

6. *Changing with vi*
   (a) The answers for this are the same as for delete except substituting `c` for `d` in each case.

7. *Yanking and Pasting*
   (a) Move to the first line of the file and type `yy`, then move to the end of the file and type `p`
   (b) Move back to the top line of the file and type `P` which will paste it above the current line.
   (c) Check you can tell the difference between the two commands.
   (d) The 'Yank buffer' only holds the contents of one yank operation. Both sets of keypresses yank the line we start on and the line below. However the first does this as two seperate operations and the 'yank buffer' only remembers the most recent. The second example yanks two lines at once, therefore placing both in the yank buffer.
   (e) You should try `ddp` and check that the text appears after being deleted.

8. *Miscellaneous*
   (a) `.` repeats the last action. In this case it is a paste operation. It could equally well have been an insert, change word or delete operation.
   (b) The `xp` command pair is useful for transposing letters.
   (c) Starting at the top of your file pressing `J` will join the following line to the current line. Repeat this until the entire file is on one line. Pressing one of the down a line keys (Such as `j`, `+` or `<RET>` should have no effect despite the illusion that there is more than one line.

Module 10

Basic X-Windows

Objectives

- On completion, you should be able to:
  - Understand the basic concepts behind networked X windowing
  - start and stop X
  - run shells and user applications under X
  - set preferences for X
  - change window managers and desktops
  - use X over a network

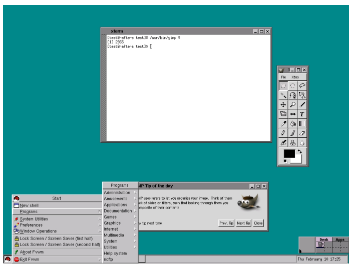
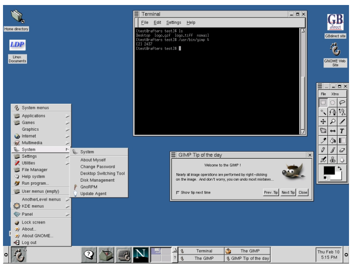
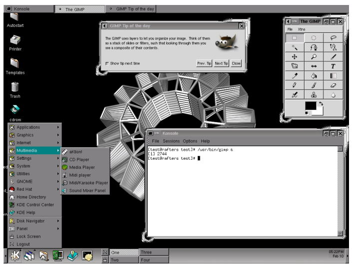
10.1 What X-Windows Is

X is a windowing system

- Provides the basic graphic functions for Linux
- Designed to provide windowing to any workstation across a network, regardless of OS
- Operates on a client-server model
- Is an *application*, i.e. not a part of the OS
- The standard Linux X server is `Xfree86`, commercial alternatives include:
  - Metro-X
  - Accelerated-X

10.2 X Needs Window Managers

- Window managers provide the controls which allow you manipulate all graphic apps, e.g.
  - move, size and stick
  - open and close
  - maximize, minimize, iconize
  - title bars
- Determine the 'look and feel' of X, e.g.
  - Win95
  - Motif
  - Next Step
- Can provide 'virtual desktops'

10.3 Window Managers Are Applications

- Linux distributions contain many window managers, e.g.

Manager	Description
`sawfish`	Used with Gnome, lightweight, programmable
`fvwm2`	Motif-like look
`fvwm2`	Win95-like configuration for `fvwm2`, Red Hat default
`twm`	Bare-bones Tab WM
`olwm`	Open Look (Sun)
`olvwm`	Virtual Screen Open Look
`WindowMaker`	Next Step look, fast and lean
`enlightenment`	Gnome-compatible WM, powerful, rich, heavy weight
`wm2`	Ultra-lean

- Window managers are X applications, thus:
  - change manager without re-starting X
  - change X behaviour without re-start

10.4 Desktop Environments

- X + WM alone don't provide everything expected of modern desktops, e.g.
  - completely integrated drag and drop
  - universal access to a clip board
- Desktop Environments bring these facilities to Linux, bundling:
  - desktop-capable window manager
  - URL-based file manager
  - facilities to share clipboard and other data between optimized apps (inc. object linking)
- Linux currently has 3 main desktop environments:
  - CDE . . . the original commercial UNIX standard
  - KDE
  - GNOME — now adopted by Sun Microsystems as the window desktop for Solaris. Helix Gnome is excelent and is available from `http://www.helixcode.com`.

10.5 Starting X

- Many possibilities
  - You may be using the graphical `xdm` tool which does it for you
  - or login to command prompt then type

    `$ startx`

    if `.xinitrc` is not setup:

        $ xinit
        $ window manager filename

10.6 Stopping X

- Stopping:

  `<CTRL>+<ALT>+<BACKSPACE>`
- Use the window manager's menus
- If you started via `xinit`, type the following in the startup xterm:

  `$ exit`
- If all these fail, switch to another virtual terminal using the following keys, then kill X from the command prompt:

  `<CTRL>+<ALT>+<F1> ... <F4>`

10.7 Running Shells (Xterms) Under X

- Even under X, the most productive way to work is often via the command line (i.e. a shell)
- The standard way to access a shell prompt under X is via a terminal emulator called an `xterm`*
- An `xterm` shell behaves like a non-X shell, except that you can cut and paste between it and X applications
- Any number of xterms can be open at the same time
- Using `ssh` (or if you are desperate, `telnet`) the `xterms` can provide shells to any number of other hosts
- To start an `xterm`:
  - From an already open xterm:

    ```
    $ xterm &
    ```
  - From a window manager menu (invariably top-level)

*Linux provides other terminal emulators for specialised hosts, but they are rarely necessary. There is also another category of emulators that provide advanced features such as transparent terminals etc.

10.8 Running Applications from an `xterm`

- Character-based apps:
  - Run exactly as they would outside X, unless the xterm itself has been misconfigured
- X applications:
  - Type the program's file name at the prompt: [*]

$ *filename* &

[*]The ampersand allows the application to run independently of the shell

10.9 Running Applications from a window manager

- Every window manager provides simple menu- based access to applications
- Application Menus are usually accessible by clicking `Mouse Button 1` on:
  - Buttons set into a task bar
  - The desktop background (root window)

10.10 Configuring X

- Default installations of Linux provide a fully functional setup for using graphic X apps
- 2 different types of X configuration that system administrators or users may need to change:
- Basic configuration of screen, mouse, keyboard behaviour, fonts
  - Could be a course in itself (classic O'Reilly manual fills a bookshelf)
  - Configuration files best edited via config tools (see next Section 10.11)
- Behaviour of desktop objects (windows, icons, taskbars, `xterms`)
  - Window manager dependent
  - Best configured via window manager preferences

10.11 Basic X Hardware Configuration

- Basic configuration for hardware is defined in the `XF86Config` file, located in `/etc/X11` *
- `XF86Config` is easier to edit using the following tools:
  - `Xconfigurator` . . . Red Hat tool sets monitor, card, screen mode, colour depth and resolution with probing
  - `XF86Setup` . . . an X application which edits most basic hardware preferences (Mouse, Keyboard, Card, Monitor, Graphic Modes)
  - `xf86config` a character-based application which prompts for the same settings
  - `mouseconfig` . . . Red Hat tool sets the mouse type with probing. Useful for setting 2-button mice to emulate 3-button types by simultaneous clicking on both buttons

*On some older systems you may find the X configuration in `/usr/lib/X11`,`/usr/X11R6/lib/X11` or `/var/X11R6/lib/`

10.12 Basic X Software Configuration

- Under X, the user can configure every conceivable aspect of graphic display
- Users may need to change:
  - Screen font sizes, styles, familes
  - Pointer behaviour
  - Screen colours
  - Window manager
- All desktop environments and many window managers provide graphic tools for changing these configurations
- They can be set, on a system-wide or per-user basis, in the following two files:
- `.xinitrc`
  - to set the default window manager and style to be used by the `startx` command
- `~/.Xresources` or `~/.Xdefaults`
  - for fonts, pointers, colours, etc
  - Read this file after changing it with `xrdb ~/.Xresources`

10.13 Networked X — The Client-Server Relationship

- X works in a client-server relationship
- The client is a user application (e.g. netscape) which needs X services to display itself on a given screen
- The server is the application which provides these services, e.g. `Xfree86`
- On a single-user Linux system, both apps reside on the same system
- On a networked Linux system the user can run an X application which is installed on a remote system but see it displayed on the local monitor, i.e.
  - The client application (e.g. `netscape`) is remote and the X server (e.g. `Xfree86`) is local

10.14 Principles of Running Remote X Apps

- The most common use for networked X is to run client apps which are installed on remote hosts
- Reasons for running X apps on remote hosts:
  - Using graphical tools to administer a remote machine
  - No local installation of the app
  - Local processing or memory are insufficient
  - No local access to data

10.15 How to Run Remote X Apps

The right way (and simplest way) to do this is to use `openssh`

- Simply type:

  `$ slogin` *`remotehost`*

- . . . then type any commands to start a graphical epplication in the terminal window; it just works.

The wrong way:

- Start the local x server:

  `$ startx`

- Enable (dangerous) lack of authentication

  `$ xhost +`*`remote-hostname`*

- Open a telnet connection to the remote host:

  `$ telnet` *`remote-hostname`*

- Set the your `$DISPLAY` *environment variable* on the remote host so that applications re-direct their graphic output to your local monitor:

  `$ export DISPLAY=oakleigh:0`

10.16 Authentication

- Xservers only allow authenticated hosts to connect
- The right way to connect to a remote host is to use `openssh`. All the other methods described here are difficult, insecure, troublesome... So don't use them.
- On a trusted LAN you might use `xhost` in an xterm
  `$ xhost +beehive`
- Or edit `/etc/X0.hosts` (0 refers to display 0):

```
$ cat /etc/X0.hosts
landlord
kebab
samosa
```

- This is dangerous
  - Allows hosts to grab your mouse and keyboard
- *Only use in a trusted environment*

10.17 Better Authentication

- Can use cookie-based authentication
- Done for you if using `xdm` or `ssh`
- Clients look in `~/.Xauthority` for cookies to feed to server
- Server must be started with appropriate argument
  - Reads *its* `~/.Xauthority` file
- Server only looks when started
  - Too late to change once running
- Both server and clients must use the same cookies
  - Involves merging `~/.Xauthority` files using `xauth`
- Hard to manage — most resort to `xdm` for local access or `ssh` for remote access.
- Documentation is not very penetrable; `ssh` is much simpler to use, and is secure.

10.18 Basic X Exercises

1. Figure out how to get an X session running

2. In an xterm window type `'xterm'` — what happens?

3. In an xterm window type

   ```
   export DISPLAY=xyz xterm
   ```

   what happens?

4. Start up another xterm.

   (a) Type:

   ```
   echo hello
   ```

   You should get 'hello' echoed.

   (b) Select the `echo hello` text so that it highlights — do this by clicking the first mouse button and dragging.

   (c) Move the mouse to another xterm window; click into it to make it active if necessary.

   (d) You should be able to paste the selected text by clicking the middle mouse button (3 button mouse) or simultaneously clicking both buttons on a 2 button mouse. Try it and see.

5. Find another machine on the same network. Use `xhost` to tell it to accept connections from your machine. Start an xterm on your machine but tell it (using the `DISPLAY` variable) to display on the remote machine.

6. Go to the `/usr/X11R6/bin` directory. Try these commands:

```
xeyes
xsnow
xfishtank
xbill
xdemineur
xclock
```

and any others that you like the idea of.

Module 11

Fundamentals of TCP/IP

Objectives

This module is intended as an introduction to the the basic concepts of IP networking. By the end of it you should understand:

- The history and uses of various protocols
- How subnetting and netmasks work
- About interfaces
- The use of ports

11.1 Fundamentals of TCP/IP Networking

Key concepts:

- Packets
- TCP vs UDP
- Services
- Subnetting inc /xx form
- Routing

11.2 History

- Developed by ARPA for university & military research
- Robust, reliable, wide area network protocol, system-independent
- Will route traffic around network outages (if routing protocols used)
- Came into widespread use in mid-late 1970s
- Popularity hugely helped by free availability of the BSD Unix implementation

  i.e. the pre-Linux reference platform
- Now the standard protocol — the Internet based totally upon it

11.3 Recap of basic IP Concepts — Components

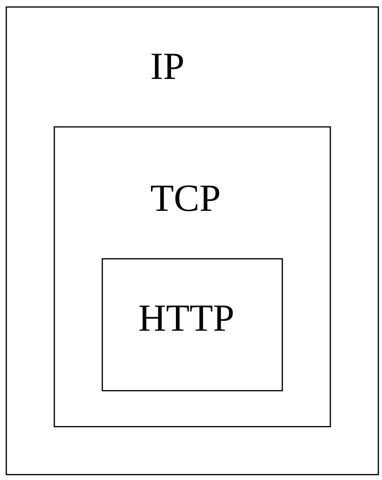
- Properly, The Internet Protocol Suite (IP Suite)
- Usually erroneously referred to as TCP/IP
- Consists of numerous protocols
- IP is used to encapsulate:
  - TCP (Transmission Control Protocol)
  - UDP (User Datagram Protocol)
  - ICMP (Internet Control Message Protocol)
  - other routing & management protocols

11.4 IP versions

- Currently at Version 4 (IPV4)
  - Entire Internet based on IPV4
  - Quickly running out of spare numbers
- IPV6 well standardised
  - Important improvements
  - Currently in miniscule use
  - Migration will occur eventually
  - Support already in Linux

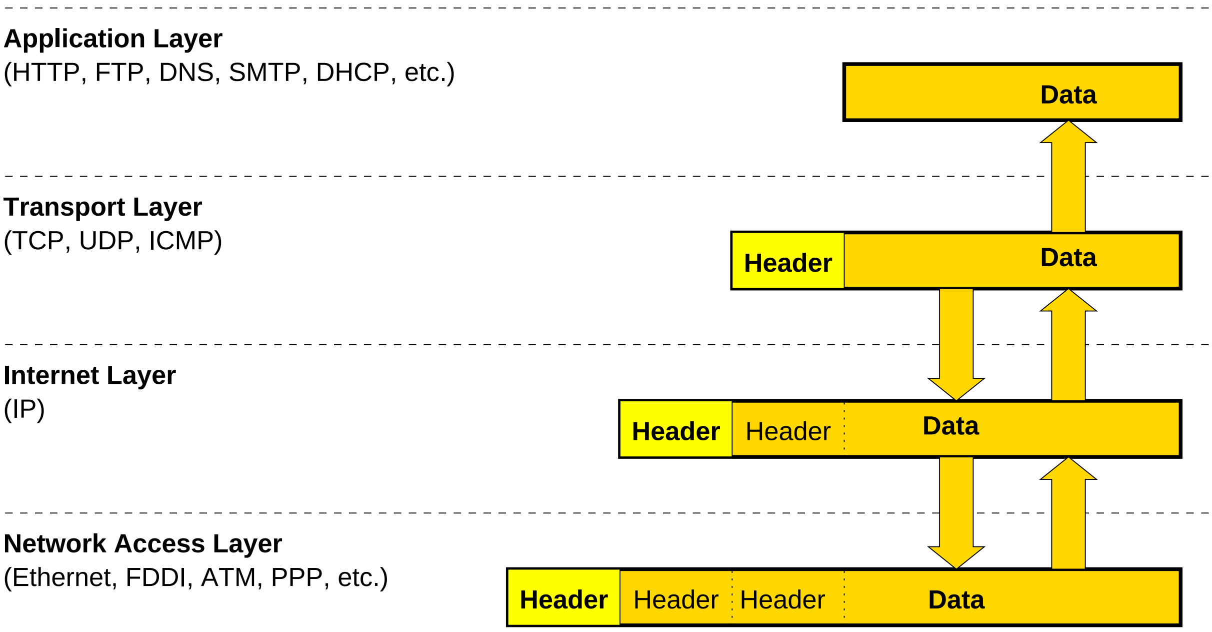
11.5 Packets

- All data transferred in packets (datagrams)
- Each packet contains various flags & admin information
  - Source address (32 bits)
  - Destination address (32 bits)
- Addresses identify hosts
  - Usually an interface on a host
- Addresses are the basis of packet routing
- Packets can be split reassembled, differentially routed, arrive out-of-order or just get lost
- Higher-level protocols (e.g. TCP) add sequencing reliability, flow control etc.



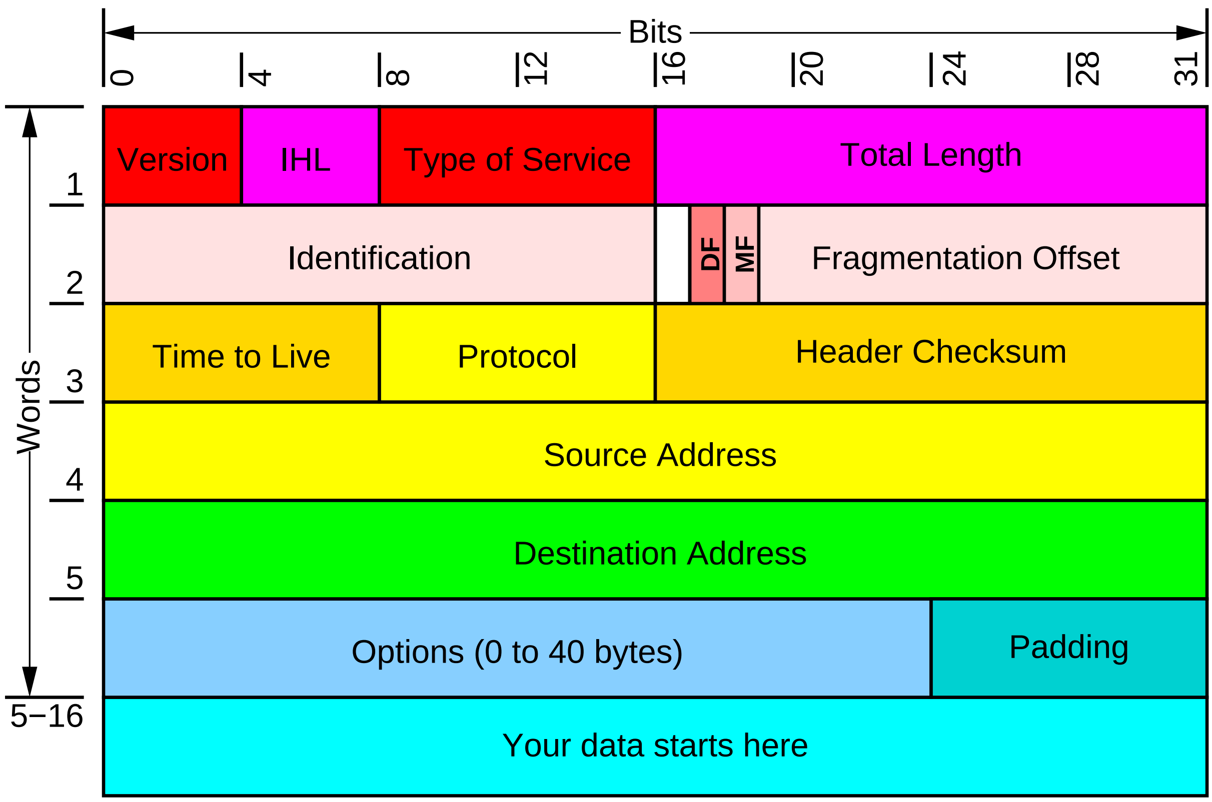
11.6 Encapsulation

- As data sent by application to network passes through layers of TCP/IP protocol stack, headers are appended to it.
- As data received by application from network passes through layers of TCP/IP protocol stack, headers are stripped from it.

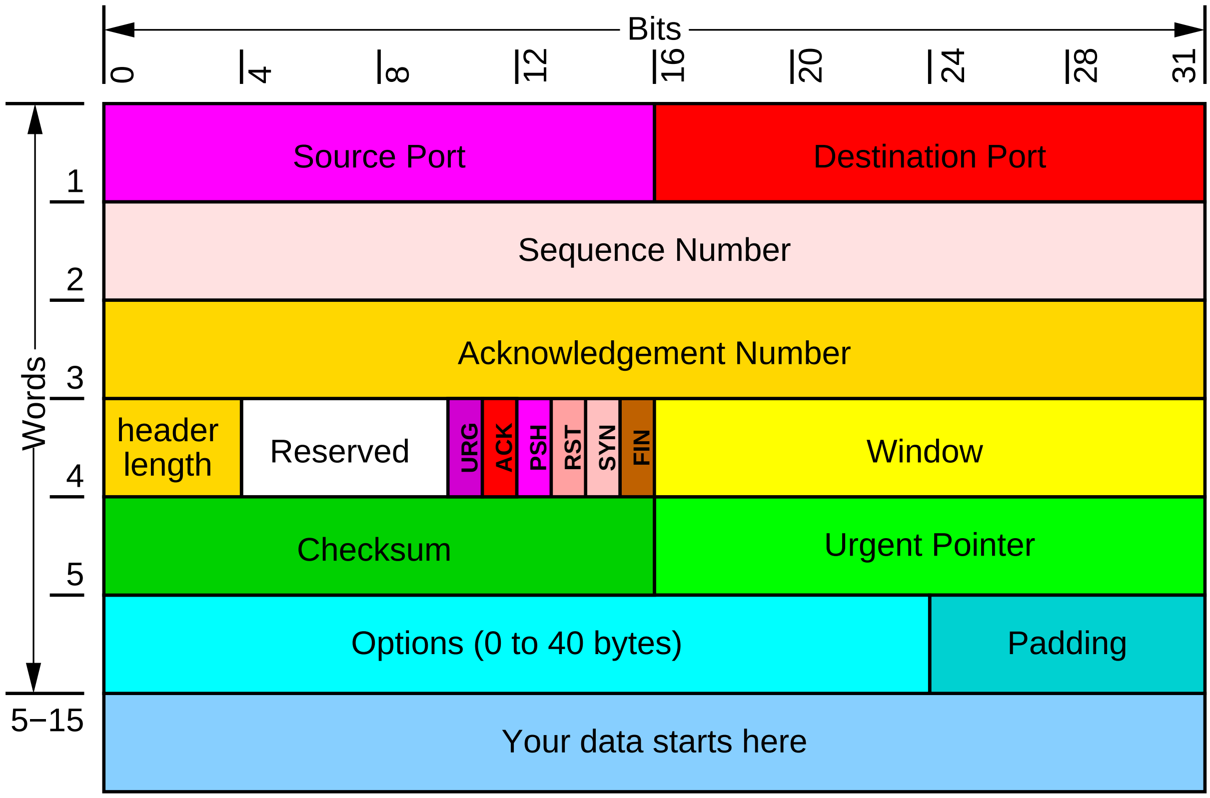


11.7 Internet Protocol Datagram

- IP uses datagram packet switching
- each IP datagram has a max. length 65,535 bytes including a 60-byte header (20-byte + optional part).
- 64 bits in an IP datagram is used to represent the address of data terminals, 32 bits for the source and 32 bits for the destination.
- version now 4; soon 6 will be common
- IHL = Internet Header length = number of 32-bit words in header
- TOS = 3 bits for precedence, 4 TOS bits: delay, throughput, reliability, cost.

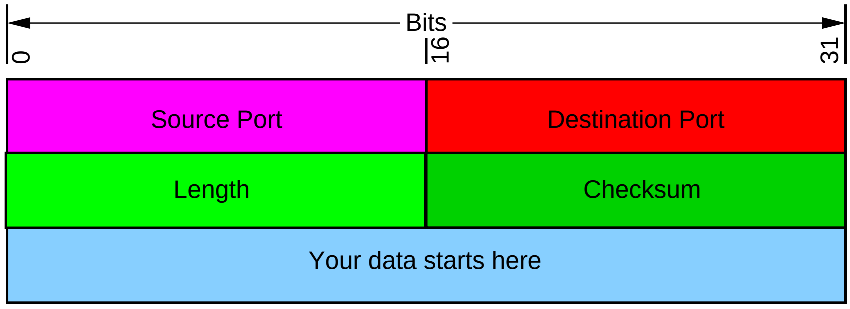


11.8 TCP Header



11.9 UDP Header

- UDP is relatively simple: no ARQ
- So no need to keep track of sequence numbers, no acknowlegdment number

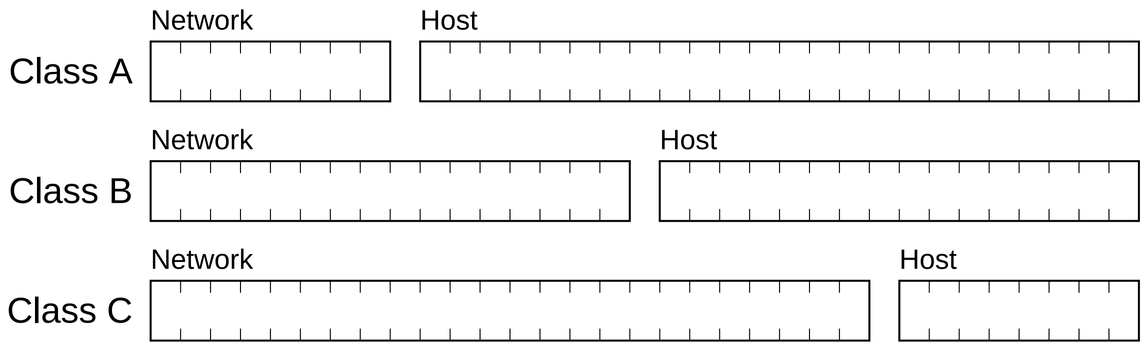


11.10 Addresses

- Addresses shown in ‘dotted decimal’ — break into 4 bytes
  - 192.168.0.129
- Four address families
  - Class A 0.x.x.x–127.x.x.x
  - Class B 128.x.x.x–191.x.x.x
  - Class C 192.x.x.x–223.x.x.x
  - ’reserved’ 224.x.x.x
- Class A network 127 is special
  - Refers to the current network (any network)
  - Current host is *always* 127.0.0.1
  - *‘loopback’* address

11.11 Addresses (continued)

- Addresses identify:
  - Network (used for routing between networks)
  - Hosts on a particular network
    - Class A 8 network bits, 24 host bits
    - Class B 16 network bits, 16 host bits
    - Class C 24 network bits, 8 host bits



- In all networks, host-parts of all zeros (0) and all ones (255) are reserved
  - Host-part zero refers to the network itself
  - Host-part all ones is 'broadcast' address (all hosts)

11.12 Netmasks and subnetting

- Netmasks split host and network part of address
- Says which machines can be reached directly
- Example:

```
Netmask 255      .255     .255     .0
Binary  11111111.11111111.11111111.00000000

IP      192      .168     .0       .129
Binary  11000000.10101000.00000000.10000001
```

- To work out the network part

```
Netmask  11111111.11111111.11111111.00000000
IP       11000000.10101000.00000000.10000001
--------------------------------------------
Result   11000000.10101000.00000000.00000000
--------------------------------------------
            192       168     0         0
```

- To work out the host part

```
Netmask  11111111.11111111.11111111.00000000
IP       11000000.10101000.00000000.10000001
--------------------------------------------
Result   00000000.00000000.00000000.10000001
--------------------------------------------
             0         0      0        129
```

11.13 CIDR: Classless Inter-Domain Routing

- Classes A, B, C are now history. WHY?
- Class C too small, running out of class B!
- Solution: CIDR
- Specify network with two sets of numbers, e.g.,
  - 15/8 is the old class A network that starts with the bits 0001111.
  - 128.32/16 is the old class B network 128.32.0.0
  - 192.168.0.128/25 is the 128 IP addresses from 192.168.0.128 to 192.168.0.255
  - 172.19.64/18 is the $2^{14} = 16384$ addresses from 172.19.64.0 to 172.19.127.255

    $14 = 32 - 18$

    $172.19.64.0 \Rightarrow 1010\,1100.0001\,0011.0100\,0000.0000\,0000$

    The netmask has 18 bits that are '1' (255.255.192.0)
    This is the range of addresses allocated to the CM labs from now on.

11.14 CIDR: Classless Inter-Domain Routing—examples

Address	Class	Network no.	Host no.
10.2.1.1			
128.63.2.100			
201.222.5.64			
192.6.141.2			
130.113.64.16			
192.168.129.49/23			
202.4.192.60/24			

11.15 Transferring Data

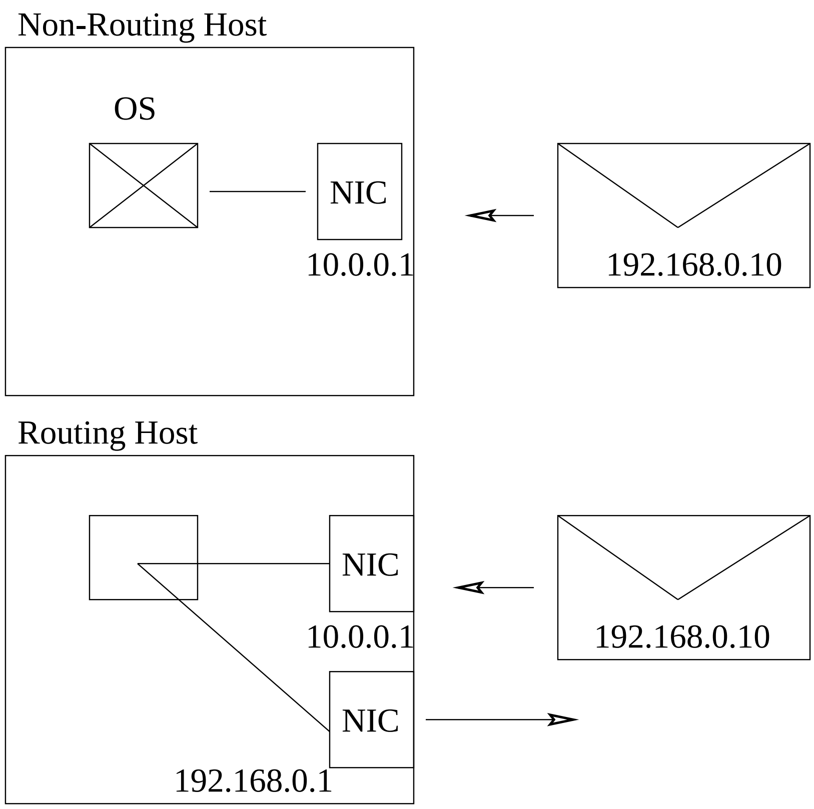
- IP allows datagrams to be sent and routed between hosts
- Contains no *application-level* data
- Data part will be one of UDP, TCP, ICMP etc.
- TCP is ‘session’ oriented data, used for long-lived connections
- UDP used for fire-and-forget messages
- ICMP used for control & testing, not seen by most applications or users
- Examples:
  - Email transferred using SMTP over TCP, (maybe many bytes, order important)
  - Web pages use HTTP over TCP
  - UDP more obscure, used for `NFS`
  - ICMP: ‘`ping`’ utility, used to test visibility

11.16 Hosts & Interfaces

- Hosts are individual computers/systems
- Each host has one or more interfaces
  - Each interface is a point of connection to a network (often a NIC or modem)
- Many hosts have a single interface, so the address is the host
- May have more than one interface
  - Interfaces could be on different networks
  - Can act as routers, forwarding packets
- Each 'interface' will have a single address

11.17 Routing

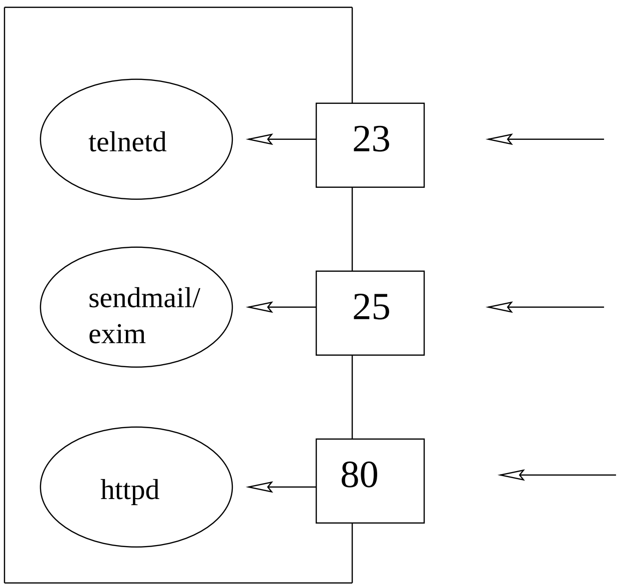
- Hosts receive packets on one or more interfaces
- Check to see if packet is for current host
  - If so, deliver to the UDP/TCP etc mechanisms
- Otherwise
  - If routing enabled *
    - Forward packet to appropriate host
  - Routing based on internal routing table
  - Manipulated by `route` command
    - Superuser only



*Often referred to as 'IP forwarding'

11.18 Ports

- Not enough just to deliver packets to hosts
- Deliver to correct applications on the host
  - Hosts presumed to be multitasking
- UDP & TCP both include port numbers
  - 16 bit numbers (0–65535)
  - Each UDP/TCP packet contains source & destination port
  - sourceport/sourceaddress & destinationport/destinationaddress uniquely identify a conversation



11.19 Ports cont..

- Many ‘well known’ ports published for client-server applications
- See `/etc/services` under Linux
  - TCP/25 — SMTP mail
  - TCP/23 — telnet (remote terminal access)
  - TCP/80 — HTTP (web protocol)
- Unix-like systems reserve ports below 1024 for super-user
- Ordinary users cannot run ‘special’ services without authorisation
- This cannot be trusted in other environments, such as Windows

11.20 Exercises

1. Using `ifconfig`, explore the interfaces available on your current system
2. Discover the IP addresses of some other machines on your network and check that you can ping them all. What `class` (A, B or C) of network are they on?
3. From the man page for `ping`, discover how to set a regular ping running every five seconds. Then investigate how you can send extra-long ping packets (try sending a ping longer than 2K bytes).
4. What ports and protocols are used to run the following services?
   - Telnet
   - SMTP
   - Printer
   - Talk
5. What happens if you `telnet` to various ports? (Try 25 or 110)
6. Use this fact to discover what mail system your machine runs, and see if it runs a webserver (Port 80)

11.21 Solutions

1. `ifconfig` by default shows a list of the currently configured interfaces including the IP addresses and netmasks.

2. `ping 10.0.0.2` will send pings to the `10.0.0.2` interface provided routing is set up correctly. You should be able to find out what class of network you are on from the IP address. See section 11.10 for details.

3. To send a regular ping every 5 seconds use

```
$ ping -i 5
```

To alter the packet size you use the `-s option` to give a size in bytes

```
$ ping -s 3000
```

4. The following values are taken from `/etc/services`

Port	Service
23	Telnet
25	SMTP
515	Printer
517	Talk

5. You should be able to talk directly to the daemon at the other end, e.g.

```
$ telnet localhost 110
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK POP3 localhost v4.47 server ready
USER lee
+OK User name accepted, password please
PASS ********
+OK Mailbox open, 2 messages
RETR 1
+OK 332 octets
Return-Path: <lee>
Received: (from lee@localhost)
        by gbdirect.co.uk (8.8.7/8.8.7) id LAA12997
        for lee@localhost; Mon, 14 Feb 2000 11:39:19 GMT
Date: Mon, 14 Feb 2000 11:39:19 GMT
From: Lee <lee@gbdirect.co.uk>
Message-Id: <200002141139.LAA12997@gbdirect.co.uk>
To: lee@gbdirect.co.uk
Subject: Test
Status: RO

This is a test.
.
QUIT
+OK Sayonara
Connection closed by foreign host.
```


6. You can sometimes find out what webserver a site is using by telnetting to port 80 and requesting the headers of the main page, e.g.

```
$ telnet www.bbc.co.uk 80
Trying 212.58.224.31...
Connected to www.bbc.net.uk.
Escape character is '^]'.
HEAD / HTTP/1.0

HTTP/1.1 302 Moved Temporarily
Date: Mon, 14 Feb 2000 11:43:06 GMT
Server: Apache/1.3.1 (Unix)
Location: http://www.bbc.co.uk/home/today/
Connection: close
Content-Type: text/html

Connection closed by foreign host.
```


Module 12

Practical TCP/IP

Objectives

After completing this module you should be able to understand and utilise:

- Firewalling principles
- Basic firewalling with `ipchains`
- Network/routing debugging procedures
- Interface configuration under Linux
- The secure shell (`sshd`, `ssh`, and `scp`)

12.1 Ping Protocols

- `ping` used to test network/host availability
- A little about its implementation
  - Uses ICMP protocol
  - Send requests of type *echo-request*
  - Receives answer *echo-reply*

12.2 Network Statistics (`netstat`) in Practice

- Show network status; many options
- Most useful: `-r` and `-n` flags (show routes, numeric addresses only)

```
$ netstat -rn
Kernel IP routing table
Destination   Gateway       Genmask         Flags  MSS Window irtt Iface
192.168.0.0   0.0.0.0       255.255.255.0   U     1500 0         0 eth0
202.80.80.0   192.168.0.1   255.255.255.0   UG    1500 0         0 eth0
192.100.100.0 192.168.0.1   255.255.255.0   UG    1500 0         0 eth0
194.217.100.0 192.168.0.143 255.255.255.0   UG    1500 0         0 eth0
192.168.1.0   192.168.0.1   255.255.255.0   UG    1500 0         0 eth0
192.168.3.0   192.168.0.1   255.255.255.0   UG    1500 0         0 eth0
127.0.0.0     0.0.0.0       255.0.0.0       U     3584 0         0 lo
```

- Note ‘gateway‘ above
  - route to networks 202.80.80.0, 192.100.100.0, 192.168.1.0 and 192.168.3.0 use gateway 192.168.0.1
  - 192.168.0.1 is a gateway (router) which knows how to access those networks
  - route to network 194.217.100.0 is via gateway at 192.168.0.143
- Often see destination of 0.0.0.0
  - ‘default‘ route
  - send all otherwise unrouteable packets to designated gateway
- *Iface* column shows which interface will be used
- Note interface for 127.0.0.0 — ‘loopback‘ interface; the host itself

12.3 `netstat` (continued)

- Also show information about connected sockets
- `netstat -a` shows no. of active connections (useful for seeing system load)
-
```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address        Foreign Address     State
tcp        0      0 *:6000               *:*                 LISTEN
tcp        0      0 linux.gazcl:domain   *:*                 LISTEN
tcp        0      0 localhost:domain     *:*                 LISTEN
tcp        0      0 *:linuxconf          *:*                 LISTEN
tcp        0      0 *:auth               *:*                 LISTEN
tcp        0      0 *:finger             *:*                 LISTEN
raw        0      0 *:icmp               *:*                 7
raw        0      0 *:tcp                *:*                 7
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node Path
unix  1      [ ]         STREAM     CONNECTED     8937   @00000082
unix  1      [ ]         STREAM     CONNECTED     8906   @0000007b
unix  1      [ ]         STREAM     CONNECTED     8933   @00000081
unix  0      [ ACC ]     STREAM     LISTENING     327    /dev/log
unix  1      [ ]         STREAM     CONNECTED     8949   @00000086
unix  1      [ ]         STREAM     CONNECTED     8926   @0000007f
unix  1      [ ]         STREAM     CONNECTED     8644   @00000059
```


12.4 `netstat` — Further Examples

- Configured interfaces
- `netstat -i`

```
Kernel Interface table
Iface MTU Met RX-OK RX-ERR RX-DRP RX-OVR TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0 1500   0     0      0      0      0  3107      0      0      0 BRU
lo   3924   0  1035      0      0      0  1035      0      0      0 LRU
```

- `netstat -p` shows processes listening on each socket *
  - Includes PID
  - Useful to kill processes hogging key ports

```
$ netstat -pn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address  Foreign Address  State       PID/name
tcp        0      0 10.0.0.1:1025  10.0.0.2:telnet  ESTABLISHED 443/telnet
tcp        0      0 10.0.0.1:1024  10.0.0.3:telnet  ESTABLISHED 442/telnet
tcp        0      0 10.0.0.1:1023  10.0.0.4:ssh     ESTABLISHED 432/ssh
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags  Type    State      I-Node PID/Program name  Path
unix  1      [ ]    STREAM  CONNECTED  662    432/ssh           @00000037
unix  1      [ ]    STREAM  CONNECTED  591    388/login -- lee  @0000002f
```

*Only supported in more recent versions

12.5 Network Traffic (`tcpdump`) in Practice

- Used to monitor network traffic
  - Need sufficient privilege to monitor devices
- Can show only particular information
  - Traffic to/from a particular host
  - Traffic on a certain port
  - Certain types of traffic, e.g. TCP, ARP, UDP
- Very configurable
  - Decide what you want to do
  - Then look at manual page

12.6 `tcpdump` Options

- Some options

`-i`	Says which network *interface* to show
`-n`	Print IP addresses *not names*
`-N`	Don't print domain name of address
`-t`	Don't print *timestamp*
`-q`	Show only minimal output (*quiet*)
`-v`	*Verbose* info (time-to-live etc.)

12.7 tcpdump Examples

```
$ tcpdump dst host 192.168.0.143 -i eth0 -n -t
tcpdump: listening on eth0
192.168.0.131 > 192.168.0.143: icmp: echo request
192.168.0.131 > 192.168.0.143: icmp: echo request
arp who-has 192.168.0.143 tell 192.168.0.131
192.168.0.131 > 192.168.0.143: icmp: echo request
192.168.0.131.1026 > 192.168.0.143.telnet: S 73945916:73945916(0) win 32120
 <mss 1460,sackOK,timestamp 238586[|tcp]> (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: . ack 3134108710 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: P 0:27(27) ack 1 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: . ack 13 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: P 27:35(108) ack 5 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: P 35:38(3) ack 55 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: P 38:41(3) ack 125 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: . ack 132 win 32120 (DF)
192.168.0.131.1026 > 192.168.0.143.telnet: P 41:42(1) ack 132 win 32120 (DF)
```

```
$ tcpdump dst host 192.168.0.143 -i eth0 -N -q
tcpdump: listening on eth0
09:56:32.947997 landlord.mysql > samosa.5660: tcp 166 (DF)
09:56:32.955822 landlord.mysql > samosa.5660: tcp 166 (DF)
09:56:32.963597 landlord.mysql > samosa.5660: tcp 182 (DF)
09:56:32.970917 landlord.mysql > samosa.5660: tcp 166 (DF)
09:56:32.979341 landlord.mysql > samosa.5660: tcp 166 (DF)
09:56:32.987218 landlord.mysql > samosa.5660: tcp 166 (DF)
09:56:32.995902 landlord.mysql > samosa.5660: tcp 555 (DF)
```


12.8 Firewalling

- Allows you to protect your machine
  - As well as machines *behind* them
- Checks packet headers before acting on them
  - Can ignore, reject or accept packets
  - Makes decision based on source, destination, or packet type
    - Or a combination
- Set up using `ipchains` under kernel 2.2
  - Older kernels used `ipfwadm`

12.9 Basic Theory

- Two main considerations
  - Port Filtering
  - Host Filtering
- Block services you don't need
- Limit services you *do* need to specific machines/networks

12.10 Basic Theory (continued)

- Firewalling can be done with `inetd`
  - `/etc/hosts.allow`
  - `/etc/hosts.deny`
  - `/etc/inetd.conf`
- Flaw in `inetd` would still let things through
- Best to drop the packets as soon as possible
  - Kernel-level filtering

12.11 ipchains

- Packet filtering set up using `ipchains`
- All the filtering is done in the kernel
  - Not by `ipchains`
  - `ipchains` just sets up/modifies the rules
- All packets entering and leaving are examined *

*Including loopback traffic which conceptually leaves the machine

12.12 `ipchains` Details

- Every packet goes through one or more '*chains*'
  - A 'chain' is a set of rules
  - Rules can accept, reject, or deny a packet
    - Can also send it to another chain
- Three default chains, *input*, *output*, *forward*
  - If a packet passes through a default chain without matching:
    - Fate is determined by the chains *policy*
    - Can be *Accept*, *deny*, or *reject*
  - If it reaches the end of a user defined chain
    - Carries on where it left off
- *forward* is for IP masquerading systems
  - Not covered here

12.13 `ipchains` Options

- Dealing with chains:

`-N`	Create a new chain
`-X`	Delete an empty chain
`-P`	Change the policy for a chain
`-L`	List the rules in a chain
`-F`	Flush (delete) all rules from a chain

- Dealing with rules:

`-A`	Append a rule to a chain
`-D`	Delete a single rule from a chain
`-I`	Insert a rule at some point in a chain

12.14 Options For Rules

- Use the following to specify packets to match

`-s`	Source address
`-d`	Destination address
`-p`	Protocol (`TCP`, `UDP`, `ICMP`)
`-j` *chain*	Jump to chain/action
`--sport`	Source Port
`--dport`	Destination Port

12.15 `ipchains` — Examples

- In most cases default chains will be sufficient
- To block all `ping` requests to our machine:

```
$ ipchains -A input -p icmp -s 0.0.0.0/0 \
> --icmp-type echo-request -j DENY
$ ipchains -L input
Chain input (policy ACCEPT):
target  prot opt     source       destination    ports
DENY    icmp ------  anywhere     anywhere       echo-request
```

- To block outgoing `ping` packets:

```
$ ipchains -A output -p icmp -d 0.0.0.0/0 \
> --icmp-type echo-request -j DENY
$ ipchains -L output
Chain output (policy ACCEPT):
target  prot opt     source       destination    ports
DENY    icmp ------  anywhere     anywhere       echo-request

$ ping -c1 landlord
PING landlord.gbdirect.co.uk (192.168.0.129): 56 data bytes
ping: sendto: Operation not permitted
ping: wrote landlord.gbdirect.co.uk 64 chars, ret=-1

--- landlord.gbdirect.co.uk ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
```

- Very simple examples but they show the theory

12.16 Removing Rules

- Rules can be removed by *number*, e.g. to delete the first rule in the *input* chain:

```
$ ipchains -D input 1
```

- or definition, e.g. delete the *first* matching rule:

```
$ ipchains -D output -p icmp -d 0.0.0.0/0 --icmp-type
  echo-request -j DENY
```

- To clear an entire chain use:

```
$ ipchains -F chainname
```

- If no `chainname` is given, it clears all chains

12.17 Implementing ipchains

- The rules are normally set up in the machines 'init scripts'
- Typically by creating a script in `init.d` that is run just before networking starts
  - Example in section 12.19
- Ensure you flush existing rules first (just in case):

  ```
  $ ipchains -F
  ```

- Generally start with the DENY rules then add what you want
- Maximum security

12.18 Save and restore

- Often useful to create a firewalling *'config file'*
- `ipchains-save` outputs a text file you can store

```
[Setup firewall rules as you want]
$ ipchains-save > /etc/ip.rules
Saving 'input'.
Saving 'forward'.
Saving 'output'.
$
```

- Can reinitialise your firewalling with `ipchains-restore` and your 'config file', e.g.

```
$ ipchains-restore < /etc/ip.rules
$
```

- Usually done in a startup script

12.19 ipchains setup script

- A sample script may look like:

```
#! /bin/sh
# Script to control packet filtering.
# If no rules, do nothing.
# Altered from the ipchains HOWTO
[ -f /etc/ipchains.rules ] || exit 0
case "$1" in
    start)
        echo -n "Turning on packet filtering:"
        /sbin/ipchains-restore < /etc/ipchains.rules || exit 1
        echo "."
        ;;
    stop)
        echo -n "Turning off packet filtering:"
        /sbin/ipchains -X
        /sbin/ipchains -F
        /sbin/ipchains -P input ACCEPT
        /sbin/ipchains -P output ACCEPT
        /sbin/ipchains -P forward ACCEPT
        echo "."
        ;;
  restart)
        $0 stop
        $0 start
        ;;
    *)
        echo "Usage: $0 {start|stop|restart}"
        exit 1
        ;;
esac
exit 0
```

12.20 Real World `ipchains`

- Connect out to a host but not in

```
$ ipchains -A input -d 192.168.0.131/32 -p TCP -y -j DENY
```

- `-y` limits matching to packets with the SYN bit set
  - Used when establishing connections
- No-one can open a connection from `192.168.0.131`
  - Can still connect to it from here ...

12.21 Interface Configuration and Management

- An interface is a point of connection to a network
- Usually a single device
  - Network card
  - PPP link
- A device can have more than one interface
  - Referred to as ‘aliases’
  - Commonly used for virtual web sites

12.22 Point-and-Click Interface Administration

- Number of ways to add/edit interface details
  - Linuxconf
  - Redhat `control-panel`
  - *'By hand!'*
- For most cases you can probably use one of the two graphical methods
- Useful to understand the configuration files behind it all
- See the *Linux Network Administrator's Guide* at `http://www.linuxdoc.org/LDP/nag2/` for much there is to know about this.

12.23 /etc/sysconfig/network-scripts

- Directory containing scripts and config files *
- ifup & ifdown activate/deactivate an interface
- Argument specifies interface to act on, e.g.

```
$ ifdown eth0
```

- ifcfg-eth* are config files for each interface
  - Should be numbered sequentially from 0
    - ifcfg-eth0 is the first interface
  - Files ending in :n (where n is a number) are aliases
    - ifcfg-eth0:0 is the first alias for the first interface

*This applies to RedHat only, you should see section 12.27 for information on other distributions

12.24 `ifcfg-ethx`

- Describes characteristics of a given interface
  - What device it should be known as (`DEVICE`)
  - IP address, network, and netmask (`IPADDR`, `NETWORK`, `NETMASK`)
  - Whether it is activated at boot time (`ONBOOT`)
  - Whether it can be controlled by normal users (`USERCTL`)
- Example:

```
DEVICE=eth0
IPADDR=192.168.0.129
NETMASK=255.255.255.0
NETWORK=192.168.0.0
BROADCAST=192.168.0.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

12.25 Altering An Interface

- It is perfectly allowable to alter interfaces while the system is running
- Requires only minimal disruption to network connectivity
  - Not a reboot
- Two simple steps
  1. Make alterations (by hand or through GUI)
  2. Restart networking
- Networking is just another service
  - `/etc/rc.d/init.d/network restart`
  - `/etc/init.d/network restart`

12.26 Adding an Interface

- Adding an alias is even easier!
  1. Add the alias
  2. Activate it
- Example: Add the following to `/etc/sysconfig/network-scripts/ifcfg-eth0:0`

```
DEVICE=eth0:0
USERCTL=no
ONBOOT=yes
BOOTPROTO=none
BROADCAST=192.168.0.255
NETWORK=192.168.0.0
NETMASK=255.255.255.0
IPADDR=192.168.0.141
```

- Then execute

```
$ ifup eth0:0
```


12.27 The ‘Proper’ Way

- Previous examples use scripts (not always provided)
- You can do everything manually
- Add an alias:
  `/sbin/ifconfig eth0:0 192.168.0.128`
- Check with `ifconfig` that it succeeded
- Setup routing to that interface:
  `/sbin/route add -host 192.168.0.128 dev eth0:0`
- Removing an alias:
  - `/sbin/ifconfig eth0:0 down`
  - `/sbin/route del 192.168.0.128`
- Adding an interface is similar . . .
- Probably want to add a route to the entire network not just the host
  `/sbin/route add -net 192.168.0.0 netmask 255.255.255.0 dev eth1`

12.28 Drivers

- Network drivers invariably handled by kernel modules
  - PCI NE2000 card handled by `ne2k-pci.o`
- Kernel cannot tell which module should be used by which interface
  - module loader uses lines from `/etc/modules.conf` (older: `/etc/conf.modules`), e.g.,

    ```
    alias eth0 ne
    alias eth1 ne
    options ne io=0x320,0x340 irq=2,12
    ```
  - Above says that interfaces `eth0` and `eth1` handled by `ne` module (NE2000 ISA)
  - Options line is module-specific; permits port/IRQ specification if not autodetected: this is common for old ISA cards.

12.29 The Secure Shell in Practice (`ssh`)

- How you use `ssh` varies across systems
- Some require stricter authentication than others
- For example, within a secure environment it may not require a password
  - Works on 'trusted host' concept
  - *Much* better than `rsh` due to encryption and server key authentication
- Can often be used as a drop-in replacement for `rsh` or `telnet`
- Has numerous advantages . . .
  - Sets up forwarding of X connections
  - Can compress the data sent
  - No passwords sent in plain text (and hence trivially read by others)
  - Not trivially hijacked using `hunt` (from `http://lin.fsid.cvut.cz/~kra/`) and other such tools

12.30 Secure Copying in Practice (scp)

- Replacement for rcp
- Much more secure
  - Encrypts all traffic
  - Uses same authentication as ssh
- Can copy local to remote, remote to local or remote to remote
- Example:

```
$ scp localfilename user@remotehost:remotefilename
```

12.31 Summary

- Wide range of network utilities available
  - Both maintenance and user-orientated
- *Very* flexible system
  - Can be hard to setup/maintain
- Pros outweigh cons
- Common jobs become second nature

12.32 Exercises

1. Network tools

   (a) Use `netstat -rn` to investigate the routes on your network. Explain each line of entry to a colleague.

   (b) Read the man page for `tcpdump`. Use it to monitor traffic on your host's network interface whilst other hosts are pinging each other.

2. `ipchains`

   (a) Use ipchains to set up the following configurations. In each case you should first set up the system by hand, check it. Then set it up so that the firewall rules are in place when the machine reboots.

      i. Block all incoming ICMP packets
      ii. Block only incoming ICMP 'echo-request' packets
      iii. Block all incoming telnet connections
      iv. Block *all* telnet connections
      v. Block all outgoing web requests (Port 80)

3. Network configuration

   (a) Using one of the admin tools (`linuxconf` or `control-panel` etc.) add an alias on your network interface so that your host can masquerade as some other host. DO NOT DO THIS IF YOU ARE NOT SURE YOU ARE USING A SPARE IP ADDRESS. Investigate what `ifconfig` and `netstat -rn` now report. Check that you can ping the alias from another host on the network.

   (b) If possible, fit an extra network card to one of the hosts (host b) and configure it to be on a different network. Check it can be pinged from its own host. Go to another host (host a) on the original network and add a route to host b's new interface, using as a gateway host b's original network interface. Check that you can ping it and then use `traceroute` to see the path taken by packets. Host b will have to have IPV4 forwarding enabled for this to work. Ask the tutor about which machine will be set up for this.

12.33 Solutions

1. **(a)** If you don't understand the output check section 12.2 or the `netstat` manpage

   (b) `tcpdump -i eth0` should monitor all network traffic. If you want to see the traffic to a particular host use `tcpdump dst host 10.0.0.3`

2. **(a)** The following are the list of rules needed to satisfy each situation. You should flush the chains before each one (`ipchains -F`).

   i. `ipchains -A input -p icmp -j DENY`

   ii. `ipchains -A input -p icmp --icmp-type echo-request -j DENY`

   iii. `ipchains -A input -p tcp -d 127.0.0.1 --dport telnet -j DENY`
   `ipchains -A input -p tcp -d 192.168.0.131 --dport telnet -j DENY`

   iv. `ipchains -A output -p tcp -d 0/0 --dport telnet -j DENY` `ipchains -A input -p tcp -s 0/0 --dport telnet -j DENY`

   v. `ipchains -A output -p tcp -d 0/0 --dport www -j DENY`

3. **(a)** -

   (b) Ask the tutor for details.

Module 13

SSH — The Secure Shell

Objectives

- After completing this section, you will be able to:
  - Use secure shell as a replacement for `telnet`
  - Understand the weaknesses of the secure shell, and in particular, how to protect your connections from being hijacked
  - Understand some of the dangers of `telnet`
  - Use `scp` to copy files
  - Understand how to create private and public keys for the secure shell using `ssh-keygen`
  - Understand how to configure the files `~/.ssh/authorized_keys`, `/etc/ssh/ssh_known_hosts` with public keys
  - Know how to configure `ssh-agent` and `ssh-add` to allow login without a password.
  - Be able to configure some important options of the client and server, including X11 forwarding.

13.1 What is the Secure Shell?

- Allows users to log into remote computers and use them interactively.
- Provides secure network access for system administrators, replacing obsolete protocols such as `telnet` and many others
- Allows tunneling of other protocols (such as X) through the encrypted connection
- Allows secure, encrypted copying of data over the Internet with `scp`
- Very useful for remote backup of servers and firewalls.
- After it is set up, it is very easy to use

13.2 But what's wrong with `telnet`?

- We still use `telnet` for some of our other classes. What could be wrong with it?
- The program `hunt`, freely available from `http://lin.fsid.cvut.cz/~kra/`, lets you very easily:
  - read passwords
  - hijack telnet sessions
  - monitor what the person is typing
- Pavel Krauz wrote the program just to show how insecure `telnet` is—try it out!
- Other tools can do this too (but just with less convenience)
- Spread the news! `telnet` is dead! Demand the Secure Shell! Use the secure shell at your work; persuade people to stop using `telnet`, except where passwords are not required, and the information is not useful (in which case, you should be doing something else!)

13.3 Cryptography

- Two main categories of encryption:
  - *public key* or *asymmetric* cryptography
    - Different key used to encrypt and decrypt the data
    - The public key used to encrypt
    - private key used to decrypt
  - *secret key* or *symmetric* cryptography
    - Same key used to encrypt and decrypt the data
    - In Secure Shell, the *session key* performs the encryption and decryption at each end.

13.4 OpenSSH and its history

- Originally, SSH was free, developed by Tatu Ylönen,
- later, increasingly restrictive licensing.
- Two versions of SSH: SSH1 and SSH2, available from *SSH Communications Security*, and *Datafellows*, at high prices ($3120.00 US for a 25 user *client only* license when I checked!)
- International group of programmers chose to develop a completely free SSH from Tatu's original SSH
  - Development moving very rapidly
  - well implemented
  - used by many system administrators round the world
  - Completely free (both in the sense of "freedom" and in the sense of "free beer"), open protocol
  - now IETF is standardising SSH
- Provides a very nice blowfish logo, seen here burying the old insecure protocols: `telnet`, `rlogin`, `rsh`.

13.5 Okay, I like the blowfish—what else does OpenSSH provide?

- OpenSSH 2.3 and onward is compatible with both SSH1 and SSH2, and interoperates with the commercial versions
- OpenSSH provides powerful encryption to the entire session
- OpenSSH 2.3 and above provides the following main programs:
  - A server `sshd` which provides services for all the clients below, as well as `sftp`, an `ftp` replacement, that works with `CuteFTP` among many other clients
  - The clients `ssh`, `scp`:
    - `ssh` is for executing programs on other remote computers, and replaces `telnet`
    - `scp` is a network copy program; works like `cp`, but over the network.
  - The utilities:
    - `ssh-keygen` for generating the private and public keys used by SSH
    - `ssh-agent` to conveniently hold the personal private key to allow you to login without typing passwords
    - `ssh-add` for adding the personal private key to `ssh-agent`

13.6 So okay, how do I use this Secure Shell?

- To use it like `telnet` is easy
- Just type `ssh` ⟨*computer name or IP*⟩
- The first time you connect, you will receive a warning something like this:

  ```
  The authenticity of host 'nms.tyict.vtc.edu.hk (172.19.64.56)' can't be established.
  RSA key fingerprint is 48:2a:cb:1f:ac:7d:21:89:39:de:47:20:cf:06:d3:44
  Are you sure you want to continue connecting (yes/no)?
  ```

  For now, enter `yes` (but see section 13.14 on page 355). The program will reply something like this:

  ```
  Warning: Permanently added 'nms.tyict.vtc.edu.hk'(RSA) to the list of known hosts.
  ```

- Enter your password when prompted
- You now have a high security, military-strength encrypted link to the other computer.*

*Provided your connection was not hijacked! See section 13.14 on page 355.

13.7 Using `scp` to copy files over the network

- Use like the `cp` command
- Put the computer name and a colon in front of a remote file
- Put a username then `@` in front of this if you want to connect as a different user from yourself
- Enough! An example of copying a file `file` from your home directory on machine *neighbour* to the current directory:

  `$ scp` ⟨*neighbour's IP*⟩`:file ~`

- Here is an another example: copying a file from my account `nickl` on the remote computer called `server` to my home directory on my local workstation:

  `$ scp nick@server:/usr/share/emacs/site-lisp/site-start.el ~`

- To copy a file called `file` from your current directory to the existing directory `/tmp/directory` on your neighbour's computer (that you have an account on), do:

  `$ scp file` ⟨*neighbour's IP*⟩`:/tmp/directory`

13.8 Useful options with `scp`

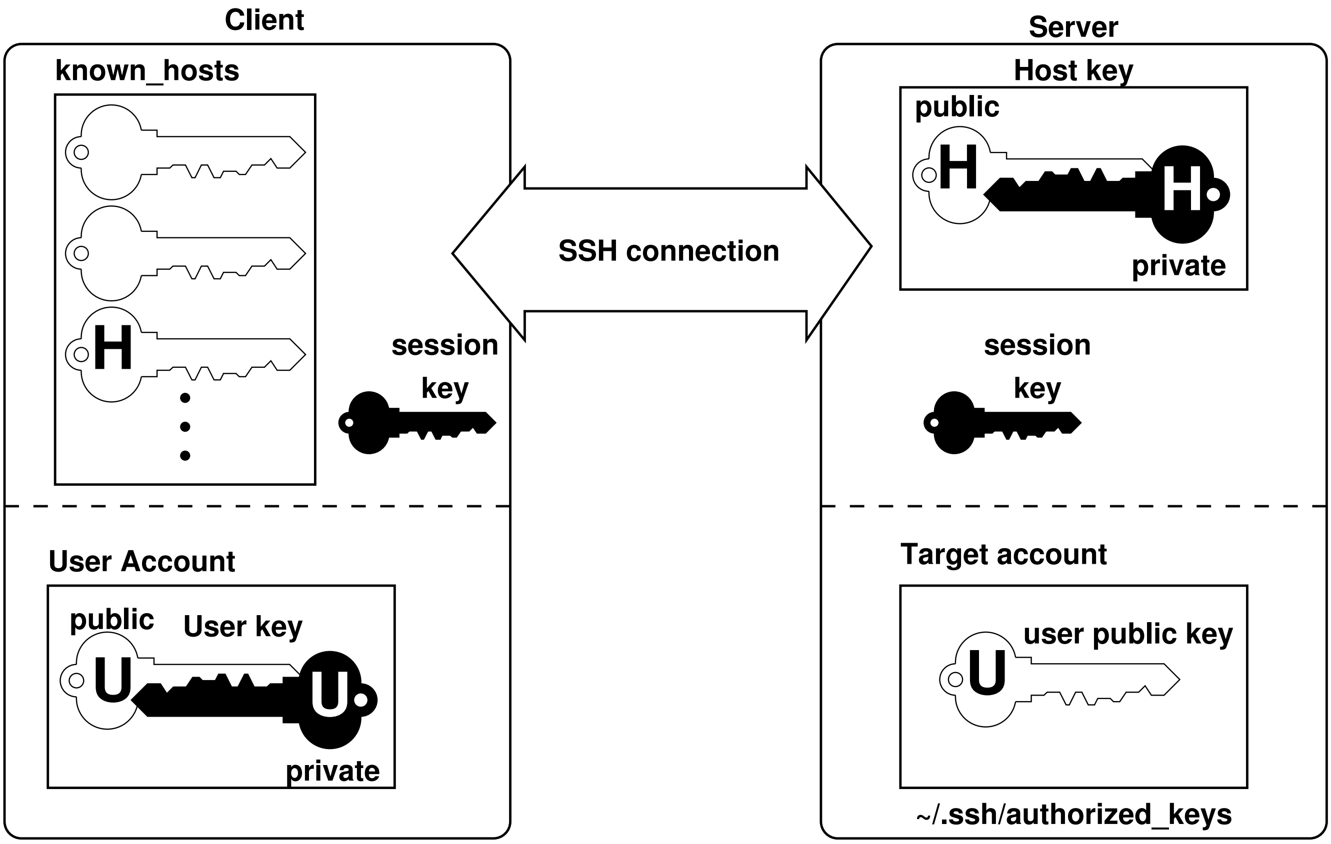
- The most useful options to `scp`:

option	*purpose*
`-p`	preserve the permissions (and owner if you are root)
`-r`	recursively copy all the subdirectories and their contents
`-C`	enable compression. Good for modem speed links.
`-P 23`	use port 23 instead of port 22. Good for transferring files to or from home through a firewall that has port 23 open, but which blocks port 22.

13.9 SSH uses public and private keys

- The Secure Shell uses a public key and a private key for each computer (public key cryptography)
- Each user can also have their own private and public keys
- Public key cryptography allows a public key (a very long number) to be freely distributed and made public
- The private key *must be kept secret*. Your personal private key is usually stored encrypted on the hard disk, protected by a long pass phrase
- The *host* keys are used to:
  - make sure the computer you are connecting to is really the one you want
  - to establish secret *session* keys used to encrypt all communication in the session, including any passwords
- The *user* keys are used to:
  - To establish the identity of the person who has initiated the session (authentication)

13.10 SSH Architecture



13.11 Overview of SSH

- The *client* computer is usually the system administrator's computer (or the person who wants to establish the connection)
- The *server* computer is often exactly that — a server the system administrator needs to administer.
- The host keys are used to:
  - authenticate the server (make sure it really is the company server, and does not belong to the competitor who wants to steal your company's secrets)
  - Establish the session key at each end of the connection.
- Note: the actual encryption of the session is done using the session keys, with symmetric encryption, since it is *much* faster than public key encryption.
- The *user* keys are optional; they allow strong authentication without passwords.
- The user private key is encrypted with a strong passphrase.
- The *agent* is a program used to hold the decrypted private key on the client, to make the private key available to SSH applications.

13.12 Steps of establishing a connection

The following steps are not complete; SSH2 is actually three separate protocols, which can be used in various ways. Here I give an approximate view of SSH establishing a connection.

- The client makes a TCP connection to the port the server is listening on (normally 22)
- The client and server announce the protocols they support, and decide on what they will use
- If the client does not have the server's public host key, then the client will be prompted as to whether they want to accept the host key. This is the warning given in section 13.6 on page 347. If you type "`yes`", then the host key will be transferred in clear text over the network, and appended to your `~/.ssh/known_hosts` file. At this point, your session is vulnerable to hijacking. To avoid this risk, transfer the public host key yourself (manually) and append it to `/etc/ssk/ssh_known_hosts` on the client as described in section 13.14 on page 355.
- Now the server and client generate and share a session key using an algorithm called the *Diffie-Hellman Key Exchange*. They use the host key pair to do this.
- The session is now encrypted, and authentication of the user now takes place, using either the user key pair, or a password.

13.13 Using `ssh-keygen` to create a personal pair of private and public keys

- When you first booted Red Hat Linux, the startup script for `sshd` generated the public and private *host keys* automatically.
- You can create your own pair of user keys: just type

  `$ ssh-keygen -t rsa`

  - accept the default file name
  - when asked for a pass phrase, type a long sentence that you can easily remember, but which will be impossible for other people to guess.

```
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nicku/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/nicku/.ssh/id_rsa.
Your public key has been saved in /home/nicku/.ssh/id_rsa.pub.
The key fingerprint is:
13:6f:6b:1f:ac:6a:21:89:f0:de:47:20:d4:06:5c:2b
```

- This will generate the files `~/.ssh/id_rsa.pub` and `~/.ssh/id_rsa`
- The private key `~/.ssh/id_rsa` *must* have access permissions *only* for you, the owner, i.e., "`-rw-------`" or "`-r--------`"

13.14 The host keys in `/etc/ssh/ssh_known_hosts` and `~/.ssh/known_hosts`

- An important step in using SSH is having the public host key of the computers you want to connect to. Unfortunately, the only really secure way is to copy it on a floppy disk from the remote machine, and take it to your own computer.
- Copy the file `/etc/ssh/ssh_host_rsa_key.pub` from the remote machine to a floppy.
- On the local machine, append the file *as one line* to the file `/etc/ssh/ssh_known_hosts`
- An alternative is just to enter `yes` when asked, as in section 13.6 on page 347
  - The public host key will be copied over the network from `/etc/ssh/ssh_host_rsa_key.pub` on the server and appended to `~/.ssh/known_hosts` on the client.
  - But: this is a serious security hazard, and cracking tools exist (`sshmitm`) to easily hijack SSH sessions before the host key is transferred,
- There are tools to automate this insecure procedure (of building the `/etc/ssh/ssh_known_hosts` file).

13.15 The file ~/.ssh/authorized_keys

- This file contains the personal public keys of user accounts that you trust (your own) to connect to your computer without a password.
- To set it up on a remote computer:
  - Copy your local personal public key to the remote computer (do not copy to ~/.ssh, or you will overwrite your own local key!) Here put the host name of the remote computer instead of "⟨*remote IP*⟩". Note: "`local $`" and "`remote $`" are shell prompts; don't type them!

    ```
    local $ scp -p ~/.ssh/id_rsa.pub ⟨remote IP⟩:
    ```
  - Log into the remote machine using `ssh`:

    ```
    local $ ssh ⟨remote IP⟩
    ```
  - Make the directory ~/.ssh if it does not exist, and change to mode 700:

    ```
    remote $ mkdir ~/.ssh
    remote $ chmod 700 ~/.ssh
    ```
  - Append ~/id_rsa.pub to ~/.ssh/authorized_keys on the remote machine, in the `ssh` session:

    ```
    remote $ cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
    remote $ rm ~/id_rsa.pub
    remote $ chmod 600 ~/.ssh/authorized_keys
    remote $ exit
    ```
- You now have installed your personal public key from your local computer account to the remote computer, allowing you to log in to the remote computer using public key cryptography instead of passwords.

13.16 The User's Public and Private Keys

- There are two computers here: the local *client* machine, and the remote *server* machine.
- When configuring SSH, most of the work is done on your local *client* computer.
- You generate a public and private key pair using `ssh-keygen -t rsa` on your *client* computer—see section 13.13 on page 354. You do *not* need to generate public and private user keys in your account on the remote server.
- Your private key is on the client. Your public key must be on the server, in the file `~/.ssh/authorized_keys`. The client does *not* need a file `~/.ssh/authorized_keys`.
- You install the secure shell agent on your client machine—see section 13.20 on page 361. You do *not* need to install it on the server.
- You add your private key `~/.ssh/id_rsa` on the *client* computer to the agent, which is *also* on the client computer, using the `ssh-add` command.

13.17 SSH1 and SSH2

- SSH1 and SSH2 use totally different, incompatible protocols.
- The commercial programs have separate programs for each protocol (pay for each separately)
- OpenSSH supports both SSH1 and SSH2.
- SSH2 is more secure than SSH1. It is best to avoid SSH1 clients and servers wherever possible.
- SSH1 uses only RSA authentication, while SSH2 uses both RSA and DSA
- See `http://www.snailbook.com/faq/ssh-1-vs-2.auto.html`

13.18 The public and private key pairs: a summary

- SSH1 and SSH2 use different files to hold the public and private keys. I've made a table to summarise their purposes:

SSH1 file	SSH2 file	Purpose
`~/.ssh/known_hosts`	`~/.ssh/known_hosts`	other host public keys added automatically when you type `yes` as in section 13.6
`/etc/ssh/ssh_known_hosts`	`/etc/ssh/ssh_known_hosts`	other host public keys added manually, as in section 13.14
`/etc/ssh/ssh_host_key`	`/etc/ssh/ssh_host_rsa_key`	host private key (`chmod 400`)
`/etc/ssh/ssh_host_key.pub`	`/etc/ssh/ssh_host_rsa_key.pub`	host public key
`~/.ssh/identity`	`~/.ssh/id_rsa`	User's personal private key
`~/.ssh/identity.pub`	`~/.ssh/id_rsa.pub`	User's personal public key (just used for copying into the `authorized_keys*` files of other servers)
`~/.ssh/authorized_keys`	`~/.ssh/authorized_keys`	Holds personal public keys of clients authorised to log into this account on this SSH server

13.19 Files and Permissions I Recommend

- If permissions are wrong, SSH will just not work. The manual pages recommend permissions, but here I have made a list from my own experience:

File	*Min Permission*	*Max Permission*
`~/.ssh`	`drwx------`	`drwxr-xr-x`
`~/.ssh/known_hosts`	`-rw-------`	`-rw-r--r--`
`~/.ssh/identity`	`-r--------`	`-rw-------`
`~/.ssh/id_rsa`	`-r--------`	`-rw-------`
`~/.ssh/identity.pub`	`-r--------`	`-rw-rw-r--`
`~/.ssh/id_rsa.pub`	`-r--------`	`-rw-rw-r--`
`~/.ssh/authorized_keys`	`-rw-------`	`-rw-r--r--`

- I use and recommend the minimum permissions. Note that SSH will *refuse* to work if:
  - the private keys are readable, writeable or executable by group/others
  - the `authorized_keys*` are writable by group/others
  - the `~/.ssh` directory is *not* readable or executable by group/others
  - the `~/.ssh` directory *is* writable by group/others

13.20 Using `ssh-agent` to log in without typing passwords

- Now you can use public key cryptography to login to the remote machine. But you need to type that long pass phrase each time! Oh, that's not convenient.
- So let's use the utilities `ssh-agent` and `ssh-add` to store the private key on your *client* computer.
- Your private key is stored encrypted on the hard disk of your client computer in your account, protected by your pass phrase.
- The agent provides a container to hold your decrypted private key in the memory of your client machine.
- You use the program `ssh-add` to add the private key to the agent once after you boot your client machine.

13.21 Setting up ssh-agent: logging in without typing passwords

- We do this on the *client* machine only. Here's what we do, once only:
  - edit your login script ~/.bash_profile in your account on the client machine, and add the line:
    `eval $(ssh-agent)`
    Note this is command substitution.
  - Log out of X and log back in again.

13.22 Using `ssh-add`: logging in without typing passwords

- We do the following every time you boot your *client* computer:
  - Open a window, and type:

    ```
    $ ssh-add
    ```

    then type your pass phrase when prompted.
- Your client user's secret key is now held decrypted by the agent in the memory of the client machine, letting you conveniently log in to your account on a remote server. If you do this:

  `local $ ssh` ⟨*remote IP*⟩

  you will connect to your account on the remote computer without having to type a password. And that's not all!

  Note: "`local $`" is the shell prompt; you do not type that.

13.23 An easier way: using `keychain`

- `keychain` is a very useful shell script that automates the management of personal private keys and the agent.
- It is also very useful in automating the use of secure shell using `cron`; see section § 3.29 on page 88.
- Get it from `http://www.gentoo.org/projects/keychain`.
- Installation is simple:
  1. get the tarball (click on the link to `http://gentoo.oregonstate.edu/distfiles/keychain-2.2.0.tar.bz2`, save it to your home directory).
  2. Unpack it into any directory

     ```
     $ tar xvjf ~/keychain-2.2.0.tar.bz2
     ```
  3. change into the directory `keychain-`$x.x$
  4. install it into the directory `/usr/bin` with:

     ```
     $ sudo install -m 755 keychain /usr/bin
     ```


13.24 What `keychain` Does

- The agent holds your user private key, decrypted
- SSH clients talk to the agent through a *socket*
  - A *socket* is a method of inter-process communication (IPC)
  - It is a special file that allows two processes to talk to each other in both directions at the same time
- When the agent starts, it creates a socket, and writes commands to standard output that create environment variables that let SSH clients know how to find the agent; for example:

```
$ ssh-agent
SSH_AUTH_SOCK=/tmp/ssh-sif17405/agent.17405; export SSH_AUTH_SOCK;
SSH_AGENT_PID=17406; export SSH_AGENT_PID;
```

- You can see this socket:

```
$ ls -l /tmp/ssh-sif17405/agent.17405
srwxrwxr-x  1 nicku nicku 0 May  2 11:19 /tmp/ssh-sif17405/agent.17405
```

- The `keychain` shell script makes sure that a file `~/.keychain/${HOSTNAME}-sh` contains these two values, and that they are correct. The algorithm is:

```
if ~/.keychain/$HOSTNAME-sh holds info about a running agent
        if that agent does not hold all required keys
                add those keys
        else
                exit
else
        terminate all running agents
        start new agent
        store socket, PID info in ~/.keychain/$HOSTNAME-sh
        add all required private user keys
```

13.25 Setting your hostname

- `keychain` will save the information about your SSH agent in the file `~/.keychain/${HOSTNAME}-sh`.
  - Previously, I set up a name server so that each of our computers would have its own name, but technical staff have disabled it.
  - The problem is that because all our machines are called `localhost.localdomain`, `keychain` will save your `ssh-agent` information in the same file name, overwriting the information about the agent in your own desktop.
  - To avoid this problem, we can give our own computer a hostname. Here I explain how you can do that.

1. Edit the file `/etc/sysconfig/network`:

```
$ xhost +localhost
$ sudo -v
$ sudo emacs /etc/sysconfig/network &
```

and edit the line

```
HOSTNAME=localhost.localdomain
```

and change it into

HOSTNAME=⟨*your name*⟩.tyict.vtc.edu.hk

where ⟨*your name*⟩ is your own name; please try to make this unique, so that it is not the same as anyone elses.

- Do *not* add spaces, or any characters except letters and digits and hyphen. Note the first character of a host name should be a letter. I suggest just use lower case letters.

2. Edit the host file, `/etc/hosts`:

```
$ sudo -v
$ sudo emacs /etc/hosts &
```

This file is used by the operating system to map host names to IP addresses and back, before the operating sytem checks the name server.

Note that on Windows, the file is in
`%SystemRoot%/System32/drivers/etc/hosts`

Edit the line

```
127.0.0.1 localhost.localdomain localhost
```

and change it to:

127.0.0.1 ⟨*your name*⟩.tyict.vtc.edu.hk ⟨*your name*⟩ localhost.localdomain local

3. Then restart networking with:

```
$ sudo service network restart
```

4. Log out and log in again.

Your hostname is now changed to ⟨*your name*⟩`.tyict.vtc.edu.hk`.

13.26 Configuring your own account to use `keychain`

- Next add these lines to the end of your login script `~/.bash_profile`:

```
[ -x /usr/bin/keychain ] && keychain ~/.ssh/id_rsa
[ -r ~/.keychain/${HOSTNAME}-sh ] && source ~/.keychain/${HOSTNAME}-sh
```

- Remove the line:

```
eval $(ssh-agent)
```

  from your log in script `~/.bash_profile`.
- Add this to the end of `~/.bashrc`:

```
[ -r ~/.keychain/${HOSTNAME}-sh ] && source ~/.keychain/${HOSTNAME}-sh
```

  Note that when you log in, `bash` will *source* your log in script, `~/.bash_profile`. Every time you start a new `bash` process, then `bash` will source `~/.bashrc`.

- Log out and log back into the computer. You will be prompted to enter your passphrase. You will never need to enter it again until you reboot the computer.

Automating network transfers using `keychain` and SSH

- You can run shell scripts from your `cron` table. See section § 3.29 on page 88. Add the line:

```
[ -r ~/.keychain/${HOSTNAME}-sh ] && source ~/.keychain/${HOSTNAME}-sh
```

  near the beginning of your script. It will be able to use the secure shell commands to connect to other computers.

13.27 Running X applications remotely

- You can run *X* applications remotely.
- Simply log into the remote machine using `ssh`,
- type the name of the graphical program, and it just works, as if it were running locally!
- The program executes on the remote computer, but the *X* protocol sends the graphics commands through the encrypted session,
- the application is displayed on your local machine, though the processing is done remotely.
- Great for running remote graphical system administration applications (remote administration)

13.28 Configuring SSH for X

Note that the settings of the RPMs from Red Hat are already configured to support X forwarding.

- The configuration file for the client is `/etc/ssh/ssh_config`
- The configuration file for the server is `/etc/ssh/sshd_config`
- Letting X work over SSH:
  - An important configuration option in `/etc/ssh/ssh_config` is:
    `ForwardX11 yes`
  - An important configuration option in `/etc/ssh/sshd_config` is:
    `X11Forwarding yes`
  - These enable X to work over SSH as described in section 13.27

13.29 Security options for the client in /etc/ssh/ssh_config

- `RhostsAuthentication no`

  disable fallback to insecure remote hosts file "authentication." The default is "yes". This option applies to protocol version 1 only.

- `RhostsRSAAuthentication no`

  disable fallback to insecure remote hosts file and host key authentication. The default is "yes". This option applies to protocol version 1 only.

- `FallBackToRsh no`

  Totally avoid `rsh` (buried on page 345!) The default is "no".

- `PubkeyAuthentication yes`

  Usually turned on by default, but if you have trouble with logging in without passwords, add to `/etc/ssh/ssh_config`.

- `UsePrivilegedPort no`

  This makes SSH work better with some firewalls. The default is "no" Also remove the SUID bit from `/usr/bin/ssh`:

  ```
  $ sudo chmod u-s /usr/bin/ssh
  ```

13.30 `rsync`: using it with SSH to mirror data

More often than `scp` I use `rsync` to transfer data.

- Advantages:
  - `rsync` can preserve the time stamps, ownership, and can transfer device files and symbolic links accurately
  - `rsync` only transfers the differences between files, not the whole file.
    - For example, if a single 50 megabyte email file has one email added at the end, and one deleted from the middle of the file, only a very small fraction of the total file will be transferred.
  - The tool of choice for mirroring web sites, ftp sites, and for downloading ISO images.
- To use it with Secure Shell, either:
  - use the option `-e ssh` with `rsync`, or
  - Add the following line to your login script:

    ```
    export RSYNC_RSH=ssh
    ```
- Most useful options (besides `-e ssh`):

`-a` Preserve **a**ll properties (permissions, ownership, use recursion to copy subdirectories, keep synmbolic links,. . .

`-v` **v**erbose; show the files as they are transferred;

`-z` use compression.

13.31 Examples of using `rsync`

- Make a copy of `~/tmp` on `ictlab` in the current directory.

```
$ rsync -avz -e ssh ictlab:tmp .
```

  - If `tmp` is a directory containing files,
    - all of the files will be copied, and
    - their ownership and permissions will be preserved as far as possible
    - If you are `root`, they will be preserved exactly
    - *Without* a trailing '`/`', the directory `tmp` will be created on the local computer in the current directory.

- Make a copy of `~/tmp` on `ictlab` in the current directory.

```
$ rsync -avz -e ssh ictlab:tmp/ .
```

  - If `tmp` is a directory containing files, same as previous example except that:
    - *With* a trailing '`/`', the directory `tmp` will *not* be created on the local computer in the current directory; the files and subdirectories of `~/tmp` on `ictlab` will be copied to the current directory.

13.32 Using `ssh` from Windows, with Cygwin

- Install Cygwin by:
  - Go to `http://cygwin.com/`, click on the "Install Cygwin now" icon
  - Install at least `XFree86`, `openssh` and `cygrunsrv`, but I prefer to install the lot (it makes Windows much less painful to me).

  Note that Henry has already installed cygwin on all the computers in our A204 laboratories.

- From the Cygwin icon on your desktop, or the Cygwin menu item at Start → Programs → Cygwin, start a `bash` shell.
- At the `bash` prompt, type:

```
$ startx &
```

  On older Cygwin installations in our laboratories, this starts a `twm` window manager session. You can type

```
$ man twm
```

  on the Windows machine (if you installed `man`), or on any Linux machine for the manual that explains how it works.

- In a terminal window in the resulting X session, type:

  `$ ssh -X` ⟨*your user name*⟩`@`⟨*IP address*⟩

- When you connect to the remote machine, you can run any graphical application on that machine, and display it locally, so you could run emacs remotely on this Windows machine in a graphical mode:

```
$ emacs assignment &
```


13.33 What else can SSH do?

- Can execute programs remotely (i.e., for doing a backup of a remote computer)
- Can do “port forwarding”, allowing amazing flexibility in sending any protocol over the encrypted tunnel, so that the application appears to be executing locally. Useful for `VNC` (`http://www.uk.research.att.com/vnc/sshvnc.html`), many other applications.
- Can provide a basic type of encrypted tunnel, a simple VPN (virtual private network)
- Many other things. There is a great book published in 2001 about it: Daniel Barrett and Richard Silverman, *SSH, The Secure Shell: The Definitive Guide* see details at `http://www.oreilly.com/catalog/sshtdg/` (the Snail Book)

13.34 Summary

- SSH allows remote login using strong encryption
- The host public key must be transferred using some other secure means: if you transfer it automatically, it is subject to a *trivial man-in-the-middle attack*.
- The *host key pair* performs two importand functions:
    - authenticate the server (only possible if you transfer the public host key from the server to the client in some secure way, such as by floppy disk)
    - begin encryption
- The *user key pair* has only one role: to provide strong authentication of the user.
- The *agent* is used to hold the decrypted private user key on the client computer. Without the agent, you would need to type your pass phrase with every `SSH` connection.
- The public user key is appended to the file `~/.ssh/authorized_keys` in your user account on the servers you will connect to. You can do this over the network once the public host key is securely copied.
- The actual encryption is done using the *session key*. The session key is never stored on the hard disk, and a new pair is generated regularly during a session.
- The session is encypted using *symmetric encryption*, because it is *much* faster than public key (asymmetric) encryption.

13.35 SSH References

- A useful guide to OpenSSH: `http://perso.club-internet.fr/ffaure/openssh.html`
- A concise guide to OpenSSH: `http://www.samag.com/archive/0910/feature.shtml`, the *SysAdmin Magazine*
- The OpenSSH press page lists many useful resources: `http://www.openssh.com/press.html`
- The Red Hat Reference Guide has a chapter on SSH: `http://www.redhat.com/docs/manuals/linux/RHL-9-Manual/ref-guide/ch-ssh.html`, or locally, `http://nicku.org/doc/rhl-rg-en-9/ch-ssh.html`
- The Snail Book FAQ: `http://www.snailbook.com/faq/` (and, of course, the Snail Book itself in the library)
- SSH FAQ: `http://www.onsight.com/faq/ssh/ssh-faq.html`
- OpenSSH FAQ: `http://www.openssh.com/faq.html`
- USENET SSH FAQ: `http://www.faqs.org/faqs/computer-security/ssh-faq/`
- ssh mailing list: `http://www.onsight.com/faq/ssh/ssh-faq-8.html#ss8.3`
- OpenSSH developer's mailing list: `http://www.openssh.com/list.html`
- Don't forget the OpenSSH `man` pages!

13.36 Secure Shell Exercises

1. Set up `sudo` with a line like this:

   your-user-name ALL=(ALL) ALL

   as described in the handout about `sudo`.

2. Check that the directories `/sbin` and `/usr/sbin` are on your `PATH`:

   ```
   $ echo $PATH
   ```

   These directories contain system administrator's commands. If you do not see them there,

   - add this command to your login script. Be careful: do not overwrite your login script!

     `$ echo 'PATH=$PATH:/sbin:/usr/sbin' >> ~/.bash_profile`

   - *source* this file into your shell process:

     `$ source ~/.bash_profile`

   - Now check your `PATH` contains those two directories:

     `$ echo $PATH`

3. Identify your IP address, and tell your neighbour:

   ```
   $ ip addr
   1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
   2: eth0: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast qlen 100
       link/ether 00:01:03:45:99:12 brd ff:ff:ff:ff:ff:ff
       inet 172.19.32.30/22 brd 172.19.35.255 scope global eth0
   ```

   You can see my address is `172.19.32.30`.

4. Connect to your neighbour's computer, then log out again:

   $ ssh ⟨*neighbour's IP address*⟩
   $ exit

5. After your neighbour connected to you, compare the content of your `~/.ssh/known_hosts` file with the public host key on your partner's computer, in `/etc/ssh/ssh_host_rsa_key.pub`. Are they different? Why or why not?

6. Use the `scp` program (see section 13.7 on page 348) to copy a file to the `/tmp` directory on your neighbour's computer.

7. Log into your account on your neighbour's computer (as described in section 13.6 on page 347) and verify that you copied the file to their machine.

8. Set up your own private and public keys as described in section 13.13. You should *use a pass phrase* for this exercise. Do this on the client machine only.

9. Set up your `~/.ssh/authorized_keys` on the server using the methods described in section 13.15. Note: your network drive is the same for your account, both on the local client and on the remote server, so you can simply append your public key to your `~/.ssh/authorized_keys` file without using `scp`. Think about it!

10. Setup `ssh-agent` on your client computer and use `ssh-add` to make your personal private key available to `ssh`, as described in section 13.20.
11. Configure `keychain` as explained in section 13.23 on page 364.
12. Demonstrate to your tutor that you can log in to your account on your neighbour's computer without typing any passwords. Show that you can execute a graphical program, such as `emacs` or `xclock`.
13. Transfer the entire content of your home directory (complete with the directory name) to the `/tmp` directory of your neighbour, using `rsync`, preserving all timestamps and permissions. Use the `--stats` option to `rsync`. Verify the the permissions and time stamps are preserved.
14. Modify one or two files in your home directory a little. Repeat the data transfer of your home directory to the same location in your same neighbour's `/tmp` directory, again using the `--stats` option. Do you notice any difference?
15. To avoid being hihacked, it is wise to transfer a host public key on a floppy disk and append it to the file `/etc/ssh/ssh_known_hosts`.
    (a) Should you take the host key from your client machine to the `/etc/ssh/ssh_known_hosts` on the server, or should you take the public key from the server and install it on your client?
    (b) What two important functions are performed using the *host* public and private key pair?
    (c) What function is performed by the *user* public and private key pair?
16. Install Cygwin on your Windows 2000 machine, and then set up the secure shell server, as described in `http://tech.erdelynet.com/cygwin-sshd.html`.

13.37 Secure Shell Solutions

Module 14

Shared File Systems

Objectives

After completing this chapter you should be able to:

- Understand basic remote file and print sharing
- Appreciate the pros and cons of Samba and NFS
- Install Samba and NFS servers
- Configure basic Samba and NFS services
- Access remote resources using SMB and NFS

14.1 NFS (Network File System)

- NFS developed by Sun Microsystems (early 80's)
- Native method for file sharing between Unix/Linux systems
- Stateless protocol
  - Means server keeps no state
  - Renders server crashes 'easily recoverable'
- Should be compatible with all Unix-like systems
- Best in trusted environment, not highly secure
- Best where all user/group IDs are same
- Often used with *Network Information Services* (NIS) to synchronise user/group IDs

14.2 NFS Basics . . . continued

- Systems are clients, servers or both
- Clients *import* shared filesystems
- Servers *export* shared filesystems
- Servers easy to implement via network daemons
- Clients require kernel modifications
- Linux systems normally work as both already
- NFS is NOT Unix/Linux specific (e.g. PC-NFS)

14.3 Exporting File Systems

- Exporting handled by daemons `rpc.nfsd` and `mountd`
- Must be running for NFS export to work
- Exported file systems listed in `/etc/exports`, format is:
  `fsname hostname(flags) [hostname(flags)]`
- Example:
  `/tmp *.blah.co.uk(ro)`
  Exports `/tmp` to all systems belonging to domain read-only *
- Important flags:
  - `ro` (read only)
  - `rw` (read/write)
  - `all_squash` (map all uid/gid to something)
  - `anonuid` (specify user ID to map to)
  - `anongid` (specify group ID to map to)
- After changing `/etc/exports`, restart NFS

  ```
  exportfs -av
  ```

  or

  ```
  killall -HUP rpc.nfsd
  killall -HUP mount
  ```

  or

  ```
  /etc/rc.d/init.d/nfs restart
  ```

*For full detail on flags use `man exports`

14.4 Viewing exports

- Use showmount:

  ```
  $ showmount -e
  $ showmount -e hostname
  ```

  ```
  Export list for landlord.gbdirect.co.uk:
  /usr/local/gbdirect/cvsroot roti.gbdirect.co.uk
  /home/adamg                 roti.gbdirect.co.uk
  /home/andylong              along2.gbdirect.co.uk
  /home/mikeb                 kebab.gbdirect.co.uk
  /mnt/cdrom                  <anon clnt>
  ```

- `NFS` uses a `portmapper` to handle requests
- This must be running (and you must have access to it) to use `NFS`
- Check that `hosts.allow` contains an entry to permit you access, e.g.
  - `portmap:  ALL`
    or
  - `portmap:  my.ip.network.`

14.5 Importing File Systems

- Mount a remotely exported directory
- Usually have to be superuser
  `$ mount hostname:/sharename /local/directory`
- If successful, the export named `/sharename` on host *hostname* is mounted on your *mountpoint* `/local/directory`
- Files accessed just as if local
- Remote host must be exporting the directory
- You must have access permission
- Your local mountpoint must exist
- Exactly like mounting a device

14.6 Samba

- Implementation of *Server Message Block* protocol (SMB)
  - Core of Microsoft's file and print sharing
  - Now *'re-invented'* as *CIFS*
- Developed in Australia by Andrew Tridgell et al
- Info, sources, distributions at www.samba.org
- High performance – competitive with NT
- Server is purely application code
  - Not part of the OS
- Provides some clients
  - `smbfs` requires OS support
  - Client module `smbfs` not part of Samba

14.7 Samba — Availability

- Samba is provided packaged with all large Linux distributions
- PDC support for Windows 2000 clients is new, currently only with version 2.2 and is available from CVS
- Samba 2.2 is currently *alpha* quality software
  - will have reached production quality by your graduation
- Nick Urbanik `<nicku@nicku.org>` has packaged the current Samba 2.2 into an RPM, ready for installation into Red Hat 7.
- Currently available from `CSAlinux:/var/ftp/pub/samba`, `http://CSAlinux.tycm.vtc.edu.hk/ftp/samba/`, `ftp://CSAlinux.tycm.vtc.edu.hk/ftp/pub/samba/`, `\\CSALINUX\pub\samba`
- Expect further updates. I will improve the RPM to work as a PDC with minimum cutomisation required.

14.8 Samba Documentation

- The book *Using Samba* is distributed free with Samba (or buy for HK$315)
- Documentation about using Samba as a PDC is currently available from `http://us1.samba.org/samba/docs/samba-pdc-faq.html` and `http://us1.samba.org/samba/docs/samba-pdc-howto.html`.
- Latest docs are available from `http://us1.samba.org/samba/docs/`
- The documentation for the `Samba` configuration file is important:

```
$ man smb.conf
```

14.9 Samba Installation

- Will vary — may come preinstalled, may come as RPMs or similar
- Key components are `nmbd` and `smbd`
  - `nmbd` is the name services daemon; mostly fit-and-forget
  - `smbd` is the samba server; listens for connections and then forks one copy per client
- Other tools & utilities exist, e.g. `smbclient`
- Configuration file is `/etc/samba/smb.conf`
- Comes with the *Samba Web Administration Tool* (`swat`); listens on port 901
- To install Nick's RPMs, do:

```
$ sudo mount CSAlinux:/var/ftp/pub /mnt
$ cd /mnt/samba
$ sudo rpm -Uhv samba*.i386.rpm
```


14.10 Samba Basics

- Most likely started as *daemons* in *init scripts*
- Can be run-on-demand via `inetd`, but unlikely
  - Gives poor performance
- Exclusively uses *TCP/IP*. Microsoft clients need to be configured for it — they may use *NETBEUI*
- Permits:
  - full file sharing, browsing and domain controller services
  - full access to printers
  - extensive customising

14.11 Access to Files and Printers

- Linux and Win/NT access controls don't match
- Various options can be set
- Attempts to match logged-on Windows Username to Linux user names and passwords
- Modern versions use encrypted passwords — takes some setting up (see documentation)
- Has concept of 'guest' users — may map to 'nobody' on Linux
- Take a look in your `smb.conf` file and read `man smb.conf`

14.12 Testing Samba

- Use `smbclient` (see screen dump below)
- May need to provide a password
- Check `DIAGNOSIS.TXT` from distribution (usually installed at `/usr/share/doc/samba-2.`x`.`x) if you have problems

```
$ smbclient -L localhost
Added interface ip=192.168.0.129 bcast=192.168.0.255
nmask=255.255.255.0
Password:
Domain=[GBDIRECT]     OS=[Unix]    Server=[Samba 2.0.3]

Sharename       Type    Comment
--------        ---     -------
www             Disk    WWW     files
software        Disk    Installable     Software
tmp             Disk    Temporary       file    space
admin           Disk    GBdirect        admin   files
printers        Printer All     Printers
IPC$            IPC     IPC     Service (Samba  Server)
okirmt          Printer
txtdj           Printer
djrmt           Printer
fax             Printer

Server          Comment
--------       -------
LANDLORD        Samba   Server

Workgroup       Master
--------       -------
GBDIRECT        LANDLORD
WORKGROUP       KEBAB
```

14.13 Smbclient

- Numerous options:

```
smbclient servicename  [password]  [-s
smb.conf]  [-B  IP addr]  [-O socket options][-
R name resolve order] [-M Net-BIOS name] [-i
scope] [-N] [-n  NetBIOS  name]  [-d  debu-
glevel] [-P] [-p port] [-l log basename] [-h]
[-I dest IP][-E] [-U username] [-L NetBIOS
name]  [-t  terminal  code][-m  max  protocol]
[-W workgroup] [-T<c|x>IXFqgbNan] [-Ddirectory]
[-c command string]
```

- Example:

```
$ smbclient //landlord/admin
Added interface ip=192.168.0.129
bcast=192.168.0.255 nmask=255.255.255.0
Password: xxxxx
Domain=[GBDIRECT] OS=[Unix] Server=[Samba 2.0.3]

smb: \> ls
  q3.dir                    85  Tue Jun 29 13:01:44 1999
  actwin2          D         0  Sun Mar  7 22:01:28 1999
  courses          D         0  Wed May 12 10:02:20 1999
  cvs              D         0  Mon Mar 22 12:36:13 1999
  domreg           D         0  Tue Sep  1 10:14:12 1998
  finance          D         0  Thu Jul  1 12:33:49 1999
  informat         D         0  Wed Jun 23 09:56:34 1999
  julie            D         0  Fri Jul  2 10:06:43 1999
............. etc ...........
```


14.14 Samba configuration File

- Three sections to `smb.conf`
  - global
  - directories
  - printers, if enabled, will export the printers known in `/etc/printcap`
- Lots of help in the book *Using Samba*, on line with installation.
  - With Red Hat 7, and Nick's RPM, it is available under `/usr/share/swat/using_samba/`
- Lots of other documentation comes with Samba:
  - Usually under `/usr/share/doc/samba-versionnumber`, e.g. `/usr/share/doc/samba-2.2.2`
- Read the man pages
- Via the web
- and others

14.15 Samba Configuration Example

- This is an example `/etc/samba/smb.conf`, suitable for use with Nick's Samba RPM:

```
[global]
  security = user
  status = yes
  workgroup = { Your domain name here }
  wins server = { ip of a wins server if you have one }
  encrypt passwords = yes
  domain logons =yes
  logon script = scripts\%U.bat
  domain admin group = @smbadm
  add user script = /usr/sbin/useradd -n -g machines
               -c Machine -d /dev/null -s /bin/false %m$
  share modes=no
  os level=65
[homes]
  guest ok = no
  read only = no
  create mask = 0700
  directory mask = 0700
  oplocks = false
  locking = no
[netlogon]
  path = /var/samba/netlogon
  writeable = no
  guest ok = no
```


14.16 Directories for Samba as a PDC

- Need some directories to hold user profiles and login scripts
  - match the above configuration

```
$ sudo mkdir -p /var/samba/netlogon/scripts
$ sudo chown -R root.root /var/samba/netlogon
$ sudo chmod -R 755 /var/samba/netlogon
```

14.17 Testing Samba

- Use `testparm` and `smbstatus`
  - `testparm` is used before starting Samba to check that `smb.conf` is ok
  - `smbstatus` reports status of Samba, all connected clients and file share modes

Notes on Testing Samba

- Note that Samba is a server implementation
- Has ftp-like `smbclient`, but file share access is provided the kernel.
- Cannot be used by Linux to *import* shared files, only export them
- Some Linuxes have import facilities too — but requires kernel support (`smbfs` module)

14.18 Exercises

1. *NFS*
   (a) Set up your local host so you can use `showmount` to show exported directories.
   (b) Find other hosts on your network which list exports.
   (c) Set up your host to export `/tmp`
   (d) Go to some other system and mount the exported `/tmp`
   (e) Play with file access on the mountpoint!, e.g. Try accessing files you normally wouldn't have access to, creating files and seeing what the ownership and permissions are on the local copy.
2. *Samba*
   (a) Locate the file DIAGNOSIS.txt
   (b) Read through it, then carefully work through *all* of its instructions to check your Samba installation.
   (c) Run `testparm` on your current `smb.conf`, pipe the output through `less` to see the results.
   (d) Run `smbstatus` and explain to your neighbour what the results mean.
   (e) Set up a share so that your `/etc` directory is exported read-only and test it with `smbclient`.
   (f) Figure out how to export users' home directories and get a colleague to test your work.

14.19 Solutions

1. *NFS*

   (a) You should ensure that the `portmap` and `nfs` services are running before using `showmount`

   (b) You can give `showmount` a hostname to query, e.g.

```
$ /usr/sbin/showmount -e somehost
Export list for somehost:
/home/adamg                 roti.gbdirect.co.uk
/home/lee                   rafters.gbdirect.co.uk
/backup                     <anon clnt>
/home/james                 oakleigh.gbdirect.co.uk
/mnt/cdrom                  <anon clnt>
```

   (c) You should add the following to `/etc/exports`:

```
/tmp    *(ro)
```

   (d) -

   (e) -

2. *Samba*

   (a) The file should be in `/usr/share/doc/samba-x.xx/docs/textdocs`

   (b) You should carry out all the test given to reach a working samba system

   (c) Check that your `smb.conf` is correct

   (d) Check the various smb manpages (`smb.conf`, `smbd`, `smbstatus` to see what the output means

   (e) You should add the following to your `smb.conf`, and restart samba with `/etc/rc.d/init.d/smb restart`:

```
[etcshare]
        path = /etc
        comment = Shared etc directory
        writeable = no
        browseable = yes
```

(f) You should ensure that the *homes* share is uncommented in `smb.conf`, and restart samba if necessary. You can test this by using:

```
$ smbclient '\\localhost\username'
added interface ip=192.168.0.135 bcast=192.168.0.255 nmask=255.255.255.0
Password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 2.0.6]
smb: \> dir
.muttrc          H  1387  Thu Jan 27 11:45:21 2000
.addressbook.lu  H  2285  Mon Jan 24 14:37:29 2000
.procmailrc      H    38  Mon Jan 24 18:11:04 2000
.newsrc.eld      H  1151  Tue Jan 25 13:24:44 2000
.mail_aliases    H    56  Thu Jan 27 16:52:13 2000
Desktop          D     0  Tue Feb  8 15:48:40 2000
.opera          DH     0  Thu Jan 27 12:53:23 2000
.balsarc         H  1391  Wed Feb  9 14:27:36 2000
.mozilla        DH     0  Wed Feb  9 10:27:15 2000
ltculogo.gif        8202  Wed Feb  9 11:49:06 2000
LANDLORD         D     0  Wed Feb 16 14:49:16 2000
nltculogo.xcf      53886  Wed Feb  9 12:54:53 2000
nltculogo.gif       8398  Wed Feb  9 13:09:16 2000
```

Module 15

Apache Basics

Objectives

On completion of this module you should be able to:

- Install the `Apache` webserver
- Perform basic configuration

15.1 What is Apache?

- Apache is the most widely-used web-server*
- Listens for requests and hands something back
- Normally the contents of a file
  - Possibly the result of a program
- Designed to be stable and configurable
  - Fast at serving synamic content
  - Use kernel based http server tux or khttpd for static content for maximum speed

*61.88% of all servers as of February 2001 (Netcraft — http://www.netcraft.com/Survey/Reports/200102/platform.html)

15.2 Installation

- Basic installation is easy
- You may be able to install from your distribution
  - Most come with `Apache`
- Otherwise just follow the download instructions from the official site
  - `http://www.apache.org/`
- Then follow the instructions in the `INSTALL` file
  - Normally just

    ```
    $ ./configure
    $ make
    $ make install
    ```
  - If you have problems check the docs
  - Available at `http://www.apache.org/docs`

15.3 How Apache Listens

- Apache runs several processes at any one time
  - Parent and several children
- Parent *‘watches over’* the children
  - Tracks how many are answering requests
  - Spawns more if free processes drop below a certain point
  - Kills spare processes if there are lots free
- Configure child numbers using *MinSpareServers* and *MaxSpareServers* directives
  - Default is reasonable for a small business
  - Tune it for busier sites

15.4 Configuration File(s)

- If compiled from source, `Apache` installs in `/usr/local/apache`
  - Earlier versions installed under `/usr/local/etc/httpd`
  - Your distribution may differ again . . . *
- Configuration file is called `httpd.conf`
  - Older versions use
    - `httpd.conf`
    - `srm.conf`
    - `access.conf`
- Controls what requests `Apache` answers
  - and how . . .

*Redhat installs config files under `/etc/httpd` and the sample web pages and logs directories under `/home/httpd`

15.5 Key Configuration Directives

- Wide range of *configuration directives*
- For a *very* basic server you need at least the following:
  - *ServerRoot*
  - *DocumentRoot*
  - *ServerAdmin*
  - *BindAddress*
  - *Port*
  - *Listen*
  - *User*
  - *Group*

15.6 *ServerRoot*, *DocumentRoot*

- Tells `Apache` where its files live
- *ServerRoot* tells `Apache` where its `conf` and `logs` directories live
  - Not always necessary
  - Good practice to have it
- *DocumentRoot* tells `Apache` where to look for documents to serve up
- Requested filenames are appended to this
- If you have

  ```
  DocumentRoot /var/www/html
  ```

  then a request to

  ```
  http://www.domain.co.uk/foo.html
  ```

  points to the file `/var/www/html/foo.html`

15.7 Is `Apache` running?

- Sometimes it is useful to check the server using the `telnet` program:

```
$ telnet csalinux 80
Trying 192.168.128.53...
Connected to CSAlinux.tycm.vtc.edu.hk (192.168.128.53).
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 05 Mar 2001 01:51:59 GMT
Server: Apache/1.3.14 (Unix)  (Red-Hat/Linux) mod_ssl/2.7.1
        OpenSSL/0.9.5a DAV/1.0.2 PHP/4.0.4pl1 mod_perl/1.24
Last-Modified: Wed, 28 Feb 2001 05:23:07 GMT
ETag: "28363-129e-3a9c8b3b"
Accept-Ranges: bytes
Content-Length: 4766
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>CSA Linux</TITLE>
 </HEAD>
 <BODY>
  ...
 </BODY>
</HTML>
Connection closed by foreign host.
```


15.8 *ServerAdmin*

- `Apache` sometimes can't complete requests
- In these cases it serves up an error page
- *ServerAdmin* is given as a contact address
- Usually set to something like

  `webmaster@tycm.vtc.edu.hk`

  - You should of course ensure that it is a *valid* email address
- Possible to specify a different error page
  - Doesn't have to use ServerAdmin

15.9 *BindAddress*, and *Port*

- Tells `Apache` which requests to answer
- By default `Apache` listens to every IP address on your machine
  - But only to the port given by the *Port* directive
- `BindAddress 192.168.0.1` tells `Apache` to ignore anything that doesn't come in on `192.168.0.1`
- `Port 8080` ignores all but the specified port
- You can use more than one *Port* directive, e.g.

```
Port 80
Port 8080
```

- If you don't specify a port then a default is used*
- You can only use one *BindAddress*!

*This is usually 80, but if you are using a binary package then bear in mind whoever compiled your package may have chosen a different value

15.10 *Listen*

- *Listen* is a replacement for *BindAddress* and *Port*
- Given IP:port or just port, e.g.

  ```
  Listen 192.168.0.1:8080
  ```

  will answer requests on the IP address `192.168.0.1` and port `8080` and no others
- To answer requests to all valid IP addresses, but only a certain port (e.g. `80`) use:

  ```
  Listen 80
  ```

- Can use more than one *Listen* directive
- Should be used instead of *BindAddress* and *Port* in new servers

15.11 *User* and *Group*

- `Apache` should normally be started as *root*
  - So it can change the user ID of the children
  - These should *not* run as root
- *User* and *Group* directives say what user/group the children should run as
  - Important security feature
- Should be set to something that has no real power on your system
  - Most people use user and group `nobody`
- Web documents should be readable by this user
- Nothing should be writeable except log files

15.12 Apache Processes

- Looking at a process list* you can see
  - The parent

    ```
    root S Jul  4  0:37 /usr/local/apache/bin/httpd -d /www
    ```
  - The children

    ```
    nobody  S  10:11  0:00 /usr/local/apache/bin/httpd -d /www
    nobody  S  10:20  0:00 /usr/local/apache/bin/httpd -d /www
    nobody  S  10:58  0:00 /usr/local/apache/bin/httpd -d /www
    nobody  S  10:58  0:00 /usr/local/apache/bin/httpd -d /www
    nobody  S  11:06  0:00 /usr/local/apache/bin/httpd -d /www
    nobody  S  11:13  0:00 /usr/local/apache/bin/httpd -d /www
    ```
- Spare processes don't use processor time
  - They are *'sleeping'*
- They *do* use memory, however
  - Negligible for a default `Apache`
  - Watch carefully the more modules you add!
  - Particularly, `mod_perl` adds a heavy memory requirement.

*Some fields from the `ps` output have been left out to aid clarity

15.13 Logging

- `Apache` can log information about accesses
- Use the *TransferLog* and *ErrorLog* directives
- `TransferLog logs/access_log`
  will log all requests in the file
  `ServerRoot/logs/access_log`
- If the filename starts with a `/` then it is treated as a proper pathname, not appended to *ServerRoot*
- *ErrorLog* is similar but controls where error messages go
  - Useful for debugging CGI scripts and misconfigurations
  - Check here first if `Apache` won't start

15.14 Customizable Logging

- Customizable logs available with *CustomLog*

  `CustomLog filename format-string`

- `format-string` consists of *'% directives'* and/or text
- *% directives* include:

`%b`	Bytes sent, excluding HTTP headers
`%f`	Filename
`%{headername}i`	The contents of headername: header in the request
`%P`	The process ID of the child that serviced the request
`%r`	First line of request
`%t`	Time, in common log format time format
`%T`	The time taken to serve the request, in seconds
`%u`	Remote username (may be bogus if return status (%s) is 401)
`%U`	The URL path requested
`%v`	The ServerName of the server answering the request

15.15 *CustomLog* examples

- To log the referer information in the file `ServerRoot/logs/referer`

```
CustomLog logs/referer "%r Refered by: %{Referer}i"
```

- *% directives* can be conditional on reply status

```
CustomLog logs/referer "%r Refered by: %200,304,302{Referer}i"
```

  - Logs the refering page only on status 200,304,302 *

- For full details consult the Apache documentation
  - Gives list of all possible *% directives*

*For full details consult the Apache documentation

15.16 Example Configuration

- A sample configuration file could look like this:

```
ServerRoot /usr/local/apache
DocumentRoot /usr/local/apache/htdocs
ServerAdmin webmaster@domain.co.uk
Listen 192.168.0.131:80
User nobody
Group nobody
ErrorLog /usr/local/apache/logs/error_log
```

- We recommend starting with the default `httpd.conf` rather than from scratch
  - Correctly configures many things for you
- The default is well annotated
  - Everything after a # character is a comment
  - Ignored by `Apache`
- `Apache` can check the syntax of its configuration
  - `httpd -t`
  - `apachectl configtest`

    . . . if you installed `apachectl` on your system.

15.17 Basic Exercises

1. *Apache Installation*
   - **(a)** Find out if Apache is installed on your machine . . . if not, install it.
   - **(b)** Check Apache is running on your system.
     - **i.** You should be able to point your web browser at `http://127.0.0.1/` to check this
     - **ii.** You might have to try `http://127.0.0.1:8080/`
   - **(c)** If Apache is not running, start it with
     `/etc/rc.d/init.d/httpd start`
   - **(d)** If Apache still doesn't appear to be running, find its configuration and log files and try to fix the error.
2. *Basic configuration*
   - **(a)** Familiarise yourself with the `httpd.conf` file.
   - **(b)** How would you change the directory where the log files are kept?.
   - **(c)** How would you change the 'root' for documents?
   - **(d)** How would you enable symbolic links to be followed on the cgi-bin directory? **Warning:** this is a *really bad* idea!
   - **(e)** Make your site only accessible on Port 8080
   - **(f)** Now make it only accessible on the IP address 127.0.0.1, and port 80
   - **(g)** Make the changes and check them.
   - **(h)** Place the following line in your `/etc/hosts` file:

     ```
     IP_ADDRESS www.test.com www
     ```

     where `IP_ADDRESS` is the IP address of your machine. You should now be able to browse `http://www.test.com/`
3. *Logging*
   - **(a)** Take a look at the access logs and familiarise yourself with the information they contain.
   - **(b)** Set up a custom log to give the time of the request, the request, referer, and number of bytes sent, as well as the time taken to serve the request.
   - **(c)** Alter your custom log to show the time taken and bytes sent *only* if a 200 status response occured.

15.18 Solutions

1. *Apache Installation*

   (a) If Apache is not installed you should be able to install it off a RedHat CD by mounting the CD and typing `rpm -ivh /mnt/cdrom/RedHat/RPMS/apache.rpm`

   (b) There are several ways to check this. One is to `telnet` to port 80 of your machine and see if you get a response.

   i. This should work for a default RedHat install, though the port number that Apache first listens on changes in various different packaging so you should try both 80 and 8080.

   (c) You can start Apache one of two ways (Which may be the same on some machines!)

   - /etc/rc.d/init.d/httpd start
   - somepath/apachectl start *

   (d) If you can't work out why Apache isn't running ask the tutor for assistance.

2. *Basic configuration*

   (a) You should make sure that you understand everything in the `httpd.conf` including those sections that are commented out.

   (b) Alter the `CustomLog` and `ErrorLog` directives to change where the log files are kept, e.g.

```
ErrorLog /var/log/myerrorlog
CustomLog /var/log/myaccesslog common
```

   (c) The 'root' for documents is specified by the `DocumentRoot` directive, e.g.

```
DocumentRoot /path/to/my/web/documents
```

   (d) You can enable symbolic links by adding `Options +ExecCGI FollowSymLinks` to the `<Directory>` section for your `cgi-bin`, e.g.

```
<Directory /path/to/my/web/documents/cgi-bin>
Options +ExecCGI FollowSymLinks
</Directory>
```

   (e) Add/Change the `Port` directive in your `httpd.conf` file to read `Port 8080`

   (f) Add the following to your `httpd.conf`: `Listen 127.0.0.1:80`

   (g) Restart the server and try to access it on both port 80 and 8080. Check that it only works as you expect and fetches documents from the correct place.

   (h) Check that you can browse `http://www.test.com`

*You may have to dig a little to find where this script is

3. *Logging*

(a) Make sure you understand what each of the columns in the access logs is for. Try `tail`ing the logs as you browse your webserver

(b) The following should create a file `newlogformat` which holds the desired log format.

```
LogFormat "%t %U %{Referer}i %b %T" newlog
CustomLog logs/newlogformat newlog
```

(c) Change your LogFormat line to

```
LogFormat "$t %U %{Referer}i %200b %200T" newlog
CustomLog logs/newlogformat newlog
```


Module 16

Apache

Objectives

On completion of this module you should be able to:

- Set up *virtual hosts* on the `Apache` webserver
- Use access controls
- Set up basic authentication
- Configure WebDAV web publishing

16.1 Two sites and more . . .

- Many companies specialise in `web hosting`
- One company may manage thousands of web sites
- One solution is to buy thousands of servers and set up each web site on each server.
  - Problem: cost
  - Must be a cheaper way
- There is a simpler way
  - Called *Virtual Hosting*
- Virtual hosting allows one server to provide many independent web sites
  - each web site has its own name
  - each web site is independent of the other
  - Apache is better at this than any other web server

16.2 Two sites and more . . . continued

- Apache can serve multiple sites easily
- Known as *'Virtual Hosting'*, e.g.

```
<VirtualHost 192.168.0.2>
DocumentRoot /www/web.test2/docs
ServerName www.test2.co.uk
ServerAdmin www@test2.co.uk
ErrorLog /www/web.test2/logs/error_log
TransferLog /www/web.test2/logs/access_log
</VirtualHost>
```

  to make apache answer requests to address 192.168.0.2 from /www/web.test2/docs

- Your machine must answer to this address*
- Apache must be listening on the address
  - Listen 80 will make Apache answer to all available addresses on port 80
  - Note that each virtual web site may have its own logging
- This is known as *virtual hosting*

*If you don't know how to set up IP aliases ask the instructor

16.3 Virtual Hosting Options

- IP-based
  - Each site must have a unique, IP address
  - Uses up valuable IP addresses
  - Site accessible by all browsers
- Name-based
  - Sites share an IP address
  - Useful if short of available addresses
  - Some browsers may have problems
- Most use IP-based hosting where possible
- Ensures maximum accessibility
- However, Internic are now requiring all sites to use name based hosting as much as possible to reduce drain on IP addresses.
- Name based hosting doesn't work with SSL encryption.

16.4 Name-based hosting

- Name-based hosting looks like:

```
NameVirtualHost 10.1.1.108

<VirtualHost 10.1.1.108>
DocumentRoot /var/www/foo/docs
ServerName foo.domain.com.hk
ServerAdmin foomaster@domain.com.hk
</VirtualHost>

<VirtualHost 10.1.1.108>
DocumentRoot /var/www/bar/docs
ServerName bar.domain.com.hk
ServerAdmin barmaster@domain.com.hk
</VirtualHost>
```


16.5 Name-based hosting (continued)

- `NameVirtualHost` tells `Apache` that an IP address can serve multiple hosts
- `VirtualHost` sections describe how documents for each site are served
  - `Apache` must be able to resolve the names in the `<VirtualHost>` directives to the IP address
- `Apache` looks at the `Host:` header to decide which documents to serve
  - Not sent by all browsers
- Requests on other IP addresses will be processed as normal
- Can use both IP-based and name-based hosting

16.6 IP-based hosting

- IP-based hosting looks like:

```
<VirtualHost 10.1.1.46>
DocumentRoot /var/www/foo/docs
ServerName foo.domain.com.hk
ServerAdmin foomaster@domain.com.hk
</VirtualHost>

<VirtualHost 10.1.1.108>
DocumentRoot /var/www/bar/docs
ServerName bar.domain.com.hk
ServerAdmin barmaster@domain.com.hk
</VirtualHost>
```

  The differences from name-based virtual hosting include:

  - The IP addresses must be different (and Apache must be listening to them!)
  - There is no `NameVirtualHost` directive
  - We may be wasting precious IP addresses!

- The Apache manual contains a very useful and complete guide to implementing virtual hosts
  - It is installed when you installed Apache
  - On Red Hat, it is available as `http://localhost/manual/`

16.7 Block Directives

- `Apache` has several *block directives*
  - Limit enclosed directives to apply to a certain set of *‘things’*
- `<VirtualHost>` is a block directive
  - Enclosed directives apply only to that virtual host
- Others are:

```
<Directory> ... </Directory>
<DirectoryMatch> ... </DirectoryMatch>
<Files> ... </Files>
<FilesMatch> ... </FilesMatch>
<Location> ... </Location>
<LocationMatch> ... </LocationMatch>
```

16.8 Block Directives (continued)

- `<Directory name>` Limits the enclosed directives to apply to everything below the directory `name`
  - `name` can be anywhere on the filesystem
  - Independent of *DocumentRoot*
- `<Location name>` is similar but is a URL path rather than a filesystem path
- `<Files name>` limits directives to files called `name`
  - Path of the file is irrelevant
  - Only checks the file name, not its location

16.9 *DirectoryMatch*, et al.

- `DirectoryMatch`, `FilesMatch` and `LocationMatch` are similar
  - Accept regular expressions as arguments, e.g.

    ```
    <FilesMatch .*\.cgi>
    ...
    </FilesMatch>
    ```
- More flexible
- Need more thought to match *only* intended files

16.10 Access Control using `.htaccess` files

- Create a file in the directory to be protected
  - Usually `.htaccess` or `.acl`
  - Can be anything
- Example:

```
AuthType        Basic
AuthName        "Members Only"
AuthUserFile    /etc/httpd/conf/auth.user
AuthGroupFile   /etc/httpd/conf/auth.group
require group   testgroup
require user    testuser
```

- Only the user `testuser`, or a user in the group `testgroup`, may access files in this directory
- Validation is done on the files
  `/etc/httpd/conf/auth.user`
  and
  `/etc/httpd/conf/auth.group`

16.11 Access Control (continued)

- Access control is *off* by default
  - Unnecessary for many sites
- Switched on by:

```
AccessFileName .htaccess

<Directory /www/web.test2>
AllowOverride AuthConfig
</Directory>
```

- `AccessFileName` identifies which filename(s) constitute an Access Control File
- Every directory in the request path is checked for a relevant file
- `AllowOverride` says that Access Control files can override authorisation directives only
  - Can have other values
  - Change behaviours through your `.htaccess` file
  - See `Apache` docs for further details
- Note that there is quite an overhead if turn on `AllowOverride`, since web server has to search for this file through each directory in the entire path of each document it fetches.

16.12 Authorisation Files

- Authorisation files are very straightforward
- Group file is `groupname:  userlist`
- For example:

```
firstgroup: user1 user2 user3
secondgroup: user2 user3 user4
othergroup: user4 user5 user6
```

- Listed users belong to that group
- Create this file by hand

16.13 Authorisation Files (continued)

- User file is a little more complicated
- Format is `username:encryptedpassword`
- For example:

```
testuser:6SlrYaxUFml
```

- Create/edit this with `htpasswd`
  - Part of the `Apache` distribution
  - Give it an authorisation file and a username

```
$ htpasswd -m /etc/httpd/conf/auth.user newuser
New password:
Re-type new password:
Adding password for user newuser
```

16.14 Access Control using `httpd.conf`

- As alternative to using `.htaccess` files, can use main `httpd.conf` configuration file for `apache`
  - centralised
- Exactly the same as using the `.htaccess` files, but put into a block directive in `httpd.conf`
- Refer to the worksheet *How to create a password protected directory on a web server* for more details.

16.15 Pros and Cons of using Access Files for Authentication

- You have a choice to put authentication configuration into `.htaccess` files or into the main server configuration.
- Each has advantages and disadvantages.

`httpd.conf`: advantages

- The server does not waste time reading all directories looking for `.htaccess` files
- The administrator can control all access to the server

`.htaccess`: advantages

- Convenient to modify the configuration; no need to reload the server, just edit the `.htaccess` file.
- Easy to delegate access control to other people

`httpd.conf`: disadvantages

- Harder to delegate authentication control to others
- Need to reload the server to read a new configuration

`.htaccess`: disadvantages

- The server needs to check every single directory, starting with the root directory on the local hard disk of the server, all the way down to the last directory. This slows the server down considerably.

16.16 How Can Users Change Their Password?

- Apache provides no solution to this directly
- Many solutions to this problem are available
- One of the best is `user_manage` by Lincoln Stein, available at `http://stein.cshl.org/~lstein/user_manage/`

16.17 WebDAV: a protocol for web collaboration

- WebDAV is a standard, open protocol for collaboration on the Web
- Allows authors to write to a web server
- WebDAV enabled software (such as Microsoft Office 2000) can edit documents directly on the web server, as if working with a local file
- Currently provides three main facilities:
  1. Locking: WebDAV prevents two authors writing to the same file at once
  2. Properties: information is stored about each file
  3. Namespace manipulation: you can copy and rename files, create collections (which are basically directories on the web server)
- Clients include:
  - Microsoft Internet Explorer 5 and later (“Web Folders”)
  - Microsoft Office 2000
  - DreamWeaver 4.0 and later
  - A number of other commercial and Open Source products. . .
  - . . . but not FrontPage 2000!
- See `http://www.webdav.org/`

16.18 WebDAV and Apache

- Apache has had support for WevDAV for some time.
- Consists of an Apache module called `mod_dav`
- Provided with Red Hat 7.0, enabled by default
- Very stable.
- See `http://www.webdav.org/mod_dav/` and `http://www.webdav.org/mod_dav/install.html`

16.19 WebDAV Configuration

- mod_dav requires a directory to store lock and property information
- You need to provide authentication
- Configuration options include:

Directive	*value*
DAV	On
DAVLockDB	lock file name
DAVMinTimeout	minimum lifetime of a lock in seconds

16.20 Apache WebDAV configuration example

- Here is an example section from the `/etc/httpd/conf/httpd.conf` configuration file:

```
DAVLockDB /var/lock/WebDAV/DAVLock

<Directory "/var/www/html/cm">
    DAV On
    Options Indexes
    AllowOverride None

    AuthType Basic
    AuthName "CM Web site management and upload"
    AuthUserFile /etc/httpd/conf/passwd
    <LimitExcept GET HEAD OPTIONS>
        Require valid-user
    </LimitExcept>
</Directory>
```

- Could instead of `<LimitExcept>...</LimitExcept>` use:

```
    <Limit PUT POST DELETE PROPFIND PROPPATCH
                   MKCOL COPY MOVE LOCK UNLOCK>
        Require valid-user
    </Limit>
```

(all on one line)

16.21 Configuring WebDAV: directories and files

- A web site controlled by WebDAV must be owned and writable by the process running Apache.
- You need to create the lock file directory, and make this owned and writable by the same user.
  - In the example above, you would do:

```
$ sudo mkdir /var/lock/WebDAV
$ sudo chown apache.apache /var/lock/WebDAV
```


16.22 What is WebDAV useful for?

- Useful for a department to collaborate:
  - provide a browsable repository of information
  - members can directly edit these resources
  - A democratised web
- Useful for Home user web publishing. Quote from *WebDAV in 2 Minutes*:

  > A home user can simplify his or her interface to a web server by interfacing with it through DAV (assuming support on the server side). First the user sets up their site on their home computer. Then, using Internet Explorer 5, they set up a Web Folder through the Add Web Folders icon. After providing information on their web server and user id, they can save to and access their directory on the server transparently by using the Web Folder on their desktop, which appears and behaves as a typical local folder.

16.23 What is the future of WebDAV?

- WebDAV is a standard Internet protocol
  - In other words, it is specified by the Internet Engineering Task Force (IETF)
- IETF is working on specifying the following features:
  - Advanced Collections: support for ordered collections, referential resources
  - Versioning and Configuration Mangement: support for maintaining a complete history of all versions of a resource.
  - Access Control: the ability to set and clear access control lists.
- This will allow WebDAV to replace many current Internet protocols, such as POP3, IMAP and CVS.
- Will have potential to help democratise the Web.

16.24 Information about WebDAV

- The best sources of information about WebDAV include:
  - The home page: `http://www.webdav.org/`
  - The WebDAV FAQ: `http://www.webdav.org/other/faq.html`
  - WebDAV in 2 Minutes: `http://www.fileangel.org/docs/DAV_2min.html`

16.25 Other useful directives

- There are around 200 Apache directives
  - More if you add modules e.g. `mod_ssl`
- The previous ones are the *essentials*
- Some other useful directives are given below:

Directive	Action
Redirect `url-path` `new-url`	Redirect Requests to `url-path` to `new-url`
RewriteRule `pattern` `new-pattern`	Rewrite requests, replace `pattern` with `new-pattern`
AddEncoding type ext	Serve up documents with extension ext with encoding type type
ForceType type	Force all documents to be served up with MIME type type
HostNameLookups on—off—double	Whether to do DNS lookups for logging purposes
ExpiresDefault	Set the default expiry time of documents

16.26 Examples

```
Redirect permanent /ents/theatre/fab-gere http://www.fabgere.com
Redirect /gbdirect/logo.gif http://www.gbdirect.co.uk/logo.gif
Redirect permanent /gbdirect http://www.gbdirect.co.uk/

RewriteEngine on
RewriteRule ^/linuxtraining.*\.htm /ltcu_moved.htm

<Location /LTCU>
AddEncoding x-gzip gz
</Location>

<Location /LTCU-plain>
ForceType text/plain
</Location>

HostNameLookups off

<Location /LTCU>
ExpiresDefault "access plus 1 month"
ExpiresByType text/html "access plus 1 week"
</Location>
```


16.27 Exercises

1. *IP based hosting*

   (a) Start with the default installation file and add an IP based virtual host:

   i. Add an IP alias for your machine (Ensure it doesn't clash with any others on your network!) Here are two methods:

   - The simplest is to use the `ifconfig` program directly:

     `$ sudo ifconfig eth0:0` *`ipaddress`*

     where *ipaddress* is the second IP address. If you want to add another alias, use `eth0:1`; use `eth0:2` for the next alias,...

   - The other method uses the `netcfg` program:

   A. Use `sudo netcfg`, then click on the Interfaces tab, select the ethernet device, and click on the Alias button.

   B. Save your changes, activate the alias, and check that your address works, first by typing `ifconfig`, then see if you can ping the address.

   C. If the interface was not started, then do so with `sudo ifup eth0`, or `sudo ifup eth0:0`

   ii. Create a dummy index page so you will be able to tell the difference between your two sites. Call the page `index.html`. Copy them to the document root for each site.

   iii. Set up Apache to serve this site and check from a browser that everything works (for both sites) as you expected.

2. *Name based hosting*

   (a) Set up your apache so that it will serve the same sites but on a single IP address (Name-based virtual hosting).

   i. First, edit your hosts table using `sudo emacs /etc/hosts`.

   ii. Add one line for each web site: put your main IP address first, then the name for the site. Example:

   ```
   10.1.1.39    www.nice.com
   10.1.1.39    www.acme.com
   10.1.1.125   sales.acme.com
   ```

   iii. set up name based virtual hosting for the sites with the same IP address. Verify that you can read them.

3. *Access control*

   (a) Create two directories on one of your sites and set up access controls so that anyone can see the main index page, testuser can see the first directory and anyone in group testgroup can see the second.

4. *WebDAV*

   (a) Configure your main directory with WebDAV, then demonstrate that you can access this directory using the Web Folders option from Internet Explorer (File → Open) on one of the Windows 2000 clients. Make sure that the directory is protected using Basic Authentication.

   (b) Demonstrate that you can edit and save a file on the WebDAV-enabled server directly using Microsoft Word.

16.28 Solutions

1. *IP based hosting*

   (a) The first thing that you will have to do is set up an IP alias for your machine so that it has two distinct IP addresses. You might find it easiest to use the Red Hat program `netcfg` for this. If you aren't sure how to achieve this ask the instructor. A list of spare IP addresses will be made available. An example from a working multi-hosted Apache is given below

```
Listen 192.168.0.3:80
Listen 192.168.0.2:80

<VirtualHost 192.168.0.3>
ServerAdmin webmaster@gbdirect.co.uk
DocumentRoot /home/www/web.llord/docs
ServerName llord.gbdirect.co.uk
ErrorLog /home/www/web.llord/logs/error-log
TransferLog /home/www/web.llord/logs/access-log
</VirtualHost>

<VirtualHost 192.168.0.2>
ServerAdmin webmaster@gbdirect.co.uk
DocumentRoot /home/www/web.trainingpages/docs
ServerName trainingpages.gbdirect.co.uk
ErrorLog /home/www/web.trainingpages/logs/error-log
TransferLog /home/www/web.trainingpages/logs/access-log
</VirtualHost>
```

2. *Name based hosting*

(a) An equivalent example using name-based hosting would be:

```
NameVirtualHost 192.168.0.2

<VirtualHost llord.gbdirect.co.uk>
ServerAdmin webmaster@gbdirect.co.uk
DocumentRoot /home/www/web.llord/docs
ServerName llord.gbdirect.co.uk
ErrorLog /home/www/web.llord/logs/error-log
TransferLog /home/www/web.llord/logs/access-log
</VirtualHost>

<VirtualHost trainingpages.gbdirect.co.uk>
ServerAdmin webmaster@gbdirect.co.uk
DocumentRoot /home/www/web.trainingpages/docs
ServerName trainingpages.gbdirect.co.uk
ErrorLog /home/www/web.trainingpages/logs/error-log
TransferLog /home/www/web.trainingpages/logs/access-log
</VirtualHost>
```

Note that the two names given `llord.gbdirect.co.uk` and `trainingpages.gbdirect.co.uk` should both resolve to 192.168.0.2

3. *Access Control*

(a) You should create a file called `.htaccess` in both directories, the first should be:

```
AuthType        Basic
AuthName        "First Directory"
AuthUserFile    /etc/httpd/conf/auth.user
AuthGroupFile   /etc/httpd/conf/auth.group
require user   testuser
```

and the second should be:

```
AuthType        Basic
AuthName        "Second Directory"
AuthUserFile    /etc/httpd/conf/auth.user
AuthGroupFile   /etc/httpd/conf/auth.group
require group   testgroup
```


Module 17

Key Configuration Files

Objectives

After completing this module, you should be able to configure the following:

- The password files `/etc/passwd`, `/etc/shadow`
- The group file `/etc/group`
- `cron` management `/etc/crontab`
- Kernel modules (`/etc/modules.conf`)
- Filesystem mounting (`/etc/fstab` and `/etc/exports`)
- System startup and shutdown scripts

17.1 `/etc/passwd`

- Stores information about users
  - Password (on some systems)
  - Id, and primary group
  - *Finger* information
  - Home directory
  - Default shell

17.2 /etc/passwd (continued)

- Colon-separated fields, e.g.

```
lee:Df18jed/nienysd:501:501:Lee Willis,Rm 1,013 567,013 765:/home/lee:/bin/bash
```

- First field is the username
- Second is the encrypted password *
- Third and fourth fields give the user ID and the primary group ID respectively
- *Finger* information is a comma separated list of information about a user
  - Typically stores full name, office room, office phone number and home phone number
- The sixth field is the user's home directory
- The user's default shell is given by the last field

*On systems which support shadow passwords this will just be an x, see 17.8 for an explanation

17.3 Editing /etc/passwd

- You should never edit `/etc/passwd` directly
  - Can lose information on multi-user systems
- Use the `passwd` command
- Normal users simply type `passwd`
  - Prompted for old password
  - Type new password twice (to avoid typos)
- Superuser can change anyone's password `passwd username`
  - Enters only the new password
  - Don't have to know the old password
- Superuser may also disable/enable accounts
  - `passwd -l username` disables or *locks* an account
  - `passwd -u username` *unlocks* the account

17.4 Other Changes To `/etc/passwd`

- `chfn` allows you to change the finger information for a user e.g.

```
$ chfn -f "Lee Willis" -o "Room 1" -p "01234 5678" -h "0123 45678"
```

- `chsh -s shell` lets you change your default shell
  - Must be listed in `/etc/shells`
  - `chsh --list-shells` will give a list of valid values
- Example:

```
$ chsh --list-shells
/bin/bash
/bin/sh
/bin/ash
/bin/bsh
/bin/tcsh
/bin/csh
$ chsh -s /bin/tcsh lee
Changing shell for lee.
Password:
Shell changed.
```

- *Note:* Both `chfn` and `chsh` require you to give your password

17.5 `/etc/group`

- Effective control of file access is one of the strengths of Linux/Unix
- One aspect of this is the concept of *groups*
- Users belong to one or more of these groups
- Access to files can be granted or denied on the basis of group priveleges
- Group membership is controlled by the file `/etc/group`

17.6 Editing `/etc/group`

- Like `/etc/passwd` shouldn't be edited directly
- Tools can change it and ensure locking
- To *create* a group with ID *gid* and name *gname*:

  `$ groupadd -g` *`gid gname`*

- To change name of group *gname* to *newname*:

  `$ groupmod -n` *`newname gname`*

- `usermod` changes the groups a user belongs to,

  e.g. to add the user `lee` to groups `www`, `project`, and `tempgroup`:

  ```
  $ usermod -G www,project,tempgroup lee
  ```

- N.B. It also removes him from any groups not listed (excluding his primary group)
- `usermod` can also change the information in `/etc/passwd`
  - Can only be run by the superuser

17.7 Important Note

- Changing user information shouldn't be undertaken lightly
- There are a number of restraints on changing usernames, IDs, and group IDs
- You can't change name while a user is logged in
- You can't change ID while user has processes running
- See `man usermod` and `man groupmod` for others
- Mostly common sense

17.8 Shadow Passwords

- *shadow passwords* are a security feature
  - Normal users could get others' passwords if encrypted versions were readable
  - Some information in `/etc/passwd` needs to be readable, but *Passwords* don't!
- Solution:
  - Keep everything except passwords in `/etc/passwd`
  - Password field contains just a single *'x'*
- Encrypted passwords are stored in `/etc/shadow`
  - Only readable by superuser

17.9 `/etc/shadow`

- `/etc/shadow` also stores other information
- Mainly password expiry information
- Can force users to change their password
- Most important benefit is increased security
- All modern systems should use shadow passwords

17.10 Scheduling Jobs (Cron)

- cron schedules jobs to run at times; specified in the file /etc/crontab

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily

5,35 * * * 1-5 root /usr/local/bin/domail
```

- The first section sets environment variables
- Cron jobs run when the current time/date matches a crontab entry
- The first 5 fields in /etc/crontab are

```
minute hour day_of_month month day_of_week
```

- * Matches all possible values
- Commas separate sets of values within a field
- Ranges can also be specified, e.g. [1-5]
- Can also specifiy steps, e.g. [0-59/5]

17.11 /etc/crontab

- /etc/crontab also specifies what *user* the job runs as, e.g.

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
```

- Second line runs the command run-parts /etc/cron.daily as root at 4:02am every day
- To run the command /usr/local/bin/domail as root *at* 10 minutes past *and* 40 minutes past the hour, *between* 9am and 6pm *on weekdays*:

```
10,40 9-18 * * 0-5 mail /usr/local/bin/domail
```

17.12 `run-parts`

- `run-parts` is a script designed for use with `cron`
- Runs all the programs in the specified directory
- Allows administrators to easily add jobs
  - Simply place an executable script/program in the correct directory
- N.B. *Not* a standard cron feature

17.13 logrotate

- Log rotation is normally handled by `logrotate`
- Run by cron, which reads `/etc/logrotate.conf` for configuration
- Example:

```
$ cat /etc/logrotate.conf
#Rotate the logs weekly
weekly
# keep 4 weeks worth of backlogs
rotate 4
# send errors to root
errors root
# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# no packages own lastlog or wtmp --- we'll rotate them here
/var/log/wtmp {
    monthly
    rotate 1
}
/var/log/lastlog {
    monthly
    rotate 1
}
```


17.14 Module Configuration

- The Linux kernel can be modular in nature
- Needs to know which devices use which drivers
- `/etc/modules.conf` contains this information *
- Typical file may look like:

```
alias eth0 ne2k-pci
alias eth1 3c509
```

- States that the device `eth0` requires the module `ne2k-pci`, and `eth1` requires `3c509`

*Warning, on some older systems this is `/etc/conf.modules`

17.15 Modules Configuration — ‘Options’

- Some modules allow you to specify options
- Mainly used for ISA peripherals, e.g. to provide I/O and IRQ information:

  ```
  alias eth0 ne
  options ne irq=10
  ```

  Specifies that `eth0` requires the module `ne` which should be passed the argument `irq=10`
- Can also specify actions to be executed when loading unloading modules, e.g.

  ```
  pre-install pcmcia_core /etc/rc.d/init.d/pcmcia start
  ```

  Run `/etc/rc.d/init.d/pcmcia start` before loading the `pcmcia_core` module

17.16 Mounting Filesystems

- Linux can store its files on multiple disks
- It decides what part of the filesystem each of these lives on using `/etc/fstab`

Logical Volume	Mount Point	FS type	Options	Dump	Check order
/dev/hda1	/	ext2	defaults	1	1
/dev/hda5	/home	ext2	defaults	1	2
/dev/hda7	/tmp	ext2	defaults	1	2
/dev/hda6	/usr	ext2	defaults	1	2
/dev/hda8	swap	swap	defaults	0	0
/dev/fd0	/mnt/floppy	ext2	noauto	0	0
/dev/cdrom	/mnt/cdrom	iso9660	noauto,ro	0	0
\\kashmir\c	/mnt/kashmir	smbfs	guest	0	0
landlord:/var/admin	/var/admin	nfs	defaults	0	0
landlord:/home/lee	/home/lee/LANDLORD	nfs	defaults	0	0

17.17 Runlevels

- Linux has several modes of operation
- Referred to as *runlevels*
- The Linux Standards Base (`http://www.linuxbase.org/spec/refspecs/LSB_1.1.0/gLSB/runlevels.html`) defines the following standard runlevels that all distributions must follow to be compliant:

0	halt
1	single user mode
2	multiuser with no network services exported
3	normal/full multiuser
4	reserved for local use, default is normal/full multiuser
5	multiuser with `xdm` or equivalent
6	reboot

17.18 Single User Mode

- Mainly used for diagnostic purposes
- Starts only a subset of the possible services, e.g.
  - No networking
  - No mail services
  - No name lookup services
    - Except `/etc/hosts`
  - No file-sharing services etc

17.19 Multi User Mode

- The ‘normal’ operating state
- All configured services are running
- Multiple users can log in
- `/sbin/runlevel` shows the previous and current runlevel of your machine

17.20 Starting up and Shutting down

- Only the *superuser* can shutdown or reboot
- `halt` will shut down the machine totally
  - For safety you should type `/sbin/halt`
- Makes sure all processes are stopped
- Stops services cleanly
- Writes unsaved data to the disk
  - 'Syncing'
- `/sbin/reboot` will shut down cleanly and reboot

17.21 Changing runlevel

- It is sometimes necessary to change runlevel
- Rare, but useful to know
- You can instruct a system to change runlevel using the `telinit` command
- Example:

```
$ telinit 5
```

- Changes to runlevel 5
- `telinit 1` takes the system down to single user mode

17.22 Initscripts

- The precise behaviour of each of the runlevels is controlled by *initscripts*
- Control which services run in each runlevel
- Live in `init.d`
  - On Debian it's in `/etc/init.d`
  - On Redhat it's `/etc/rc.d/init.d`
- Each file here is a script that can be called with an argument, `start`, `stop`, or `restart`

17.23 rcn.d

- The contents of the directories rc*n*.d control which services start and stop in runlevel *n*
- The directories hold symbolic links to the files in init.d
- The links are named informatively
- To start service *abc* you would create a link typically named Sxx*abc*, to init.d/*abc*
- The xx specifies the order to run the scripts, e.g. S00foo will be run before S90foo
- Links that stop a service are of the form Kxx*abc*

17.24 Initscripts — An example

- Consider the following: *

```
lee @ 12:22:08 /etc/rc.d/rc3.d ls -l S*

lrwxrwxrwx ... S01kerneld -> ../init.d/kerneld
lrwxrwxrwx ... S10network -> ../init.d/network
lrwxrwxrwx ... S15nfsfs -> ../init.d/nfsfs
lrwxrwxrwx ... S20random -> ../init.d/random
lrwxrwxrwx ... S30syslog -> ../init.d/syslog
lrwxrwxrwx ... S40atd -> ../init.d/atd
lrwxrwxrwx ... S40crond -> ../init.d/crond
lrwxrwxrwx ... S40portmap -> ../init.d/portmap
lrwxrwxrwx ... S40snmpd -> ../init.d/snmpd
lrwxrwxrwx ... S45pcmcia -> ../init.d/pcmcia
lrwxrwxrwx ... S50inet -> ../init.d/inee
lrwxrwxrwx ... S55named -> ../init.d/named
```

- We can see that the first thing started is `kerneld`, followed by `network` services, `nfs` services, etc
- There are also a series of `Kxxyyy` scripts which shut down the services in a sensible order

*Unimportant information has been removed from the screen dump so do not be alarmed if this doesn't look like you'd expect!

17.25 Restarting Services

- Can be necessary to restart a particular service, e.g. so it can re-read a modified configuration file
- This can be done without a complete reboot
- It must, however, be done by the superuser
- To restart samba (`smb`) we can do the following:

```
$ cd /etc/rc.d/init.d
$ ./smb restart
```


17.26 Exercises

1. *Passwords*
   (a) Find out whether your machine is using standard or shadow passwords?
2. *Users*
   (a) Add a new user (`useradd`) and set them up with the correct Full Name, password, home directory. Set their default shell to `csh`
3. *Groups*
   (a) Create a new group and add your user to this group
   (b) Now remove both the user and the group. How would you ensure that all files belonging to that user have been removed?
4. *Scheduling*
   (a) Add a cron job to eject your CDROM drive at 5 minutes past every hour and put it back in at ten minutes past the hour
5. *Mounting*
   (a) Set up your `fstab` so that

   ```
   $ mount /dev/cdrom
   ```

   will automatically mount your CD drive under `/mnt/cdrom`
6. *Runlevels*
   (a) Switch your machine between runlevels 3 and 5. What is happening? What happens if you change to runlevel 6?
   (b) Make sure your machine runs the same set of services in both runlevels
7. *Stop, Start and Restart Services*
   (a) Check you can stop, start, or restart services
   (b) Can you do this as a normal (ie non-root) user?

17.27 Solutions

1. *Passwords*

   (a) Your machine will have an `/etc/shadow` file if it is using shadow passwords. The password field will be set to an 'x' in `/etc/passwd`.

2. *Users*

   (a) The following would set the details for the user Lee Willis

   ```
   $ useradd leewillis
   $ passwd leewillis
   Changing password for user leewillis
   New UNIX password:
   Retype new UNIX password:
   passwd: all authentication tokens updated successfully
   $ chfn -f "Lee Willis" leewillis
   Changing finger information for leewillis.
   Finger information changed.
   $ chsh -s /bin/csh leewillis
   Changing shell for leewillis.
   Shell changed.
   ```

   The home directory should be set up properly (`/home/leewillis`), if not you can change it with

   ```
   $ usermod -d /home/leewillis leewillis
   ```

3. *Groups*

   (a)
   ```
   $ groupadd newgroup
   $ usermod -G newgroup leewillis
   ```

   (b) To remove the group, the user and the user's home directory

   ```
   $ groupdel newgroup
   $ userdel -r leewillis
   ```

   There are a few important points here! Firstly there may still be files in the filesystem belonging to that user. To locate them all you should have done

   ```
   find / -user leewillis -exec rm -f {} \;
   ```

   prior to removing the user. You should also have located all files belonging to the group and re-parented and/or removed them before removing the group

4. *Scheduling*

   (a) The following lines should acheive the desired effect

   ```
   05 * * * * root eject /dev/cdrom
   10 * * * * root eject -t /dev/cdrom
   ```

5. *Mounting*

   (a) The entry should look like

   ```
   /dev/cdrom       /mnt/cdrom       iso9660 noauto,ro     0 0
   ```

6. *Runlevels*

(a) You can change runlevels by using `telinit 5` and `telinit 3`. All non-relevant services are stopped and the new ones started each time you change runlevel. Runlevel 6 reboots the machine!

(b) You should ensure that the directory listings for `/etc/rc.d/rc3.5` and `/etc/rc.d/rc5.d` are the same. This should ensure that the same services are started/stopped when entering either runlevel.

7. *Start, Stop and Restart services*

(a) -

(b) -