Hand on II

Keystrokes

Ctrl-c

Stops a command or program that is currently being executed.

Ctrl-d

Removes you from the current environment, this will log you out of the system if you are at a shell prompt.

Ctrl-s

Temporarily halts the current command being executed, e.g. scrolling of text on the screen.

Ctrl-q

Resumes the command that was halted by **Ctrl-s**.

Ctrl-u,

Cancels what you have just typed, so that you can start again.

Ctrl-h, Del

Deletes the last character typed and moves back one space.

Directory Commands

cd

Stands for **change directory**, e.g.

```
cd /usr/home/(you)
```

takes you to your personal directory. The first **/** refers to **root**.

Typing **cd** without a path takes you back to your home directory, i.e. where you arrive when you first log on.
Typing **cd..** takes you up one directory,

df -k

Stands for **disk free**, gives you the amount of space available on the disk that you are currently on.

mkdir

Means 'make directory', e.g.

```
mkdir user
```

creates a directory called **user** in the directory you are in when you issue the command.

pwd

Stands for **print working directory** and prints the directory that you are in to the screen.

rmdir

Means **remove directory** e.g.

```
rmdir user
```

removes the directory **user** provided that it is empty!

System Commands

*

This wildcard character matches any number of characters and is useful in searches, e.g.

```
g*
```

matches all files beginning with 'g'.

>

Redirect output from a program to a file, e.g.

```
ls -l > listing
```

redirects the listing of **ls -l** into a file called 'listing'.

<

Redirect output from a file into a program, e.g.

```
mail user_account < listing
```

redirects the greeting letter called 'hello' to John, rather than you having to type it.

|

Pipe output from one program to another, e.g.

```
who | wc -l
```

gives a count of the users on the system.

>>

This **append** adds the input to an existing file without overwriting the original, e.g.

```
postcript >> letter
```

adds the contents of 'postscript' to an existing file called 'letter'.

chsh

Means **change shell** and changes the shell that the user is using. The user will be prompted for a password since the 'passwd' file is being changed, then the user will need to type the path to the shell e.g. **/bin/bash**.

echo

This 'echoes' arguments to the screen, e.g.

```
echo $SHELL
```

displays the value of the environment variable SHELL. This could return **/bin/tcsh** (Linux often uses this shell) or **/bin/bash**. **echo $PATH** displays the current path.

exit, Ctrl-d

Logs you out.

jobs

This lists the jobs running under the current shell in 'job ID' order. You can type **bg %jobid** to put a particular job running in the background. **Ctrl Z** also suspends a job. Typing **fg %jobid** brings the job back to the foreground.

kill

This **kills** a process e.g.

```
kill 5173
```

kill the process which has been given the temporary number 5173. This process number is found by using the **ps** command. Do not use **kill 1** as this kills the system scheduler! If a process refuses to die you can type **kill -KILL [PID]** to stop a process immediately without any tidying up on exitting. Finally, **kill -HUP [PID]** tells the process that an event has occurred, or a configuration file change has occurred and needs to br reread.

man

The **manual** command is very useful for finding out comprehensive information on an individual command e.g.

```
man cd
```

gives all the information on the command **cd**. Typing **man -k mail** lists the Unix commands that relate to the word **mail**.

passwd

Allows you or the administrator to change passwords.

printenv

The 'prints the environment' variables to the screen.

ps

The **process status** command shows the programs currently running. **ps -a** shows all the processes being run by all users. An example is the following:

```
ps -ef | grep erpcd
```

where '-ef' gets the process number and pipes it to **grep** which filters on the following word, in this case for the program 'erpcd'.

The following information is shown:

- **PID** Process ID.
- **TTY** Each shell opened has a 'character special' called a 'tty' (held in '/dev').
- **STAT** State, either 'S', sleeping, or 'R', running.
- **TIME** CPU time that the process is taking up.
- **COMMAND** The command running.

Typing **ps x** shows all the processes relating to X windows, whereas **ps ax** shows all the processes being run by everybody. Typing **ps ux** gives even more information such as the user.

setenv

The command **set environment variable**, sets aside a small amount of memory to hold paths etc. e.g.

```
setenv GUI /usr/utility/gui_r4
```

sets a variable 'GUI' with the path that follows to the actual program. This program can now be run by typing 'GUI'.

```
setenv DISPLAY :0.0
```

sets an X window session locally.

These settings are commonly setup permanently in the user's **.profile** (located in the '/etc' directory). This can be edited with any text editor.

The following are common environment variables:

- **SHELL** The current shell.
- **HOME** The current user's home directory.
- **HOSTNAME** The name of the computer.
- **DISPLAY** The X display that the applications are to use.
- **LD_LIBRARY_PATH** The search path for libraries.
- **PATH** The search path for applications.

If you wish to append directories to the path then type **setenv PATH ${PATH} : /search/here**. In order to use it then you need to cache the new path by typing **rehash.**

The DISPLAY variable is made up of three parts 'hostname : displaynumber : screennumber'. The hostname is the computer, whilst the other variables are '0' unless several machines are connected. X windows looks to this variable to find out where to send the X Windows traffic.

set path

Sets a path where regularly used programs or data are found e.g.

```
set path=($path /usr/utility/gui_r4/bin)
```

sets the path '/usr/utility/gui_r4/bin'.

Some commands used to set the environment come from the C shell. In order to check which shell you are running type **echo $SHELL**, if this does *not* return '/sbin/csh' then you type **/bin/csh**.

su

The command **switch user** switches the login user to another user, e.g.

```
su root
```

switches to the 'root' login.

top

Gives a constantly updating view of the top 20 processes (a real time version of 'ps'), i.e. those that are using the CPU the most.

who

This displays the users currently logged on the system.

whoami

Displays who you are currently logged on as. (e.g. 'root', a user etc.)

xhost +

Opens an X window for a program to run in. After issuing this you would then run the program (e.g. Netscape).

File Commands

.

The **dot** is not a command as such. If a file is spelled with a dot at the beginning, Unix treats it as a hidden file. Configuration files are often preceded with a dot.

cat

The **concatenate** command displays a file, e.g.

```
cat bankletter
```

displays the contents of 'bankletter' on the screen.

```
cat > newletter
```

takes whatever you type and redirects it into the file 'newletter', **Ctrl-d** gets you out of it.

```
cat >> existing
```

takes whatever you type and appends it to an existing file called 'existing' funnily enough.

chmod

The command **change mode** changes the mode or permissions, of a file or a directory. When you do an **ls -l** you will see in the first column, a line of 10 characters looking something like 'drwx-w-rw-'. The 'd' means 'directory' (you could have '-' for file, 'l' for link to a file, 'b' for a 'block special', 'c' for a 'character special', 'p' for a 'named pipe', or 's' for 'socket'). The next three characters refer to the permissions of the login user, in this case the user has read, write and execute access to the directory. The next three characters refer to the permissions of the group and the final three characters refer to the permissions of all users. The **chmod** command can be used in various ways as shown by the following examples:

- **chmod go-rwx newletter** removes read, write and execute permissions for users in the group (g), and all other users, for the file 'newletter'. Using a '+' instead of '-' adds the permissions. You can also use 'o' for others, or 'u' for user.
- **chmod 766 newletter** causes the file 'newletter' to have read, write and execute permissions for the user, read and write permissions for the group members and read and write permissions for all other users. Why? Well, the 7 represents 111(binary) and 6 represents 110(binary) for each set of three 'rwx's. 'r' being set is given binary 1, 'x' being not set is given binary 0. Read permission is '4', write permission is '2', execute permission is '1' and no permissions is given with '0'.
- **chmod 700** ***dirname*** results in **drwx------** for the directory which restricts access to everyone

bar the owner.

- **chmod 664 *filename*** gives **-rw-rw-r--** that allows you and your group to read and edit the file but all others can only read the file.
- **chmod 600 *filename*** gives **-rwx------** creates a private file that only you can see and edit.

You can change permissions for groups of files with one command by using wildcards such as *****.

chown

Use this to **change ownership** of a file e.g.

```
chown dmuthoni myfile
```

changes the ownership of the file 'myfile' to dave. This can only be carried out by the owner of the original file. A way around this is for the recipient to copy the file, then the copied file becomes their own.

chgrp

Use this to **change group ownership** of a file.

compress

This compresses a file e.g.

```
compress myfile
```

results in a file called 'myfile.Z'. The command **uncompress** can be used to uncompress the file.

cp

The **copy** command copies files from one directory to another, or to the same directory with a different name, e.g.

```
cp bankletter /home/dmuthoni/bankletter1
```

copies the file 'bankletter' from the directory that you are currently in, to the '/user/dave' directory with a new name 'bankletter1'.

file

This returns information on the content of a file, e.g.

```
file myletter
```

might return 'ASCII' to say that Unix guesses that 'myletter' contains ASCII.

find

This **finds** a file or directory, e.g.

```
find / -name apache -print &
```

this finds a file with name 'na' starting the search from the 'root' and printing the result to the shell window, whilst still allowing you to carry on using it.

grep

This stands for **global regular expression and print** and is a search utility, e.g.

```
grep "325 Victory"
```

searches the current directory for files containing the text '325 Victory'.

gzip

GNU zip compresses files to create a '**.gz' file.

head

This command followed by a filename, displays the first ten lines of that file.

less

This is a way of displaying a file, it will give a percentage of file so far displayed at the bottom of the screen, and you can progress through reading the file by pressing the space bar.

ln

The command **link**, links files and directories, e.g.

```
ln -s/export/home/fred usr/fred
```

creates a copy of 'fred' in the '/export/home/' directory in the 'usr/fred' directory. A 'hard link' is like a Windows 'shortcut', there can be a number of them, with different names and they take up little space. A 'soft link' is identified with the '-s' switch and creates a copy of the file elsewhere.

lp (for System V) or lpr (for BSD)

The command **line printer**, prints a file, e.g.

```
lp newletter
```

prints 'newletter'.

lpstat -a all

The **line printer stats** command checks the printer queue in System V Unix.

ls

This **lists** the contents of the current directory, e.g.

```
ls /etc
```

lists the files and sub-directories of the current directory.

- **ls -l** gives a long list of directories including file sizes, permissions, type etc. Using the **-a** switch causes 'all' files to be listed including those hidden files starting with **.**.
- **ls -c** lists files by creation time.
- **ls -p** marks directories with a slash at the end of the name.
- **ls -x** displays the list in rows across the screen.
- Using **ls | more** is useful for large directories as it stops the screen scrolling, you press the 'Return' key to advance one line at a time, or press the space bar to advance one page at a time.

You can type 'ls' and then define one or more directories for it to list.

more

This is another way of displaying a file, it will give a percentage of file so far displayed at the bottom of the screen, and you can progress through reading the file by pressing the space bar. Whilst in **more**, if you type **v** you will be taken straight to the **vi** editor.

mv

This **moves** a file from one directory to another or renames it in the same directory, e.g.

```
mv bankletter bankletter1
```

renames 'bankletter' to 'bankletter1'.

rm

'remove' a file, e.g.

```
rm oldletter
```

removes the file 'oldletter'.

************ Using **rm -rf** recursively removes all files and directories below the one that you are in. Using **rm -i** gives you the option of cancelling or confirming the command.

sort

Sorts the contents of a file, e.g.:

```
sort -o outfile infile
```

The contents of 'infile' are sorted in alphabetical order and fed into a new file called 'outfile', as defined by the switch '-o'.

tail

This means the **tail end**, this dynamically displays the file that is being written to in real time, e.g.

```
tail -f /var/log/dmesg
```

shows the file 'dmesg' which is being written to.

tar

The command **tape archive** is an file archiving command. It creates a single uncompressed archive file from several, ideal for sending data over networks. Often files are archived, and then compressed using 'gzip'. E.g.

```
tar -tvf tarfile
```

displays the contents of a tarfile.

```
tar -xvf tarfile
```

extracts the contents of a tarfile. 'x' is extract, 'v' means 'verbose' and 'f' means the file.

```
tar -xvf tarfile target
```

extracts the file target from the tarfile.

touch

This just creates an empty file for appending to later on e.g.

```
touch log
```

creates an empty file called 'log' that needs to be available for another program to write to it perhaps.

uncompress

This command **uncompresses** a 'gzip' file, e.g.

```
uncompress myfile.gz
```

uncompresses the file 'myfile.gz'.

wc

The command **word count** counts the words in a particular file, e.g.

```
wc letter
```

counts the number of words in the file 'letter'

Networking Commands

arp

Displays the 'Address Resolution Protocol' table e.g.

```
arp -a
```

displays all arp entries for all connected devices.

```
arp -d <ip address>
```

deletes the arp entry for that particular IP address.

ftp

The command **file transfer protocol** attaches you to another IP device e.g.

```
 ftp 41.204.160.12
```

attaches you to the device with address 141.205.15.154. You are normally presented with a login and password screen.

Commands that are used in FTP are:

- **dir** - directory listing.
- **quit** - quit from ftp.
- **cd** - change directory.
- **get** or **mget** - get a file (or multiple files).
- **put** or **mput** - put a file (or multiple files).
- **bin** - sets up your system to receive binary files.
- **hash** - displays hashes whilst files are being transferred.
- **lcd** - local change directory changes the directory on your local machine to which you are sending and receiving files. This is useful as it saves you having to quit ftp to carry out the directory change.

The **Hosts** file can be found in the directory '/etc'.

netstat

This stands for **network statistics**, e.g.

```
netstat -r
```

displays the routing table of the Unix box.

```
netstat -a
```

displays alll network information.

Unix uses **routed** to listen to RIP in order to discover the Default Gateway.

ping

Ping an IP device e.g.

```
Ping 196.200.223.254
```

telnet

Ctrl-6 and then **Ctrl-]** gets you to the **telnet>** prompt where typing **close** gets you out of telnet.

ifconfig

This displays the IP configuration of the box, e.g.

```
ifconfig -a
```

displays all IP configuration.

If you want to look at the routing process you can type:

```
ps auxw | grep route
```