


Exercise 1 Getting Help

Unix provides man pages (manual pages) for practically all installed software. The man page should always be the first point of reference in determining how a program works, and for working out why it doesn't work. In addition, man pages exist for a lot of the common configuration files as well.

Q 1. Browse the man page for `man`. Read about the `-k` option in particular

```
vmuser@Ubuntu-2013:~$ man man
```

Q 2. Using `man -k <keyword>` determine what command to use to do the following:

- a) print the current working directory
- b) make directories
- c) list the directory contents 
- d) move or rename a file
- e) copy a file or a directory
- f) remove a file (delete)
- g) remove a directory which is empty
- h) search for files within a directory tree
- i) display lines with a file/files containing a particular pattern

Exercise 2 Getting Around the File System

Q 1. Where am I? On some Unix systems the default prompt will not show the current working directory, or it maybe displayed in a cryptic manner. The `pwd` command returns the Present Working Directory:

```
vmuser@Ubuntu-2013:~$ pwd
```

```
/home/vmuser
```

Q 2. Listing Files. The `ls` command is useful for getting a list of files in the current directory, or any specified directory. Look at the man page for `ls` now:

```
vmuser@Ubuntu-2013:~$ man ls
```

After reading the man page, use `ls` to perform the following tasks using only one command, without piping:

- a) List all files, including hidden files, showing all details
- b) List file sizes in "human readable" format
- c) List all files, sizes in human readable format, sorted by file size, *From smallest to biggest*
- d) From your home directory, list all files in `/etc/` from oldest to most recent, sorted by *modification time*

Q 3. Working with Directories. Making, changing, and deleting directories are common tasks that must be performed. Examine the man pages for `mkdir` and `rmdir`. There is no specific command to rename a directory or file, however this behaviour can be achieved by moving the directory from one name to another. The `mv` utility allows for files and directories to be moved.

Extra Technical Information The change directory command `cd` does not have a man page as it is built into the command shell (`bash`). Information on `cd` is available in the `bash` man page. Note, most of what is in the `bash` man page is not examinable in this unit.

After reading the man pages, do the following tasks:

- a) Create a sub directory in your home directory called **weekone**
- b) Change into that sub directory
- c) Verify that you are in the **weekone** directory by displaying the current working directory
- d) Verify that the directory is empty using `ls`
- e) Using the `..` short cut, change to the parent directory
- f) Rename **weekone** to **notweekone**
- g) Remove the **notweekone** directory
- h) Verify that the **notweekone** directory has been deleted

Exercise 3 Dealing With Files

Q 1. Viewing Files. It is a common task to view the contents of a plain text file. A number of programs are capable of doing this:

- **cat** short for concatenate
- **more** a program to display a page of text at a time
- **less** a program to display a page of text at a time, with more features than **more**
- **head** displays a number of lines from the start (head) of a file
- **tail** displays a number of lines from the end (tail) of a file

Look at the man page for these five tools. They are all useful for specific tasks, but most of the time you will want to use **less**.

For example, to view the `/etc/passwd` file you would do:

```
vmuser@Ubuntu-2013:~$ less /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh

contd...
```

After reading the man pages try and answer these questions:

- a) Name two differences between **more** and **less**?
- b) How would you display the first 5, and only 5, lines of a file?
- c) How would you display the last 100 lines of a file, and any following lines that get appended?

Q 2. Deleting Files. To delete files in Linux, use the **rm** command. Look at the man page for **rm** then:

- a) Create a file using **vim** in your home directory
- b) Use **ls** to verify the file was created
- c) Delete the file using **rm**
- d) Again, using **ls** verify the file was deleted

Q 3. Copying and Moving Files. To copy or move files in Linux, use the `cp` and `mv` commands. Browse the man pages for these commands and then create a file, copy it and then move it.

Q 4. Text Searching. Searching for text within a file is another common task that must be performed. A number of tools exist for text manipulation and searching which will be used in later weeks. However, for simple searching within a group of files, the `grep` utility is sufficient.

Begin by browsing the man page for `grep`. At this time it is not necessary to know about extended expressions or regular expressions.

After browsing the man page, do the following:

- a) Display all lines that contain the word *db* in the file `/etc/nsswitch.conf`
- b) Find lines containing *monthly* in `/etc/logrotate.conf`, making sure to print 2 lines before and 2 lines after the matching line
- c) Print the number of lines containing *false* in `/etc/passwd`

Exercise 4 Input/Output Redirection

Reference: <http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html>

The previous exercises aimed to introduce a number of Unix command line tools and show off some of the functionality offered. However, the true power of the command line is visible when multiple commands are tied together using input/output redirection. Redirection is possible because most Unix command line tools (by default) read input from Standard Input (stdin) and direct the output to Standard Output (stdout). A second output source called Standard Error (stderr) also exists.

There are two types of redirection, redirection to a file or device (which is just a file), or redirection to another application. This is made possible by one of these operators:

- > Write stdout to a file. Overwriting the file if it exists, creating the file if it doesn't exist
- >> Concatenate (append) stdout to an existing file. If file does not exist, create a new file
- 2> Write output of stderr to a file. Overwriting the file if it exists, creating the file if it doesn't exist
- | Pipe the stdout of one command to stdin of another. This is one of the most useful redirection operators

Additional examples are shown at the above URL.

- Q 1.** a) Use redirection to save a copy of the directory listing of your home directory to a file named `homedirlist`
- b) Use piping along with `cat` and `less` to concatenate `/etc/passwd` and `/etc/group` together and display the output one page at a time. *Caution. Do not do this as root in case you overwrite the system files by accident*

Exercise 5 Combining it All Together

The power of the Unix command line is realised by combining commands together. This is called piping. Piping allows you to redirect the output of one command to the input of another. So instead of displaying to the screen, it becomes the input for the second command.

To pipe the output of one application to the input of another we use the pipe operator (`|`). For example, to pipe the output of `tail -n 100` (which is bigger than the screen) into `less` we would do:

```
vmuser@Ubuntu-2013:~$ tail -n 100 /var/log/messages | less
```

- Q 1.** For this exercise we want to get a list of all the files in `/usr/lib` that have the word *lib* somewhere in their name. However, there are many files that contain *lib* so we want to display one page of text at a time.

The three commands we will use are: `ls`, `grep` and `less`.

Use your knowledge of `ls`, `grep` and `less` along with piping to produce the desired result.

Note: Unix provides a much more powerful file searching utility, `find`. `find` will be introduced in a later week.

Exercise 6 Editing Files

The most common command line based text editor in Linux is `vi` or its improved cousin, `vim`. There are alternatives to `vi` such as `nano` (which is installed in the S block level 5 environment) or `emacs`. While `vi` can be difficult to use initially it is a very powerful text

editor and is likely to be installed on any Unix like operating system, whereas some of the other alternatives may not.

Exercise 7 Symbolic Links (symlinks)

Most Unix file systems allow the creation of file 'links'. These are commonly referred to as *symlinks* even though file links can be either 'hard' or 'symbolic'. In most cases a symbolic link is the type of link you want to use, only use hard links when necessary.

A symlink allows you to create one or more pointers to a single real file or directory. When an application or process tries to access a symlink, the filesystem automatically follows the link to the real file and proceeds as normal. The utility of symlinks will become more apparent as you work through the semester.

Try the following simple exercise to demonstrate how symlinks work:

- Q 1. Take a look at the man page for `ln`
- Q 2. Using a text editor such as `vi`, create a file called `realfile.txt` in your home directory containing any piece of text you like. Exit from `vi` ensuring that you save the file.
- Q 3. Use the `ls` command to verify the file was created.
- Q 4. Using the `ln` tool, create a symlink to `realfile.txt` called `linkfile.txt`
- Q 5. Use the `ls` command to verify the symlink was created.
- Q 6. Use the `cat` or `less` command to view the contents of `linkfile.txt`
- Q 7. The contents should be the same as what you saved in `realfile.txt`

Exercise 8 Command Line Time Savers

As the Unix command line has been around for 30 years, there has been ample time to improve the user interface and the user experience. Two of the most common and useful features are Tab Completion and the Bash History.

Part I — Tab Completion

If you need to type in a particular file name or path, and that filename or path exists, you can usually partially type some of the name, then press the *tab* key to autocomplete the name. For example:

1. The command that you would use to configure a network interface is `ifconfig`
2. At a command prompt try typing `ifc` and then press *tab*

Now, suppose you wish to view the contents of the file `/etc/udev/rules.d/70-persistent-net.rules` using the `more` command. Normally you would use the command:

```
vmuser@Ubuntu-2013:~$ more /etc/udev/rules.d/70-persistent-net.rules
```

However, you can use tab completion to save some typing. Press *tab* at the end of the end of each of the following lines (but do not press *enter* until the end):

1. `mor`
2. `/e`
3. `ud`
4. `ru`
5. `70`

Notice that the generated command line is missing `net.rules` from the end, this is because more than one file matches the search string of `70`. Press *tab* twice and bash will reveal all of the files that match. Finish the command line by typing `n` and then pressing *tab*.

Part II — Bash History

`bash` automatically keeps a record of every single command you type, internally, this history is stored in the file `~/.bash_history` (`~/` is a short way of referring to the current users home directory). From the command line you can scroll through the history by using the *up* and *down* arrows.

1. By now you should have typed a number of commands and therefore, have some commands stored in the history. View the contents of the bash history file.
2. Use the arrows to cycle through the history

Being able to use the arrows to go back to recent commands is very useful, consider the situation if you made a mistake typing the command `cd tiles` instead of `cd files`. Rather than retyping the entire command, you could just press the *up* arrow once, then correct the mistake.

However, as the history grows in size, sometimes it is easier to be able to search the command line history. You can tell **bash** to search this history by pressing *Control + r*

1. Generate some command history by doing each of these commands (dont worry about the actual commands or output for now). Remember, use tab completion to save your fingers!
 - (a) `dmesg`
 - (b) `ifconfig`
 - (c) `tail /var/log/messages`
 - (d) `head /etc/passwd`
 - (e) `tail /etc/group`
 - (f) `ls /etc`
 - (g) `cat /proc/cpuinfo`
2. Now we want to re-run the `tail /var/log/messages` command without retyping the command, and without scrolling up using the arrow keys
3. Press the *Control* and *r* keys, notice that the bash prompt has changed
4. Type `/var`
5. You should notice the command appear, if it doesnt, type some more of the command
6. When the command you want fully appears, press *enter*

Exercise 9 Homework

Learn about and experiment more with `vim`. A good handy cheat sheet is available here:

<http://www.fprintf.net/vimCheatSheet.html>