Module 2

A Linux Session

1. Log in to your Linux system using either a graphical user interface or `ssh`. If using a graphical login, open the *Terminal* application to bring up a shell (command prompt).
2. What level of user are you – normal or root? (Check the prompt, or use the `id` command.) If you are logged in as root, create a normal username for yourself[1]. Log out and then log in using that.
3. Change your password. Don't forget it!
4. A home directory has been created for you. What is it called? Use the `pwd` command to find out.
5. Use the `stty -a` command to see which special characters the terminal understands.
6. Make sure that the following terminal characters work to modify your command-line:
    1. `BACKSPACE` to delete the last character
    2. `^W` to delete the last word
    3. `^U` to delete the entire line
    4. `^C` to interrupt a command

Getting Help

Use the `man` command to get help on these commands:

- pwd
- ls
- man

To see the next page, hit `SPACE`. To get help when viewing the manual, type `h`. To quit the manual, type `q`.

Use the `whatis` command to get a simple summary of these commands:

- id
- stty
- whatis

Use the GNU **info** command to learn about the bash shell.

Exiting the Shell

Exit the shell you are in by typing `^D` (at the beginning of a line). Check that either the Terminal window disappears, or that your *ssh* session ends.

Log Out

If logged into a GUI, log out of the system.

1 To add a user "Joe Dunne", as root use the command "`useradd -d /home/joe -c "Joe Dunne" -m joe`" and then change his password using "`passwd joe`".

peter@petermunro.org 087 689 1951

Module 3 – Getting Started

1. Display the message '`hello world`' using the `echo` command.
2. Test again using the '`echo -n`' option. When might this be useful?
3. Display the current date and time.
4. List who is on the system.
5. Display the file `/etc/services` using `cat`.
6. Use `cd` to change to `/usr`. Verify where you are with `pwd`. Now return to your home directory. Check that you have done so.
7. List the contents of the root directory ('`/`') without changing to it first.
8. Now `cd` to the root directory, and list its contents.
9. Display a long listing of the root directory.
10. Now return to your home directory, and again display a long listing of the root directory from there.

peter@petermunro.org 087 689 1951

Module 4 - The Filesystem

Absolute or Relative?

Are these pathnames absolute or relative?

1. `file1`
2. `dir1/file1`
3. `/home/joe/dir1`
4. `/home/joe/dir1/file1`
5. `./file1`
6. `../dir1/file1`
7. `/etc/passwd`
8. `../../etc/passwd`
9. `.`
10. `..`
11. `../../..`

If your current directory is `/home/joe`, which of the following would refer to `file1` within a directory called `dir1`?

1. `dir1/file1`
2. `/home/joe/dir1/file1`
3. `./dir1/file1`
4. `../joe/dir1/file1`
5. `../../home/joe/dir1/file1`
6. `/home/joe/./dir1/./file1`
7. `/usr/../home/joe/dir1/file1`
8. `../../home/../home/joe/../joe/dir1/file1`

Disk Free Space

Find out how many 'real' disks underpin the Linux filesystem on your machine. Ignore any 'virtual' filesystems (those with simple names like `tmpfs` or `udev`) and just count the ones with real device names (like `/dev/sda1` for example).

How much disk space is available on these disks?

peter@petermunro.org 087 689 1951

Module 5 - Shell Metacharacters

What do these patterns match?

1. `*`
2. `?`
3. `??`
4. `??*`
5. `[abcdefg]`
6. `[a-zA-Z0-9]`
7. `[^a-zA-Z0-9]`
8. `*[0-9]`
9. `[09-]`
10. `?[0-9]`
11. `*.[0-9]`

How would you match filenames...

1. that are three characters long? Test this by listing all files in `/etc` with three characters (use "ls -d ..").
2. that contain only three letters? Test this again for `/etc`.
3. that start with a letter?
4. that *don't* start with a lowercase letter?
5. whose second character is a letter?
6. that start with an 'f' and end with a number?
7. that contain the word 'photo-' followed by three or more digits, followed by ".jpg"? (eg photo-001.jpg, photo-002.jpg etc)
8. that have 8 characters, a dot, then 3 characters?

peter@petermunro.org 087 689 1951

Module 6 - Working with Files and Directories

Copy Files

1. Copy the file `/etc/passwd` to your home directory.
2. Change directory to your home directory (if you are not there already).
   Create a subdirectory called `copies`, and another called `empty`.
   Using one command, copy the files `/etc/hosts`, `/etc/services` and `/etc/protocols` into your subdirectory `copies`.

Move/Rename Files

1. Move the `passwd` file in your home directory into the subdirectory `copies`.
2. Create a new subdirectory `copies2`.
   Using one command, move all the files in `copies` into `copies2`.
3. Rename the directory `copies2` to `morecopies`.
4. Move the directory `morecopies` into the directory `copies`.

Remove Files

1. Remove the file `passwd` within the `morecopies` directory.
2. Performing an interactive remove on the `morecopies` directory, remove just the `protocols` file.
3. Remove the directory called `empty`.
4. Remove the entire directory tree `copies`.

peter@petermunro.org 087 689 1951

Module 7 - Viewing Files

cat

Use `cat` to display the file `/etc/mime.types`. Evidently the `cat` command is reading from the file `mime.types` and displaying the contents to the screen.

What happens if you just type '`cat`' on its own? Where does the `cat` command read from now? How do you stop it?

more & less

Now use `more` to display the file `/etc/mime.types`. Display the help page ('`h`'). Try moving down (1) a line-at-a-time, (2) a page-at-a-time, and (3) back a page. Finally, (4) quit `more`.

Do the same again, but this time use `less`.

file

Run the file command on the files in `/etc`. What two types of files are the most common in the `/dev` directory?

wc

Use word count to find out how many lines, words and characters are in the file `/etc/services`. Now do the same for the file `/etc/hosts`. What changes when you run '`wc`' on both of these in the same command?

How would you count how many users are registered on this Linux system?

peter@petermunro.org 087 689 1951

Module 9 - More on the Filesystem

Hard Links

1. Create a directory `mydir`. Copy the file `/etc/motd` (message of the day) into mydir. In the same directory, create a link to this file, called `motd1`.
   View it with `cat` – are the contents the same? Check both with '`ls -l`'. Are their attributes the same? What about the link count?
2. Create a new link, to the same file, in the parent directory. What happens to the link count now? What happens when you remove one of the links?
3. Try to create a link to the directory itself. What happens?
4. Try to link to a file on another device (check with '`df -h`'), for example `/run/sshd.pid`. What happens?

Symbolic Links

1. Create symbolic links to both of these (ie one to the directory, and another to the file on a different device). List them with '`ls -l`'. What do you see? What is the 'length' of each symbolic link?
2. Try to create a symbolic link to a file that doesn't exist? Does it work?

peter@petermunro.org 087 689 1951

Module 10 - Knowing How the Command Line is Processed

Identifying Shell Special Characters

Which characters are special to the shell in these command lines?

1. `echo System upgrade starting...`
2. `ls -l`
3. `echo *`
4. `ls file[0-9]`
5. `date ; who`
6. `cp "file containing spaces" newfile`

Quoting Special Characters in the Shell

1. Create files with the following names (use '`touch`'):
    1. *
    2. a filename containing spaces
    3. "quoted"
2. Display the following messages using `echo` – try them each first *without* quotes:
    1. This is a *
    2. I have $400
    3. This is a quote mark: "
    4. This line contains many spaces
    5. This is one message
       on two lines

peter@petermunro.org 087 689 1951

Module 11 - Shell Variables

Creating Shell Variables

1. Create a shell variable called `favecolour`, setting it to your favourite colour.
2. Display the value of this variable: "My favourite colour is XXX"
3. Now create a new sub-shell, by typing 'bash'. This creates a new shell as a separate process.
4. Try to display the value of your variable in the new shell. What happens?
5. Return to your original shell by exiting the sub-shell.
6. Now make your variable available as an environment variable. Create another new sub-shell. Test to see that your environment variable is now available to this new shell.
7. If you change the variable in the sub-shell, what happens if you try to display it in the original (parent) shell?

Using Variables

1. Use 'echo' to display the value of these variables:
    1. HOME
    2. HOSTNAME
    3. PATH
2. Use the 'set' command to list all variables defined in this shell.
3. Use the 'export' command to list all environment variables known to this shell.

peter@petermunro.org 087 689 1951

Module 12 - Setting Up Your Shell Environment

Modifying .bashrc

Modify your `.bashrc` to display a message whenever a new sub-shell is created. Test that this works.

Modifying .bash_profile

To set up environment variables so that they're available when we log in, we put them in `.bash_profile`. This gets executed whenever we login, but not when a sub-shell is created.

Modify your `.bash_profile` to set an environment variable, and check that this gets set when you login, and inherited by sub-shells.

peter@petermunro.org 087 689 1951

Module 13 - I/O

Redirecting Output

1. Use the *cat* command to list the contents of the file `/etc/passwd`.
2. Now use redirection to *copy* the `/etc/passwd` file to your current directory.
3. Run '`cat`' without any arguments. What happens? How do you terminate input (ie tell '`cat`' that end-of-file has been reached)?
4. Now use *cat* to create a file with contents that you type in from the keyboard. Terminate the input (end-of-file). Now switch on shell echoing (`set -x`). Run the command again. What does the '`cat`' command see as its arguments? [To switch echoing off, use '`set +x`'.]

Redirecting Standard Error (stderr)

1. Run the command: '`wc -l /etc/*`'. Notice that this produces many errors. These can be thrown away by redirecting the error channel into `/dev/null`. Do this and check that it works.
2. This time, direct the errors into a file. Check that the file contains the errors.
3. Now redirect both the output and the errors, but to different files.
4. Finally, redirect both the output and the errors into the same file, making sure that one does not overwrite the other.

peter@petermunro.org 087 689 1951

Module 14 - Pipes & Filters

Using *grep*

1. Use *grep* to find:
    1. All users who use the `/bin/bash` shell. (Hint: use `/etc/passwd`)
    2. All processes ('`ps -ef`') owned by user `root`.
    3. A summary of all IP addresses in use on this system. (Use `ifconfig -a`)
    4. A summary of all CPU model names available on this system. (Use the file `/proc/cpuinfo`)

Using *cut*

1. Use `cut` on `/etc/passwd` to list all usernames registered on this system.
2. Next, use `cut` on `/etc/passwd` to list all the home directories instead.
3. Now use `cut` to list each username, together with their home directory.

Using *sort*

1. Pipe the output of `ls -l` (run on a directory with some files in it, such as `/etc` or `/usr`) through *sort*. What is the output sorted by?
2. Modify your command to *sort* by *link count* (column 2).
3. Now modify your command to sort files by size. Which are displayed first, largest or smallest? How would you reverse the sort order? Try it.
4. Which option to *sort* allows you to ignore case?

peter@petermunro.org 087 689 1951

Building Commands using Pipelines

Create command pipelines to perform the following:

1. List the number of users registered on this system (counting system and application users).
2. List the number of users currently logged in.
3. List the total number of processes (‘`ps -ef`’) at the time the command is run.
4. List all processes, listing only two fields: process ID, and the process' name.
5. List all files in /usr/lib, sorted by size, largest first.
6. List the 5 largest files in /usr/lib, largest first.
7. Produce summary disk information: list devices, percentages in use and the mount points to produce the following (use `df -P`):

```
tmpfs 1% /dev/shm
/dev/sda1 70% /boot
/dev/sda6 59% /home
/dev/sda7 88% /opt
```

8. Replace the spaces in your output above by colons (:).

Extra Challenges

1. Count how many processes are owned by each user. It should produce output similar to this:

```
    104 root
     72 peter
      2 avahi
      2 68
      1 smmsp
        ...
```

2. Find the 10 most popular words in the file ulysses.txt. This should produce output something like this:

```
  13683 the
   8163 of
   6693 and
   5896 a
   4874 to
        ...
```


peter@petermunro.org 087 689 1951

Module 15 - Regular Expressions

Regular Expressions

Create a text file containing the following words, one per line (or try http://files.petermunro.org/somewords):

```
barber
barbershop
barbed wire
barbarella
barker
barking
barn
carton
cartoon
door mat
Existence
```

1. Use `grep` to display only lines starting with the letter 'b'.
2. Use `grep` to display only lines starting with '`barb`'.
3. Use `grep` to search for all lines starting with either 'b' or 'c'.
4. Use `grep` to display only lines starting with '`bark`' or '`barn`' only.
5. Use `grep` to search for all lines ending with 'on'.
6. Use `grep` to search for all 6-letter words.
7. Use `grep` to search for all lines containing more than one word .
8. Use `grep` to find all lines containing both 'ar' and 'n' in that order.
9. Use `grep` to search for all lines starting with 'bar' and ending in a vowel.
10. Use `grep` to search for all lines starting with 'bar' but *not* ending in a vowel.
11. Use `grep` to search for all lines starting with a lowercase letter.
12. Use `grep` to search for all lines *not* starting with a lowercase letter.

Solutions: Regular Expressions

```
1. grep '^b' words
2. grep '^barb' words
3. grep '^[bc]' words
4. grep '^bar[kn]' words
5. grep 'on$' words
6. grep '^......$' words
7. grep ' ' words
8. grep 'ar.*n' words
9. grep '^bar.*[aeiou]$' words
10. grep '^bar.*[^aeiou]$' words
11. grep '^[a-z]' words
12. grep '^[^a-z]' words
```


peter@petermunro.org 087 689 1951

More Regular Expression Work

Using Regular Expressions in Applications

1. **Man Pages** View the *bash* manual (`man bash`). To search, hit forward slash. The cursor moves to the bottom of the screen awaiting your search expression.
   Now search for the next heading, or in other words, enter a regular expression that searches for all lines starting with a capital letter. Hit RETURN to search.
   Having found an occurrence, to go to the **n**ext one, hit '**n**'. And again. And again. Notice that the pager (in this case *less*) places each occurrence at the top of the screen.
   To go back to the previous occurrence, hit '**N**' (SHIFT-n).

2. **The vi Editor** Open *vi* on the *ulysses.txt* file. To search, again hit '**/**'. To repeat forward and backward searches, the commands are the same as before ('**n**' and '**N**').
   Search for these patterns (and test with **n**/**N**):

   1. Any line starting with two dashes (--).
   2. Any line ending in a colon (:).
   3. Any line starting with an asterisk (*).
   4. A dollar sign ($), anywhere on the line.
   5. A digit (anywhere on the line).
   6. A string of three or more digits, anywhere on the line.
   7. Any line containing exactly five characters.

peter@petermunro.org 087 689 1951

Module 16 - Comparing Files

cmp

1. Copy your `ulysses.txt` file to `ulysses2.txt`. Open `ulysses2.txt` in an editor (eg `vi`), go to the end and change a single character near the end of the file.
2. Now run `cmp` to compare the two files. It should report the difference.

diff

1. Run `diff` on the same two files. How does it report the differences?
2. Create two small directory structures and compare them with `diff -r`. (To create them, you could:
    1. Recursively copy say /etc/xinetd.d to your home directory;
    2. Recursively copy that to, for example, xinetd2.d.
    3. Edit a file in xinetd2.d.)

peter@petermunro.org 087 689 1951

Module 1 - Creating a Shell Script

Create a Shell Script

Write a shell script called *hello*. The script should display the words "hello, world".

Make sure you've added the 'shebang' line.

Type the command name to try to run it. Does it run? If not, why not? Can you run it by calling it as "`bash hello`"?

Create a Personal 'bin' Directory

Create a new directory called *bin* in your home directory. Move your script into it. What do you have to do now to have the script execute whenever you type its name? Make these changes and check that the script runs.

peter@petermunro.org 087 689 1951

Module 2

Accessing Command-Line Arguments (Positional Parameters)

Write a Command that Agrees with You

Write a command that works as follows. Whatever you say, it agrees with you:

```
$ i think Linux scripting is fun
I think Linux scripting is fun, too!
$

$ i think it is time for a break
i think it is time for a break, too!
$
```

Think about how this command is going to work:

- what will the name of the command be?
- how will your script print out all of the arguments?

Write a Command that Displays Information about How it was Invoked

This command should display information including: the number of arguments it was invoked with; its command name; argument one; all the rest of the arguments, like this:

```
$ checkscript one two three
The number of arguments is: 3
My command name is: checkscript
Argument one is: one
All the arguments: one two three
$
```


peter@petermunro.org 087 689 1951

Module 3 - If Statements

Using an if Statement

1. Write a script called `hasusername` that tells you whether the specified user has a username on the system:

```
$ hasusername joe
joe has a username on the system
$

$ hasusername billy
billy does not have a username on the system
$
```

2. Write a script called ison that checks whether a given user is logged in right now:

```
$ ison peter
peter is logged in
$

$ ison joe
joe is not logged in
$
```


peter@petermunro.org 087 689 1951

Module 4

Reading Input from the User

Modify your *hello* script from earlier to do the following:

- Prompt the user for their name
- When the user enters their name, collect it into a variable
- Print out "hello joe, how are you today?" (or whatever the user's name is)

peter@petermunro.org 087 689 1951