SCC 150 Practical
Week 14: Using the Linux Shell Bash

Task 0: Preamble / Installation

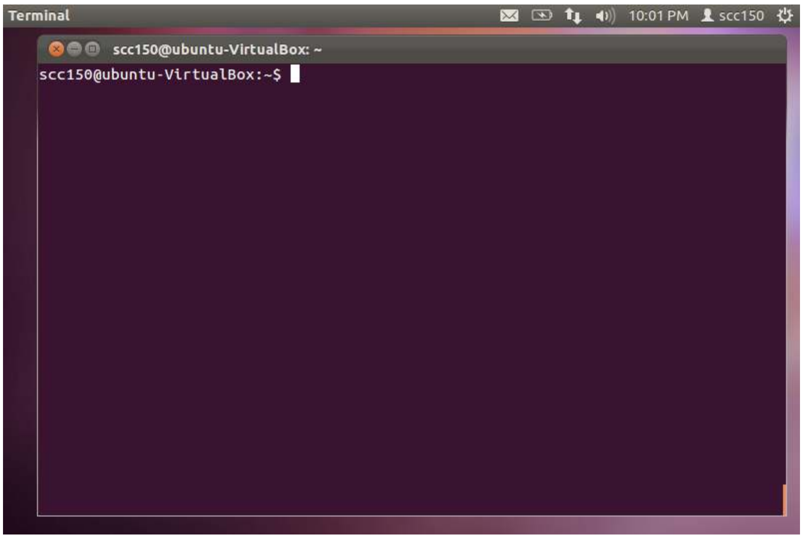
We will be using VirtualBox, a virtualisation utility, to run Linux within MSWindows for this practical. VirtualBox is already installed on your machine. Before you can run Linux you need to do the following:

- In a file explorer (right click on the windows icon of MSWindows' dock bar), move to the root of the C:/ hard drive by typing C:/ in the address bar of the file explorer.
- You should see a list of directories, one of which is named 'Utz'. Open the 'Utz' directory by double clicking on it.
- Utz should contain a directory called SCC150, **copy this directory to your MSWindows desktop**. (This is a critical step.)
- Open the SCC150 directory on your desktop, and double click on the file "SCC150.vbox". This file should appear with the icon:

Task 1: Finding and manipulating files in the console

Exercise 1.1: Finding the H: drive, creating a directory for your coursework there.

Once Linux has launched, log in as SCC150 (password SCC150), and open a console from the left-hand menu. You should get something like:



```
scc150@ubuntu-VirtualBox:~$ 
```

Typing F11 (or double clicking on the window's top bar) will switch the console to full screen. F11 will revert to windowed mode.

→ In the console, type

```
pwd[RETURN]
```

(print working directory), where [RETURN] is the RETURN key. What comes out is the *home directory* of the user scc150.

→ Now type:

```
cd De[TAB][RETURN]
```

(change directory), where [TAB] is the tab key. Now type [RETURN] a few times, and notice how the prompt of your shell (the string printed by default for each new line) has changed. This prompt can be configured, and on other unix systems might look different.

→ Now type:

```
pwd[RETURN]
```

Notice how your working directory has changed.

→ Now type

```
cd s[TAB][RETURN]

ls
```

What you should see is the top-level content of your H: drive.

IMPORTANT NOTE: The Linux VM will be reset between each reboot. Any file you want to keep must be saved to your H: drive (path ‘/home/scc150/Desktop/sf_win_h’). Anything else will be lost.

→ Now type

```
find . -name "*.c" -or -name "*.java"
```

This should print all C and java files you have on your H: drive. If the list is too long you can send the output of the 'find' command to the 'less' command using:

```
find . -name "*.c" -or -name "*.java" | less
```

The '|' character is called "pipe". It redirects the output of 'find' to 'less' rather than printing in on the console. It is a powerful construct that is extensively used. We will see more about this in the lectures.

→ Now type:

```
mkdir SCC150_linux

cd SCC150_linux
```

You have just created a directory called "SCC150_linux" on your H: drive. All your files for the Linux coursework should go there. Under MSWindows, use a file explorer to navigate to your H: drive, and check the directory you have just created is indeed there.

→ Now, back to the Linux console, type

```
pwd
```

You should get an output like:

```
/home/scc150/Desktop/sf_win_h/SCC150_linux
```

→ Now type

```
cd ../../
```

"../" means you are going back one level in the directly hierarchy. "../../" corresponds to two levels.

If you type

```
pwd
```

You should get an output like:

```
/home/scc150/Desktop
```

→ Using 'cd' and 'pwd' practice navigating inside the VM's file system and that of your H: drive until you feel confident about these two commands. Remember to use 'cd[RETURN]' (or 'cd ~[RETURN]') to go back to your home directory if you are lost. You may want to use 'cd -' for a change, and try to discover what it does.

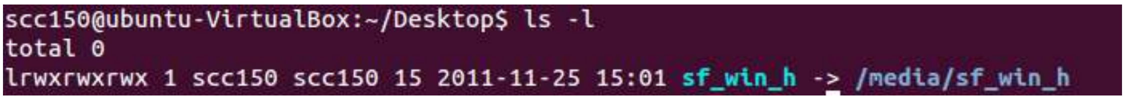
Exercise 1.2: Creating a symbolic link from your desktop to your new directory

We will now create a symbolic link (a shortcut) to your SCC150_Linux directory.

→ Before we do that, go back to your Desktop directory '/home/scc150/Desktop' and type

```
ls -l
```

You should get an output like



```
scc150@ubuntu-VirtualBox:~/Desktop$ ls -l
total 0
lrwxrwxrwx 1 scc150 scc150 15 2011-11-25 15:01 sf_win_h -> /media/sf_win_h
```

Note the "->" symbol. This means that 'sf_win_h' is in fact a shortcut to the directory '/media/sf_win_h', where your H: drive is mounted.

→ In the console now type:

```
ln -s sf_win_h/SCC150_linux .
```

Try to use the TAB key to get '~/Desktop/sf_win_h/SCC150_linux' to expand.

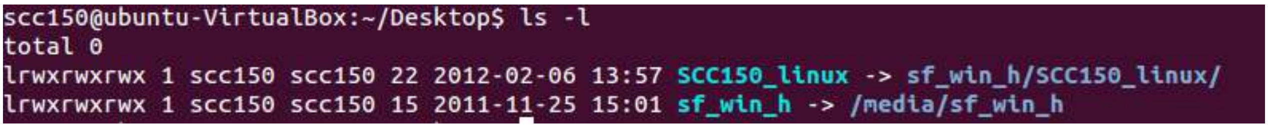
'ln' is a command to create file shortcuts in Unix systems. The -s options means the link is "symbolic", which means a special "symbolic" file has been created on your Desktop pointing to the right location. The alternative are "hard links", which would look as if 'SCC150' was a directory directly on your Desktop. Hard links are however only possible within the same file system (which is not the case of your H: drive). You normally will no need to use hard-links (They can create situations that are quite confusing.) You can get more detail with 'man ln'.

The dot '.' at the end of the line means "this directory" and asks 'ln' to create the link in the current directory (/home/scc150/Desktop).

→ Now check you have a working symbolic links. Type

```
ls -l
```

You should get an output like



```
scc150@ubuntu-VirtualBox:~/Desktop$ ls -l
total 0
lrwxrwxrwx 1 scc150 scc150 22 2012-02-06 13:57 SCC150_linux -> sf_win_h/SCC150_linux/
lrwxrwxrwx 1 scc150 scc150 15 2011-11-25 15:01 sf_win_h -> /media/sf_win_h
```

→ Now do

```
mkdir SCC150_linux/practical_week_14

cd SCC150_linux/practical_week_14
```

→ What do these two lines do? How can you check they worked properly?

Exercise 1.2: Fun with less, grep, zless, zgrep

In this exercise we will use the less and grep commands from the lecture (please refer to your notes for info on their role and basic workings), and will discover some of their variants, zcat and zgrep.

→ We will first download a text from the Internet using 'wget'. In your 'practical_week_14' directory, type:

```
wget www.gutenberg.org/cache/epub/11/pg11.txt
```

'wget' is a command line tool to retrieve web resources as a browser does, but with no interaction. It is quite powerful, and can download a whole web site (not advised if the site is large) or follow hyperlinks.

→ Check you have a file called pg11.txt in your lab directory using the console. Which command do you need to use to do that?

→ Find out what the size of "pg11.txt" is using the command line. If you don't remember how to do it, type 'man ls' to access the documentation of 'ls'. You can search for the keyword 'size' in the man documentation by typing '/size[RETURN]. Type 'n' in man afterwards to find each occurrence of 'size' in the documentation.

→ Use one of the commands of the lecture to print the beginning of the file on the console. Which command is it? Which book does the file contain?

→ Using **grep** find out how often the words "Alice", "Queen", and "head" can be found in "pg11.txt".

We will now compress pg11.txt directly on the command line, and use variants of grep and cat, called zgrep and zcat to work on the compressed version.

→ Type

```
gzip pg11.txt

ls -lh
```

What extension has the file that is now in your directory? What is its size?

→ Try to view pg11.txt.gz using "head". Does it work? Now try the following:

```
zcat pg11.txt.gz
```

and then the following

```
zcat pg11.txt.gz | head
```

The first line uses zcat, a variant of cat that decompresses the file(s) passed as argument before printing it (them) on the screen. The second command passes the decompressed output on to "head" to only print the start of the decompressed file(s).

→ Now use grep to search for 'Alice' in pg11.txt.gz. What do you get? Why?

→ Try the same with zgrep. Does it work? Type 'man zgrep' to understand what zgrep does.

Note: On your installation of Ubuntu, 'less' will be able to view the compressed file. That is because it uses a pre-processor (/usr/bin/lesspipe) to uncompress gzipped files on the fly. In systems with no pre-processor for 'less', 'zless' would be able to work as less on compressed files.

→ Have a look at '/usr/bin/lesspipe ' (using less of course). In what kind of language is lesspipe programmed in? Can you recognise constructs? Commands?

Exercise 1.3: Fun with sed

We will now touch on a powerful text tool called 'sed' (stream editor).

We will first uncompress 'pg11.txt.gz' back into an uncompressed version.

→ Use 'man gzip', to find out which command line tool to use for the decompression. On the command line use this command on 'pg11.txt.gz'. Check with 'ls -hs' the name and size of the resulting file.

To work more easily we will rename 'pg11.txt.gz' into something more descriptive.

→ Type

```
mv pg11.txt alice_in_wonderland.txt
```

'mv' means move, and is the command used to move and rename files on unix. With 'ls', check what is now the new name of pg11.txt.

→ Now type:

```
sed 's/Alice/Bob/g' alice_in_wonderland.txt
```

You should not be able to read much.

→ Using the same technique as we have seen with 'zcat' in exercise 1.2, redirect the output of sed to less. Do you notice anything changed? Were all instances of 'Alice' modified? Why?

'sed' is a powerful streaming editor to change a txt file automatically. In the above line, 's/Alice/Bob/g' is the sed script and means 'substitute (first 's') Alice by Bob globally ('g' at the end).

→ We will now save the result of our substitution in a new file. Type:

```
sed 's/Alice/Bob/g' alice_in_wonderland.txt > bob.txt
```

Now use 'ls' to look at the content of your directory. Try to view bob.txt using less and head. The '>' above serves a similar function to '|' we have seen earlier, but instead of redirecting output to another command, it redirects it to a file, here bob.txt.

Important: Be careful! If 'bob.txt' already existed, the above command will just overwrite it, with no warning. The console does not use any "Trash" folder, and a deleted file is thus permanently lost.

To learn more about sed, use your favourite browser and have a look at

" Sed - An Introduction and Tutorial" by Bruce Barnett, at
http://www.grymoire.com/Unix/Sed.html

You can try some of the example out for practice.

That is the end for this practical. Congratulations!

--- End ---

SCC 150 Practical
Week 16: Using regex to mine logs

Task 0: Preamble / Installation

- type "man ssh" and read the start of the documentation, learning what ssh is and what it can do.

- read the introduction of https://help.ubuntu.com/8.04/serverguide/C/openssh-server.html and learn about what sshd is, and how it works together with ssh.

- Download the file secure.log.gz from the SCC150 moodle page. Using command line tools, uncompress the file, and browse through it to understand what it contains.

Exercise 1: Distribution of SSH intrusion attempts – grep and sed

1.1. Using grep print out all the lines where the sshd server deamon complains about an invalid user attempting to break into the machine.

1.2. Read http://www.grymoire.com/Unix/Sed.html#uh-4 to learn what \1 can do for you in sed. (You might want to refer to the previous practical about sed too.) Note how the brackets (and) are escaped in the examples: This corresponds to old regular expressions. You might want to use the -r option to use the modern regex syntax (see man sed for more info on this). If you decide to use old regular expressions, note that + won't be recognised by sed.

Using the grep command from 1.1, and pipelineing the result to sed, extract the IP addresses from the computers from which invalid ssh attempts originated.

1.3. Using the example of 'SCC150_Week15_1' with sed, sort and uniq, compute the distribution of intrusion attempts across IP addresses. From which address did most the attacks came?

1.4. Read about the command 'host' with 'man host'. Using host on the first IP address, find out in which country this address is registered.

1.5. Read http://tldp.org/LDP/abs/html/commandsub.html#COMMANDSUBREF about command substitution with backquotes (`..`). Then type:

```
test=`ls *`
```

```
for i in $test; do echo $i; done
```

Can you explain what happened?

1.6. Using command substitution and a loop similar to the previous one, apply the command host to all the unique IP addresses from which attacks are originating. (You will need to modify your answer to 1.3. to return unique IP addresses with no frequencies).

Exercise 2: Distribution of SSH intrusion attempts – awk

2.1. Look at http://www.bolthole.com/awk2.html to remind you of what awk can do. Do the same as in 1.1. with awk.

2.2. Using what we have seen during SCC150_Week16_1, use awk to replicate the result of 1.3 although without sorting the IP according to their frequencies. Try not to use any other command (uniq, sort, etc.).

Exercise 3: Distribution of SSH username attempts – awk and uniq

Using awk and sort, print the distributions of usernames used to perform ssh intrusion attempts, printing username with their frequency from the most frequent ones to the least frequent.

SCC 150 Lent Coursework
Linux and Shell Scripting

Submission:

Deadline: Friday 27 April 2012 at 6pm (Week 21)

You should submit your answer as an electronic document to the moodle website of the SCC150 module (https://mle.lancs.ac.uk/course/view.php?id=150), as you have done for the Michaelmas coursework of this module. You will find detailed instructions on how to do so at https://mle.lancs.ac.uk/mod/resource/view.php?id=3703.

Format of submission:

- Do not include the text of the questions in your submission. Simply number your answers with the number of the question your are answering (e.g. 1a, 1b, etc.)
- Do include your full name and library card number in your submission.
- Clearly distinguish code from explanation in your submission. I suggest you frame any code as done in this document.

Question 1: pipelining and indirection [5 marks]

Consider the following shell command line:

```
echo "Hello SCC150 Class" | sed -r 's/Hello/Good Bye/' > helloGoodBye.txt
```

1a. Explain in a few sentences what this line does. [2 marks]

1b. Where is the result of the sed command stored? Which command can you use to view this result once the above command has executed? [2 marks]

1c. What is this result? [1 mark]

Question 2: regular expressions [6 marks]

Preparation: Download the text of the Magna Carta on the web site of the SCC150 module (MagnaCarta.txt).

We want to print all the lines containing the word 'king'. We first use

```
grep -E "king" MagnaCarta.txt
```

2a. Does this work? Explain in a few sentences what the problem is. [1 mark]

2b. Correct the above problem. Which updated command do you use? [1 mark]

Tip: You might want to refer to the first lecture of week 16 on regular expressions.

2c. Does your new version print lines contains 'King' with a capital K? How would you change it to do that, while still printing lines with a lowercase 'k'. [2 marks]

2d. Write a regular expression that matches words that:

- start with the same letter as your first name and
- end with the same letter as your first name and
- contain as many letters as your first name (although not necessarily the same letters).

[2 marks]

Note: You can test your regular expression by using grep in interactive mode, i.e. by typing

```
grep -E "some_regex"
```

where you replace some_regex by your regular expression. Grep will then repeat any line typed on the console that matches the passed regular expression. Type control + D (end of file character) to end grep.

Question 3: shell scripts and argument passing [6 marks]

3.1 Write the following code into a file called: my_shell_script.sh

```
#!/bin/sh
echo '$# is:' $#
echo '$0 is:' $0
echo '$1 is:' $1
echo '$2 is:' $2
echo '$* is:' $*
```

Make this file executable using 'chmod +x' (see Week 14, 2nd lecture).

Execute the shell script on the terminal, and try passing it different numbers of arguments (only use alphanumerical characters for your arguments to avoid issues with special shell characters). Observe the output you obtain.

Based on your experimentation explain in a few sentences what the variables $#, $0, $1, $2, and $* contain in a Bourne Shell script (the particular shell we are using here).

[3 marks]

3.2 Write the following text in a file called greeting.sh, and make the script executable.

```
if [ "$1" = "Francois" ]
then
   echo "Hello Francois"
   exit 0
fi
echo "???"
```

When executed, the script should return the following output:

```
scc150@ubuntu-VirtualBox:~/SCC150/CW2$ ./greeting.sh
???
scc150@ubuntu-VirtualBox:~/SCC150/CW2$ ./greeting.sh  Francois
Hello Francois
scc150@ubuntu-VirtualBox:~/SCC150/CW2$ ./greeting.sh Josh
???
```

3.2.a. **Change the code of greeting.sh so that ??? is replace by a full English sentence of your choosing.** [1 mark]

3.2.b. **Further modify greeting.sh so that it recognises your first name, in addition to 'Francois', and replies a personalised message.** The message to your first name should be different that the one for 'Francois'. [2 marks]

Question 4: HTML scrapping [6 marks]

Write the following code in a file called 'scrapping.sh', make the file executable.

```
#!/bin/sh
wget -q -O - "www.google.co.uk"
```

This script retrieves the webpage of www.google.co.uk and prints it on the console (standard output). The '-O -" is used to print the webpage on the console rather than storing it in a file (which would be the default). '-q' is used to turn off wget's own messages on the console. The returned webpage is written in a language called HTML. It is a kind of programming language that tells a browser how the webpage should be displayed. Here we have no browser, so the HTML code is directly printed on the console.

4.1. The above script can only retrieve Google's front page.

Modify the script so that it retrieves any URL passed as argument.

Explain what you have done in one or two sentences.

(You might want to look at Question 3.1 above.)

For instance

```
./scrapping.sh "scc.lancs.ac.uk"
```

should return the HTML code of SCC's front page.

[1 mark]

4.2. We now want to extract any hyperlink found in the retrieved HTML document that is returned by the above script. In HTML, hyperlinks are encoded using the following syntax:

```
<a href="http://www.scc.lancs.ac.uk/info/contact_details">Contact Us</a>
```

The string "http://www.scc.lancs.ac.uk/info/contact_details" is called a URL (Unified Resource Location), and is used to identify resources on the web (web pages, but also images, documents, dynamic information, etc.). To make sure each URL appears alone on a line, we are going to replace each double quote (") with a newline character ('\n', ASCII Character 10 on a Unix machine).

Modify scrapping.sh so that each quote is replaced by a newline in the output text using the command sed. [1 mark]

Explain what you have done in one or two sentences. [1 mark]

Tip: You might want to look at the first lecture of week 15 for an example on how to do this. You will need to escape the quotation character " using \" in your sed expression so that sed does not look for a matching quotation mark.

4.3. We are now going to extract the URLs contained in webpage returned by scrapping.sh.

Using grep, modify scrapping.sh to print each line containing a url. [1 mark]

Explain what you have done in one or two sentences. [1 mark]

For instance this is what I get on FSF's gnu website:

```
scc150@ubuntu-VirtualBox:~/SCC150/CW2$ ./scrapping.sh "www.gnu.org" | tail
http://shop.fsf.org/
http://www.fsf.org/join
http://donate.fsf.org/
http://flattr.com/thing/313733/gnuproject-on-Flattr
http://www.fsf.org
http://www.fsf.org/
http://www.fsfe.org
http://www.fsfla.org/
http://fsf.org.in/
http://creativecommons.org/licenses/by-nd/3.0/us/
```

4.4. When using your final script on www.google.co.uk or on scc.lancs.ac.uk do you notice any problem? **Explain in a few sentences how you could address them in your script.** (You do not need to provide the code of the actual script for this question.) [1 mark]

Total number of marks: 24

— END —