Introduction to Linux Part 2

Daniel Grose
STOR-i, Lancaster University

Lancaster University

Shell Scripting

STOR-i

excellence with impact

Lancaster University

Introduction

- Shell:
  - A program that interprets commands
  - An interface between the terminal and the kernel
- Script:
  - A program that runs in the shell
  - Allows you to write an application by “sticking together” other applications
  - Good for running repetitive tasks

STOR-i
excellence with impact

My First Shell Script

```
#!/bin/bash
echo "Hello World!"
```

Shebang – Tells the shell what application should be used to run the script

echo – Prints a message to STDOUT (as we saw yesterday)

Lancaster University

My First Shell Script

- Let's create the script
  - Start your preferred editor (gedit and nano are probably easiest for new users)
  - Type the script
  - Save it in your home directory (~) with the name my-first-script.sh
- Now let's check it works!

STOR-i
excellence with impact

Lancaster University

Executing My First Shell Script

- Executing a script can be done a number of different ways:
  - **$ bash my-first-script.sh**
  - **$ sh my-first-script.sh**
  - **$ ./my-first-script.sh**
  - **$ my-first-script.sh**
- The last two need some additional work to actually work however!

STOR-i
excellence with impact

Executing My First Shell Script

$ ./my-first-script.sh

- Needs to be made executable
- We can do this using the **$ chmod** command
- Specifically, **$ chmod +x ./my-first-script.sh**

$ my-first-script.sh

- Needs to have the ~ folder added to the PATH environment variable and be made executable
- Generally, not a great idea

Command Substitution

- Consider the very simple backup command
  - **$ tar zcvf backup.tar.gz ~**
- Creates a tar archive (gzipped) of all the user's home directory
- Don't use this to backup, as it's not good!
- So, let's make it better…

Command Substitution

- We can use command substitution to run a shell command and store it to a variable

```
#!/bin/bash
OF=my-backup-`date +%Y-%m-%d`.tar.gz
tar -zcvf $OF ~
```

- What is function of `date +%Y-%m-%d` ?
- Why is it useful in this context?
- For command substitution we use backticks (`) not quotes (‘)

Variables

- As we've just seen, scripts can have variables
- No data types
- Can contain a number, character, or a string of characters

```
#!/bin/bash
STR="Hello!"
here=`pwd`
echo "$STR you are at $here"
```

- What will the output of this script be?

Passing Arguments

```
$ ./myscript.sh arg1 arg2 arg3
```

```
#!/bin/bash
echo There are $# arguments to $0: $*
echo first argument: $1
echo second argument: $2
echo third argument: $3
```

- Arguments typed after a command on the command line are passed to the script as variables
- $* returns a single string containing all of the arguments
- $1, $2 etc refer to a numbered argument
- $0 refers to the script/command itself
- $# returns the number of commands in the command line
- What is the output of the above script?

Lancaster University

Setting Variables Interactively

- Using the **$ read** command:
  - Reads a line from STDIN
  - Assigns each word to the corresponding variable

```
#!/bin/bash
echo "What is your name?";
read MYNAME;
echo "Hello $MYNAME - hope you're well."
```

STOR-i
excellence with impact

Revisiting Our Backup Script

```
#!/bin/bash
echo "Extension of files to back-up";
read extension;
OF=my-backup-`date +%Y-%m-%d`.tar.gz
tar -zcvf $OF `find ~/ -name
  "*.${extension}" -print`
```

- A more specialised backup script
- **$ read** asks the user what extension the files have they wish to backup
- **$ find** searches for files in the ~ directory with that extension

Lancaster University

Exercise

- Create a backup script that has the following features:
  - When the script is executed, the user will be prompted to specify the location of the files they wish to back up
  - The script will then ask the user for a path to store the backup
  - The script will then ask the user to provide a name for the backup
  - The provided name will then be appended with the date and time of the backup
- Create a copy of the script that uses the folder to be backed up as an argument, rather than prompting the user for the location

STOR-i
excellence with impact

Lancaster University

Conditionals

- Decide whether or not to perform an action
- Decision is taken based on the results of an expression

```
#!/bin/bash
T1=“foo” T2="bar”
if [ "$T1" = "$T2" ]; then
echo expression evaluated as true
else
echo expression evaluated as false
fi
```

STOR-i
excellence with impact

Lancaster University

File Comparators

- -e file exists
- -f file is a regular file (not a directory etc.)
- -s file is not zero size
- -d file is a directory
- -L file is a symbolic link
- -r file has read permission
- -w file has write permission
- -x file has execute permission
- f1 -nt f2 f1 is newer than f2
- f1 -ot f2 f1 is older than f2

STOR-i
excellence with impact

Other Comparators

Integer Comparators:

- -eq is equal to
- -ne is not equal to
- -gt is greater than
- -ge is greater than or equal to
- -lt is less then
- -le is less than or equal to

String Comparators:

- = is equal to
- != is not equal to
- -z is null, zero length
- -n is not null

Lancaster University

For Loops

- Loops are used to perform actions until a condition is met, or until all data has been processed
- The For loop iterates across a list, executing the body of the loop for each value in the list

```
for i in 0 1 2 3 4 5 ; do
echo The number is $i
done
```

- The loop assigns the value of each list item to the variable i

STOR-i
excellence with impact

Lancaster University

For Loops and Command Substitution

- For loops can be used with command substitution. What do these loops do?

```
for FILE in `ls` ;
do echo The file name is $FILE
done
```

```
for i in `seq 1 10` ;
do echo "i=$i"
done
```

STOR-i
excellence with impact

While Loops

- While loops execute while a condition evaluates to true:

```
#!/bin/bash
i=0;
while [ $i -lt 10 ]; do
echo "i= $i"; let i=$i+1;
done
```

- While the value of i is less than ten, print the value of i, increment it, and loop

Shell Arithmetic

- Integer arithmetic can be performed using the **$ let**, **$ expr**, or **$ bc** commands

```
#!/bin/bash
BIGNUM=1024
let RESULT=$BIGNUM/16
echo $RESULT
```

```
#!/bin/bash
BIGNUM=1024
RESULT=`expr $BIGNUM /
16`
echo $RESULT
```

```
#!/bin/bash
BIGNUM=1024
RESULT=`echo "$BIGNUM /
16" | bc`
echo $RESULT
```

STOR-i
excellence with impact

Lancaster University

Exercises

- Write a simple shell script that takes a path of a directory as a command line argument and list all files and folders inside the given directory.
- Write a script that asks for the user's age. If it is equal to or higher than 18, print a message saying that this user is allowed to drink alcohol. If the user's age is below 18, print a message telling the user how many years he or she has to wait before legally being allowed to drink.
- Write a script that takes exactly one argument, a directory name. If the number of arguments is more or less than one, print a usage message. If the argument is not a directory, print another message. For the given directory, print the ten biggest files. (tip, you will need to use the following commands *du -a, sort, head*.)

STOR-i
excellence with impact

Lancaster University

Tools and Miscellany

STOR-i

excellence with impact

Lancaster University

Global Regular Expression Print

- Commonly known as grep
- Searches for a regular expression in specified files
- Prints the lines that match the pattern
- **$ grep [opts] <regexp> <filename>**
- For example:
  - **$ grep -n "^\<V[a-z]*\>" ~/names.txt**
  - Print all of the lines in names.txt that have first names beginning with V, as well as the line number

STOR-i
excellence with impact

Lancaster University

Regular Expressions

- Commonly known as regex or regexp
- Series of characters that specify a search pattern
- Supported by many utilities including grep, vim, sed and awk
- Can be incredibly powerful, but comes with added complexity

STOR-i
excellence with impact

Lancaster University

Regular Expressions

- Most characters will match themselves except:

Character	Meaning/Usage
*	zero or more matches of the previous element
\	Used to escape special characters
.	Matches any character except new line
^	At the start of the expression, means “start of line”
[ ]	Matches a single characters in the square brackets
[^]	Matches a single character that isn’t in the square brackets
$	At the end of the expression, means “end of line”
\<	Matches the empty string at the beginning of a word
\>	Match the empty string at the end of a word
|	Performs a logical OR on two expressions (only in egrep!)

STOR-i
excellence with impact

Lancaster University

Examples of Regular Expressions

- Match lines where any sequence of three letters that starts with r and ends with n appears:
  - **$ grep r.n myfile.txt**
- Match lines that end with the word finish:
  - **$ grep "finish$" myfile.txt**
- Match lines that begin with “\begin”:
  - **$ grep "^\\begin" filename.tex**

STOR-i

excellence with impact

Stream Editor

- Commonly known as sed
- Pattern matching engine that can perform manipulations on lines of text
- Works on a line by line basis
- sed is a batch editor
- **$ sed 's/foo/bar/g' myfoofile.txt > mybarfile.txt**
- Substitute all instances of “foo” in the contents of myfoofile.txt with “bar” and save the output to mybarfile.txt
- s means substitute
- g means “global”, so for the entire content

Lancaster University

Deleting content with sed

- You can delete content using the d function
  - **$ sed '3,7d' myfile.txt**
    - Deletes lines 3 to 7 from myfile.txt
  - **$ sed '/dogs/d' myfile.txt**
    - Deletes any lines that contain match the regular expression “dogs”

STOR-i
excellence with impact

Exercises

- Use the following command to download grepdata.txt to your home directory.
  - **$ wget www.lancaster.ac.uk/~fraserj/stor601/grepdata.txt**
- Use grep to print all lines that contain a phone number with an extension (the letter x or X followed by four digits).
- Print all lines that begin with three digits followed by a blank.
- Use sed to replace all ‘High School’ with ‘Secondary school’.
- Delete any email addresses that are from @example.com

Lancaster University

Secure Shell

- Commonly known as ssh
- Allows a user to connect to a shell on a different machine
  - **$ ssh fraserj@ius.lancs.ac.uk**
- You are presented with a password prompt if the remote machine supports password authentication
- To connect to a remote machine, ssh server must be installed and running on the remote computer
- You can also use an ssh client program on other operating systems to connect to the remote computer (e.g. PuTTY)

STOR-i
excellence with impact

Lancaster University

Secure Copy

- Commonly known as scp
- Allows you to copy files and directories from one machine to another using ssh
- Copy a single file to a remote host:
  - **$ scp ~/file.txt user@remotehost:/path/to/file.txt**
- Copy all .tex files to a remote host:
  - **$ scp ~/*.tex user@remotehost:/path/to/directory/**
- Copy the data directory from your home folder to a remote host:
  - **$ scp -rp ~/data/ user@remotehost:/~/data/**
- Copy a single file from a remote host to your computer:
  - **$ scp user@remotehost:/path/to/file.txt ~/file.txt**
- -rp means “recursively” and “preserve file attributes”

STOR-i
excellence with impact

Lancaster University

rsync

- Synchronises files and directories from one location to another whilst minimising data transfer
- Speeds up file transfer by only copying the differences between two files
- Performs synchronisation over the network
- Preserves permissions, ownership, symbolic links etc
- **$ rsync [options] source destination**

STOR-i
excellence with impact

Lancaster University

Using rsync to backup

- **$ rsync -avh /home/fraserj/ /media/disk/backup/**
  - Copies the contents fraserj's home directory to /media/disk/backup
  - -a – Archive mode. Same as -rp on scp
  - -v – Verbose mode. Shows more information
  - -h – Output numbers in human readable format
  - -e – Use ssh to connect to remote destination
  - --exclude='.bak' – Exclude specific files

STOR-i
excellence with impact

Lancaster University

STOR-i Ubuntu VM

STOR-i
excellence with impact

STOR-i Ubuntu VM

- The STOR-i Ubuntu Virtual Desktops are a new resource giving access to a full Ubuntu Linux desktop remotely.
- Each student has dedicated access to there own instance of the Virtual desktop, which in effect is your own personal virtual machine hosted on fast University hardware, accessible from almost any internet connected device.
- The VMWare Horizon Client application can be launched from AppsAnywhere. Access through these client applications is the preferred way of accessing your Ubuntu virtual desktop, however access from a web browser also gives a very good experience.
- The machines have the following specification:
  - Ubuntu 22.04 LTS
  - 4 x Intel Xeon CPU Cores
  - 16GB RAM
  - 80GB Disk
- Additionally, you have sudo rights with your standard account for additional installations and configuration changes.

Lancaster University

How to Access

- Install the VMWare Horizon Client from Apps Anywhere on your STOR-i laptop
- Navigate to https://mylab.lancaster.ac.uk in your preferred web browser
- Click the … menu button next to the “STOR-i Ubuntu 22.04” and click “Launch from Client”
- Log in using the same credentials you use for your STOR-i laptop

STOR-i
excellence with impact