University of Reading

WELCOME TO OUR WORKSHOP!

Before we begin, please do the following:

- Login to your PC
- Download the slides from:
  research.reading.ac.uk/rse/knowledgebase/unix-basics-training-materials/
- Work through slides 3 - 6 so that you're logged in to the Computing Cluster when we start the course

UNIX BASICS

An introductory workshop to the Unix operating system for novices

OBJECTIVE

University of Reading

- The objective of the course is to leave the student with enough knowledge of Unix commands to be able to
  - navigate the file structure
  - create directories and files
  - use a basic text editor
  - read files and modify them
  - search for files and file contents
  - chain commands
  - write simple bash scripts and execute them

COURSE OVERVIEW

- Getting started
  - Customising the display
  - Logging in and using the Unix shell terminal
- Lecture 1
  - Navigating the directory structure and creating directories and files
- Practical 1
  - A short exercise using materials from Lecture 1
- Short Break
- Lecture 2
  - File system operations and writing simple bash scripts
- Practical 2
  - A short exercise using materials from Lecture 2
- Tips and Tricks

GETTING STARTED

- Choose ‘MobaXterm’ from AppsAnywhere and click on ‘visit website’
- Click on the blue ‘Portable Edition’ button which starts the download
- Open your ‘Downloads’ folder and **extract** the MobaXterm zip folder
- Double click the MobaXterm executable in the **extracted** folder
- You should now have the MobaXterm Window.
- You can customise the appearance of the terminal window by going to Settings-> Configuration and click the ‘Terminal’ tab.
- Here you can customise the terminal window to your needs.

LOGGING IN

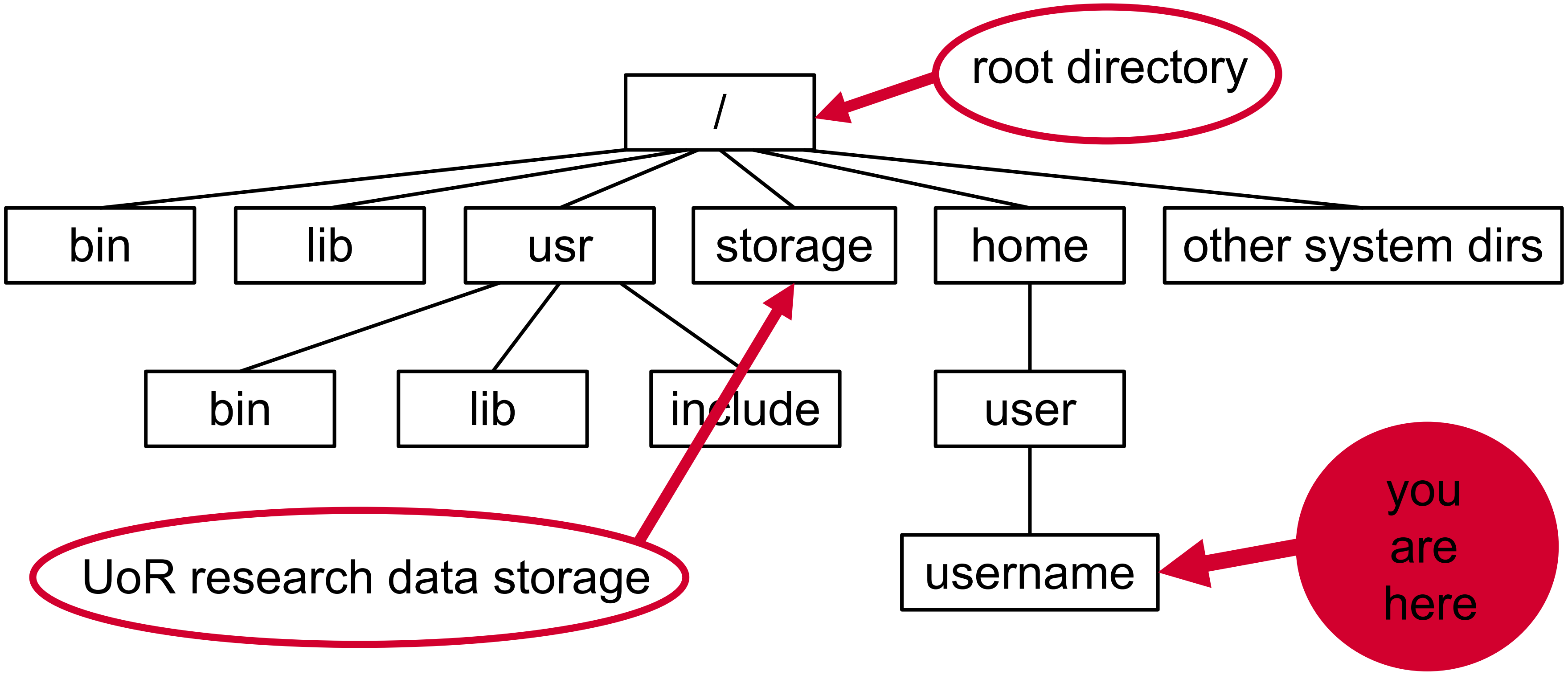
- Click ‘Start Local Terminal’ and then at the screen prompt enter

```
ssh -X cluster.act.rdg.ac.uk
```

- Enter your password and you’ll see a message that you’re connected to the ’Reading Academic Computing Cluster’
- You’ll see your bash shell prompt, which looks something like this:

```
[username@racc-login-0-1 ~]$
```

UNIX DIRECTORY STRUCTURE



WHERE AM I? - PWD

- ‘pwd’ is short for ‘print working directory’ (*not password!!!*)
- The ‘pwd’ command shows the directory in which you are on the system
- Example:

```
[sxs98pmh@racc-login-0-1 ~]$ pwd
/home/users/sxs98pmh
```

- In the example, the screen output shows the path which is your **home** directory, it is your working directory right after you log in.
- Your working directory might be shown in your shell prompt, here it is ‘~’ which is a shortcut for the user’s home directory

WHAT FILES DO I HAVE? - LS

- ‘ls’ is short for ‘list contents’
- Typing ‘ls’ and pressing the ‘return’ key shows you the contents of your home directory
- Example:

```
[sxs98pmh@racc-login-0-1 ~]$ ls
Desktop        Music      Videos
Documents      Pictures   test.txt
Downloads      Public
```

- Your home directory might be empty if this is the first time you logged in. If this is the case and you see no output, try the ‘ls’ command by e.g. listing the root directory with ‘ls /’

CREATING A DIRECTORY - MKDIR

- The command to create a directory in the current folder is simply: ‘mkdir directory-name’

- Example:

```
[sxs98pmh@racc-login-0-1 ~]$ mkdir mydir
[sxs98pmh@racc-login-0-1 ~]$ ls
Desktop        Music        Videos
Documents      mydir        Public
Downloads      Pictures     test.txt
```

NAVIGATING DIRECTORIES - CD

- ‘cd’ is short for ‘change directory’
- ‘cd mydir’ navigates into the ‘mydir’ directory
- Remember Unix is case sensitive!
- The prompt shows that you are in the ‘mydir’ folder:

```
[sxs98pmh@racc-login-0-1 mydir]$
```

- Of course there are no files in this folder yet!
- ‘cd ~’ returns to your home directory from any location on the system.
- ‘cd ..’ goes up one directory

- Note: in subsequent slides we shorten the prompt `[sxs98pmh@racc-login-0-1….]$` to just ‘`$`’. Where you see a ‘`$`’, a command entry into the terminal follows

KEYBOARD TRICKS

- Example:

```
[sxs98pmh@racc-login-0-1 ~]$ ls mydir
```

The ‘Return’ key executes the above command.

The ‘up arrow’ recalls the command again.

The ‘home’ key puts the cursor at the beginning of the line.

- Now ‘ls’ can be deleted and replaced with ‘cd’ so it is like this:

```
[sxs98pmh@racc-login-0-1 ~]$ cd mydir
```

- With this keyboard trick you don’t have to retype previously used commands

MORE TIPS AND TRICKS

- Another aid to speedy bashing is to copy and paste to the command line.
- To save using the mouse you can press ‘shift + insert’ keys together

- Bash features an auto-complete: start typing the name or path for a file then press the ‘Tab’ key. If only one possibility exists then bash will complete it in the command line. If more possibilities exists bash will show you the various options.

- Try a random directory: ‘ls /v(tab) w(tab) h(tab)’
- You should now have a line that looks like this: ‘ls /var/www/html/’

University of Reading

CREATING FILES - TEXT EDITOR

- Various text editors are usually available on a linux machine, the choice is personal preference. Some are command based only, some have graphical user interfaces.
- Some popular editors include: vi/vim, emacs, nano and gedit
- We will now use gedit to create a simple file called hello.txt.

- In your mydir directory, type ‘gedit’ to launch the editor.
- Type a text line, e.g. ‘Hello World’ in the editor window and then save the file as ‘hello.txt’.
- Then close the editor again and check with ‘ls’ if the file has been created successfully.

READING FILES – MORE AND CAT

- The commands ‘more’ and ‘cat’ read a file and display it to the screen.
- ‘cat’ is short for ‘catalog’ and displays the entire file
- ‘more’ displays the file one page at a time

- Example (in your mydir directory):

```
$ more hello.txt
Hello World
$ cat hello.txt
Hello World
```

- The difference here is that for large files ‘more’ will prompt you to show another page, whereas ‘cat’ will read the entire file to the screen.
- When reading a large file with ‘more’, you can use the ‘Enter’ key to move down line by line or the spacebar to move down a whole page.

University of Reading

COPYING AND MOVING FILES: CP AND MV

- The ‘cp’ command copies a file
- Example (in your mydir directory):

```
$ cp hello.txt hello2.txt
```

- The ‘mv’ command moves a file to a new filename. This is a way of renaming files.
- Example (in your mydir directory):

```
$ mv hello2.txt hello3.txt
```

CAUTION! This command is irreversible and you can corrupt or lose your files in the worst case scenario. If in doubt, always use ‘cp’ first!

REMOVING FILES - RM

- It goes without saying that the ‘rm’ command is one of the more dangerous commands, but you need to use it to remove files.

- Example (in your mydir directory):

```
$ rm hello3.txt
```

- This removes the hello3.txt that we created previously.

- Be careful, ’rm’ has no undo option and there is no trash folder in linux!

University of Reading

COMMAND OPTIONS

- Most commands have options which allow more functionality. These options are often also called ‘switches’ or ‘flags’.
- The basic form is: command [*options*] [*file*|*dir*]

- Examples:
- The ‘cp’ command can also copy directories using the ‘–r’ flag. It will *recursively* copy the directories and files within.

```
$ cp -r mydir mydir2
```

- The ‘–r’ flag can also be used for other commands, such as ‘rm’ , to manipulate directories rather than files.

FILE PERMISSIONS

- If you want more information about your files, such as size and permissions, use the '-l' switch with the 'ls' command
- Example:

```
$ ls -l mydir/hello.txt
  -rw-r--r-- 1 sxs98pmh sxs     14 Feb 16 09:20 hello.txt
```

- -rw-r--r-- are the file permissions. 'r' for read, 'w' for write and 'x' for execute. The first set of three (here it's rw-) after the first dash applies to the user (you), the second to group access and the third to everyone.
- The file is readable and writable (by you) but not executable and readable to other group members and everyone.
- For a directory the first dash is replaced by a 'd'.
- We will manipulate the file permissions in the second part of the course.

THE ECHO COMMAND

- The ‘echo’ command is a very simple but useful tool: all it does is output its argument to the screen.

- Example:

```
$ echo Mary had a little lamb
Mary had a little lamb
```

REDIRECTING OUTPUT: ‘>’ AND ‘>>’

- With the > and >> operators you can redirect screen output to a file.
- To use them simply add them after a command with a destination filename.
- We previously used the ‘echo’ command which outputs its arguments to the screen. We can *redirect* the output of echo into a file using ‘>’.

- Example (in your mydir directory):

```
$ echo Mary had a little lamb > mary.txt
$ ls
mary.txt
```

University of Reading

APPENDING FILES WITH '>>'

- The append file operator '>>' is used in a similar way to the '>' operator.
- Example (in your mydir directory):

```
$ echo Its fleece was white as snow >> mary.txt
$ ls
mary.txt
$ more mary.txt
Mary had a little lamb
Its fleece was white as snow
```

WILDCARDS

- First, let’s create some dummy files so we can see wildcards in action.
- HINT: use the up/down arrows to create the below more easily.

- Example (in your mydir directory):

```
$ echo "poem1" > poem1.txt
$ echo "poem2" > poem2.txt
$ echo "poem3" > poem3.txt
$ echo "poem1" > poem1.doc
$ echo "poem1" > poem1.dcsv
$ echo "poem1" > poem1.pdf
```

- Check if the files are there with ‘ls’

WILDCARDS

- If you have many files and subdirectories you'll probably want to filter the results to only show the files/folders you are interested in.
- In Unix there are two 'wildcard' operators: '*' and '?'
- The asterisk '*' represents any set of characters, and the placeholder '?' is any character in a certain position.

- For example, to show only text files one could type: 'ls *.txt'
- To show only files called 'poem1', try: 'ls poem1.*'
- For several files called poem1.txt, poem2.txt, poem3.txt etc one could use '?' to filter for these: 'ls poem?.txt'

DANGEROUS COMMANDS

- While flags and wildcards are a very handy tool to extend the functionality of simple commands, some care has to be taken when using them.

- Examples:

DO NOT USE THOSE UNLESS YOU ARE ABSOLUTELY SURE!!!

```
$ rm *
```

This removes all files in current folder.

```
$ rm -rf *
```

This forcibly removes all files and subfolders in current folder.

```
$ rm -rf .
```

This forcibly removes the current folder and all files and subfolders.

University of Reading

MISSION ABORT – CTRL-C

- Sometimes you mistype a command or issue a command which takes a very long time or appears to hang.
- The way to get out of this is to press the ‘ctrl’ key and the ‘c’ key at the same time.
- You should then see the terminal prompt return.
- You can try this with the ‘sleep’ command, which just results in inactivity for the prescribed time (100 seconds in the example below).

- Example:

```
$ sleep 100
```

Now press ctrl and c to abort the above command.

HOW TO GET HELP - MAN

- All Unix commands come with on-board help which you can access via the ‘man’ command, which is short for ‘manual’
- Let’s try it for the ‘ls’ command:

```
$ man ls

 LS(1)
  NAME
       ls - list directory contents
  SYNOPSIS
       ls [OPTION]... [FILE]...
  DESCRIPTION
       List information about the … … …
```

- Almost all inline help follows the above format.

HOW TO GET HELP - MAN

- The key points to realise are the syntax of the command and the options or ‘switches’

```
SYNOPSIS
     ls [OPTION]... [FILE]...
```

- Switches are sub commands which modify the output
- Earlier we used ‘ls –l’. Here we used the ‘-l’ switch to ask for the long format of list command
- For the file we’ve created, this translates to: ‘ls -l mydir/mary.txt’
- Switches can be combined together e.g.: ‘ls –lr’, which means list in long format and in reverse order.

OTHER WAYS TO GET HELP

- As well as the ‘man’ command, nearly all Unix commands provide embedded help on their usage.
- Depending on the command, switches in the form ‘–h’, ‘--help’ and ‘-?’ can be appended to the command to show the commands’ usage.
- For example the help option for the ls command used earlier can be invoked by simply using ‘ls –help’
- The ‘more’ command help can be invoked by typing ‘more -?’
- The ‘man’ command usage can be invoked by typing ‘man –h’
- Invocation of help is not standardised, but one or other of the above will nearly always work.
- If all else fails, just use the ‘man’ command!

TIPS AND TRICKS

- Bash maintains a history of everything you typed, in a hidden file called .bash_history in your home directory.
- You can see this file (and other bash initialisation files) if you type ‘ls -a .ba*’
- The history file can be useful to trace back long commands that you want to reproduce but have not kept a record of.
- The dot in front of the filename represents a hidden file. These files are usually hidden from view but the ’-a’ flag reveals them to the ‘ls’ command

PRACTICAL 1

- Work through the examples in the lecture (use alternative file/directory names if you have followed along):
  - Use the commands ‘pwd’ and ‘ls’ to look at your home directory. Note that your home directory might be empty if you are logging in for the first time. Keep using these commands throughout this exercise!
  - Create the mydir directory and practice navigating with ‘cd’
  - Practice the ‘echo’ command and create, read and append the ‘mary.txt’ file using the example commands from the lecture
  - Practice copying and removing files and searching for files with wildcards. Create some dummy files for this exercise.
  - Have a look at manuals of some of the commands we’ve used with ‘man’ and see what other options are available.

University of Reading

USING GREP TO SEARCH FILE CONTENTS

- The command ’grep’ is a "general regular expression parser“
- Do not be put off by this mouthful – think of it as a search engine that can look inside files for bits of text.
- The simplest example of its use is to find a word or phrase in a bunch of files within a directory.
- Search within a single file:

```
$ grep lamb mydir/mary.txt
```

- Search within a directory:

```
$ grep lamb mydir/*
```

USING THE PIPE OPERATOR TO CHAIN OUTPUTS

- The ‘pipe’ operator is used to ‘chain’ the output of one command into another. Combined with grep (for example) it can be used to filter or generate new output.
- Example:

```
$ ls *.txt | grep Mary mydir/*
```

- The two operators can even be combined thus:

```
$ ls *.txt | grep Mary mydir/* >
  only_Mary_files.txt
```

- You now have a file containing a list of files which only contain the word Mary!

CREATING AN ALIAS COMMAND

- We will create a new command called 'list'. If you type 'list' before you follow the steps below, you will get an error that the 'list' command doesn't exist.
- The new 'list' command will be an alias for 'ls –l' which is used frequently and typing it every time can be tedious.
- We can make this alias permanent and ready to use at each login by putting it into the .bashrc file.
- The .bashrc file is located in your home directory. This is a system file that allows you to customise your shell environment.

- Example:

```
$ gedit .bashrc
```

CREATING AN ALIAS COMMAND

- If the .bashrc file isn't yet present, the previous command will create it. In this case it will be an empty file. At the end of the document type the alias command alias list='ls -l'
- Save and close the file.
- Activate your .bashrc file with the 'source' command.

```
$ source .bashrc
```

- The above happens automatically at login.
- Now you have essentially created a new command for 'ls -l'. When you type "list" now at the prompt you should get the long format directory list as if you had just typed 'ls -l'.

University of Reading

BASH SCRIPTING

- We will now write a simple bash script that incorporates the ‘echo’ command as well as the ‘ls’ command.
- So far, commands have been typed into the terminal screen.
- Unix provides a way of executing commands in a file. We shall use the extension “.sh” to denote that this is a shell script.
- A bash script reads a file from top to bottom executing one line at a time.

- Example:

```
$ gedit myscript.sh
```

BASH SCRIPTING CONTINUED

- Enter the following into your script:

```
#!/bin/bash
echo Your directory contains the following files;
ls -l;
echo Done;
```

- Then save and close the file

BASH SCRIPTING: PERMISSIONS

- Have a look at the script with the long format list command:

```
$ ls -l myscript.sh
-rw-r--r-- 1 sxs98pmh vis 85 Oct 23 16:33 myscript.sh
```

- Have a look at the file permissions: the file is readable and writable (by you) but not executable. If you try to execute it, you'll get a 'permission denied' error.
- In order to run any script, we have to make it executable ('x'). This can be done with the 'chmod' command.

```
$ chmod u+x myscript.sh
$ ls -l myscript.sh
-rwxr--r-- 1 sxs98pmh vis 85 Oct 23 16:33 myscript.sh
```

- The batch file can now be executed by typing

```
$ ./myscript.sh
```

PRACTICAL 2

- Practice everything you've learned in the second lecture.
- Follow the examples to practice 'grep'.
- Create a shell script using gedit that:
  - Creates a directory.
  - Lists home directory contents and puts it in a file in the created directory.
  - Uses 'grep' to look for a word you have used in your example and pipe the output to the screen.
  - Optional: add extra functionality to the script if you have time.

University of Reading

THE END!

- Happy Bashing! ☺

- Please give us your feedback by following this link below:

- https://forms.office.com/Pages/ResponsePage.aspx?id=xDv6T_zswEiQgPXkP_kOX7ArvOm3cbpHnixhCNWKRS9UMVNBWkdWUklHN0kzMjcxQ0wwV0UwSlhaRS4u