GETTING STARTED - LEARNING THE UNIX SHELL

UNIX is an operating system first developed in the 1970s, and has been under constant development ever since. It's been developed as Linux, it is also hidden behind the shiny user interface of MacOS and iOS (Mac, iPhone, iPad etc.). *So it's still very much with us and relevant today*!

UNIX is a stable, multi-user, multi-tasking system for servers, desktops and laptops. This means you can run multiple jobs at once, start and kill processes, have multiple users, remote access servers, install and configure network services (such as web servers, databases) and so on. Linux is at its heart a UNIX operating system, extended with a (choice of) graphical user interface. Many operating system services and all applications you interact with run on top of the UNIX OS. Unlike Windows and MacOS, nearly all the software is open source, so you can even contribute to its development!

> Classic quote: "*It's a UNIX system, I know this…*", Jurassic Park (1993) - although, as you'll soon agree, Hollywood doesn't know a UNIX system…

Today, you're going to become or hone your skills as 'a power user'.

The absolutely core thing you need to do today, is ensure you don't try to 'just get by' using just the mouse and windows GUI - *this is the fast path to learning nothing useful today.* Instead, if you get stuck, *ask the people around you and the staff for help, we're learning together and that's why we're here - to help you learn!*

- We're going to learn how to use 'the shell', a powerful extensible command line user interface that is the 'power user' interface for interacting with UNIX and the basis *for most system administration* on any sensible OS platform! Specifically, bash an open source replacement for the Bourne shell developed by Bell labs in 1979.

We are going to use a Linux (Ubuntu) which is installed on all the lab machines in SCC's labs. Find out how to access virtual lab machines and install on your own computer.

TODAY'S TASKS

We're going to use recommended parts of an open source 'Software Carpentry' tutorial on GitHub that nicely introduces key features to get you started with UNIX shell.

There are tons of 'headroom' in this task, so if you're already experienced with UNIX shells, you *can skip* to the interesting parts (*but I'd advise reading through quickly to be careful not to overestimate your prior knowledge!*)

> https://swcarpentry.github.io/shell-novice/

This is a *great* introduction to the shell that goes well beyond the core set of features you'll need on this course to get started with programming. We'd like you to ensure that you understand the following key shell features to make the coming weeks and indeed all future labs in your courses easier:

Part A

Click on `terminal` on the toolbar (or hotkey `Ctrl + Alt + T`), which should open a window with a UNIX bash shell (like everything on Linux, you can configure the terminal font, size and appearance and hotkeys as you wish). Minimally, you should work carefully through these sections working towards *understanding how the commands work*, *a useful subset of command arguments*, and why they are there in the order that they are...

1. Download the files needed to complete the tutorial (start here)
2. Ensure you absolutely understand how the UNIX structures files, and files into directories into a `tree`, and how to create, rename and delete files.

How does the tutorial example file system compare/ differ to that on the machine in front of you? Especially you should know and understand `pwd`, `cd`, `ls`, `mv`, `cp`, `rm` (*care there is no undelete on UNIX!*), `mkdir`, `rmdir`, `less`, and `man` (*very very useful*). **Keeping a 'lab notebook' and making notes will help you remember all this! It's easy to forget what you did, especially if you don't do it very often!**

Learning checklist:

- ☐ Can you find the path to your home folder using *only* the terminal, and *only* the graphical browser? Make a folder in your home folder, can you see it in the browser. Make a folder Using the graphical browser, can you see it in the terminal?
- ☐ Have you made a folder for your files, e.g. called `src`, or by course `scc111`, `scc121` etc. (*not using the graphical UI nor mouse*)
- ☐ Navigate to the root of the file system in the shell, listing the files in each folder as you go (hint: `cd ..`). Navigate back to your home folder using `cd` (with no arguments).
- ☐ Use a `wildcard`, e.g. can you list all the files in `/usr/bin` starting with a particular prefix (starting with `l`)? How many files or programs are there? *Yes, most of the commands you are running are just other programs that ship with UNIX...*
- ☐ List all the files in your home folder in 'long format'. List all the files in your home folder *including* any hidden files and folders. Note the permissions (`rwx`), file owners and how folders (directories) are flagged.
- ☐ Create a new file in your home folder called 'README.md' using `touch`. Open in a text editor and add some notes on what you've just done so you can remember next week (e.g. `code`, `vim`, or `nano`). Save it, then view its contents in the terminal (hint: try `more` and `less`)
- ☐ **IMPORTANT** you should know where your Uni wide 'h drive' is mounted in the lab file system (what's it's `path`) and how to open, copy or move files there. Use `cp` and a `wildcard` to copy all of the files and folders to your `h` drive. Hint: checkout `man ls` (the manual) and particularly `cp`'s `-r` option. *You'll want to do this because 'h' drive files are backed up, and any files you save on the lab machines will be lost when the machine is rebuilt and upgraded.*

Part B

One spectacular feature of UNIX is the ability to send the output of programs to files, pass files as input, and even chain programs together so the output of one becomes the input to another. *Amazingly powerful!*

- ☐ Learn about output `>` file redirection (work through the pipe and filter tutorial), then do a *long listing* of your home folder redirecting the output to a file called `my-files`
- ☐ Can you use `cat` and `wc` utilities to count the number of files listed? How many lines, words and characters were listed? What happens if you pipe `my-files` through `sort`?

By the end of this section you should be feeling confident in using the UNIX shell and basics of `bash`.

Listing your current folder, changing folder to the parent, and back. Plus the basics of copying, renaming, and deleting files using (just) the shell. Editing and saving files, and taking a backup in your `h` drive.

Part C

Here's a few tasks to get you practicing some of your new skills with the shell effectively.

1. `curl` lets you fetch Internet files such as a web page as raw content. You can then process this content e.g. by redirecting to a file `>` or piping into something else `|` . e.g.

```
curl http://lancs.ac.uk
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://www.lancaster.ac.uk/">here</a>.</p>
<hr>
<address>Apache Server at lancs.ac.uk Port 80</address>
</body></html>
```

(you can see our website has moved from this)

`grep` lets you look in a file (or input passed into it using a pipe) for a pattern, for example:

```
curl https://www.lancaster.ac.uk | grep h1
```

- [ ] returns what ?
- [ ] How many level 2 headings (`h2`) are in the same web page?

2. Download some open government data (e.g. lancaster_brownfieldregister_2019_06_17_rev1.csv) using `curl` .

   You could look at how many lines were downloaded, it's file size, and you may find it helpful to view the first few lines, e.g. if you saved it as `brownfield.csv` like this:

```
wc -l brownfield.csv
ls -lh brownfield.csv
head -3 brownfield.csv
```

- ☐ Using `grep` , and possibly but not necessarily, other tools (grep has lots of cool options, see the `man` page!), can you write a one liner which finds out how many sites are `"not permissioned"` . *Note that if you need to include a space in a pattern or filename, you'll need to put it in double quotes* `""` , or use `\` to 'escape' the spaces.

3. Fancy a bit more of a challenge? Download the Lancashire MIDAS rainfall data for 2020, 2021, 2022. This gives the hourly rainfall for nearby Morecambe for the years 2020-2022 inclusive.

   - ☐ can you use `tail` to strip off the explanatory headers (first few rows) and get just the first 24 hours of data (from 1st January) each year
   - ☐ strip out the rainfall column `prcp_amt` from the CSV files, *taking care to remove the header line and the* `end data` *footer line* - hint: `cut -d, -f<#>` - where <#> is the *field number* or numbers you want will help!
   - ☐ Add up the rainful for each of the years - hint: `paste -sd+ - | bc` would merge all the input lines into `paste` and add them together separated by plus `+` . This output is then used by the calculator `bc` to total the numbers.

HACKER EDITION

Want more?

MORE SHELL TASKS

1. Extend the rainfall task above. How can you total the rainfall for all 3 years at once as a single command?
2. Learn how to create a loop to run a command many times or on many files. Can you use a loop to move the rainfall files to a subfolder?
3. Or even, create handy scripts of commands (yes, shells have a basic programming language!). Create a shell script e.g. called `backup.sh` for making a backup of your files to your home folder with a filename '`backup-<today's date>`'.
4. Explore `bc` a little further. How can you use bc to convert decimal numbers to hexademical, and hexadecimal numbers to binary? See `man bc` .
5. Learn about how to use regular expressions with `grep` . This is fantastically useful for searching through files, or filtering the output of an application to pick out specific output matching a pattern using *a pipe*, such as error messages.

INSTALLING THE COMPILER

RECOMMENDED You can use a lab machine running in a virtual environment over the network from more or less anywhere! This has **all the software and tools you'll need instantly**! Access at `http://mylab.lancaster.ac.uk`. *Care: you are logging into a virtual Linux machine, the keyboard may behave differently and this can affect how easy it is to type your password, especially if you have symbols!*

If you want an offline environment, you'll need the following for SCC.111 lab exercises:

- a C/C++ compiler, we recommend the free `gcc` (gnu-C) compiler.
- You'll also need python version 3 (latest stable release will be fine) at the time of writing this is 3.12.x.
- You'll also need `git` revision control software

Depending on your platform and what's installed on your computer, you have various options for getting this *more or less* easily using a software package manager:

LINUX

If you're running Linux use your package manager, e.g. with `apt` we'd recommend the following packages:

```
sudo apt update
sudo apt install build-essential
sudo apt-get install manpages-dev
```

You need to be administrator (your user in the `sudoers` file) to be able to do this.

WINDOWS

If you're running Windows, then there is a 'Windows Subsystem for Linux', which gives you the *same environment* on your Windows machine:

- `https://learn.microsoft.com/en-us/windows/wsl/install`

MAC

On the mac platform, you can install the full Xcode development environment (large) using the 'app store', then find the 'install command line tools' menu option. But at a minimum you need the compiler and development tools and libraries (recommended unless you want to be doing full Mac application or iOS development). From the terminal, run:

```
xcode-select --install
```

You will need to accept the license to be able to use `gcc` but you will be clearly prompted to do so.

VERIFYING YOUR INSTALL

When this is installed, you should be able to type:

```
gcc --version
```

Which should produce output such as:

```
gcc (Ubuntu 13.2.0-23ubuntu4) 13.2.0
Copyright (C) 2021 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

INSTALLING THE RECOMMENDED CODE EDITOR

We will be recommending you use *Visual Studio Code*, which is a *free* modern highly configurable text editor that has loads of helpful plugins and syntax colour highlighting. This is cross-platform and will work on Windows, Mac and Linux:

- `https://code.visualstudio.com/download`
- In the lab we have pre-installed the C programming extension and the Python programming extension - you can install these easily from *within the editor* (read the help on managing extensions)
- Some of us are fond of the `vi` (read: classic UNIX editor) family of editors e.g. vi-improved, or vim; if you want to go there, definitely do the cool interactive tutorial. Another options is gnu emacs,

which is massively extensible and customisable and also has a loyal following.

MORE LINKS

- Instructions for installing and testing GCC on Ubuntu - should be the same more or less on newer versions of Ubuntu
- Installing GCC on other platforms