CITS4407 Open Source Tools and Scripting
Semester 1, 2021
Week 2 workshop – Bash and Linux introduction

Before starting this workshop, work through the "Accessing Linux" worksheet, to make sure you can access a Linux environment.

1. The bash terminal

If you follow the instructions in the "Accessing Linux" worksheet, you should currently have a terminal window in front of you, showing a `bash` prompt – the prompt usually looks something like this:

```
[bobjones@linuxbox ~]$
```

This is *prompt* is a sign that the shell is ready to accept input. The exact prompt you see may vary, but normally it will end with a dollar sign, and from now on, we'll just show a dollar sign as an abbreviation for it.

Type any random string you like into the terminal, and hit enter.

You should see something like:

```
bash: fnorrrrrd: command not found
[bobjones@linuxbox ~]$
```

This is fine – it means bash is working, but didn't recognize the command you typed.

It will be (almost) impossible for you to do permanent damage to a computer by typing incorrect commands, so don't be afraid to experiment. The worst that is likely to happen is that you see an error message.

Try typing the following commands (for each one, type it after the prompt and then hit "enter"):

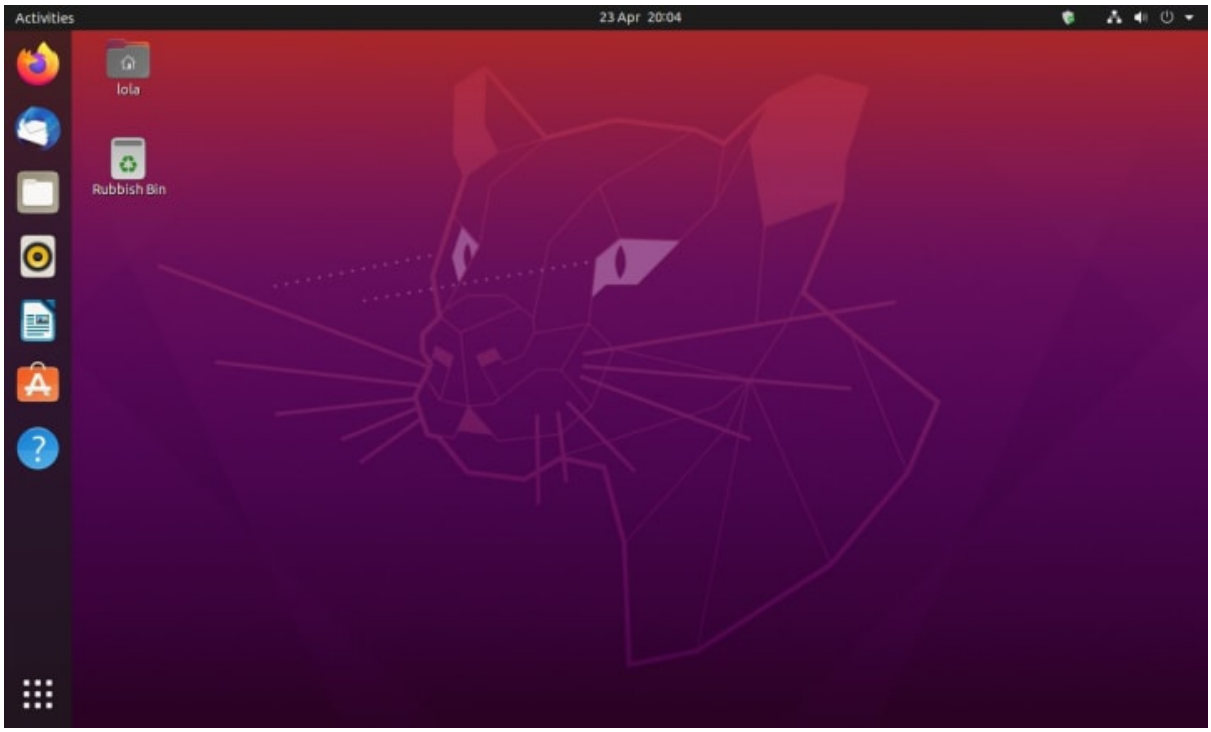
- `date`
- `ls /`
- `ps`

What do you see, in each case?

After typing a few commands, try hitting the "up" arrow on your keyboard – the commands you've already typed should appear in reverse sequence. This is Bash's *command history*, a record of commands you've entered. Hitting the "down" arrow should take you back to a plain prompt.

2. The Linux desktop

If you're using a University lab machine or VirtualBox on a MacOS X computer, you will be able to see the Linux *desktop* environment – something like this, for MacOS users:



We won't require you to use a Linux desktop environment, but it may be useful to know how to use it if you have one.

Take a look through the menus available to you, and see if you can find out how to launch a web browser, a terminal window, and a *text editor*.

3. Bash built-in commands

When you typed "`date`" or "`ls`" or "`ps`", this was effectively asking Bash to look for a program with that name, run it, and show you the results (if any).

But bash also has a number of *built-in* commands, which we'll use through the semester.

Type the following:

```
$ echo hello there
```

You should see the words "`hello there`" printed in the terminal. "`echo`" is not the name of a separate program, but is a command built in to Bash.

Type the following:

```
$ help
```

Bash will display *all* its built-in commands. You can type to see what the "`exit`" command does, and similarly for any other built-in command.

4. Looking up documentation

Try typing each of the following commands, to see what they do:

- `man pwd`
- `man ls`
- `man cd`
- `man touch`
- `man mkdir`

They should display *manual pages* for different Unix commands in what's called a *pager* (an environment where you can use the "Page Up" and "Page Down" keys to scroll back and forth through a file). In each case, you can type the "`q`" to quit the pager and get back to the Bash prompt.

If you ever can't work out how to get back to a bash prompt – there is no harm in just closing the terminal window you're using, and opening another.

Some additional information on the `man` system can be found here.

Finally, try typing `man bash`, and start to page through the manual for Bash itself. The manual is *very* long – if you hit `shift-G`, the pager will take you all the way to the end, and should show you how many lines of text are in the manual (down the bottom of the screen).

We will be exploring quite a few of Bash's features, but it is a very complex program, and we won't have time to look at them all. But if you are every curious, the bash "man page" is the place to find out what they all are.

5. The Unix file system

Just like on Windows or MacOS X computers, a Linux computer stores files on *disks*, and organizes the files into what's called a *file system*.

Any file system has a *root directory* – a top-level directory which contains all other files and directories. (On Windows, there are multiple "root" directories, but on Linux, there is just one.) The root directory is wrtten as follows:

```
/
```

– that is, with a forward slash. Linux also uses a forward slash to separate directory names in the *path* to a file or directory (whereas Windows uses a backslash).

Type the following commands

```
$ echo hello there > /tmp/hello.txt
$ less /tmp/hello.txt
```

We will look more at what the "`>`" symbol in the first command is doing later – but in brief, you've created a file in the "`/tmp`" directory, which is where Linux stores temporary files – files which will be deleted when the system starts up; and then you've used a *pager*, "`less`", to view it.

Linux adopts what are called the "Unix File System" conventions for where it stores files of different sorts. Home directories for users, for instance, are normally stored in a directory called "`/home`", and temporary files, as we've seen, are stored in "`/tmp`".

You can read more about the Unix File System conventions here.