The Basics

Max Carter-Brown
Anglia Ruskin University, Wellcome Sanger Institute
max.carter-brown@aru.ac.uk

Contents

1. Outcomes

By the end of this document you should be able to:

- Open a terminal on your computer
- Understand and navigate the file system on your computer

2. Setup

To make best use of bioinformatics tools we need to set our computers up properly. We need to interact with the computer on a more basic level - we need to use what is called a terminal. A terminal is a text based prompt where a user can input commands and get results back. Simply put, you type text and the results of the command you executed are either written to the screen or to a file.

2.1. Mac

You are in luck for this bit - all you need to do is find your terminal application. On my machine it's here: '/System/Applications/Utilities/Terminal.app'. Or open Spotlight and search there.

2.2. Windows

You need to install Windows Subsystem for Linux (WSL). Note that Windows PowerShell is *not* what we want. PowerShell, although it sounds great, is not widely used at all in bioinformatics and you will find yourself running into issues very quickly if you use it.

If you are on Windows 10 version 2004 or higher, or Windows 11 you will need to install WSL via PowerShell.

Shell 1: Install WSL via PowerShell on a Windows machine

```
wsl --install
```

For more instructions, please see this link for the Linux distribution installation and this link to install Windows Terminal. On there, it will tell you also how to set up Ubuntu (the default Linux distribution) as your default operating system on the terminal.

2.3. Google Chromebooks

You should have something already in-built. Check out this link for more information on installation.

3. The terminal and the filesystem

Go ahead and open up your new terminal. Depending on your machine you will be running a certain kind of shell. The shell is the program you use which interprets the commands you will be writing. It is usually BASH or ZSH, though many shells exist (Figure 1). Your shell will have a bit to the left of a `$` or `>`, which might include the date, and/or your location in your file system. To the right of the `$` / `>` is where you type your commands.

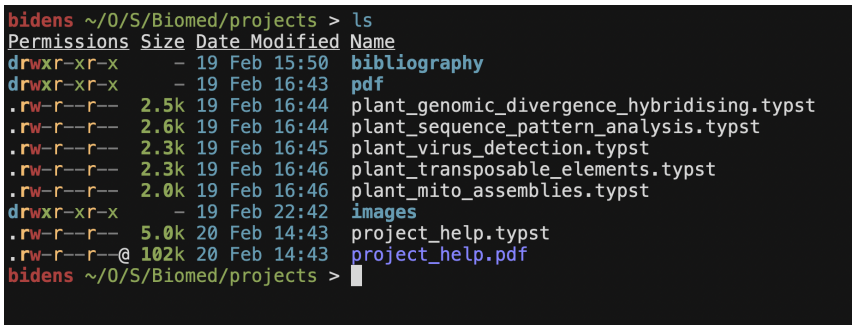

Figure 1: My shell set up. It will be a bit fancier than yours, but it does the same thing.

When you are starting out, it is good to know where you are in your system. By that I mean your file system. When you open up your terminal, it places you inside your file system somewhere (usually your home directory). This is equivalent to you opening up Finder (on a Mac) or File Explorer on Windows. They are visual guides to where all the files on your computer are stored. The terminal reflects this too, but uses text and not fancy graphics. We can work out where we are in our file system by 'Printing the Working Directory'.

Shell 2: Print the working directory by typing this command.

```
pwd
```

Now we know where we are. But what is here? The next command lists files in a directory.

Shell 3: Listing files in the directory you are in.

```
ls
```

This command should output some text which shows you what is in your directory. It will not look like Figure 1, but it should display the same information that is in the 'Name' column.

Shell 4: Adding flags to our `ls` command modifies its behaviour. Hashed lines are comments, not code to be executed.

```
# these are the same
ls -t -h -o -r
# a friend of mine told me about these flags whilst doing my PhD
ls -thor
```

You notice that I added some text there. `-t` sorts output by time. `-h` makes the output human readable (i.e. adds K for kilobytes and M for megabytes etc). `-o` outputs list in long format. `-r` reverse sorts, so coupled with `-t` I am reverse sorting by time - neat right?

`ls` is a program which has the simple task of listing files. It can list files anywhere though, not just where we happen to be. All we have to do is point `ls` to the directory where we want it to list files.

To do this, you have to be familiar with what a file system is a bit more. In Figure 2 I have made up a toy example which starts at the root of our file system and contains different directories.

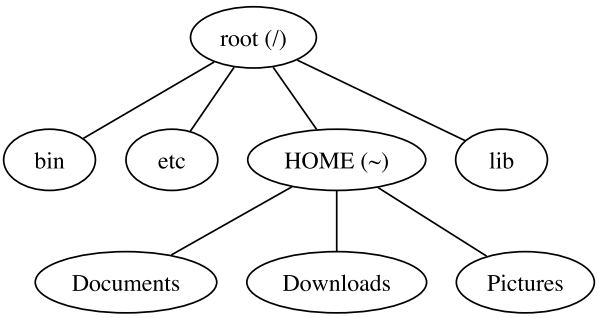

Figure 2: My made up file system. The root (denoted by a forward slash `/`) is the origin of the file system, and it usually has lots of stuff in it we won't worry about. Under the root, we have HOME, which is our home directory (denoted by a tilde `~`). Under HOME are our familiar directories - where my documents live, and where my downloads are.

As mentioned earlier, when we open our terminal we are usually placed in our home directory. But now I want to list the files in the directory 'above' where I am. I want to ascend the file system tree up one level. How do I do this? There are actually two ways in our made up example.

Shell 5: Two ways of doing the same thing. The `..` is a code for 'go up one level'. The `/` is the symbol for root. It so happens that when we go up one level from HOME, we reach the root! So these two lines of code do the same thing here. If you run these on your machines however, you are likely to get a different result as your root is likely to be many levels above your current directory.

```
ls ..
ls /
# note you can also pass the flags like you did before
ls -thor ..
```

I will introduce one more command here, `cd`. `cd` changes the directory we are currently in. We use it if we want to 'physically' travel around the file system. Running `cd` by itself will return you to your home directory (which may not be what you want). Usually we run `cd` with a path - that is, a pathway to where we want to go to.

Shell 6: Changing directory. The first line takes us home. The second line takes us up one level, and the third line takes us in to the Documents folder (provided the Documents folder is in the directory you are currently in!)

```
# change directory
cd
cd ..
cd Documents
```

By using these two commands (`cd` and `ls`) we can explore any part of our file system that we have permission to view.

Shell 7: Exploring our toy file system shown in Figure 2.

```
# this might display:
# Documents  Downloads  Pictures
ls
# go into Documents
cd Documents
# have a look around
ls
# go back to where you were
cd ..
# go up again
cd ..
# have a look around
# this might display:
# bin  etc  HOME  lib
ls
```

4. Over to you

1. List the files in your current directory.
2. List the files in the directory two levels above your current directory. How many files and/or directories are there?
3. Change directory to the root directory and print the working directory using `pwd`. What does it say?