Introduction to Unix shell

- 2015/16 Part III Systems Biology SEB module
- 10 Feb 2016, 10:00-13:00
- Bioinformatics Training Room, Craik-Marshall Building, Downing Site
- Avazeh Ghanbarian, Alexey Morgunov

Introduction

Unix shell is a command line interpreter that provides a user interface for directing the operation of the computer by entering commands as text for a command line interpreter to execute, or by creating text scripts of one or more such commands. In plain English, it is a powerful way of telling your computer what to do. You can read more about the history of Unix shell here http://www.softpanorama.org/People/Shell_giants/introduction.shtml.

Developing skills for coding in any language consists of the following components:

- *Logic* - understanding the syntax, how commands and scripts are structured and how components fit together. This is something one has to learn.
- *Awareness* - knowing what commands, methods and tricks exist and what they can be used for. This is like checking your inventory of LEGO bricks - you need to know what you have in order to start thinking how to put them together to build what you want.
- *Practice* - and a lot of practice. Learning how to combine the bricks together to solve increasingly more complex problems is best achieved through continuous practice.
- *Google and Stack Overflow http://stackoverflow.com/* - what coding really is about. It is likely that unless you are doing something very very novel, someone else has run into the same problem and has a solution. Find it and use it, don't reinvent the wheel. This is an important part of the learning and practice process.

In this tutorial we focus on explaining the *Logic* component and on building some *Awareness* about existing commands and methods in Unix shell. Finally, we give some exercises for *Practice* and leave it up to you to familiarise yourself with how to search for answers if you get stuck.

If you have previous experience with Unix shell. Skip to the *Exercises* section below and try your skills at it.

Basics

The syntax of commands:

```
[command] -[options] [file or folder]
```

N.B. The angular brackets [] do not need to be typed. They are used here as a placeholder of specific type (e.g. a filename).

Very useful starting points:

```
man [command] #manual entry for the command ('q' to exit)
which [command] #locate the program aliased to the command
ls #list files in the directory
ls -l #long information
ls -lh #human readable format
ls -lht #sort by time
ls -A #include hidden files
pwd #print working directory
cd [folder] #change directory into folder
cd ~ #change to home folder
cd .. #move up a directory
```

Working with files and directories:

```
mkdir [name] #create a new directory
cp [file1] [file2] #copy file1 to location file2
cp [file] . #copy file from its location to working directory
mv [file1] [file2] #move file1 to location file2, e.g. rename
rm [file] #delete file
rmdir [directory] #delete directory
```

Careful when using `rm` recursively (`rm -r`) - it is better and safer to use `find` instead, e.g. to remove all files with `.pdf` as their extension in the current working directory: `find . -name '*.pdf' -delete` . (The star in `*.pdf` here means all files that end in `.pdf` .)

Working with text files

Viewing file contents:

```
clear #clear the terminal screen
cat [file] #outputs file contents in terminal window
less [file] #one page at a time, space for next, (q)uit
gedit [file] #open text editor, also 'emacs', 'vi'
head -N [file] #display top N lines of file
tail -N [file] #display bottom N lines of file
wc [file] #word, line, character and byte count
```

License