

Exercises:

Introduction to Unix

Version 2023-01

# Licence

This manual is © 2018-2023, Simon Andrews.

This manual is distributed under the creative commons Attribution-Non-Commercial-Share Alike 2.0 licence. This means that you are free:

* to copy, distribute, display, and perform the work
* to make derivative works

Under the following conditions:

* Attribution. You must give the original author credit.
* Non-Commercial. You may not use this work for commercial purposes.
* Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under a licence identical to this one.

Please note that:

* For any reuse or distribution, you must make clear to others the licence terms of this work.
* Any of these conditions can be waived if you get permission from the copyright holder.
* Nothing in this license impairs or restricts the author's moral rights.

Full details of this licence can be found at

<http://creativecommons.org/licenses/by-nc-sa/2.0/uk/legalcode>

### Exercise 1: Connecting to a Linux Server

* Find the server address, username and password which have been assigned to you
* Connect to the server via the web interface and check you can see your desktop

### Exercise 2: Basic Unix commands

* Run the ls program to see what files and folders are in your home directory
* Run ls -l to get the output in “long” format with the owner, size and file type listed
* The figlet command draws pretty graphical representations of text you supply, something like this:

\_\_\_ \_ \_ \_

|\_ \_| | | \_\_\_\_\_ \_\_\_\_\_ | | (\_)\_ \_\_ \_ \_\_\_ \_\_

| | | |/ \_ \ \ / / \_ \ | | | | '\_ \| | | \ \/ /

| | | | (\_) \ V / \_\_/ | |\_\_\_| | | | | |\_| |> <

|\_\_\_| |\_|\\_\_\_/ \\_/ \\_\_\_| |\_\_\_\_\_|\_|\_| |\_|\\_\_,\_/\_/\\_\

* Read the man page for figlet to work out how to use it (man figlet)
* Get the program to write your name. If you put spaces in your name you’ll need to put your name into quotes.
* Find the correct switch to add to the command to get your name centred in the terminal
* xcowsay is a graphical program which makes a cow say something
* Run xcowsay -t 0 "I am a graphical program"
  + Note that you can’t enter more commands in the terminal until you click on the cow to make it go away
  + Read the man page to find out what the -t 0 means
* Look at the help page for the multiqc program by running multiqc --help Note that there isn’t a man page for this since it isn’t a core piece of software.

### Exercise 3: File system basics

* Check where your working directory currently is by running pwd
* List the files folders in the directory using ls -l
* Use mkdir to create a folder called compare then run ls -l to check that you can see it
* Use cd to move into the seqmonk\_genomes/Saccharomyces cerevisiae directory in your home directory. Make sure you use tab completion to write the folder names.
* Run ls -l to see what folders you can see. Each of these represents a different genome assembly of the worm genome.
* Using ls list the contents of directories containing a 4 in their name (ls \*4\*)
* Use the head command to simultaneously show the first line only of all of the I.dat files in any of the subdirectories (\*/I.dat)
  + Are the chrI sequences all the same length?
* Use cd to move into the EF4 directory, then use less to look at the contents of Mito.dat
  + See if you can find the first rRNA gene (type /rRNA to search in a less session)
  + What is its position?
* Using cp copy Mito.dat into the compare directory in your home directory
  + It will be cp Mito.dat ~/compare/ where the ~ means your home dir
* Use cd to move back to the ~/compare/ directory
  + Use nano to edit the Mito.dat file
  + Change Mito to Mitochondrion in the ID and AC header lines at the top of the file
  + Save the file with Control+o and then exit nano with Control+x
  + Use mv rename the file from Mito.dat to Mitochondrion.txt
* Using ln –s create a symlink from the original Mito.dat file to the same filename in your current directory (the compare directory). Remember to use tab completion to write the folder/file names.

ln -s ../seqmonk\_genomes/Saccharomyces\ cerevisiae/EF4/Mito.dat .

* Run diff Mitochrondrion.txt Mito.dat to see what differences it can find between the two versions of the file.

### Exercise 4: Redirection and Bash Loops

* Go into the FastQ\_Data directory and look at one of the fastq files using less
  + Less is clever enough to realise that the file needs to be decompressed so you can just pass the file to less directly
  + Now validate that one of the files can be successfully decompressed
    - Run zcat on the file, but…
    - Throw away the STDOUT output (using > /dev/null) so that you just see errors or warnings
* Calculate the signatures of all of the fastq files using the sha1sum program (with a number 1 in the middle, not the letter l)
  + Start by running sha1sum on one fastq file to see how it works
  + Now run it on the entire contents of FastQ\_Data using a wildcard \*fastq.gz (rather than a loop)
    - Write the results (STDOUT) to a file in your home directory using >~/signatures.txt
    - Write any errors to a different file in your home directory (2>~/errors.txt)
* Use nohup to run the fastqc program on all of the fastq.gz files (\*fastq.gz)
  + Check the nohup.out file to see that it has finished.
* Once the fastqc jobs have finished, run multiqc . (note the dot to specify it should run in the current directory) to assemble the fastqc output into a single report.

### If you have time

* Write a bash loop which will go through every .dat file in seqmonk\_genomes and will count the number of lines containing rrna (case insensitive). The process will be:
  + Move to the seqmonk\_genomes/Saccharomyces cerevisiae folder
  + Use a shell wildcard which will find all of the .dat files (\*/\*.dat)
  + Write a loop to iterate over these. For each one
    - Use echo to write out the name of the file plus a space (check for how to not include a newline at the end)
    - Use grep to get the lines containing “rrna” (check for case insensitive)
    - Use wc to get and print the number of lines of hit (check how to just get the line count)
    - Run the loop and save the results to a file called rrna\_count.txt
* Convert every fastq.gz file in FastQ\_Data into a fastq.bz2 file
  + Read the file with zcat
  + Pipe it to bzip2 (with the option to write to stdout)
  + Redirect the output to a new file with .bz2 on the end

### 