

# **Exercises: Linux Bootcamp**

## Licence

This manual is © 2018, 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 3: Understanding how PATH works

- Use the `echo` command to show what is in your current `PATH` (remember it needs to be called `$PATH` when you use it)
  - Which directories are being searched?
- Use the `which` command to find where on your file system the `figlet` program is installed.
- Try using `which -a` to see if there are any other programs called `figlet` later in the `PATH`
- We have given you a second `figlet` program which is in `/opt/figlet/figlet`. Run this by providing a direct call to it (giving its full path). Check that it runs OK.
- Modify your `PATH` to add `/opt/figlet/figlet` before the current directories (use the `export PATH=...` function, and make sure the existing `$PATH` is still included)
- Rerun `which figlet` to show that the version in `/opt` is now found first
- Run `figlet` from the command line and see that the `/opt` version is now the default.
- Use nano to edit the end of your `.bashrc` file to permanently add `/opt/figlet` to your `PATH`. Open a new shell to check that this still works.

### Exercise 4: Redirection and Bash Loops

- Go into the `FastQ_Data` directory and look at one of the fastq files using `less`
  - `Less` will not directly read `fastq.gz` files, so you'll need to use `zcat` on the file and then pipe the result to `less`
  - Now validate that one of the files can be successfully decompressed
    - Run `zcat` on the file, but...
    - Throw away the `STDOUT` output so that you just see errors or warnings
- Create a profile of the configuration files in `/etc/` using the `shasum` program
  - Start by running `shasum` on `/etc/profile` to see how it works
  - Now run it on the entire contents of `/etc/` using a wildcard (rather than a loop)
    - Write the results to a file in your home directory
    - Write any errors to a different file in your home directory
    - Have a look at the errors to see why it might have failed in some cases
- 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
  - Work out a shell wildcard which will find all of the `.dat` files
  - 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`
- [If you have time] 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 in a different folder
  - Maybe add an `echo` statement so you can see which file it's processing

## Exercise 5: Installing OS packages using apt

- Use apt to install the `clustalw` multiple alignment tool and the `clustalx` graphical interface
- How many additional packages were needed to satisfy the dependencies for each tool
- Use the `clustalx` tool to align the rRNA sequences in `Align_Data/reference_sequences.txt`
- You can use the `apt-file` program to see which files have been installed by a particular package. Use this to see what the `clustalw` package installed
  - Install `apt-file` with `apt install apt-file` (as root)
  - Build the file cache with `apt-file update`
  - List the files for `clustalw` with `apt-file listmore / clustalw`
  - Look at the directories the files are installed into

## Exercise 6: Binary and script installation

- Install the Blast search tool from NCBI.
  - Find the appropriate distribution file to download from the project web site at <ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/>
  - Use the `md5sum` program to calculate a hash signature for the file you downloaded and compare the answer to the `.md5` file on the NCBI site. If they're the same then the file download was successful
  - Move the extracted files to `/opt/blast/`
  - Add `/opt/blast/bin` to your `PATH`
  - Use `ldd` to see what libraries the `blastn` program links to
  - Go into the `Align_Data` folder and build a blast index of your references sequences using `makeblastdb -dbtype nucl -in reference_sequences.txt`
  - Search your new database with the `test_seq.txt` sequence by using `blastn -db reference_sequences.txt -query test_seq.txt`
  - Which species does the test sequence most likely come from?
- Install FastQC from <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
  - You will need to check that you have java installed before `fastqc` will run
  - Run `java -version` to see if you have it. If not then use apt to install the `default-jre` (java runtime environment)
  - Download the zip file into your home directory and unzip it
  - Move the unzipped data to `/opt/FastQC`
  - Change the permissions on the `/opt/FastQC/fastqc` launch script to be executable
  - Check what interpreter the launch script is using and that it exists.
  - Create a symlink from `/opt/FastQC/fastqc` to `/usr/local/bin/fastqc` so that `fastqc` appears in your `PATH` (`/usr/local/bin` should already be there)
  - Check that the install works by performing a `fastqc` analysis of all of the files in `FastQ_Data`. Run `fastqc --help` if you're not sure how to run the program

## Exercise 7: Installation from source code

- We are going to be compiling from source so we need to install the basic command line toolset. Use `apt` to install the `make` `te` package for the OS which contains these.
- Install `samtools`, a library for manipulating BAM and CRAM mapped sequence files.
  - Go to <http://www.htslib.org/download/> and download the latest sourcecode for `samtools`
  - Extract the data from the file. Note the type of compression the tar file uses and make sure you put the appropriate switches onto your `tar` command
  - Move into the new directory which was created and go through the standard autotools build process. We will install `samtools` into `/opt/samtools`
    - `./configure --prefix=/opt/samtools`
    - `make -j 2`
    - `make test`
    - `make install`
  - Note that the `configure` will fail because of missing dependencies (probably a couple of times!). Each time, read the error, install the missing dependency using `apt` and then run `configure` again until it completes successfully.
  - After the install has completed, add the appropriate directory to your `PATH` so that `samtools` is accessible everywhere
  - Have a look at the results of `ldd` on the `samtools` program binary, can you see the different libraries which you had to install?

## Exercise 8: Installing R packages

- R is not yet installed on these systems. It should be available in the package repository under the name `r-base-core`. Install this package using `apt`
- Compilation of packages requires some additional libraries, so also use `apt` to install `libxml2-dev` and `libcurl4-openssl-dev` and `libssl-dev`
- Install the CRAN `beanplot` package
  - You want to install this for all users, so start an R session as root (using `sudo`)
  - Use `install.packages` to install the package directly from CRAN
  - Check that you can load the library using `library(beanplot)`
  - Check it works by running:
    - `beanplot(rnorm(1000), rnorm(1000)+2)`
- Install the Bioconductor package `GenomeGraphs`.
  - Find the install instructions on the Bioconductor website
  - The install may take a while as there are a large number of dependencies
  - Make sure you can load the library using `library(GenomeGraphs)`
  - Make sure it works by running
    - `mart <- useMart("ensembl", dataset="hsapiens_gene_ensembl")`
    - `gene <- makeGene(id = "ENSG00000095203", type="ensembl_gene_id", biomart = mart)`
    - `gdPlot(gene) asdas]`
- Install the `intensitydiff` package from github
  - Download the latest release as a tar.gz from <https://github.com/s-andrews/intensitydiff/releases/>
  - Install the package (as root) using R CMD INSTALL

## Exercise 9: Installing Perl modules

- Install the `Date::Calc` module using the `cpan` program
- Try to spot the download, compile, test and install phases of the installation
- Find out how many days you've been alive with:
  - `perl -MDate::Calc -e 'print Date::Calc::Delta_Days(1973, 9, 29, 2018, 11, 5)'`
  - Use your own birthday for the first 3 arguments, obviously though
- Manually install the `Digest::SHA1` module
  - Find the package on <http://search.cpan.org>
  - Download the tar.gz file
  - Extract the contents and cd into the newly created directory
  - Go through the standard manual install
    - `perl Makefile.PL`
    - `make`
    - `make test`
    - `sudo make install`
  - Check it worked with
  - `perl -MDigest::SHA1 -e 'print Digest::SHA1::sha1_hex(12345)'`

## Exercise 10: Installing Python3 packages

- Install the `multiqc` python package (and program) using `pip3`
  - Use `apt` to install `python3-pip`
  - Use `pip3` to install `multiqc` use `sudo` so it's available to everyone
  - Run `multiqc` on the `FastQ_Data` folder you ran `fastqc` in before
    - `multiqc .` (you need the dot at the end)
    - Look at the `multiqc_report.html` file which is generated
- Install the `ColourScience` package from <https://github.com/colour-science/colour>
  - Download the latest release tarball
  - Uncompress it and move into the directory
  - Run `python setup.py install` to install it
  - If you're feeling ambitious, try installing it into a non-standard directory

## Exercise 11: Using conda

- Install `miniconda` from <https://conda.io/miniconda.html>
  - Download the `.sh` script
  - Run it
  - Accept the default location so it installs in your home directory
  - Allow it to add `conda` to your `PATH`, and start a new shell so it takes effect
- Add the `bioconda` channels to your `conda` install
  - `conda config --add bioconda`
  - `conda config --add conda-forge`
- Install the `circos` package into a new environment called `course`
  - What other packages are pulled in?
  - Can you start the environment and get `circos --version` to run
  - What do you get when you run `which perl` ?

- Run `source deactivate` and try `which perl` again

## Exercise 12: Using singularity

- Install the singularity-container package using apt
  - Check the version with `singularity --version`
- Go to <https://singularity-hub.org> and do a search for hello-world (the one by vsoch). Find the address for the container
- Pull the container down onto your machine, calling it `hello-world.simg`
- Start a shell inside the container
- List the contents of your home directory in the shell
  - Can you see all of your files?
- List the contents of `/opt` in the shell
  - Can you see the software you installed in there before?
- Look at the contents of `/singularity` in the container, Work out what it does?
- Run the container as an application (exit the shell and run `./hello-world.simg`) do you get what you expected?



## [Optional] Exercise 15: Troubleshooting

- You have been given 4 programs called `broken1`, `broken2`, `broken3` and `broken4`. All of which are in `/usr/local/bin/`. None of them currently run.
- Alter the programs or system so that you can run all of them by just calling their name

## Exercise 16: Installing Linux in a VM

- You have been provided with two ISO images
  - An Ubuntu live ISO
  - A CentOS install ISO
- Pick one of these and install it into a VirtualBox VM
  - Set up a new VM
  - Give it some memory (2GB) and disk (20GB)
  - Add the ISO to the virtual DVD drive
  - Start the image and step through the installation
- Once your new image is running see if you can replicate some of the exercises you ran before in your own VM instance.

## [Optional] Exercise 17: Creating a cloud Linux instance

- Create an amazon AWS login for yourself if you don't have one already
- Go through the process of creating and connecting to an EC2 linux instance
  - Create a key pair and download the `.pem` file for it
  - Create an EC2 instance and launch it
  - Use the key file to connect to your instance
  - Try out some of the previous exercises to show that they work in this environment