Intersect Australia Ltd
PO Box H58, Australia Square
Sydney NSW 1215

T +61 2 8079 2500
F +61 2 9262 3040

ABN 67 131 752 657

Introduction to UNIX for HPC

Exercises

Joe Thurbon | Luc Small | Joachim Mai | Jeff Christiansen |

24 May 2013 | 3.0

Shell Primer

Unit 1: Run Spot, Run

Exercise 1(a) – Running commands

The purpose of this exercise is to demonstrate how BASH parses (interprets) a command line and runs commands based on that.

`regurgitator.sh` is a command that tells you about what its arguments are. It was written specifically for this course to show how BASH interprets commands and augments.

Run these commands sequentially:

1. **`regurgitator.sh`**
2. **`regurgitator.sh here are four arguments`**
3. **`regurgitator.sh My forename is`** *`YourFirstName`*
4. **`regurgitator.sh My name is`** *`"YourFirstName YourLastName"`*

Now try out a few standard UNIX commands. Some of these commands also have arguments which will cause a change in behaviour. Run these commands and try to figure out what the commands do:

- **`ls`**
- **`w`**
- **`w <buddy_login_id>`**
  - **`buddy_login_id should be the login of your buddy, e.g. hpc22)`**

- **`finger`**
- **`finger <buddy_login_id>`**
- **`date`**
- **`uptime`**

Exercise 1(b) – Flags and Parameters

Run the command:

- **`regurgitator.sh –d three arguments`**

Run the command:

- **`regurgitator.sh –u here are "four arguments"`**

Run the command:

- **`regurgitator.sh –n 3 here are five arguments`**

Now we'll examine the contents of the training directory. Try out a few standard UNIX **commands with flags** and compare the difference between the outputs:

- **`ls`**
- **`ls –l`**
- **`ls –a`**
- **`ls --all`**
- **`ls –la`**
- **`ls --all –l`**

Now try out a few **commands line parameters with flags** and compare the difference between the outputs:

- **`ls --format=horizontal`**
- **`ls --format=single-column`**

Exercise 1(c) – The Calendar

Using the online manual (man), work out how to use the **`cal`** command to print the calendar for the month containing your birthday.

Unit 2: Where am I?

Exercise 2(a): Finding your way around

The purpose of this exercise is to gain familiarity with the basic commands used to interrogate, navigate and manipulate directories.

(1) Examine the contents of your home directory. You should see, at least, the **`training`** directory. Using the **`man`** command, determine how to make **`ls`** give you more information about each entry. (NOTE: you should see the **`regurgitator.sh`** script).

(2) Examine the contents of the training directory without changing your working directory.

(3) Change your working directory to the training directory.

(4) Examine the contents of your home directory. You should now be able to do this in three different ways (HINT: Absolute path from root, my home directory, home directory of a specified user)

Exercise 2(b): Finding your way around (part 2)

(1) Run the **regurgitator.sh** command. Note that if you just type in **`regurgitator.sh`** it won't work. You will have to 'find it'. (HINT: "`..`" is the parent of a directory).

You can use the **`-d`** option to tell you the working directory (NOTE: The working directory is not the same as the path that was used to run the program).

(2) Use the UNIX standard command to find out what your current working directory is

(3) Examine the contents of the current directory using the "working directory" option

(4) Examine the contents of your buddy's home directory using the "parent directory" option

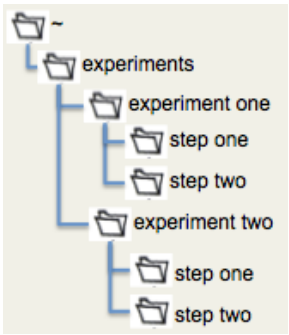
(5) Use `ls` to see all of the files in all of the directories in your home directory. (HINT: `man ls`, you're interested in the recursive option.)

Exercise 2(c): Making Directories

The purpose of this exercise is to learn how to create directories.

(1) Change the working directory to your home directory

(2) Re-create the directory tree as shown below.



(3) Use 'ls' or 'tree' to verify that you have created the correct directory structure (should only require one command)

Exercise 2(d): Moving and Copying Files

The purpose of this exercise is to practice copying files.

(1) Copy the `regurgitator.sh` from your home directory into each of the directories created in the last exercise. For each directory, try to specify the path to the `regurgitator.sh` file differently (e.g. absolute path, relative path, .., ~, with the current working directory as the home, etc.)

(2) Make a copy of the **`regurgitator.sh`** script in your home directory and call it **`copy_of_regurgitator.sh`**

(3) Move the file **`copy_of_regurgitator.sh`** to the experiments folder. Examine the experiments folder to ensure it has been moved there. Examine the home directory to ensure that the file **`copy_of_regurgitator.sh`** no longer remains there

Unit 3: It's what's inside that counts

Exercise 3(a)

(1) Go to the "training" directory in your home directory

(2) Look at the contents of poem.txt (HINT: **`man cat`**)

(3) How many words are in the poem? (HINT: **`man wc`**)

(4) Print the contents of poem twice in one command

(5) Have a look at the contents of long_poem.txt (HINT: **`man less`**)

(6) Look at just the end of long_poem.txt (HINT: **`man tail`**)

(7) Look at the file with the non-printable characters on it. (HINT: **`man cat`**). Can you work out what they are?

(8) How much space is taken up by all the files that you are looking at?

(9) How much space is taken up by each all the files that you are looking at?

Unit 4: It's all just files

Exercise 4(a)

(1) Go to the "training" directory in your home directory

(2) Have your fortune told (HINT: **`fortune`**). Do it again and see a new fortune

(3) Have your fortune told, and save (*HINT: send the output to a new file*) it to a file called **`my_fortune.txt`**. Look at the file.

(4) Do (3) again. Observe the contents have changed.

(5) Have your fortune told, and save it to the same file, but don't lose the first one (*HINT: append the output to an existing file*)

(6) Take the long poem and remove all the redundant lines **and save to a new file called copy_long_poem.txt (i.e. do not overwrite the existing long_poem.txt)**. Look at the file afterwards to ensure all redundant lines have been removed.

(HINT: you'll need to pipe two commands together and redirect the output to the new file)

Unit 5: Products of our environment

Exercise 5(a): Checking the environment variables

(1) Work out the values for some useful variables "PATH", "HOME", "TEMP", "PS1"

Do you recognise that last one?

(2) At the command line, view the environment

(HINT, if it is too big to see all of it, remember that the "|" command will send the output of **`env`** to another command, and that **`less`** will paginate output)

Take a look at the following environment settings and try to work out what they are for:

`"HOST", "SHELL", "USER", "CPU", "LANG", "OSTYPE", "MACHTYPE"`

(3) View the environment, search for any "**`PATH`**" settings and send the output to a file called **`path.txt`**

(HINT: This will require 2 commands (man "**`grep`**") and redirection of the output to a file)

Exercise 5(b): Manipulating the Environment

(1) Set a variable **`MYNAME`** to your first name, and view the contents.

(2) Set the variable **`MYNAME`** to your last name, and view the contents.

(3) Without typing your last name in again, set your variable to your first name followed by your last name, separated by a ":" sign (HINT: **`MYNAME`** will appear on both sides of the = sign)

(4) Change the working directory to your home directory. Move the **`regurgitator.sh`** file from that directory to the **`bin`** directory. Type **`regurgitator.sh`** from the current directory (NOTE: should fail as no such file is found)

(5) Change the path so that you can run the **`regurgitator.sh`** from that directory, without specifying a qualified path to it. (HINT: set the **`PATH`** variable, but recall the lesson of (3))

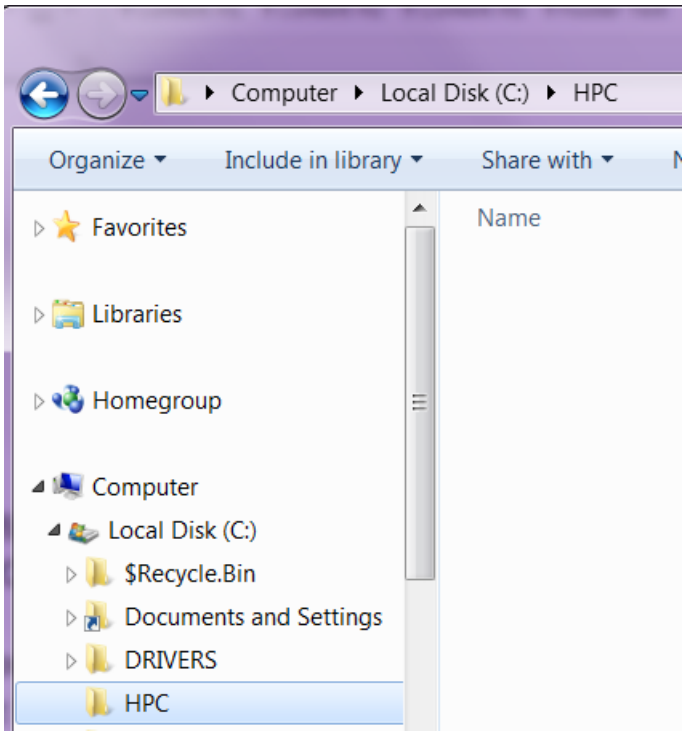
(6) Change your prompt to whatever you want! (HINT: set the **`PS1`** variable, man **`PS1`** for details)

Data Module

Unit 1: Making a move (using FTP)

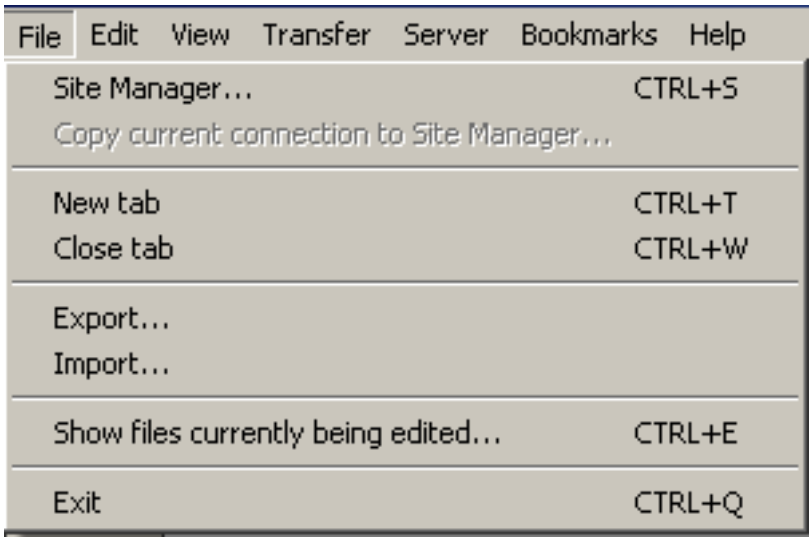
Exercise 1(a) – Connecting to the HPC machine

(1) Firstly create a folder on your local machine where you can store the files that you will later copy from Octane. To do this, open Windows Explorer, and navigate to the C: disk. Then click on "New Folder" and create a new folder called "HPC" (**`New directory name=C:\HPC`**). This directory will be empty.



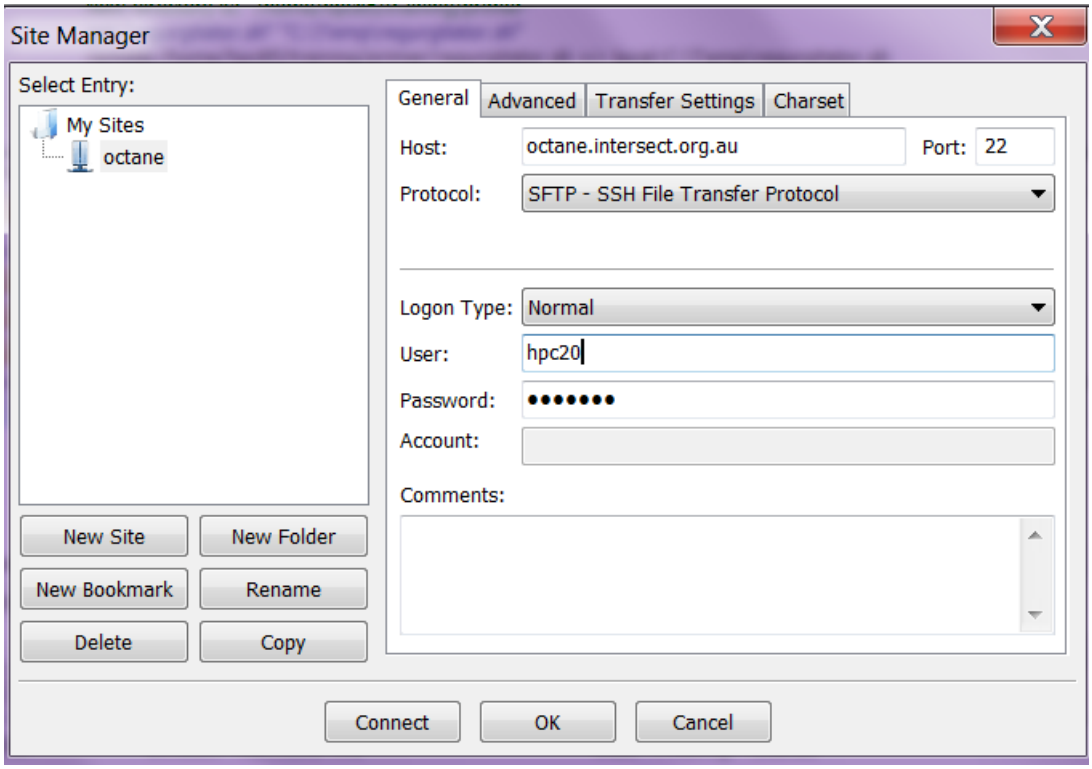
(2) Download *FileZilla*, if it isn't preinstalled on your system. To do so, browse to **`http://filezilla-project.org/download.php?type=client`** and click **`FileZilla_n.n.n_win32-setup.exe`**. Double click the file once it has downloaded and follow the installation procedure.

(3) Open FileZilla by selecting *Start Menu > FileZilla FTP Client > FileZilla*.



(4) Go to the *File > Site Manager* menu.

(5) Click *New Site*. Type in **`Octane`** as the new site name.
(6) Type **`octane.intersect.org.au`** as the *Host*.
(7) Type **`22`** as the *Port*
(8) Select the **`SFTP - SSH File Transfer Protocol`** as the *Protocol*.
(9) Select **`Normal`** as the *Logon Type*.
(10) Type your username under *User.* The dialog box should look like this:

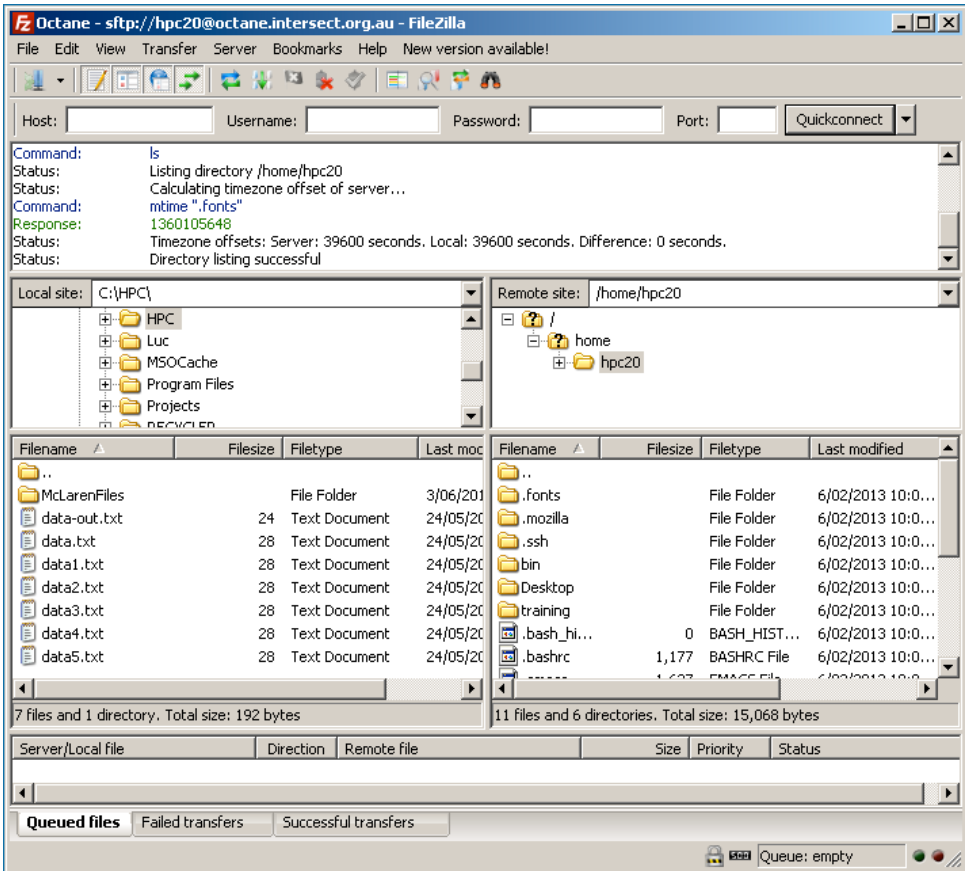


(11) Click *OK* to close off the dialog box.

(12) Select *Octane* from the Site Manager icon (the left-most icon on the top toolbar). This will initiate a connection to the HPC machine.

(13) As this is your first time connecting to *Octane,* you will be presented with a dialog box asking you to accept the security key. Click OK to accept

(14) Once authenticated, the *FileZilla* window will look like this:



(15) The left-hand pane represents the file system on your local machine. Navigate to the directory you created earlier **`C:\HPC`**

(16) The right-hand pane represents the file system on the remote HPC machine (**`octane.intersect.org.au`**). When you connect, by default, the right-hand pane will show the contents of your home directory on the HPC machine. You will recognise some of the files you modified or created earlier in the exercises here

(17) To transfer files between your local machine and the HPC machine you can simply drag-and-drop between the two panes.

Exercise 1(b) – Creating and Transferring files

(1) Create a file in Notepad and save to **`C:\HPC\data.txt`** and type the following 5 lines into it:

```
1, 4

2, 3

3, 4

4, 4

5, 2
```

We'll pretend this file is your dataset.

(2) Transfer **`C:\HPC\data.txt`** to your home directory on the HPC machine. To do this, switch to FileZilla and browse to **`C:\HPC`** in the left-pane. The right-pane should already be in your home directory.

(3) Drag **`data.txt`** from the left-pane to the right-pane.

(4) SSH to the HPC machine using *Putty* (see the previous modules for guidance).

(5) At the command prompt type:

```
file data.txt
```

(6) The following text will be displayed:

```
data.txt: ASCII text, with CRLF line terminators
```

(7) The **`file`** command returns information about the file. The first part, "**`ASCII text`**", tells us this file is a text file. The latter part "**`with CRLF line terminators`**" tells us this file is in DOS/Windows format. In order to use it with the HPC machine, a UNIX machine, we'll need to convert it to use the right line terminators.

(8) At the command prompt type:

```
dos2unix data.txt
```

(9) The file will now be in UNIX format. Now try repeating this command:

```
file data.txt
```

(10) This command will return the following text. The absence of detail about line terminators indicates that it has UNIX line endings:

```
data.txt: ASCII text
```

(11) At the command prompt type:

```
cat data.txt > data-out.txt
```

(12) Now return to *FileZilla* and right click on the right pane. From the context menu, select *Refresh*. The listing of files will be refreshed from the HPC machine and consequently, you'll see **`data-out.txt`** in the file listing.
(13) Copy **`data-out.txt`** to your local **`C:\HPC`** folder by dragging from the right pane to the left.
(14) Switch to *Windows Explorer* and browse to **`C:\HPC`**.
(15) Open **`data-out.txt`** in Notepad. You'll note that the text is on the one line because we forgot to convert **`data-out.txt`** into DOS/Windows format.
(16) Switch back to your SSH session.
(17) At the command prompt, type:

```
recode latin1..dos data-out.txt
```

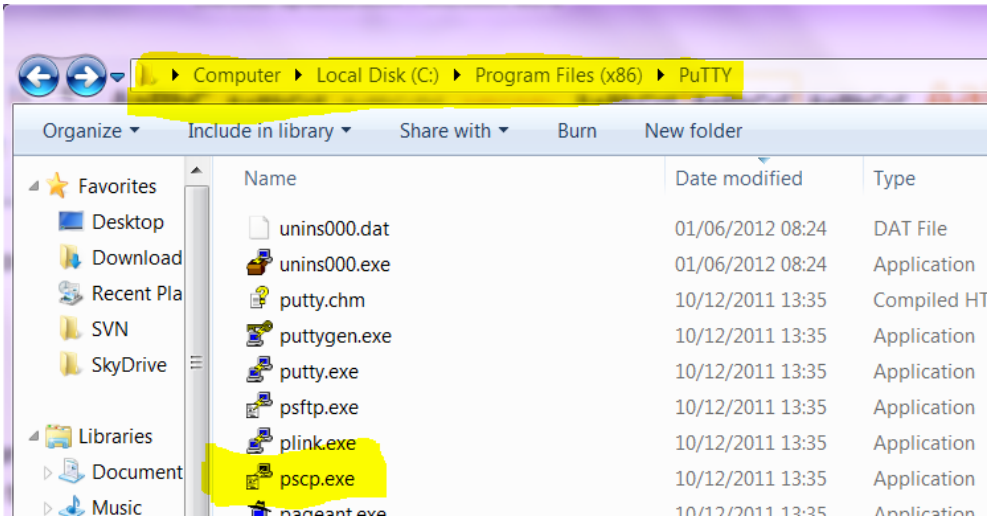
(18) In *FileZilla* copy across **`data-out.txt`** to your local **`C:\HPC`** folder once more.
(19) Open **`C:\HPC\data-out.txt`** in Notepad once more. This time the text should span multiple lines.

Unit 2: The Second Date (using SCP)

Exercise 2(a) – Transferring files from your local machine to the HPC machine using PSCP

(1) Using Windows Explorer, check where Putty and PSCP (Putty Secure Copy) have been installed on your local machine. Typically they will have been installed into a location similar to the two listed below. The location depends on where your University's system administrator has installed them:

a. **`C:\Program Files (x86)\Putty\`**
b. **`C:\Program Files\Putty\`**



(2) Open a Windows Command Prompt by locating *Run* in the Start Menu, then typing **`cmd`** into the resulting dialog box and clicking *OK*.

(3) Check that folder that contains the PSCP program is already included in your PATH by typing the following command:

```
echo %PATH%
```

(4) If the path that contains the PSCP program is not included in the PATH, then add that folder to your PATH by typing the following command:

```
set PATH=%PATH%;"C:\Program Files (x86)\Putty"

* The exact location of the path should match the path as per instruction (1) above
```

(5) Create 5 dataset files in Notepad, all containing the same text:

```
1, 4

2, 3

3, 4

4, 4

5, 2
```

(6) Save the files into the **`C:\HPC`** folder, naming them: **`data1.txt`**, **`data2.txt`**, **`data3.txt`**, **`data4.txt`**, and **`data5.txt`**.

(7) Return to the Windows Command Prompt
(8) Change to the **C:\HPC** directory:

```
cd C:\HPC
```

(9) Transfer the files to your home directory on the HPC machine:

```
pscp *.txt <user_name>@octane.intersect.org.au:/home/<user_name>
```

(10) Enter your password when prompted and wait for **pscp** to finish running.
(11) Switch to a *Putty* session to the HPC machine, or start a new one.
(12) Verify that the files are on the HPC machine:

```
ls -ls ~
```

Exercise 2(b) – Transferring files from the HPC machine to your local machine using PSCP

(1) Open a *Windows Command Prompt* by locating *Run* in the Start Menu, then typing **cmd** into the resulting dialog box and clicking *OK*.
(2) Ensure that PSCP is in your PATH by typing the following command:

```
set PATH="C:\Program Files\Putty";%PATH%
```

(3) Change to the **C:\HPC** directory:

```
cd C:\HPC
```

(4) Make a destination directory **HPCFiles** in the **C:\HPC** directory:

```
mkdir HPCFiles
```

(5) Change to the **C:\HPC\HPCFiles** directory:

```
cd HPCFiles
```

(6) To transfer the files from the HPC machine to your local machine type:

```
pscp <user_name>@octane.intersect.org.au:/home/<user_name>/*.txt .
```

(7) Enter your password when prompted and wait for **pscp** to finish running.
(8) Use the Windows list directory command, **dir**, to verify that the files have been downloaded:

```
dir
```

Unit 3: (W)get it on! Using Wget

Exercise 3(a) – Downloading and unzipping a dataset on the HPC Machine using WGET

(1) In a web browser, browse to http://data.gov.au/dataset/national-public-toilet-map/

(2) Scroll down 1/3 of the page and locate the **Download** section.

(3) Rather than clicking on the **`Toiletmap.zip`**, right-click on it and select *Copy Link Address / Copy Shortcut*. (The name of this menu item is browser specific; the objective is to copy the URL to the dataset to the clipboard.)

   a. The URL will have a format similar to this: http://data.gov.au/bye?http://raw.data.gov.au/610/Toiletmap.zip

   b. Ignore the part of the URL up to and including the ? mark, and only use the following part, i.e. http://raw.data.gov.au/610/Toiletmap.zip

(4) Switch to your *Putty/SSH* session on the HPC machine

(5) Change directories so that you are in your home directory (e.g. **`cd ~`**)

(6) Enter the following command:

```
wget <pasted URL to dataset>
```

e.g. **`wget http://raw.data.gov.au/610/Toiletmap.zip`**

(7) **`wget`** will download the file, should reach 100% complete status and then finish

(8) List the current working directory to verify that the file (e.g. **`Toiletmap.zip`**) has downloaded:

```
ls -ls
```

(9) Create a directory where you will extract (unzip) the contents of the ZIP file to

```
mkdir Toiletmap
```

(10) The file has a **zip** extension, so well need to unzip it to make use of its contents. Unzip the file to the Toiletmap folder to check out the contents of the ZIP file:

```
unzip Toiletmap.zip -d ~/Toiletmap
```

(11) If you list the current working directory again, you'll find there's a new folder **Toiletmap/** containing the uncompressed files:

```
ls -ls
```

(12) Use the disk usage command (**du**) to get an idea of the size of the original ZIP file, and the extracted files (dataset) you are working with. Notice the difference between the size of the ZIP file (**Toiletmap.zip**) and the extracted files in the **Toiletmap** directory:

```
du -hs Toiletmap*
```

(13) Change to that directory:

```
cd Toiletmap
```

(14) Use the disk usage command (**du**) to get an idea of the size of the individual files you are working with:

```
du -hs *
```

Unit 4: Things need to change, and quick! (Editing files in place)

Exercise 4(a) – Editing a file in-situ

Please take the following steps to complete the unit exercise:

(1) Switch to a *Putty* session to the HPC machine, or start a new one.
(2) At the command prompt, get a directory listing of all of the data files you created

```
ls data*.*
```

(3) Choose which data files you would like to edit (e.g. **`data1.txt`**). At the command prompt, type:

```
nano data1.txt
```

(4) This will open the file you created in Exercise 1(b) of the Data Section.
(5) Remove the last line of the file. Use the arrow keys to navigate and the backspace key to delete characters.
(6) Add a line at the top of the file, consisting of the word "Dataset".
(7) Save the file by hitting **Control**+**O** (this is "oh" and not zero!).
(8) Exit **`nano`** by hitting **Control**+**X**.
(9) At the command prompt type:

```
cat data.txt
```

(10) Verify that your changes have taken effect.

Unit 5: What's mine is yours (Permissions and Ownership)

Exercise 5(a) – Changing ownership and file permissions

(1) Switch to a *Putty* session to the HPC machine, or start a new one.
(2) On the command line type the following to change to the temporary directory:

```
cd /tmp
```

(3) Use **`nano`** to create a text file containing a secret message in this temporary directory. Name the file *<your name>*`_secret.txt`.
(4) On the command line type:

```
ls -ls <your name>_secret.txt
```

(5) Confirm that the file is "world readable" by looking at the file listing.
(6) Have a buddy type the following at his/her command prompt:

```
cat /tmp/<your name>_secret.txt
```

(7) He/she will be able to see the contents of the file because it is world readable.
(8) At the command prompt type:

chmod g-r,o-r /tmp/<*your name*>**_secret.txt**

(9) This will remove the read access permission for the group and the world. You can check that this has taken effect by repeating the **ls -l** command as above (Step 4).
(10) Have a buddy type the following at his/her command prompt once more:

cat /tmp/<*your name*>**_secret.txt**

(11) He/she will **not** be able to see the contents of the file this time.

Exercise 5(b) – Create your first script, make it executable and run it!

(1) Switch to a *Putty* session to the HPC machine, or start a new one.
(2) Navigate to your home directory (**cd ~**)
(3) Use **nano** to create a text file in your home directory. Name the file **my_script.sh** and type in the following two lines:

```
#!/bin/bash

echo "Yippee!"
```

(4) Save the file and exit **nano**.
(5) Check the permissions in the file by doing **ls -l**
(6) Attempt to run **my_script.sh** by typing the following:

my_script.sh

(7) It won't run because execute access has not been granted.
(8) At the command prompt type:

chmod u+x my_script.sh

(9) Again check the permissions in the file by doing **ls -l**. Notice what has changed after using the **chmod** command!
(10) Now attempt to run the script again — it should work without issue.

Intermediate HPC

Please take the following steps to complete the unit exercise:

Exercise 1 – Monitoring the queue with qstat

(1) List all jobs using **qstat -a**

(2) Choose a job from the list, not its job number, and get full details about the job using **qstat -f *<job-number>***

(3) Select a user and list all his/her jobs using **qstat -u *<user-name>***

Exercise 2 – Submit a very simple job script with qsub

(1) Create a job script in your home directory using **nano**. Request a small amount of resources (walltime, ncpus etc). The job should print the date then sleep for 60 seconds then print the date again. (See below for a sample script to adapt).

(2) Queue the job using **qsub *<job-script.pbs>***. Note the job number.

(3) Verify the job is queued using **qstat**.

(4) Once the job has run, files will be created containing any text sent to STDOUT and STDERR by the job. These will take the form ***<job-script>*.o*<job-number>*** and ***<job-script>*.e*<job-number>***, respectively. Take a look at these files using **cat**, **less**, or **nano**.

Sample script

You can use this script as a basis for completing the exercise. Down the track you can adapt it to run your HPC jobs.

```
#!/bin/bash
# Request resources
# * 10 minutes wall time to run
#PBS -l walltime=00:10:00
# * 1 node, 1 processor
#PBS -l nodes=1:ppn=1
# * 100 megabytes physical memory allocated to job
#PBS -l mem=100mb
# Specify a project code (for accounting)
#PBS -P a40
# Set email address
#PBS -M fred@intersect.org.au
# Send an email when jobs
# begins (b), gets aborted (a)
# and ends (e)
#PBS -m abe
# Move to directory job was submitted in
cd $PBS_O_WORKDIR
# Specify the job to be done
date
sleep 60
date
```

Exercise 3 – Putting it all together

In this exercise we'll align 70 similar but different short strings of text using a software package called ClustalW2, which was designed for aligning amino acid sequences in proteins. The 22 different amino acids that make up proteins can be coded with a one-letter code, so we have a 22 letter alphabet to play with in this exercise. The program will align the 70 strings of letters and introduce gaps wherever necessary to produce an alignment across the whole set.

ClustalW2 is installed on octane as a module so we'll write a PBS job script drawing on all the UNIX skills you've learnt over the past day and a half. Prior to that though, to provide an idea of the outputs expected, we can also run ClustalW2 using a graphical user interface provided by the European Bioinformatics Institute (EBI) in Cambridge UK.

(1) Using the Windows browser, make the following directory: C:\HPC\clustal

(2) Download the following file that contains the 70 sequences (in FASTA format which is required for ClustalW2): https://raw.github.com/IntersectAustralia/TrainingMaterials/master/IntroToUnixHPC/70_text_strings.txt and save these to C:\HPC\clustal

(3) Visit the web interface for ClustalW2 at the EBI: http://www.ebi.ac.uk/Tools/msa/clustalw2/

(4) Follow the instructions on-screen:

   a. Upload the input file **`70_text_strings.txt`** (or copy and paste into the input sequence box)

   b. Click on the 'Submit" button

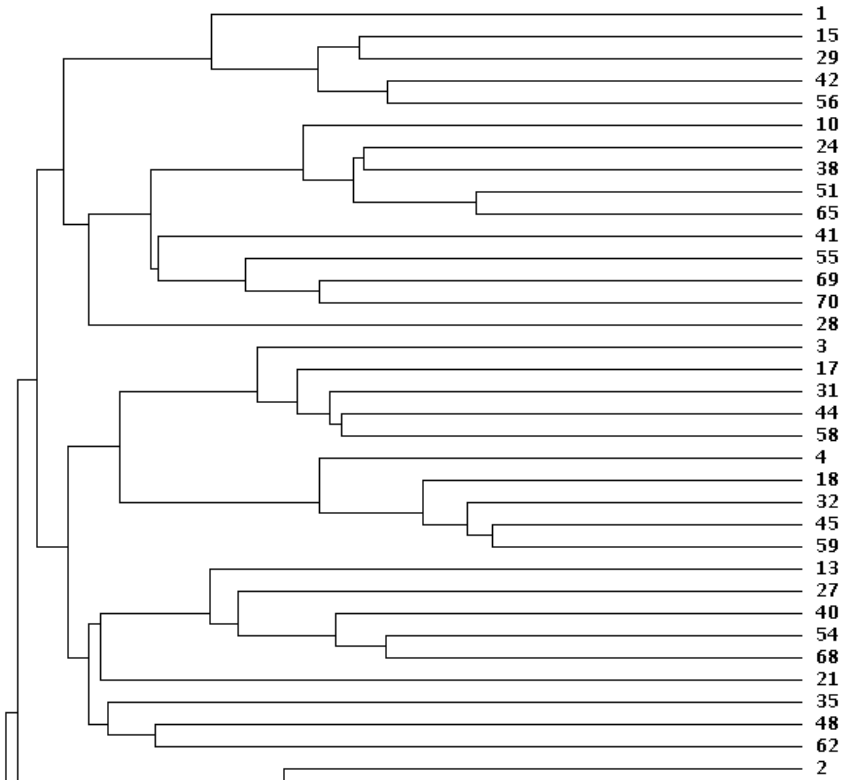
(5) The results are returned showing the sequences aligned (and gaps introduced to force the alignment) such as:

```
CLUSTAL 2.1 multiple sequence alignment

15              -BAKINGMADADDE-BRAI-----NEEDIANVICARWITNESSESP--ETEREST----- 46
```

```
29              -BAKINGMADA---------------EDIANVICARWITNESSESP--ETERES------ 36
42              -BAKINGMADA---------------EDIANVICARWITNESSE------ERES------ 32
56              -BAKINGMADA---------------EDIANVICARWITNESS----------------- 27
1               -BAKINGMADADDEBRAIN-----EDEDIANVICARWITNESSESP-RETEREST----- 48
24              BARKINGMAD--DLEBRAI--NE---RDIANVIWITN---ESSESP-TERRESTR----- 44
38              BARKINGMAD--DLEBRAI--NE---RDIANVIWITN---ESSESP------STR----- 39
51              BARKI-------DLEBRAI--NE---RDIANVIWITN---ESSESP------STR----- 34
65              BARKI-------DLE-----------RDIANVIWITN---ESSESP------STR----- 28
10              BARKINGMAD--DLEBRAI--NEDWARDIANVIWITN---ESSESP-TERRESTR----- 47
69              BARKINGMADADDLEBRAINEDEDWARDIANVICA-------SESP-RE--TSTR----- 45
70              -ARKINGMAD--DLEBRAI--DEDWA--IANVICA-------SESP-R----STR----- 36
55              BARKINGMADADDLEBRAINEDEDWARDIANVICARWITNESSESP-RE--TSTR----- 52
```

(6) You can also click on the 'Guide Tree' option to see on a tree structure how related the 70 different sequences are to each other. In the part of the tree shown, sequences 45 and 59 are the most related (and then 32 is the most related to both 45 and 59) etc:



Now we will perform the same task using HPC. This will require:

1. getting the 70 input data sequences onto octane,
2. submitting a ClustalW2 job into the HPC queue using this input data,
3. writing the output data to somewhere in your area on octane, and then
4. getting the output data off octane.

Note that there are numerous approaches to tackle the problem, and no one right answer, and you'll need to draw on the UNIX skills you've learnt over the past days to achieve this.

Note also that ClustalW2 has been installed on octane as a module called **`clustalw`**. To get a module to work on octane in a PBS script, the following commands can be used:

`source /usr/share/modules/init/bash`
(this initialises the module's environment. Note that if you are going to be a user of orange we would normally have a ~/.bashrc script for you that is executed at log-in and includes this so you wouldn't have to run/include this)

`module load` `<module_name>`
(this command is used to load a specific module)

ClustalW2 also has a specific command line syntax of its own, so for this example, use the following line in your script to start it, to specify the input text file, and to specify values for the PWMATRIX and PWGAPOPEN parameters:

```
clustalw2 -INFILE=<input_file_name> -PWMATRIX=BLOSUM -PWGAPOPEN=1000
```

After writing your script and submitting your job to octane, you should end up with some output files with the following extensions:

`.aln` – this is a text file and can be viewed in the normal way you view .txt files using UNIX (hint: `more, cat`)

`.dnd` – this is a file to read into an interactive tree viewing program such as MEGA http://www.megasoftware.net (hint: to do this, you'll need to get the .dnd file off octane and onto your own machine first, and also install MEGA on your local machine in order to view the tree).