UNIX/Linux Practical 1
Introduction

This session is designed to introduce you to the underlying UNIX based filesystem and computing infrastructure used within the School of GeoSciences.

1. What? Why?
- Introducing Linux – a sophisticated UNIX operating system for data analysis and processing.

2. Connecting to Linux in GeoSciences from Windows Labs
- Connecting to the School's XRDP service

3. Files and Directories
- Listing, viewing, copying, making. How your workspace is structured.

4. Help!
- How to get it. Is it helpful? Other places to look.

5. File Utilities (Big Data Downloads)
- Useful tools that use the command line - **`tar`**, **`gzip`**, **`ftp`**.

6. Further File Utilities (Zip Files, Automated Download, Web PDFs)
- More useful tools for file-handling and working remotely - **`zip`**/**`unzip`**, **`curl`**, **`wget`**, **`pdftk`**.

7. Resource Monitoring Utilities
- Finding out how much disk space you have. Finding other users.

8. Searching
- Finding files. Finding files with a specific bit of text in them.

9. Full Applications
- What are available? How do I start them? Transferring machines: **`ssh`**

10. Log Out/Off or Close Session?
- Full logout versus leaving a session suspended in memory to return to later (from anywhere!)

This practical will allow you to familiarise yourself with the basics of working with UNIX (Linux) in the School of GeoSciences. The practical workbook is designed for you to work through during the practical session in the lab, when there are people around to help. **It is NOT a definitive reference!** Much more information is available on the web, including that given at:

`http://www.geos.ed.ac.uk/~gisteac/wkzero`

There is a set of attainment targets on the next page. Make sure you can do everything in the list. Many students **will** be using UNIX throughout the year. It is therefore enormously important that you speak to one of the demonstrators if you are confused or have been unable to complete the tasks.

Making sense of this Workbook

Throughout this practical workbook any commands you have to type, will be given in **`bold courier font`**, and prompts or responses from the computer will be in `plain courier font`, for example:

```
[snnnnnnn@server ~]$ pwd
[snnnnnnn@server ~]$ cd /
[snnnnnnn@server /]$ ls -al
```

Other text to note, or buttons to click or be aware of, will be introduced in bold paragraph **body** text.

s*nnnnnnn* represents your unique login (e.g. matriculation number preceded by an **s**.)

NB For staff this may be a first initial immediately proceeded by surname up to 8 characters (e.g. jjohnson, jjohnso1, etc.) and for visitors with an initial v*n*fsurna (e.g. v1jjohns).

~ (known as **tilde**) represents your home directory. In the command prompt above note that this will change to reflect whichever directory you are currently working in.

Note that UNIX is **case sensitive** - all commands must be typed **exactly** as they are shown on this sheet or they will not work!

Tasks for you to do will have a letter in the left hand margin - a), b), c) etc.

Targets for this Session

You should be able to:

- Connect to the Geosciences XRDP service from a Windows lab PC
- Know where to find information regarding connecting from home/other platforms
- Locate your home directory
- Change directories
- Create directories
- View simple text files; use simple file viewers (pagers)
- Know how to find quick help on commands/programs
- Download and retrieve data via FTP or similar mechanisms
- Monitor disk space used, and find out about other users on the system
- Search for (and through) files; simple pattern matching
- Start full Linux applications and understand about different types of application
- Connect to different machines using ssh
- Log out of a Linux session and out of XRDP

And you should know:

- How your directory system is structured and the notation used
- Where to go for help on UNIX/Linux (real people and the web!)
- Why you would use UNIX/Linux
- When to switch off

1 What? Why?

Operating Systems – An Introduction

A computer's operating system is a set of programs which provide control over basic computer functions and operations – hence the name. The applications used by human operators are built upon the operating system and released in versions compatible with particular operating systems.

Most University computers run a version of Microsoft's Windows operating system. For many users and tasks this is perfectly adequate however for some tasks, or for advanced users, a system which provides much greater control – and which provides this relatively easily, is required.

For more powerful computing the School therefore provides a selection of machines running UNIX based operating systems. UNIX systems power most internet web servers as well as mission-critical infrastructure and other systems.

The most common way of accessing a UNIX system is by connecting from a Windows lab PC however there are dedicated UNIX or dual (Windows/UNIX) desktop workstations within the School, and also Apple Mac computers (which these days also run a version of UNIX) in selected locations across the University. It is easy to connect to the School UNIX servers from any of these systems.

Introducing Linux

The specific flavour of UNIX used in GeoSciences is Scientific Linux (which, for those who need to know about such things, is based on the Red Hat *style* of Linux distribution.)

What does Linux offer?

Primarily UNIX/Linux offers a powerful command line environment which allows rapid instruction and repeated commands rather than tediously pointing and clicking many times; these can also be easily documented for instructing other future users. Graphical applications are also available but the command line offers much greater precision and efficiency in terms of control. Commands can also be issued in a programmatic fashion at pre-set times offering great power to users as well as considerable time savings.

While Windows has improved its offering in recent years (e.g. PowerShell has been added in addition to the age-old and very capable DOS command line), UNIX/Linux is also a free and open source platform meaning that users can obtain the actual computer code used to create programs in addition to the programs. This underlying code can be edited by anyone, and suggested improvements passed back to original authors. Knowing exactly what a program is doing, or being able to find out, is invaluable, both for awareness/understanding but critically for making work reproducible and extensible (i.e. can be developed further).

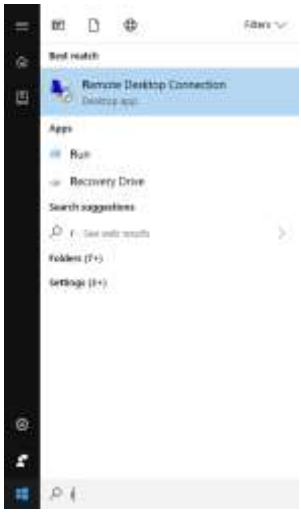
2 Connecting to Linux in GeoSciences from Windows Labs

XRDP:The School's remotely accessed Linux environment

The main mechanism currently supported for connecting to GeoSciences Linux systems is using the **XRDP** service. On a Windows lab PC this is achieved using Windows' built-in **Remote Desktop** client as follows. Find the **Remote Desktop Connection** *client* program by clicking on the Windows **Start** button to the bottom left of the screen.

You should see the cursor appear to blink in the adjacent **Search** text box. You can now type R (R usually better than r!) to begin searching. Usually this will be sufficient to locate the **Remote Desktop Connection** client, it being a feature of Windows itself. If you have difficulty however then you can find it via: **Start > Windows Accessories > Remote Desktop Connection**.

We would normally recommend explicitly locating programs in the menu system (you can scroll the mouse wheel to make this much easier) however given the extra features in modern Windows searching may be significantly quicker.



In/on Windows 7 you can find the RDC client under **Accessories**, or use **Search**. In/on Windows 8 you will likely need to use **Search**! For full instructions, including how to connect Apple Macs and Linux machines to the GeoSciences servers see the information at:

https://www.ed.ac.uk/geosciences/intranet/it/linux-servers
https://www.ed.ac.uk/geosciences/intranet/it/linux-servers/xrdp

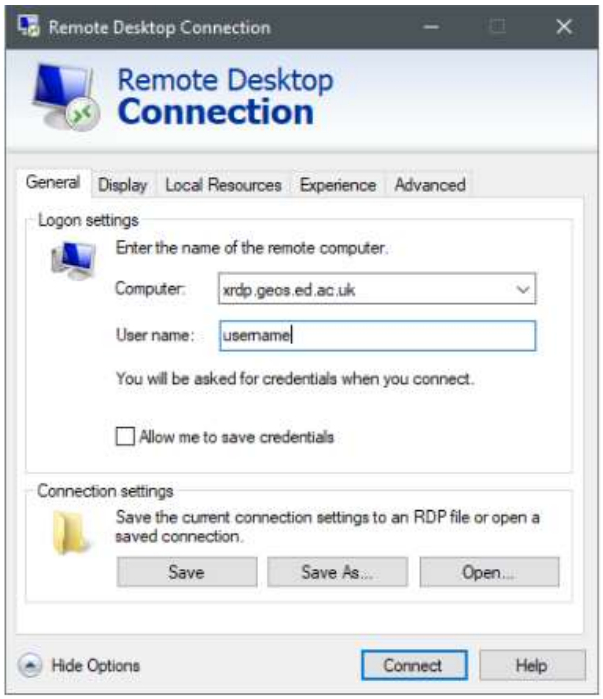
Notes for expert users:
You can also connect to the School's **ssh.geos.ed.ac.uk** service with a suitable secure shell client such as **MobaXterm** or **mosh**. This can be useful if internet bandwidth or allowances are extremely limited when working remotely. **MobaXterm** includes both command line and *X server* for graphical display; **mosh** is text only allows for intermittent connections however. You will initially log in to the School's basic SSH gateway and will need to connect (with **`ssh`**) in turn to a fuller compute server to undertake any processing work.

Connecting with Remote Desktop Connection – Setting the settings…

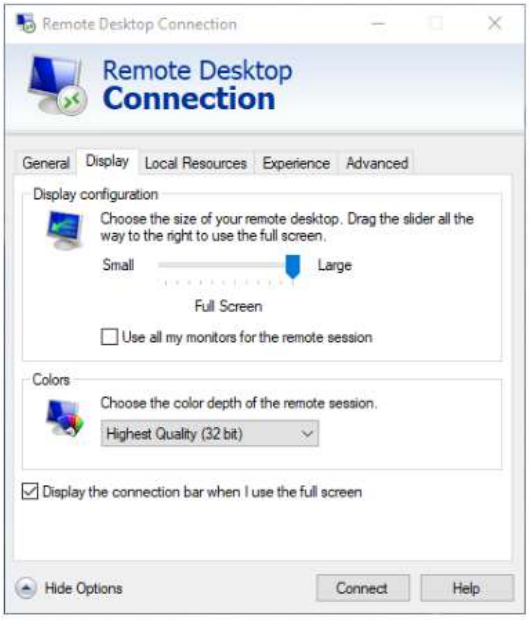
The following pop-up box should appear. For **Computer:** enter the address of the service **xrdp.geos.ed.ac.uk**

Don't proceed just yet but instead click on the left-hand drop-down **Show Options**. **NB** If you make a mistake you can simply make another attempt. Once the **Options** are shown and the box expands you can enter your **UUN username**, e.g. s1234567 (note just the username is needed here without any @ed or @sms.ed etc. on the end.)

NB Entering a **username** is key as it allows the system to distribute users effectively over the pool of servers behind the scenes thereby ensuring a balanced load on the servers and minimal risk of server crashes!



Now let's examine the **Display** tab before we proceed…

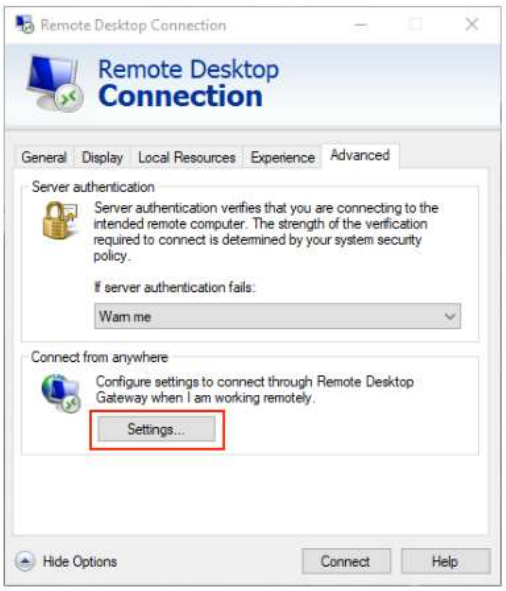


Here you can control the size and type of display used. The default setting is **Large** and this may be useful however you will need to switch back and forth between the Remote (Linux) Desktop and the local one in order to refer to these or other notes. You may wish to re-open any browser tabs in a browser running on the Linux system (the usual open source alternatives are available – Firefox, Chrome). You can however minimise the window or control its size later by moving the mouse pointer to the top middle of the screen to access a hidden title bar with **Close, Minimise** and, most useful, **Restore Down** *gadgets* (i.e. clickable buttons).

Are we done yet…?! Sort of…

In a lab setting where the computer is physically part of the University network already you may not need to select the following **Advanced** settings however it is worth noting these for working from elsewhere across the University or from home, when on fieldwork etc. If you already have a VPN connection (or even eduroam wifi access) you may not need to do this but why not take this one time today if using your own device to set it up to work, worldwide?

Click the **Advanced** tab and then the **Settings…** button.



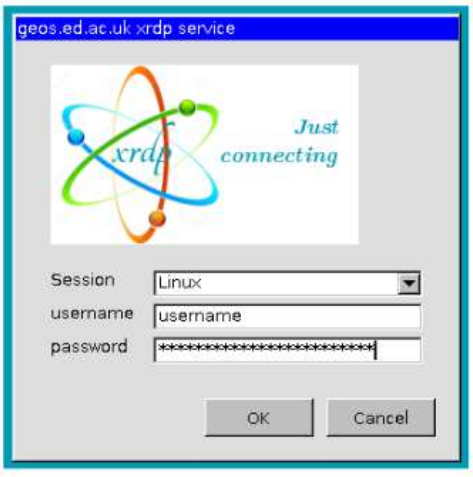
Select **Use these RD Gateway server settings** and enter **rd-gateway.is.ed.ac.uk** for the **Server name** and **Allow me to select later** as the **Logon method**. Now uncheck/untick **Bypass RD Gateway server for local addresses**. Since we are connecting to a non-Windows machine and Remote Desktop is originally a Windows technology we should leave the bottom box blank (staff or PhD students with a fixed desktop Windows PC may however also need this setting if remotely accessing their office PC from a laptop for instance).

Click **OK**, then **Connect** once back at the main RDC prompt. You will then be prompted for a **username** and **password** in order to connect to the Remote Desktop gateway (we'll finally connect to the actual Linux server in a minute!). Enter your username in the form **username@ed.ac.uk** along with the password you use to connect to Windows.

If you get a warning about a certificate you can safely ignore it and just click **Yes** since you are working in GeoSciences under these instructions, although you should read all such notices carefully when working online more widely.

While the set-up has perhaps been a little laborious XRDP itself offers a number of advantages in that we can log out and go home while programs are still running, and we can resume working on the GeoSciences server cluster the following day – useful for example should we have a large data processing job to complete which takes longer than a single working day. While there are other methods we can use to achieve this, the XRDP system also allows the state of open applications to be preserved upon logging out and back in, offers a full virtual Linux desktop experience, and all via a secure connection.

Thankfully we are now ready to connect to the GeoSciences server cluster. You should see the GeoSciences XRDP service login prompt. If required enter your **username** (just the username – no internet *domain* contextual information required this time) and then your **EASE password** (EASE is used to authenticate to GeoSciences Linux).



Click **OK** and wait for the virtual desktop environment to appear!

Note a number of useful features. Similar to Windows there is a **Computer** *icon* (i.e. clickable image). You can use this to open the **Caja** file manager. **NB** note that sometimes certain directories are not visible by default and need to be accessed via the command line first!

Most programs can be found from the menus which in this system are from a more traditional (non-Windows) layout at the top of the screen.

The **Applications** menu is particularly useful. The right-hand **System** menu is for logging out. Beside the menus are a selection of icons including another shortcut to the **File Manager**. The most useful icon though is the **Terminal** one – the black box, currently the middle of the three.

We will make extensive use of this for the rest of the session thus click on the **Terminal** icon now. By default this appears with a dark background which may be useful for avoiding eye strain during long periods of computer-based working (remember to take regular breaks!) If however you find some text hard to read against the black background you can quickly invert this to have a white background by going (in the **Terminal** window) to **Edit > Profile Prefefences > Colours** tab and then selecting e.g. **Black on white** from the **Built-in schemes** drop-down list.

We will now look at using the command line interface to demonstrate the power of UNIX. Many of you will make use of the most common of these commands to manage your work during taught classes or research.

If you have not got a **Terminal** window to appear, or are not happy at this point – please ask for help!

3 Files and Directories

The Bash (Shell) Prompt

When you open up a Terminal window you should see a largely blank window apart from a semi-cryptic bit of text known as the bash prompt. The command line is known in UNIX terminology as a shell (sh) and for historical reasons the default shell is bash. The prompt shown awaits your commands to be typed *at* it, but gives some helpful information to help with this.

```
[snnnnnnn@server ~]$
```

If you remember from page 2, **snnnnnnn** is your username, shown at (@) the host or **server** name (i.e. the name of the server logged into – currently usually one of the **baltic** servers but perhaps other School servers such as the gateway machine **adder** or *compute* servers **server** or **achray**.)

Far more usefully, the tilde (**~**) tells you that you are in your home directory. This has a full address which we will ask you to *translate* (as it were) in the next section in order to bolster your Linux knowledge!

Space – the frontier between ALL commands! Use spaces!

A key point before we proceed is that computers (despite recent advances in so-called Machine Learning) are really rather dumb and thus all commands and any other parameters (known as *arguments* or *flags*) must all be specified separated by spaces so that the computer has a chance to identify where one command or argument ends and the next begins. In these, and other exercises, if it looks like a space then it probably is one and should be typed as such! **NB** This is different from e.g. web browsers which simply render text as instructed using special mark-up 'tags' and which otherwise ignore spaces! When working at the command line however you ***need*** spaces!

Listing Files and Directories

a) One of the most useful commands in UNIX is **pwd** - **p**rint **w**orking **d**irectory. It tells you where you are in the directory structure, and the full ***path*** to that location.

NB In UNIX/Linux (and in fact in DOS, i.e. anywhere outside of Windows Explorer) folders are known as ***directories***.

To use the **pwd** command simply type it as is at the bash prompt and press **Return** or **Enter**: This command takes no arguments (further control information or parameters) and therefore no spaces are required here.

```
[snnnnnnn@server ~]$ pwd
```

What is the address of (*path* to) your home directory?

……

……

Be aware that drive letters such as **M:** (or **G:** or **T:** etc.) are a Windows-only feature and that when using UNIX you will need to reference your home directory as /home/s*nnnnnnn*.

Fortunately the Linux desktop environment provides convenient shortcuts to your home directory and each time you start a Terminal session you are logged 'into' (located within) your home directory automatically to start with.

b) You will also need to know what files and directories there are within your home directory. The command for this is **ls** (i.e. list). Try it now. Are there any files in your home directory?

……

……

ls is a command which can optionally take a further **argument** (separated from the command by a space) and the output from it can be changed by the use of control **flags**. For an example we shall explore the **wkzero** directory held on our main shared teaching resource, ***netdata***.

```
/geos/netdata/wkzero
```

Look at the example below and try it yourself. Notice it takes a single argument (the path to a directory location) and so a single space is required between the command and argument.

→
```
[snnnnnnn@server ~]$ ls /geos/netdata/wkzero
demodir  if.txt  jabberwock.txt  nation_data.txt  protocols  xyzpoints.txt
```

Now let's try a more advanced version with two spaces required!

```
→ [snnnnnnn@server ~]$ ls -al /geos/netdata/wkzero
total 68
drwxr-xr-x  3 omacdona netdata  4096 Sep 14  2010 .
drwxrwsr-x 54 omacdona netdata  4096 May 29  17:10 ..
drwxr-xr-x  3 omacdona netdata  4096 Sep  7  2011 demodir
-rwxr-xr-x  1 omacdona netdata  1734 Sep  7  2011 if.txt
-rwxr-xr-x  1 omacdona netdata  1353 Sep  7  2011 jabberwock.txt
-rwxr-xr-x  1 omacdona netdata  8296 Sep  7  2011 nation_data.txt
drwxrwxr-x  2 omacdona netdata  4096 Sep 12  2016 protocols
-rwxr-xr-x  1 omacdona netdata 34965 Sep  7  2011 xyzpoints.txt
```

The first version used **`ls`** with an **argument** only (the *address of* or *path to* a directory) - the command means "list the files that are in the directory **`/geos/netdata/wkzero`**".

The second version uses both an **argument** and **flags**. The flags are **`-a`** and **`-l`** (you can stick flags together as above **`-al`**). **`ls -a`** means show all the files, including hidden ones (hidden filenames in UNIX start with a **`.`** character). **`ls -l`** means give **l**onger (more) information on each file (or l for long-form perhaps).

In the example above you should note the two extra entries **`.`** and **`..`** . A single **`.`** means "the present directory" in this case **`/geos/netdata/wkzero`**. A double **`..`** means the directory above this one – "the parent directory" or in this case **`/geos/netdata`**. We will come back to this notation later on…

So **`ls -al /geos/netdata/wkzero`** means "show all the files in the directory **`/geos/netdata/wkzero`** and give a long description of them".

In the long description you can see the read and write permissions of the files, what type of file (file or directory) they are, who owns them, how big they are etc. For example:

```
drwxr-xr-x  3 omacdona netdata  4096 Sep  7  2011 demodir
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
```

The explanation for this follows:

Column 1: the **`d`** in this column shows that **this file is a directory** – normal files have a **`-`** symbol here.
Columns 2-10 show the **permissions** set up for this file – see the full explanation below
Column 13 shows the number of links to the file (you need not worry about this)
Columns 15-22 record the user name of the **owner** of this file – **`omacdona`**
Columns 24-30 record the UNIX **group** (of users) to which the file belongs – **`netdata`**
Columns 34-37 show the **size** of this file in bytes or characters
Columns 39-50 show the **date** and, if recent, the time this file was **last modified**
Columns 52-57 show the **name** of this file – **`wkzero`**

Columns 2, 3, and 4 show that the user (**owner**) has permission to **r**ead, **w**rite/delete, and e**x**ecute (run commands on) this file. Columns 5, 6, 7 and 8, 9, and 10 respectively show that both the UNIX **group** and all **other** users only have permission to **r**ead and e**x**ecute for this file. They cannot alter it or delete it. We also commonly say that the file is world-readable (i.e. readable only by anyone in who has access to the Geosciences server that is!)

c) Using the commands above, find out if there are any files in **`/geos/netdata/wkzero/demodir`** and if there are, note their name, owner, and size below.

……

……

……

……

Simple Viewing of Files

d) Viewing simple text files is easy – you can use one of these 3 *pager* commands (and others!):

`more` ***`filename`*** displays ***`filename`*** one page at a time
`head` ***`filename`*** displays the first few lines of ***`filename`***
`tail` ***`filename`*** displays the last few lines of ***`filename`***

You should have found a file called **`if.txt`** in **`/geos/netdata/wkzero/`**. What are the first few lines?

(**Hint:** A filename is itself just a path to some data so when referencing files outwith the current folder (as we are doing here) the full path – including filename – should be specified and *passed* to the command as its argument – e.g. if entering a command to act on a file it would take the form:
my_chosen_command /top-folder/sub-folder1/sub-folder2/…/myfile.txt.)

……

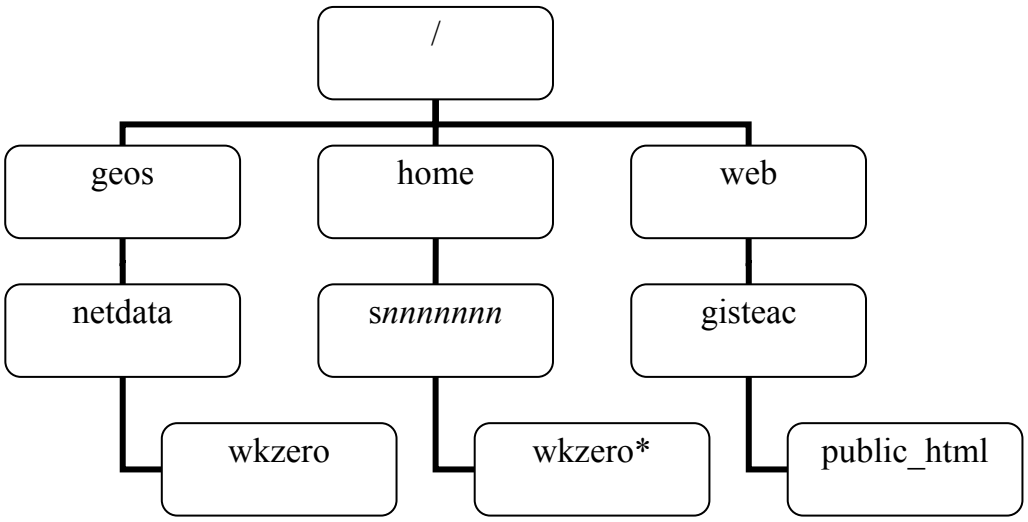
……

……

……

As well as `more` there is actually also a similar command `less` which is set as the default text viewing (pager) program on the version of Linux used within the School! It works mostly the same way but requires you to press **`q`** to quit once you are finished reading. This prevents you scrolling to the end of the file and having the text viewer program close automatically as can happen with `more`.

UNIX/Linux Filesystem Structure (Example)



The above diagram shows a simplified extract of the GeoSciences Linux filesystem. At the top (of the *inverted tree* structure) is the home directory of the root (or super) user – this is the administrator or filesystem owner. Following along the branches from this 'root' are various folders and subfolders (or directories in UNIX terminology).

You should recognise the shared folder netdata as part of this with one of its subfolders wkzero shown which you used earlier. The other wkzero* marked with an asterisk (*) is meant to signify one lying within your own home directory – *we haven't actually created this yet but in fact will get you to create this next!*

NB The user **gisteac** (note there is no '**h**'!) is a very useful user to remember in GeoSciences! In particular, shown here are gisteac's web folders at `/web/gisteac/public_html` which contain much useful info relating to geospatial computing. Any publicly accessible info stored here can be reached over the web as per the URL given at the very top of this practical, i.e. at **www.geos.ed.ac.uk/~gisteac**. **Note** the special use of the tilde character (**~**) here to signify the main (home) web directory of a user.

Creating and Deleting Files and Directories

Managing your directories and files is an important aspect of working with UNIX (or indeed any computer operating system). You only have a limited amount of disk space yet putting all your files together in one directory is only going to lead to large amounts of wasted time searching for previous work! Making a directory is easy – use the command **`mkdir`**. For example:

→ `[snnnnnnn@server ~]$ mkdir wkzero`

will create a new directory called **`wkzero`** – **NB** in *your* current directory. Remember the need for spaces to separate commands from arguments. Using **`mkdir`** create some other new directories to put *your* work in. How about **`webpages`** and **`papers`** for starters?

e) Once you have made these new directories, check their properties using **`ls`**.

……

Having directories isn't much use if you are stuck in your home directory. To change directory use **`cd`**. For example:

```
→ [snnnnnnn@server ~]$ cd wkzero
→ [snnnnnnn@server wkzero]$ pwd
  /home/snnnnnnn/wkzero
→ [snnnnnnn@server wkzero]$ cd ..
→ [snnnnnnn@server ~]$ pwd
  /home/snnnnnnn
```

Remember the **`..`** notation from before (p.13)? Here it is *used* (i.e. in the command) to change to the parent directory of the directory that you are currently in (i.e. to move to one level above). **NB** Also notice how the prompt reflects the current directory and changes from `~` to `wkzero` and then back again in the above example. **NB** Keep an eye on this prompt changing and use this feature during the practical to ensure that you are issuing commands from within the correct folder!

f) Now use the **`cd`** command to *change **back*** to *your* **`wkzero`** directory and add two more new subdirectories there (i.e. within **`wkzero`**!) called **`ftpdata`** and **`docs`**. Use **`ls`** and a handy command called **`tree`** to check everything is OK.

……

……

If a command fails first check that you are in YOUR **`wkzero`** folder and not our reference/source version! (You should have *your* username in the path – and **not** `netdata`!) Use **`pwd`** to check, or **`cd ~/wkzero`** to go 'home'!

Note! You can create directories in your UNIX home directory from within Windows using Windows Explorer (My Computer). Your home directory usually appears in Windows, mapped to **M:** for most GeoSciences or research postgraduates, or perhaps to another letter (usually **T:**) for undergraduates or students formerly undergraduates at Edinburgh.

When using Windows to access UNIX network drives – **be careful!** Windows lets you put spaces in your directory names. UNIX is happy enough with this, but it can make moving around in directories within UNIX more awkward since you may need to enclose the directory names in quote marks **" "**. You may also find some software will object to UNIX paths with spaces in their names; much better to use an underscore **_** or hyhen **-** instead.

To delete a file in UNIX you use the **`rm`** command. For a directory you use the **`rmdir`** command. **Be very careful when using `rm`! Once a file is gone, it is gone forever!** It is much better to **always use `rm` with the `-i` flag**. This means that UNIX will ask you to confirm the deletion of a file or directory. Much safer… **Remember** that with *network* drives, e.g. if you use Windows Explorer to delete a file or directory on your **M:** drive, the same rules apply – it will not go to the Recycle Bin, it will be gone forever!

→ Use **`rmdir`** to delete one of your new directories and **`mkdir`** to recreate it.

UNIX Symbolic Links; Group Projects – Beware…

You should also look out for UNIX/Linux *links* (known more fully as *symbolic links* or *symlinks*) which will appear simply as folders under Windows. If you are collaborating with several colleagues you should always use **`rm`** or, better still, **`unlink`** to remove any unwanted links so as not to accidentally delete any files or sub-folders which you have write access to! Deleting symlinks from Windows could be rather risky to say the least! Symlinks can be identified shown in cyan (light blue) when listed in a Terminal window.

Sneaky (useful!) tips:

1) Using the up arrow button on the keyboard will scroll through the commands you have typed before so you can re-use them… You can also move the cursor along a recalled command to edit it with minimal effort.

2) In a similar vein, the Terminal offers a copy and paste facility even across operating systems (i.e. between Linux and Windows and vice versa). Either go to the **Edit** menu and **Copy** and **Paste** as required (or use **Shift+Ctrl+C** for copy and **Shift+Ctrl+V** to paste – no shift key required for those in Windows…) **NB** Some other editors are even more rapid allowing you to simpy highlight text to copy, and right-click to paste. If you highlight past the end of a line a 'newline' is also included and any text (e.g. commands) submitted to the computer with one click!

3) If you are typing a file or path name, you can use auto-completion by hitting **`<tab>`**.

4) You can use a handy version of **`rm`** to delete any files and folders safely and in one operation. Use **`rm -ir`** (or **`rm -iR`**).

Copying and Moving Files and Directories

You can copy files and directories not only within your own working area (i.e. in your home directory, or any directories below that in the hierarchy) but also from other users' directories so long as *you have the correct permissions*.

To copy a file or directory, use the command **cp**. For example, *once again* check you are in your **wkzero** folder then do the following:

→ [s*nnnnnnn*@server wkzero]$ **cp /geos/netdata/wkzero/if.txt .**
→ [s*nnnnnnn*@server wkzero]$ **cp /geos/netdata/wkzero/if.txt 2nd_if.txt**

g) But what do they do? Use **ls** to list all the files in your directory. What has happened? So what does each command above do? Be sure and ask if not clear before moving on!

……

……

……

……

Copying a whole directory is very similar - you just add a flag **-R** which stands for **recurse** the directory i.e. look at all the files within the directory and copy them as well, keeping the directory structure the same as that of the original. Try:

→ [s*nnnnnnn*@server wkzero]$ **cp -R /geos/netdata/wkzero wkzero_copy**

h) Then use **ls** again. What has happened? Again ask if not sure before moving on!

……

……

Moving and **renaming** a file or directory are the same thing in UNIX and the command used is **`mv`**. If you rename a directory then the ***path*** (or address) of all the files within that directory will move as well. Try the example below:

→ [s*nnnnnnn*@server wkzero]$ **ls wkzero_copy/**
demodir if.txt jabberwock.txt nation_data.txt protocols xyzpoints.txt
→ [s*nnnnnnn*@server wkzero]$ **mv wkzero_copy/ not_wanted_now**

→ You have now moved (renamed) the directory so the following command fails:

[s*nnnnnnn*@server wkzero]$ **ls wkzero_copy/**
wkzero_copy/: No such file or directory

→ You can also move groups of files around within your directory structure. Having renamed **`wkzero_copy`** we should move the files out of it to somewhere more logical: So:

→ [s*nnnnnnn*@server wkzero]$ **mv not_wanted_now/*.txt docs/**
→ [s*nnnnnnn*@server wkzero]$ **ls docs/**
if.txt jabberwock.txt nation_data.txt xyzpoints.txt

The **`*`** character is a **wildcard** that means "all characters".

NB The final / is often not required, it simply indicates to the user that they are referencing a directory. Thus:

→ [s*nnnnnnn*@server wkzero]$ **ls docs**
if.txt jabberwock.txt nation_data.txt xyzpoints.txt

will work just as well!

To keep your home directory (and i.e. thus **M:** or **T:** drive) tidy and well-organised you should **m**o**v**e your **`PCInduction`** folder from yesterday (which should be in your uppermost top-level ***home*** directory *if you created it*) into *your* **`wkzero`** folder.

We can do this as follows:

→ [s*nnnnnnn*@server wkzero]$ **mv ../PCInduction PCInduction**

Note we could also have issued this as any of these:

[s*nnnnnnn*@server wkzero]$ **mv ../PCInduction .**
[s*nnnnnnn*@server wkzero]$ **mv ~/PCInduction PCInduction**
[s*nnnnnnn*@server wkzero]$ **mv ~/PCInduction .**

The tilde character (**~**) is a short-hand reference for your own home directory. Thus we can more usefully write this command as:

[s*nnnnnnn*@server wkzero]$ **mv ~/PCInduction ~/wkzero/PCInduction**

NB This last command can be issued from ***anywhere*** within the directory structure!

4 Help

Help in UNIX can be a little unfriendly until you get used to it! There are lots of sites on the web that will be clearer and easier to follow for beginners. You should, however, know how to get help for specific commands – and once you get used to the syntax and structure of the pages, the help system might even start to grow on you…

Help is accessed with the **man** (short for manual) command and the help pages are exactly that – online versions of the UNIX manual. The structure is **man <command>** where **<command>** is the name of the command you want help with. Try:

→ `[snnnnnnn@server wkzero]$` **man man**

Press **<Space>**, **<Enter>**, or the `cursor (arrow) keys` to *scroll* through the content of the manual pages, and **q** to quit or exit. This should give you a better idea of how to use help! Notice that the help system uses the default text viewer, **less**, described earlier.

Another useful feature of man involves the **-k** flag. This allows you to specify a keyword, which man will look for in the help pages. Try:

→ `[snnnnnnn@server wkzero]$` **man -k transfer**

i) This will return quite a lot of possible commands – lurking in the list is one we will use next – **ftp**. What does the **man** page for this say it is for?

……

……

5 File Utilities (Big Data Downloads)

FTP (File Transfer Program/Protocol)

FTP is a long-standing way of transferring files from a remote computer/site – either a standalone PC, workstation, or server, to your machine. The following example uses a demonstration FTP site set up for this practical. Due to the insecure nature of FTP many sites will now require Secure FTP (sometimes known as SFTP) or will provide a more involved (registration probably required), but more modern, web interface. FTP is still a useful facility however, and still used by many internet sites and geographic data providers.

Type the following (when asked for a user name type **anonymous** and in line with 'anonymous ftp' convention, enter your ***email address*** when asked for your password):

→ [s*nnnnnnn*@server wkzero]$ **cd ftpdata**
→ [s*nnnnnnn*@server ftpdata]$ **ftp**
→ ftp> **open ftp.ed.ac.uk**

```
Trying 129.215.70.239...
Connected to ftp.ed.ac.uk (129.215.70.239).
220-    Welcome to the University of Edinburgh Anonymous FTP server
220-=========================================================================
220-
220-The following anonymous ftp servers are also available:
220-
220-    ftp.ed.ac.uk            University of Edinburgh (this server)
220-    ftp.ucs.ed.ac.uk        Information Services (old EUCS)
220-    ftp.epcc.ed.ac.uk       Edinburgh Parallel Computing Centre
220-
220-When requested for a username enter 'ftp' or 'anonymous'.  If you have
220-problems, try using a dash (-) as the first character of your password.
220-If you still have problems or wish to make a comment then send email to
220-ftpmaster@ed.ac.uk. The local time is Thu Sep  5 13:19:28 2013.
220-
220-All transfers are logged.  This server supports automatic taring and
220-compressing/uncompressing of files during transfer.
220-
220-
220 lewis.ucs.ed.ac.uk FTP server (Version wu-2.6.2(2) Mon Oct 12 14:39:33 BST
2009) ready.
```

→ Name (ftp.ed.ac.uk:s*nnnnnnn*): **anonymous**

You should get a response like this:

```
331 Anonymous login ok, send your complete email address as your password
```

→ Password: **s*nnnnnnn*@sms.ed.ac.uk**

```
230-This service is managed by Information Services.  It holds information
 which may be useful to system managers and space is provided for
 individuals and groups upon request.  Upload facilities are also
 available.  Anyone can make use of this service.
.
.
.
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
```

You are now connected to the Edinburgh FTP server – **ftp.geos.ed.ac.uk** - and have limited rights to list and download files. Remember that you are now a user of the FTP application; it has its own set of commands (though they are very similar to UNIX commands).

Type the following:

```
→ ftp> ls
  227 Entering Passive Mode (129,215,17,244,233,181).
  150 Opening ASCII mode data connection for file list
  -rw-r--r--   1 root     root         2647 Feb 19  2014 INSTRUCTIONS-FOR-USING-
  THIS-SERVICE
  drwx-wx-wx   3 root     root         4096 Sep  9 14:54 eduplоad
  drwx-wx-wx   2 root     root       499712 Sep  9 11:06 incoming
  drwxr-xr-x  45 root     root         4096 Feb 19  2014 pub
  226 Transfer complete
→ ftp> cd pub/geos
  250 CWD command successful
→ ftp> ls
  227 Entering Passive Mode (129,215,17,244,177,138).
  150 Opening ASCII mode data connection for file list
  drwxr-xr-x   2 309643   root         4096 Sep 10  2008 wkzero
  226 Transfer complete
→ ftp> cd wkzero
  250 CWD command successful
→ ftp> ls
  227 Entering Passive Mode (129,215,17,244,205,239).
  150 Opening ASCII mode data connection for file list
  -r--r--r--   1 root     daemon        365 Sep 10  2008 jefferson.tar.gz
  226 Transfer complete
```

The commands you have typed above ONLY apply to the remote system – you are still in the same directory on your local UNIX server. The command sequence above shows you some of the files you can download. Often you will first have to set the **transfer type**. You are going to download the file **jefferson.tar.gz** which is a compressed binary file. So just to be sure type:

```
→ ftp> type binary
  200 Type set to I
```

Then, use the **get** command to begin transfer:

```
→ ftp> get jefferson.tar.gz
  local: jefferson.tar.gz remote: jefferson.tar.gz
  227 Entering Passive Mode (129,215,17,244,137,65).
  150 Opening BINARY mode data connection for jefferson.tar.gz (365 bytes)
  226 Transfer complete
  365 bytes received in 0.0503 secs (7.26 Kbytes/sec)
```

Then, to log off from FTP and close the network connection:

```
→ ftp> bye
  221 Goodbye.
```

You should now have your own copy of the file **jefferson.tar.gz** transferred from the FTP Server running the Edinburgh FTP Service. While modern day FTP servers will often allow you to obtain files via a web browser, the next part is key knowledge for UNIX users.

GZIP (GunZip)

So you've got a file that looks like it might have some data in it. But what kind of a file is it? And how do you get at the data? As is often the case for data files on UNIX, this file is actually a collection of files specially encoded to save space (e.g. when dealing with high-res images or large scientific datasets). This is a two stage process that will introduce you to two more useful UNIX utilities.

→ First make sure the file **jefferson.tar.gz** (**mv** if necessary) is in *your* **wkzero/ftpdata/** as it should be. **NB** You can launch applications (e.g. ftp) from any chosen destination directory, in this case we used the **ftpdata** directory, to make life easier once finished with FTP. Make sure you are in the **ftpdata** directory now.

The file **jefferson.tar.gz** has been compressed using the **GunZip** utility. This allows you to compress files so that they take up less disk space (and transfer faster across FTP). To look at the data, we first have to reverse this compression, using the command **gzip**:

```
→ [snnnnnnn@server ftpdata]$ ls -l
  total 4
  -rw-rw-r-- 1 snnnnnnn snnnnnnn 365 Sep 11 12:00 jefferson.tar.gz
→ [snnnnnnn@server ftpdata]$ gzip -d jefferson.tar.gz
→ [snnnnnnn@server ftpdata]$ ls -l
  total 12
  -rw-rw-r-- 1 snnnnnnn snnnnnnn 10240 Sep 11 12:00 jefferson.tar
```

Note what has happened. The **-d** flag told GunZip to decompress the file, so it loses its extension and gets bigger. You can see which flags to use to compress files by typing **gzip -h** (h for help).

Now you have uncompressed the file, but it is still **archived – it is one file that contains many other files**. You thus need another utility to get the archived contents out again.

TAR (Archiving Tool)

To get the files back out from their archived form, you need to use the **tar** command. The instructions below are a **very simple example** of using **tar** to extract files - you can do much more with it! Have a look at either the online help or the UNIXHelp web pages…

To extract the files, type the following:

```
→ [snnnnnnn@server ftpdata]$ tar -xvf jefferson.tar
  jefferson.txt
```

Ok in this case there is only one file but usually you will have many files – e.g. datasets, images, etc. (The name tar originally stood for tape archive/archiving, involving many files.)

The flags:
-x Extract files from the archive.
-v Verbose- provide feedback to the user.
-f The file to extract from will be specified by the user and is the next argument.

6 Further File Utilities (Zip Files, Automated Download, Web PDFs)

Working with Windows .zip files; Minimising Upload/Download

While the UNIX gz format is common in the scientific and computer programming worlds, often there is a need or preference for the, possibly more common, Windows zip format.

On the GeoSciences Linux servers there are therefore tools to work with files in this format.

→ `[snnnnnnn@server ftpdata]$` **`zip jefferson.zip jefferson.txt`**

would compress `jefferson.txt` to become `jefferson.zip`. **NB** The command requires the output zipped file to be specified first!

You could also compress a whole folder of files in the same way, or specify a set of files to zip into one compressed file. This makes it much easier to distribute multiple files intended for use by Windows users without having to download all the files to your local Windows PC, zip up and re-upload the compressed zip file to a web server. You can simply do the work on the GeoSciences server and move the result directly to the web server. This is handy if working with limited internet connectivity where only a command line Terminal connection is available, or perhaps if broadband allowances or speeds are low and files are big in size.

→

`[snnnnnnn@server ftpdata]$` **`unzip jefferson.zip –d jefferson2`**

Would then decompress that file to a new copy of the file in a new folder `jefferson2`. Alternatively you can leave out the folder specification and will be asked whether you wish to overwrite any existing files. You can find more information by simply running zip with no arguments specified (this is commonly the case with UNIX/Linux commands).

Remote/Automated File and Web Page Retrieval (e.g. via WWW)

Instead of obtaining files or datasets from special file-servers via FTP, you may have to retrieve files via the more conventional world-wide web (WWW) and save them to your GeoSciences home directory. If you wish to automate downloads or are working with very limited internet connectivity then you can use special Linux command line tools to allow the GeoSciences server to download these files for you. You can then issue other commands via Linux to perform processing or analysis on these and obtain results leaving your broadband allowance or internet bandwidth unaffected, or even automate the entire process.

Introducing `curl` and `wget`

The **`curl`** and **`wget`** commands can be used to retrieve files from any web URL, e.g. with **`curl`**, [**NB** Type these all on one continuous line!]:

→ `[snnnnnnn@server ftpdata]$` **`curl https://www.geos.ed.ac.uk/~gisteac/wkzero/protocols.txt -o protocols.txt`**

→ `[snnnnnnn@server ftpdata]$` **`curl https://www.geos.ed.ac.uk/~gisteac/wkzero/protocols.html -o protocols.html`**

If you omit everything from the output –o flag onwards the file contents will just be displayed.

The **wget** command functions in a similar way. Each command has both advantages and disadvantages over the other, but **wget** also offers a useful *recursive* facility where it is possible to download not only a web page, but other pages or links contained within it, e.g, [**NB** Again, type this all on one continuous line]:

→ [s*nnnnnnn*@server ftpdata]$ **wget http://www.geos.ed.ac.uk/~gisteac/wkz ero/protocols_all.html -r**

→ Notice the **-r** at the end of the line. Use **ls** and **more** to see what you get from the above commands!

(→) **NB** It is also possible to use these commands non-interactively in pre-written script files to *automate* file retrieval across the web (via HTTP/S), or by FTP as we used earlier, or in fact by a number of other data transfer protocols including Secure FTP (SFTP). This also means you can set a large download going and log out (we'll see more on this later). The ability to automate such tasks is a major strength of these tools and of sophisticated operating systems such as UNIX or Linux. Windows does offer some such sophisticated tools but is still catching up in many respects and arguably does not cater so well for multiple users (despite its everyday prevalence!)

Be careful to use **wget** (and **curl**) responsibly. You may not be able to use all functions without having first been given the appropriate access to a website! Also, if you try to bulk save people's websites en masse too often they may well complain!

Working with PDFs remotely (For WWW-based PDF distribution)

There are an unlimited number of tools that you may wish to use to analyse large datasets downloaded to your GeoSciences home directory, or that you might wish to run on the School servers. As a slight change however let's look at a useful tool for working with PDF files, a very common method of distributing information over the web either for local colleagues (saving clogging email inboxes) or for distribution of information world-wide.

While saving or printing to PDFs is now common place, often you may require to join multiple PDFs together into one document, or to separate a single PDF into multiple files. E.g. you may wish to stitch together pages scanned individually on a School or University multi-function scanner-copier machine, or perhaps scan a selection of documents in one go then split into separate documents later.

You perform such tasks with **pdfunite** and **pdfseparate**. The tool **qpdf** is also available which allows you to encrypt/decrypt/optimise PDF files. Here is an example of using **pdfunite** to merge a set of PDF files, from **netdata**, into a single PDF in your current directory:

→ [s*nnnnnnn*@server ftpdata]$ **pdfunite /geos/netdata/wkz ero/protocols/*.pdf protocols.pdf**

Don't worry if you don't fully understand this just now – do ask though!! – the point is to know you can do it and to have a document (this one!) of where to find the tools. You can review the output of this from a File Manager window and/or by running a PDF reader.

7 Resource Monitoring Utilities

Despite the ability to download large datasets automatically, you only have a limited amount of disk space on UNIX/Linux. You can find out what disk space you have available by using the **quota** command:

→ `[snnnnnnn@server ftpdata]$ quota`

NB Don't be alarmed if this takes a while to respond!

Using the **-s** option/flag will give you a more useful breakdown.

→ `[snnnnnnn@server ftpdata]$ quota -s`

In some versions of Linux the quota command has been known to display potentially misleading user information (i.e. home dir) however the *numbers* should reflect *your* quota!

→ **NB** If quota, or another command, ever misbehaves (disk access can be slow due to the physical traversing involved) then you can simply use **Ctrl+C** to abort the stuck command. I.e. press and hold **Ctrl** (Control) then press **C** whilst still holding **Ctrl** to abort the command.

To find out how much disk storage space you are using, use the **du** command. See **man du** for fuller details.

→ `[snnnnnnn@server ftpdata]$ du -sk`

will give you the total amount in kilobytes of disk space used in the current directory which should still be `ftpdata` unless you have changed this yourself! (Running this command in your home directory will tell you your **total** disk usage.)

Replacing the `k` with an **h** will give you a more readable form.

→ `[snnnnnnn@server ftpdata]$ du -sh`

Alternatively you can use SI units (where 1 Kb = 1000 bytes instead of 1024 bytes). This may help you manage your data volumes as you will appear to have more MBs or GBs etc!

→ `[snnnnnnn@server ftpdata]$ du -s --si`

Note that this uses a secondary (sub) flag signified by a double-hyphen.

Finally, running the following command in your *home* directory will give you a breakdown of the total space used by each file and directory located there, hence type:

→ `[snnnnnnn@server ftpdata]$ cd ~`
→ `[snnnnnnn@server ~]$ du -sh *`

Alternatively you can achieve the same anywhere as follows:

→ `[snnnnnnn@server ~]$ du -sh ~/*`

j) Using **quota** and **du**, find out how much storage space you are using and how much you have left:

……

k) Sometimes you may find that other users share space with you or control files to which you require access. You can find out about other users (i.e. who they are!) on the system in two ways – **who** and **finger**.

```
[snnnnnnn@server ~]$ who
```

will give you a list of who is currently logged on to the machine you are working on - it will also tell you how they are logged in. Who is logged on to the same UNIX machine as you? And which remote machine are they using to access it?

……

……

……

You can use a related command if you forget who you are – **whoami**

l) **finger** gives a much bigger set of information about a user. It allows you to search for users based on their username – the example bellow shows the result for **gisteac**. You may require to identify a user in order to negotiate space or access with them so this can be very handy to identify a user behind a username!

```
[snnnnnnn@server ~]$ finger gisteac
Login: gisteac                          Name: MSc in GIS teaching
Directory: /home/gisteac                Shell: /bin/bash
Last login Wed Mar 11 16:47 2015 (GMT) on pts/49 from vpn2-089.vpn.net.ed.ac.uk
No mail.
No Plan.
```

See if you can find out the login name and home directory of another user using the **who** and **finger** commands.

……

……

The information in the **Plan:** comes from a special file in the user's home directory called **.plan** (a hidden file) – not everybody has a plan… ☺

8 Searching

m) You can search for all sorts of things in UNIX – usually it will be files or bits of text. The commands you would normally use are **grep** and **find**. But what do the commands do? Time for you to do some work. Use **man** to find out what **grep** and **find** actually do…

……

……

……

……

……

……

……

……

……

……

……

n) Found out? Try these commands - what do they do?

```
[snnnnnnn@server ~]$ find /geos/netdata/wkzero -name "*.txt"
[snnnnnnn@server ~]$ grep -n 'stan' /geos/netdata/wkzero/nation_data.txt
```

……

……

……

……

……

9 Applications

The way you interact with UNIX software applications can vary. Some will operate simply at the command line in one go, sometimes interactively, others may take over the screen with a text-based menu or screen system, others offer full-blown graphical capabilities.

Textual: Command Line or Screen based

When you start a command line based application, control and response is **transferred from UNIX to the application**. You have already used one application of this sort – **`ftp`**. Remember how the command prompt changed from `[snnnnnnn@server ~]$` to `ftp>` ?

Another program that you may wish to use if working in particularly challenging environments is the UNIX email program **Alpine**. The important point to note with all these is that **once you are running the application, the commands will be specific to that application**. As a quick example, do the following:

→ `[snnnnnnn@server ~]$ alpine`

o) Note how this time the command prompt format then changes significantly to that of a different type of program. How would you describe this?

……

Type **e** to exit the initial greeting if shown, then type **q** and then **y** to quit. Other useful text
→ only programs include simple text editors such as **`pico`**, **`nano`** and **`vim`**, or the original UNIX line editor **`ed`** – sometimes useful within code for replacing one bit of text with another. The editor that we recommend (and use in much teaching) is **`emacs`** which we'll look at next…

Graphical: Graphical User Interface (GUI) based

p) GUI based applications are very different. Here, the application usually opens a new window for itself, and your **interaction with the application is through that window, not the command line**. For example:

`[snnnnnnn@server ~]$ emacs`

will start up **`emacs`** in a new window. Your interaction with **`emacs`** is now through the GUI, not the command line. Move the window if necessary so that you can see the Terminal window. What has happened to the command line? We will come back to this later…

……

Some other useful graphical applications may include web browsers such as **`firefox`**, PDF readers such as **`acroread`**, or full-blown programs such as GIS and Remote Sensing software. We'll look later in more detail at running some of these, however simple command line access is also an invaluable option for quick or remote/low-bandwidth working.

SSH: Connecting to programs running on specific servers

You may require access to other UNIX/Linux machines e.g. if they have different software or greater processing power. This is common when using UNIX workstations. You can achieve this by connecting with the **ssh** (secure shell) command from a Terminal window.

You can also connect to a School UNIX server remotely. The way to do this is to connect to **ssh.geos.ed.ac.uk** using an SSH client such as **MobaXterm** or **mosh** (good for patchy signals/mobile working) or perhaps the built-in ssh software if using an Apple Mac). This ensures that you will have proper access to the correct School gateway server which may change in future while the address, ssh.geos.ed.ac.uk, should remain the same.

Once logged in you will be connected to the gateway server **adder**. This contains very few programs and is merely acting as an access point (gateway) to the other cluster of dedicated servers. You will thus need to **ssh** to another server containing whichever software applications you require. For example let's connect to the compute server, **burn**.

```
[snnnnnnn@server ~]$ ssh burn
```

Other servers exist which you may wish to connect to at some point, e.g. **achray**. You may be asked for your username and password, or just your password, **each** time you make another ssh connection from an existing session. Don't worry if you are given a message about host authenticity each time you connect to a new GeoSciences *host*; when asked if you wish to continue, you can safely type **yes** then press **<return>**. If ever in doubt though, you can always check with IT.

Each time you log in, you are connected to that machine via a *brand new* connection, just the same as when you first open a terminal window on a PC. You will thus once again start in your home directory each time you make a new connection. To close the connection to the remote machine use **exit**: Control will pass back to your original machine (or ssh window) and so on if you have multiple logins.

```
[snnnnnnn@server ~]$ exit
```

10 Log Out/Off or Close Session?

You can close and logout an individual Terminal session by typing **exit** as above. This is recommended as it will close the session 'cleanly' – i.e. without any unclosed programs or files. It is always best to clean up after yourself! To close the whole graphical XRDP session and all programs you can go to the System menu and select **Log Out**. You will be asked to confirm.

You can also go to the top of the screen and wait for the blue Remote Desktop drop-down bar to appear and click Close (X). The major strength of XRDP and closing this way is that when you login later any open programs should remain running. (**NB** There is also a tool called screen which can be used to keep programs open when running at a command line only using an SSH client.)

That's all for this practical. You should safely file these worksheets and use them for guidance during the coming months, and certainly during your dissertation!