Linux practical, CDT 2018

Susan Hutchinson
Department of Statistics,
University of Oxford

October 2018

Contents

List of Figures

1 Introduction

First, a very brief look at the Fedora 28 Linux desktop. Next, a longer introduction to the Unix/Linux command line.

This file available at **http://www.stats.ox.ac.uk/pub/susan/cdt/Exercises.pdf**. I recommend that you open a copy on the desktop as many of the links are clickable.

2 The Linux desktop

The first time you log on, a little set-up is needed. I have suggested an action for each window

Welcome Next
Typing Next
Privacy Slide both to **[OFF]** | Next
Online Accounts Next
Start Using Fedora Click on this
Getting Started Close this

You should then see the standard Linux Desktop as in Figure 1.

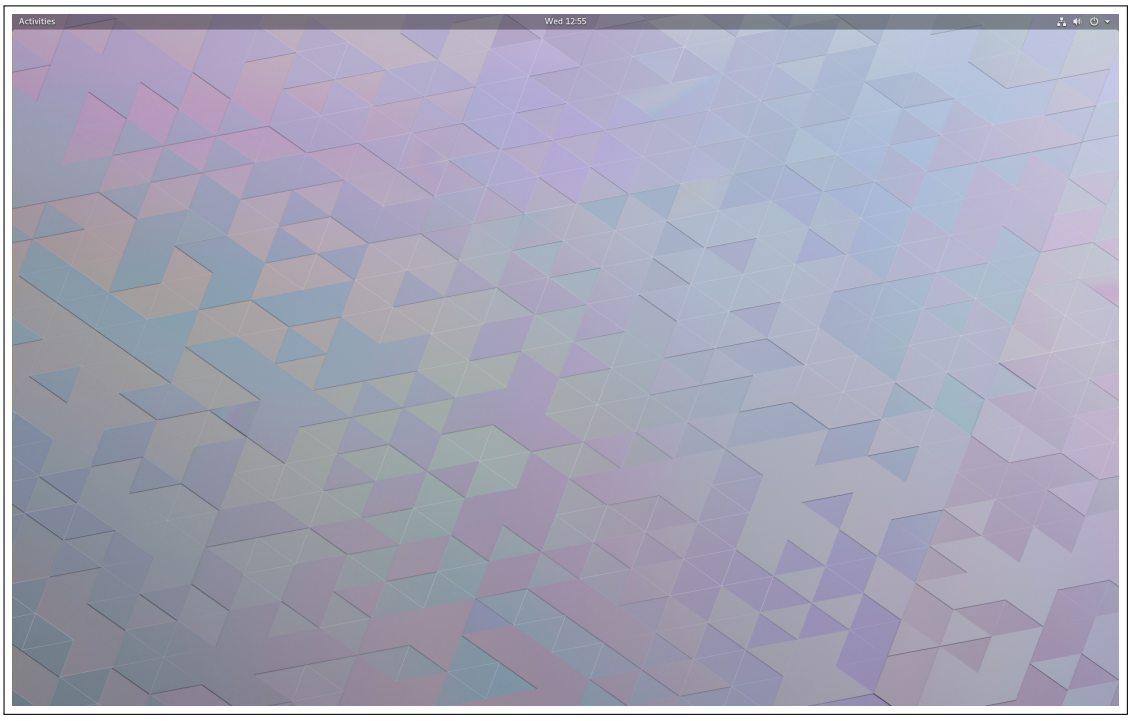

Figure 1: Adding a terminal window to the Favourites bar.

We will now explore the Linux desktop and explain how to do some simple tasks.

2.1 Starting applications

Press the Windows key (or in Linux, the Super key) to start applications. You should see a screen like Figure 2.

The bar on the left side is the Favourites bar which contains

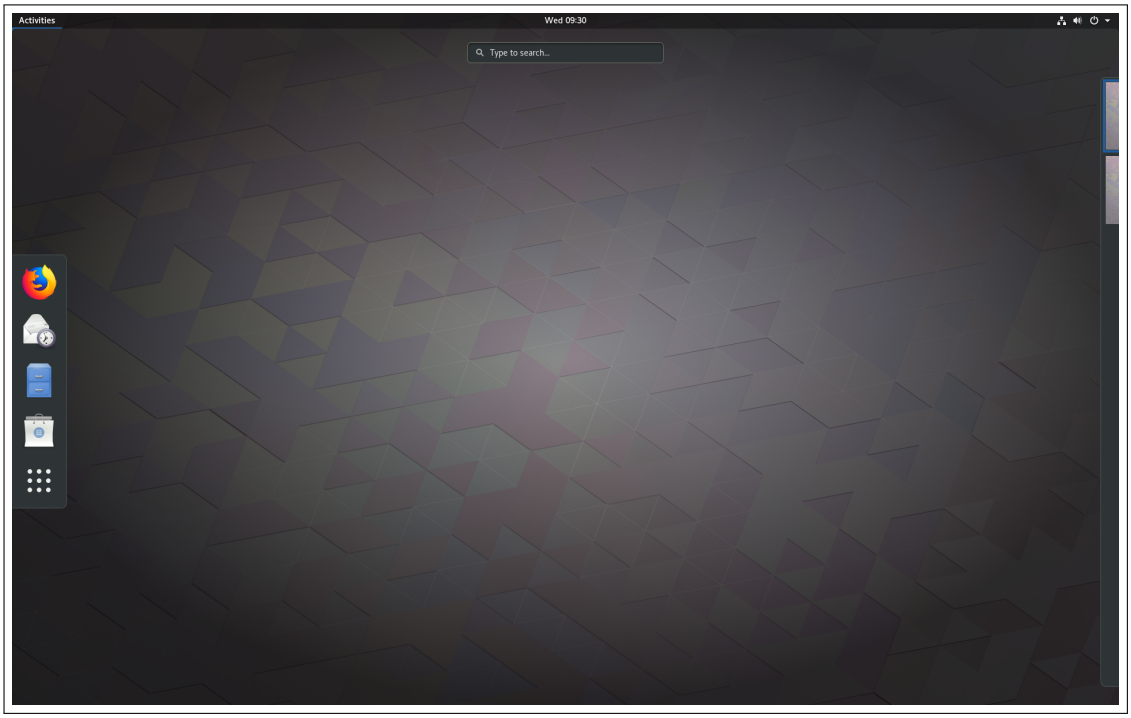

Figure 2: Starting applications

Firefox - the web browser

Evolution - an email client

Files - the Nautilus file browser

Software - Add and remove software [Not configured for standard desktops]

Show Applications - display available applications

To remove items right click on the icon on the Favourites bar and select **Remove from Favourites**. I suggest you remove **Evolution** and **Software** as these won't be needed.

2.2 Adding applications to Favourites

To add an application such as a terminal window or Google Chrome to the Favourites bar, click on **Activities** on the top left corner and then enter the name of the application in the **Type to search** box that appears in the middle of the top of the screen as in Figure 2.

Once you have found the application you want, right click on its icon in the Favourites bar and select **Add to Favourites**. It should now appear in the Favourites bar permanently. If you decide to remove it, right click on it and select **Remove from Favourites**.

If you haven't already, add terminal window to Favourites. I prefer the simple Terminal window, but there are several others.

2.3 Finding files

Open the file browser (it looks like a filing cabinet) and you should see something like Figure 3.

The file browser opens in your home directory. You can explore other locations by selecting 'Other Locations'. Click on 'Computer' to have a look at your computer's files and

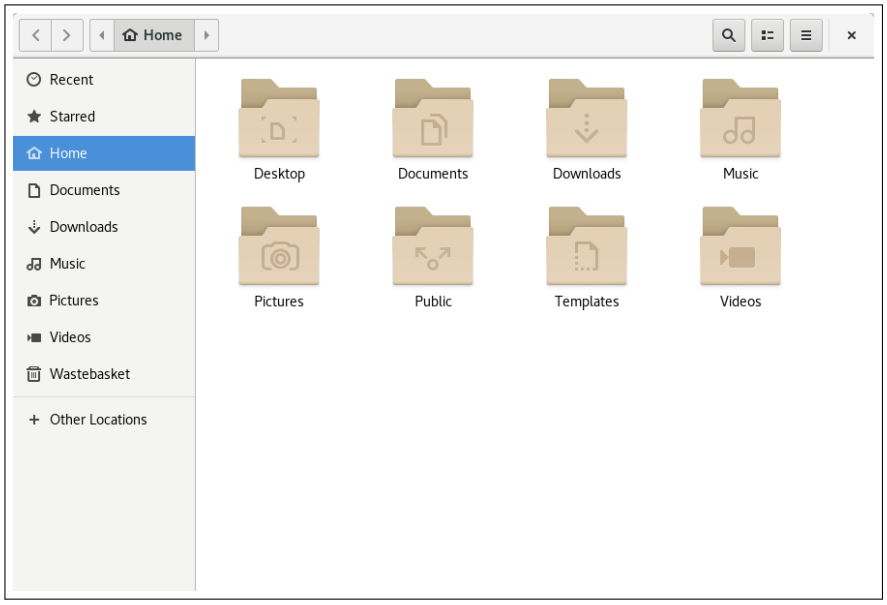

Figure 3: The nautilus file browser.

directories. There may be some places where you can't go or files you can't view. They will have a **X** on them.

On the bar at the top of the Files window towards the right is a small box with a 2x2 grid. Click on this. This box allows you to change the file browser view. One click will make it display more details about each directory. This is called the list view.

See if you can change the list view to include the owner and permissions, and changes 'Modified' to 'Modified Time'. This is done by right clicking by **Name** and selecting options. It should look like Figure 4.

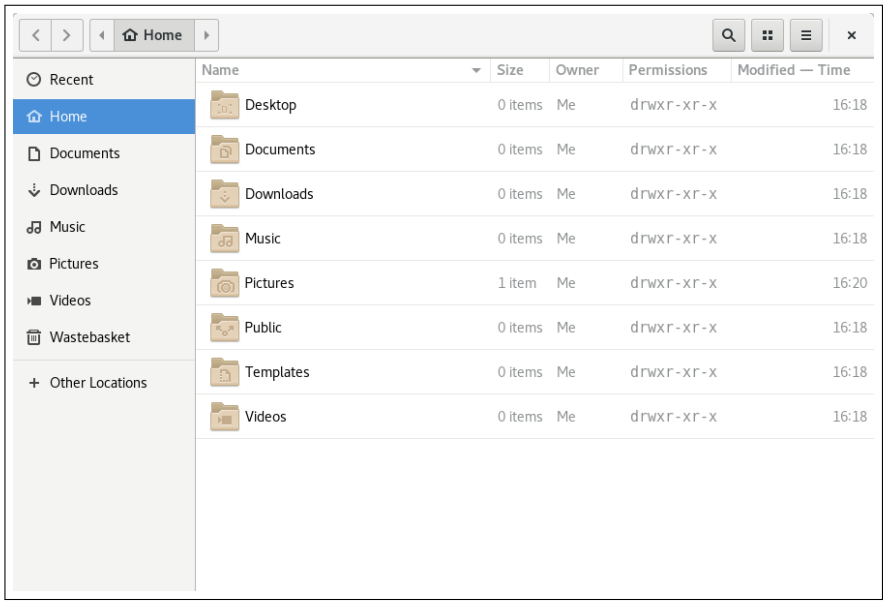

Figure 4: The nautilus file browser – modified.

2.4 Multiple workspaces

If you hit the Windows key (or Super key) you can create new workspaces. If you have never used multiple workspaces before they can take a little getting used to. I find them incredibly useful as they allow me to organise my work into different tasks. See if you can open Firefox in one workspace screen and the file browser in another. To move windows between workspaces, right-click on the bar at the top of the window and select 'Move to Workspace Down' or 'Move to Workspace Up' as appropriate. After pressing the Super key, it's also possible to drag and drop windows between workspaces using the overview on the right side.

As an aside, almost all commands that you start by clicking on an icon can be started using a command instead. If you still have a Firefox browser window open, close it. Now enter this command

```
firefox &
```

2.5 Logging out

On the top right corner is power icon and down arrow. Expand the down arrow selections, click on your name and you should then see an option 'Log Out'.

3 The Linux command line

Before you start

i. Make sure you have added the terminal window to Favourites.
ii. Open a terminal window

Download some files

In a terminal window enter these commands exactly as they appear. The commands will all be explained later.

Download some files and directories which will be used during these exercises. Although it is possible to use a browser to download this file you can also do this from the command line.

```
cd
curl http://www.stats.ox.ac.uk/pub/susan/cdt/CDT.tgz >CDT.tgz
```

to download the files and then

```
tar -xvzf CDT.tgz
```

to unpack them.

and you should start firefox. The `&` will be explained later.

In the following sections, Linux commands are introduced and explained, then examples and exercises are provided to illustrate some commonly used features.

3.1 Where am I?

This exercise explores Linux file system navigation.

Note that all commands are typed in lower case. There are very few Linux commands which have any uppercase (CAPITAL) letters.

You will be using these commands:

Command	**Purpose**
`pwd`	Print working directory. In other words, "where am I?"
`ls [options]` *`directory`*	List files. If used on its own, it lists everything in the current working directory (where you are currently located).
`file` *`filename`*	Tells you what sort of file the file named *filename* (for example) is.
`cd`	Change directory. In other words, change my current location.
`man` *`command`*	Command manual pages.

Table 1: Navigation and file query commands

Right away we will see how quiet Linux commands are by default. Try typing in

```
cd
```

at the prompt and you will get no output at all. This does not mean that anything has gone wrong. For many commands, no output means successful completion.

A digression on prompts. You can customise your prompt to look however you like. We won't do that now, but you may notice that it changes as you move around the file system.

Not all commands are silent. Try

```
pwd
```

You should get a response like: `/homes/`*`user`*. In all examples please replace *`user`* with your username. Now try

```
ls
```

You should now see a listing of all the files in the directory `/homes/`*`user`*. Let's find out about some of the contents of `/homes/`*`user`* using the `file` command.

```
file TestDir
```

The shell tells you that this isn't a regular file, it's a directory. In other words it's a special file which acts as a holder for yet more files (like a folder in Windows).

```
file longfile.txt
longfile.txt:  ISO-8859 text
```

so this file is a simple text file.

3.1.1 Absolute and relative paths

We will make use of the following character shortcuts.

Symbol(s)	Description	Purpose
.	One dot	The current directory.
..	Two dots	The directory above the current directory.
/	A slash or forward slash	The root or top directory.
~	The tilde character	The home directory.

Table 2: Symbols used by `cd`

Let's explore the idea of relative and absolute path names using the cd command. Change into the directory `TestDir`.

```
cd ~/TestDir
pwd
cd ..
pwd
cd ..
pwd
```

and so on until you can't go any further (you won't see an error, you just stop going anywhere). `..` is a special location which means up one level. All directories contain a `..` so you can go up a level. The exception is called `/` or sometimes "the root" or just "slash". You can't go any higher than / so `..` doesn't take you anywhere. Note that there is another special directory called `.` (a single dot) which means "current location".

During the above task you went up the directories one level at a time. Now let's reverse the process and go back to the Desktop directory one level at a time. You should be in "/". You don't need the pwds but it may help you see what is going on.

```
cd homes
pwd
cd user
pwd
cd TestDir
pwd
```

Remember always to replace *user* by your own username.

We're now going to make use of two things, the `ls` command and the knowledge that the file called `/usr/share/icons` is a directory, to further illustrate the concepts of absolute and relative pathnames.

```
cd
cd ../../usr/share
```

```
ls icons
```

and you will get a listing of the contents of the directory.

```
ls /usr/share/icons
```

and you should get the same list of files.

The absolute (i.e. complete) location of the icons directory is `/usr/share/icons`. We have just asked to see what is kept inside it in two different ways. The first is a relative pathname while the second is the full or absolute path.

Imagine the `icons` directory is a particular house, say 42, High Street, Abingdon and I ask you to deliver a letter there. I could tell you to deliver the letter to "42 High Street, Abingdon": the full/absolute address. No matter where you are in the UK, that's enough information. However, if you were already in Abingdon I could tell you to deliver the letter to the relative address of " 42, High Street" or even better, if you were standing on the high street just "number 42" would be enough.

The `ls icons` command worked because you were already in the `/usr/share/` directory. It wouldn't work from another location. The command `ls /usr/share/icons` command will work from anywhere (although it's more long winded). Let's prove it by changing our current location using the cd command.

```
cd
cd TestDir
pwd
```

you should get `/homes/`*`user`*`/TestDir` i.e. you have moved into the TestDir directory.

```
ls /homes/user/TestDir
```

should give you a list of files in that directory. You could use `ls` on its own without the name of the directory because you have already moved there with `cd`. Let's see what happens when we make a mistake:

```
ls TestDir
```

should give you an error saying there is `No such file or directory` which is correct. The command fails because `TestDir` on its own is a relative path and you've started from the wrong place.

Try to answer/do the following:

1. Were you just using absolute or relative paths in the last example?
2. Now try to get back to the root (or `/`) directory with one command only using an absolute path.
3. Now get back to the `/homes/`*`user`*`/TestDir` directory using one command only.
4. What are the contents of the `/` directory? From your home directory use only one command to find out.
5. Make sure you can see your data directory: `/data/`*`host`*`/`*`user`*.

See section 6.1 on page 19 for answers.

Finally, a short cut! In order to change to a directory in your home directory, use the ~ character which represents your home directory to move to `TestDir` from any location.

```
cd ~/TestDir
```

3.2 File and directory manipulation

Now we're going to create a directory and put some files there.

Command	Purpose
`cd`	Change directory – or change my current location.
`mkdir` *directoryname*	Create a directory called *directoryname*.
`touch` *file1 file2*	Create one or more empty file(s) called *file1*, *file2* ...
`cp` *file1 file2*	Copy *file1* to *file2*. Also used to copy directories.
`ls`	List files.
`rm` *file1*	Remove (or delete) a file *file1*. Also remove directories.

Table 3: File and directory manipulation commands

Now execute these commands:

```
cd
mkdir directory1
cd directory1
touch file1 file2 file3 file4
```

Remember that words in italic should be replaced by names that you have chosen. Experiment to see what happens if you are not in your home directory. What happens if you try to create a directory in `/usr/bin`? Is there anywhere outside your home directory where you **are** allowed to create directories? [Hint: look at the top level directory – you should be able to create a files and directories in one of those. The name of the directory might also be a clue.]

Use the `cp` command to copy one file to another and then use `ls` to check that you have done what you want. Then delete a file using

```
rm file1
```

Now we are going to copy one directory to another. The commands you need are

```
cd
cp -r directory1 directory2
```

Use ls to make sure you have done what you want. The new directory should contain exactly the same files as the old one. Note use of the -r option. This makes cp copy

the contents of a directory — this is known as a recursive copy. Finally remove the new directory with

`rm -rf directory2`

Note that this is a dangerous command and should be used with care!

Use `ls` to check that this has worked.

You should now be familiar with these simple file manipulation commands. Remember that in Linux the `rm` command really does delete files. There is no Recycle Bin to retrieve files that were deleted by mistake.

3.3 Viewing files

In these exercises we explore commands to view the contents of files.

Command	Purpose
`cat file`	Show the while contents of a file called *file*.
`more file`	Display the contents of *file* a screenful at a time.
`less file`	Display the contents of *file* a screenful at a time, but with more options. For example, after starting `less` enter `G` to go straight to the end of a file and then move backwards.

Table 4: Viewing the contents of files

Use the following commands to look at the contents of the file `google.txt`.

```
cd
cd Files
cat google.txt
```

This is not very useful if the file is more than a screenful.

```
more google.txt
```

Note that <space> takes you to the next page and `q` will quit before the end of the file. Now try

```
less google.txt
```

See if you can get to the end of the file with a single command. Use google to help if you like. Then use `q` to exit.

Speed things up

By now you may be fed up with all this typing! When using the command line in a terminal window, there are ways to make life easier for you:

Filename and command completion

- <tab> key completes commands and filenames

Arrow keys allow us to:

- recall previous commands
- change previous commands

3.4 Help commands

These exercises explore various ways to get help with and about commands. If you know

Command	Purpose
`man` *command*	Read the manual page for a command. So `man ls` would give details of the `ls` command and `man more` of the `more` command.
`apropos` *word*	Search the manual pages for names and descriptions which contain *word*. So `apropos copy` would list all the commands that have the word `copy` in the description.
`which` *command*	Display the location of the command being used.
`whatis` *command*	Gives a brief description of a command.

Table 5: Wild cards and globbing

what command you need, you can use the `man` command to find out the details of that command. Try it with a few of the commands you have used already. Not all commands have as many options as `ls`!

```
man ls
```

to find out details of the `ls` command.

i. What option is used to display modification time?
ii. What option is used to display the size of a file?
iii. How can you reverse the order of the sort so that the largest/most recently changed file is at the bottom of the list?
iv. Check that they do what you expect.

Sometimes you might not be sure exactly what the command is. In that case you can use the `apropos` command which finds all command descriptions which match a given word. So to find out what commands there are to manipulate files are available use

```
apropos file
```

Note that the output from this command is very long! In a future session I can explain how make this more useful.

Sometimes you need to know where Linux stores command. Use which to display the location of the file. Try it with `less`, `more`, `cp`, `apropos`:

```
which less
which cp
```

```
which R
which pdflatex
```

Did you notice that `R` and `pdflatex` are stored in different places? The `/usr/local` directory is used to share frequently used application so that we can provide a more up-to-date version than that which comes with a standard installation.

Finally you may have seen a command and want to know briefly what it does. Use the `whatis` command to find out. Try this on some commands you have learnt.

4 Logging on to a remote computer

Command	**Purpose**
`ssh` *host*	Log on to a different system, named *host*.
`ssh` *user*`@`*host*`.stats.ox.ac.uk`	Log on to a different system from outside the department, named *host*.
`ssh-keygen`	Generate an ssh public/private key pair to enable moving between computers without entering a password.

Table 6: Logging on to a different system

During the course, you will be using servers to run jobs. You will need to login to these servers to create and start your jobs; and you will need to be able to transfer data between your desktop systems and these servers.

There are five CDT servers. They are

greyheron, greywagtail : 758GB RAM, 96 (hyperthreaded) processors, 16TB /data directory

greypartridge, greyplover : 758GB RAM, 48 (not hyperthreaded) processors, 16TB /data directory

greyostrich : 758GB RAM, 48 (not hyperthreaded) processors,22TB /data directory, four NVIDIA TESLA K80 GPU cards (eight GPUs in total). This server should be used only for GPU work.

I'll refer to these as the `grey*` servers from now on.

From any Statistics computer the short form of the host name can be used. So

```
ssh greywagtail
```

would be used to log on the CDT server `greywagtail`. For all the following examples `greywagtail` has been used, but this can be replaced by any other server.

On each `grey*` server you should find two directories where you can store data:

`/data/`*host*`/oxwasp/oxwasp18/`*user*
`/data/`*host*`/not-backed-up/oxwasp/oxwasp18/`*user*

Data in the first directory is a backed up daily, data in the second, never. There is a system-wide limit of 400GB changed data per day for backups so please, if you are moving a lot of data around, check with other members of the group to make sure they are not doing the same thing.

Try this now. Log onto greywagtail and make sure you can find your data directories.

Note that every time you log onto a server (or any other computer in Statistics) you are prompted for your password. We are going to speed things up now by removing the need for passwords between Statistics computers.

4.1 Remove the password prompt between Statistics systems

Set up ssh keys so that you are not prompted for a password each time you move between Statistics systems. In a terminal window on your desktop do the following:

i. Enter the command

```
ssh-keygen -t rsa
```

Generate a public key/private key pair. The following output should appear

```
Generating public/private rsa key pair.
Enter file in which to save the key (/homes/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /homes/user/.ssh/id_rsa.
Your public key has been saved in /homes/user/.ssh/id_rsa.pub.
The key fingerprint is:
20:30:e6:b0:5f:a0:aa:4f:d0:09:c7:b0:f4:25:45:4e
                                  user@gate.stats.ox.ac.uk
The key's randomart image is:
+--[ RSA 2048]----+
|o.=.oE           |
|.X.+=            |
|= +.o..          |
|.= o . .         |
|o +     S        |
|..               |
|. .              |
| o               |
|  .              |
+-----------------+
```

ii. `cd`

iii. `cd .ssh`

Note the dot before the directory name. Now check whether the file `authorized_keys` exists. If it does use this command:

```
cat id_rsa.pub >>authorized_keys
```

if the file doesn't exist use

```
cat id_rsa.pub >authorized_keys
```

Now check that this works:

```
ssh greywagtail
```

The first time you connect you will see

```
The authenticity of host 'greywagtail (163.1.210.96)' can't be
established.
ECDSA key fingerprint is
3a:b1:d2:0d:a3:09:cf:46:e9:43:04:87:ac:f3:8e:10.
Are you sure you want to continue connecting (yes/no)?
```

Enter `yes`.

Now from `greywagtail`, use `ssh` to log onto `greyplover`. You should not be asked for a password.

5 More advanced command line features

5.1 Looking at parts of a file

We have already used `cat`, `more` and `less` to look at the contents of files. But what if we just want to look at parts of a file, or find out some characteristics of a file such as the number of lines or words?

Command	**Description**
`head` *file*	Display the first 10 lines of a file.
`tail` *file*	Display the last 10 lines of a file.
`wc` *file*	Counts the number of characters, words and lines in a file.

Table 7: Displaying parts of a file

Using

```
cd
cd Advanced
cat longfile.txt
```

you see all of `longfile.txt`; using

```
less longfile.txt
```

gives you a screenful at a time. Use the spacebar to move on a screenful, G to go to the end of the file and q to exit. Now use

```
head longfile.txt
tail longfile.txt
```

to look at the first 10 lines with `head` and last 10 lines with `tail`.

Now use the `wc` command to find out the number of lines, words and characters there are in `longfile.txt`.

5.2 Using wildcards to match file names

File globbing or wildcard expansion allows you to use special characters to match more than one file or directory name.

Character	What it matches
`*`	The * (asterisk or star) matches any number of characters or none.
`?`	The ? (question mark) matches exactly one occurrence of any character.
`[ ]`	Matches any characters in a given range.

Table 8: Wildcard characters

Change to the directory called `WildCards`.

```
cd ~/WildCards
```

Now experiment with wild card characters. What do the following match?

```
ls foo?
ls foo2*
ls foo[1-2]
```

What command would you need to match only the files `foo20` and `foo2bar`? [Hint: you might need to use more than one wild card character.] The answers are at in section 6.1 on page 19. Now use

```
wc -l *
```

to find the length of each file. Note that the -l is a hyphen (-) followed by the lowercase letter `l`. You should see output like this:

```
  1 foo
  2 foo1
  2 foo10
  1 foo2
  1 foo20
  3 foo2bar
 10 total
```

Now see if you can create a match so that `wc -l` just shows the files with a 1 (the number one) in their name. Again a possible answer is at the end.

5.3 Pipes and redirection

In this exercise we are going to explore two very powerful command-line features which increase the flexibility and range enormously. We will use these commands:

Command	Description
`du -sk`	Display the sizes in Kilobytes of all the files in a directory.
`grep`	Search for the characters with a given pattern in file.
`sort`	Sorts the contents of one or more files.
`tail`	View the last few lines of a file.
`wc`	Counts the number of characters, words and lines in a file.

Table 9: Commands used in our pipe

Command	Purpose
`>`	Sends the output from a command to the named file. If the file already exists the previous contents will be lost. If the file doesn't exist it will be created.
`»`	Appends the output from a command to the named file. If the file doesn't exist it will be created.
`<`	Reads input from the named file. NB This option is rarely used.
`|`	Uses the output from one command as the input to the next.

Table 10: Characters used for pipes and redirection

and the following characters:

Almost all Unix/Linux commands use standard input for receiving instructions and standard output for displaying the results. Commands make use of special data streams to move input and output to and from the command. STDIN can be though of as a gateway into the code, STDOUT is a gateway for output. Note that there is a third standard data stream called STDERR (standard error) which commands often use to print error messages and warnings. We won't mention STDERR again today. By default STDOUT gets routed to the screen display. It is also easy to connect a commands STDIN to the keyboard device for example. The STDIN allows a program to ask you questions and you can type responses.

Redirection is a way of "grabbing" STDOUT or STDIN and forcing it to go somewhere other than the default. The most common instance is to redirect STDOUT into a new file. This is extremely useful because it means you can run a command and save the results automatically. To redirect STDOUT use the > symbol followed by the target so command > file.txt redirects the output from command into a new file called file.txt. To redirect STDIN into a command use < like this: command < file.txt.

Connecting commands together with pipes is one of the most powerful features of Linux. Linux does not have a command to count the number of files in a directory but it does have one command to list the files (ls) and a second command (wc) to count the number of lines in a list.

You could therefore use a pipe (|) to glue ls and wc together:

```
ls | wc -l
```

Note that we are using the -l (that is a hyphen followed by the lower case letter l) argument or option for wc. To get information about any command use

`man command` eg man wc

5.4 Some examples

In the directory `Searching` there are three small text files. Check their contents using `cat`. If we wanted to search for all occurrences of "green" in the files `fruit` and `veg` we would use

```
grep green fruit veg
```

The output is displayed on the screen. To store the output from the command in a file use

```
grep green fruit veg > out.txt
```

which would store the output of the `grep` command in a file called `out.txt`. Check this with

```
cat out.txt
```

If you run the command again, this time using

```
grep green fruit veg >> out.txt
```

you should see that there are now two copies of the output in the file. It is also possible to use < redirect the input from the keyboard to a file. For example

```
cat < out.txt
```

also works.

In this short example we can count how many times green appears, but when there are many matches it would be useful to use `wc` to find out. We can redirect the output from the command into a file and then check the length. Note that in all cases the `-l` option to `wc` is a minus sign followed by the lower case letter `l`.

```
grep green fruit veg > out.txt
```

and then run `wc` on the file

```
wc -l out.txt
```

but it would be much more efficient to join the two commands together with a pipe. Use

```
grep green fruit veg | wc -l
```

Here the two commands `grep green fruit veg` and `wc -l` are joined together by a special symbol called a pipe.

Now we're going build a longer command which will find the 5 largest files in a directory. When building pipes of commands it often helps if you make sure each link in the pipe works before adding the next. First display the size of all the files in the `/usr/bin` directory.

```
du -sk /usr/bin/*
```

Now sort this output by size, so that the largest are first.

```
du -sk /usr/bin/* | sort -nr
```

Now display the final 5 lines which will be the 5 largest files.

```
du -sk /usr/bin/* | sort -nr | head -5
```

Now how you would you find the 5 largest files in the `/usr/bin` directory beginning with the letter 's'? See section 6.1 on page 19 for a solution.

5.5 Running commands in the background

When you start a command such as `rstudio` from a terminal window it is good practice to add an & (ampersand) character after the command to keep access to the command line. Compare Figure 5 and Figure 6.

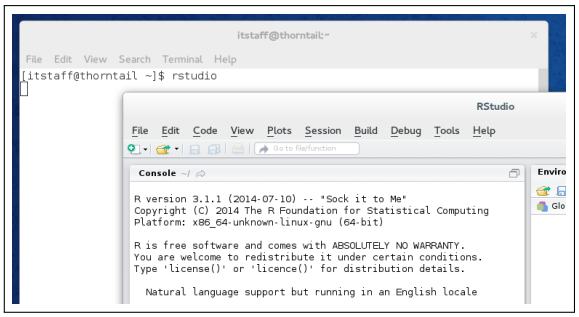

Figure 5: Starting `rstudio` without &.

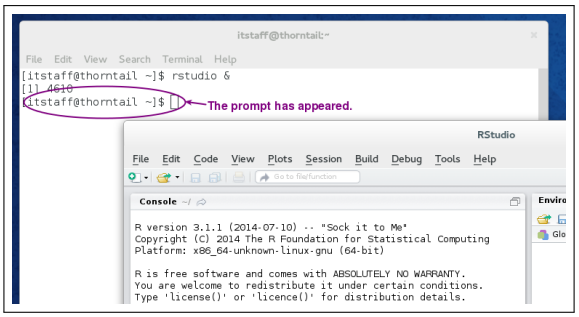

Figure 6: Starting `rstudio` with &.

5.6 Where should I store things?

There are two places where files should be stored:

- Your home directory
- The local disk

The contents of these folders can be seen either by using the **Files** application from the Favourites bar or from a terminal window with the commands

`cd` Change to your home directory

`cd /data/host/user` where `host` is the name of your computer and `user` is your username. So if your computer is `treeowl` and your user is `smith` then you would use `cd /data/treeowl/smith`.

Note that if you are using the **Files** application, once you've found the location you want, you can right-click and select **Open in Terminal** as in Figure 7 which will then open a terminal window which starts in that directory.

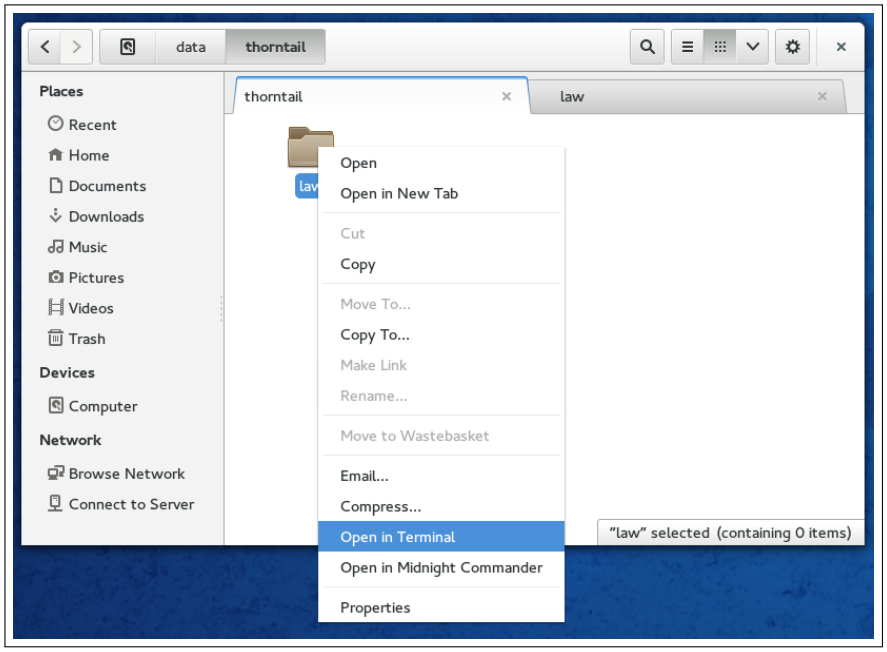

Figure 7: Opening a terminal from **Files**.

6 Conclusion

In future sessions we will look more at remote access to the grey* servers. If there are other topics that you would like covered, please let me know.e

6.1 Linux answers

Absolute and relative paths

1. A relative path – which is why it didn't work.
2. `cd /`
3. `cd /homes/`*`user`*`/Testdir` or `cd ~/Testdir`.
4. `ls /` will display the contents of the root directory.
5. Use `ls`.

Using wildcards to match file names

`ls foo*` matches

```
foo1 foo2
```

`ls foo2*` matches

```
foo2 foo20 foo2bar
```

`ls foo[1-2]` matches

```
foo1 foo2
```

To match just the files `foo20` and `foo2bar` use

```
ls foo2?*
```

To use `wc -l` just to display the files with a 1 in their name

```
wc -l *1*
```

Sorting and searching – some examples

To find the 5 largest files beginning with 's' in the `/usr/bin` directory use

```
du -sk /usr/bin/s* | sort -n | tail -5
```