1 Using the command line

1.1 Adding applications to Favourites

Make sure you have added the terminal window to Favourites.

To add an application such as a terminal window to the left side Favourites bar, click on the **Activities** button on the top left corner and then enter the name of the application in **Type to search** box that appears in the middle of the top of the screen as in Figure 1. Once you have found the application you want, right

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

click on it 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**.

1.2 Speeding things up

Don't forget that 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

1.3 Where am I?

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 called *filename* (for example) is.
cd	Change directory. In other words, change my current location.
man *command*	Command manual pages.

Table 1: First commands

Right away we can 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/jones`. In all examples please replace `jones` with your username. Now try

```
ls
```

You should now see a listing of all the files in the directory `/homes/jones`. Let's try finding out about some of the files. Take the file `Desktop`.

```
file Desktop
```

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 WELCOME.txt
WELCOME.txt: ASCII text
```

so this file is a simple text file.

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

```
cd
cd ../../usr/share
ls info
```

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

```
ls /usr/share/info
```

and you should get the same list of files.

The absolute (i.e. complete) location of the info directory is **/usr/share/info**. 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 name. Imagine the **info** 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 info`** command worked because you were already in the **`/usr/share/directory`**. It wouldn't work from somewhere else. The command **`ls /usr/share/info`** 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 Desktop
pwd
```

you should get **`/homes/jones/Desktop`** i.e. you have moved into the Desktop directory.

```
ls /homes/jones/Desktop
```

should give you the list of files in that directory. In fact 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 deliberately do something wrong:

```
ls Desktop
```

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

Let's expand the idea of relative and absolute path names using the cd command. Make sure you are still in the Desktop directory before you start (check with **`pwd`**).

```
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 "/". Note that you don't have to do the pwds but it may help you visualize what is going on. You can also use **`ls`** to have a look around each level if you have time.

```
cd homes
pwd
cd jones
pwd
cd Desktop
pwd
```

Try to answer/do the following: Were you just using absolute or relative paths?

1. Now try to get back to the root (or `/`) directory with one command only using an absolute path.
2. Now get back to the `/homes/jones/Desktop` directory using one command only.
3. What are the contents of the `/` directory? From your home directory use one command only to find out.

1.4 File and directory manipulation

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

Command	Purpose
cd	Change directory. In other words, 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*. Can also be used to copy whole directories.
ls	List files. If used on its own, it lists everything in the current working directory (where you are currently located).
rm *file1*	Remove (or delete) a file called *file1*. Can also be used to remove whole directories.

Table 2: File and directory manipulation 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 âĂŞ 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.

1.5 Viewing files

We're going to 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.

wget http://www.stats.ox.ac.uk/pub/susan/linux/Files.tgz

to download the files and then

```
tar -xvzf Files.tgz
```

to unpack them.

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 3: Viewing the contents of files

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

```
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. Then use **q** to exit.

1.6 Help commands

Command	Purpose
man *command*	Read the manual page for a command. So **man ls** would give details of the **man** 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 4: Finding out about commands

If you know 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.

1. What option is used to display modification time?
2. What option is used to display the size of a file?
3. How can you reverse the order of the sort so that the largest/most recently changed file is at the bottom of the list?
4. 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.

1.7 Logging on to a remote machine

Command	Purpose
ssh *hostname*	Log on to a different system.

Table 5: Logging on to a different system

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

```
ssh greyheron
```

would be used. The four CDT "grey" servers are **greyheron, grepartridge, greyplover, greywagtail.** If you need access to Statistics servers from outside the department either use

```
ssh gate.stats.ox.ac.uk
```

from a terminal window (or PuTTy on Windows) and then

```
ssh greyheron
```

or connect to the VPN and use **ssh greyheron.stats.ox.ac.uk**. Note that it is possible to set up ssh keys so that you are not prompted for a password each time, but it is beyond the scope of this short introduction.

Once on a different system you will have access to your file in your `/homes` directory but *not* files in your `/data/`*hostname*`/`*username* directory.

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

`/data/`*hostname*`/`*username*
`/data/`*hostname*`/not-backed-up/`*username*

Data in the first directory is a backed up daily, data in the second, never. However there is a system-wide limit of 300GB changed data per day for backups so please, if you are moving a lot of data around then check with other members of the group to make sure they are not doing the same thing.

1.8 Submitting jobs on a remote machine

Once you have logged into a remote system you may well want to submit jobs that will last several hours, possibly several days.

Command	Purpose
`screen`	Connect and disconnect from a session from multiple locations and allow long-running processes to persist without an active shell session.

Table 6: The screen command

Start the **`screen`** command and run whatever command you need to run.

Once the job is running you can then use the sequence

CTRL-a `d`

to detach from the screen process. You can then logout. To reattach the screen session use

`screen -r`

There is a longer **`screen`** tutorial here: http://www.rackaid.com/blog/linux-screen-tutorial-and-how-to/.

1.9 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 2 and Figure 3.

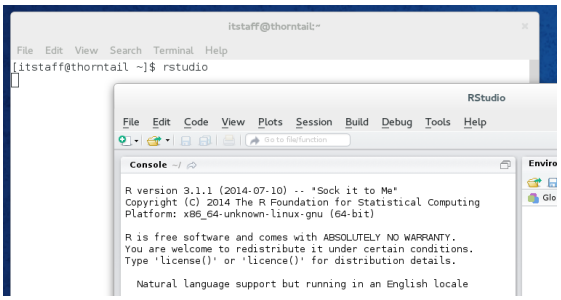

Figure 2: Starting **`rstudio`** without &.

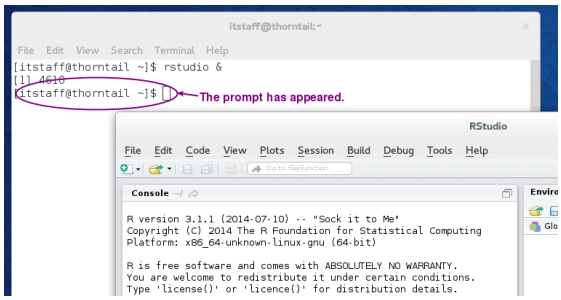

Figure 3: Starting **`rstudio`** with &.