Introduction to Linux

Some of this material borrows very heavily from the excellent BioLinux tutorial here. Both BioLinux and CloudBioLinux are used extensively in this training. Some other parts of the material borrow from slides provided by Andy Law. Please take some time to read Law's Laws, which are amusing.

Don't be afraid. Be brave. Here is the Linux command-line:

Using the command line
The real power on Linux/Unix systems is the command line. Many programs and facilities are available through graphical options on Linux, but all programs and facilities can be accessed by the command line. Sometimes it is easier to do things through graphical interfaces, but sometimes it is easier using the command line. This is especially true when you start to work with large numbers of files or are considering automating processes.

1. The command ls lists files in a directory.
By default, the command will list the filenames of the files in your current working directory. At the moment, this is probably your home directory.

If you add a space followed by a **–l** after the **ls** command, it alters the behavior of the command – it will now list the files in your current directory, but with details about them including who owns them, what the size is, what kind of file it is, etc. The full meaning of the information returned to you when using the –l option with the ls command will be explained during the course.

Exercise
Type the command ls
Type the command ls –l

What do you see that is different?

2. The command man provides help for a command.

There are many options you can provide with the ls command that modify what kind of information is returned to you. By typing **man ls** you get access to the manual page for this command. Almost all Linux commands have a manual page, and it is worth referring to them to find out what options are available. Many jobs can be made easier by using the right command options.

Open the help for the command **ls** by typing **man ls** and look at some of the information provided. Close the man page by typing the letter q.

If you do not know the name of a command to use for a particular job, you can search using man –k. For example, **man –k list** gives a list of a number of commands that have the word “list” in their description. You can scroll through these, or try making the search more specific. For example: **man –k “list directory”**

will return only four commands. You could then look at the man pages for each command to decide which was best for the job at hand.

Exercise
Type the command **man cp** - what does this tell you?

Type the command **man mv** - what does this tell you?

Type the command **man rm** - what does this tell you?

What is the difference between **cp** and **mv**?

Remember to hit "q" to exit from the command.

---

3. Basic Linux/Unix tips for filenames

a.) Certain characters should not be used in filenames in Linux/Unix.
If you stick with letters, numbers, hyphens, underscores and full stops, you will be fine.

b.) Linux/Unix cannot deal with spaces in filenames!
Make sure your filenames do not contain them. Filenames with spaces in them are a common problem when transferring files to Linux/Unix from computers running Windows, or Mac operating systems. If you end up with filenames with spaces in them, you will need to enclose the entire filename in quotation marks so that Linux/Unix understands that the space is part of the name.

Alternatively, you can “escape” the space using a backslash. For example, if I have a file called **"my document"** Linux/Unix will see this as two filenames, “my” and “document”.
But you could write either of the following to make it understand you mean a single file:

- “my document”
- my\ document

We advise that you change the name of such files to remove the space.

c.) Assume that everything is case specific
Linux/Unix systems consider capital letters different from lower case letters. The filename **myFile** is not the same as the filename **Myfile** or **myfile**.

---

4. Changing directories
The command used to change directories is **cd**. If you think of your directory structure, (i.e. this set of nested file folders you are in), as a tree structure, then the simplest directory change you can do is move into a directory directly above or below the one you are in.

To change to a directory one below you are in, just use the cd command followed by the subdirectory name:

cd subdir_name

To change directory to the one above the one your are in, use the shorthand for “the directory above” ..

cd ..

If you need to change directories to one far away on the system, you could explicitly state the full path:

cd /usr/local/bin

If you wish to return to your home directory at any time, just type cd by itself.

cd

If you get lost and want to confirm where in the directory structure you are, use the **pwd** command (push working directory). This will return the full path of the directory you are currently in.

Exercise
Change directory from your home directory to the directory training by typing **cd training**

Type **pwd** to see the full path to where you are.

Type **ls** to see which files are in the directory

Type **cd rnaseq**

Type **pwd** to see the full path to where you are

Type **ls -l** to see what is in this directory

Go back to your home directory by typing **cd**

Change directory into the /usr/local/bioinf directory by typing **cd /usr/local/bioinf**

List the files in this directory. Many of the programs in /usr/local/bin are bioinformatics programs.

Go back to your home directory by typing **cd**

---

5. Tab completion
Tab completion is an incredibly useful facility for working on the command line. One thing tab completion does is complete the filename or program name you want, saving huge amounts of typing time.

For example, from your home directory, you could type:

cd tra

and hit the tab key. If there is only one directory with a name starting with the letters “tra”, the rest of the name will be completed for you. Here this would give you:

cd training

Exercise
Return to your home directory if you are not already there by typing **cd**
Type **cd tra** and use tab completion for the rest of the command.
You will now be in the training directory.

Type **cd /usr/local/** and use tab completion for the rest of the command.

What happens?

---

6. Command history and filename completion

Using the functionality of your command history and command line can save you a lot of typing! If you use the up arrow key when you are at the prompt in your terminal, you can see previous commands you have run. This is particularly useful if you have mistyped something and want to edit the command without writing the whole command out again.

You can view past commands using the command **history**.

By default, **history** will return a list of all of the commands run. You can add a number as a parameter to the command to ask for longer or shorter lists. For example, to return the last 30 commands run, you would type:

history 30

To re-run a command listed by the history command, you can just type the command number, preceded by an exclamation mark. E.g.

!12

Exercise
Type **history**
Run one of your previous commands using ! followed by the number of the command.

---

7. Making a directory
To make a new directory, you use the command mkdir (make directory). For example:

mkdir newdir

would create a new directory called newdir.

Exercise
Start in your home directory (which command will take you to your home directory?)
Make a new directory called testdir
Move into the new directory testdir by typing **cd testdir**.
Using information in the Linux/Unix shorthand and shortcuts section, try to move back into your home directory

---

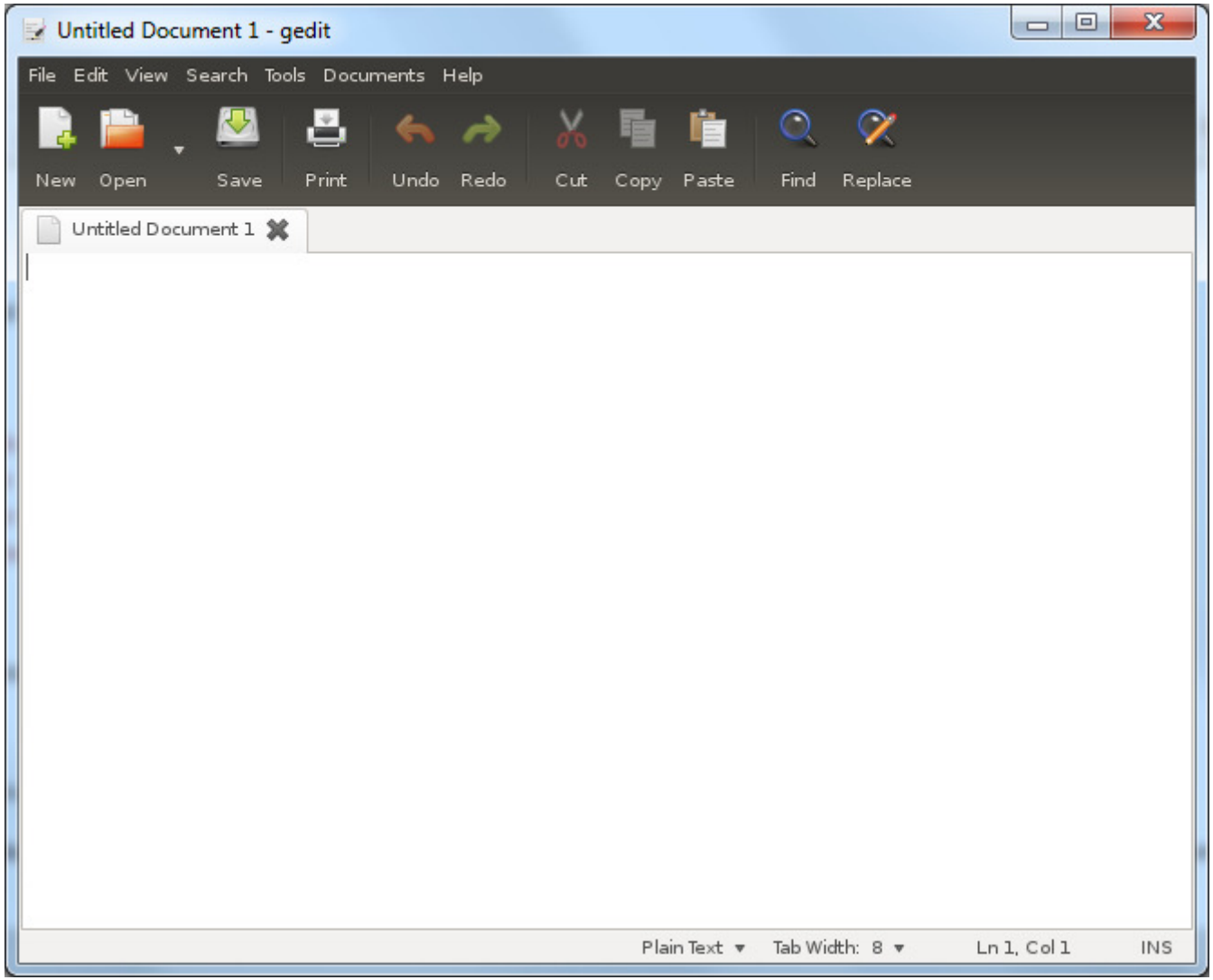
8. Using text editors

One of the text editors on your instance is called **gedit**. We can start gedit by typing:

gedit &

The & ensures that the command is run "in the background", and you can continue to work in your terminal whilst the editor is running.

Opening up the text editor may take some time as the server you are using is in the US, and the PC you are accessing it from is right in front of you. Eventually you will see this:



Note the editor may seem slow and unresponsive. This is an internet issue, not a linux issue. If you were using a local linux server, this would be as fast, or faster, than Windows.

Type some text into the editor, click "Save", Choose **ubuntu** in the "Places" section, give the file the name **myfile.txt** and hit save.

9. Reading text files

There are many commands available for reading text files on Linux/Unix. Among the most common are **cat**, **more**, and **less**.

The command **cat** can be used for concatenating files and reading files into other programs, so it is a very useful facility. However, **cat** streams the entire contents of a file to your terminal and is thus not that useful for reading long files. The **more** and **less** commands show the contents of a file one page at a time. **less** has more functionality than **more**!!

With both **more** and **less**, you can use the space bar to scroll down the page, and typing the letter q causes the program to quit – returning you to your command line prompt.

In addition, once you are reading a document with more or less, typing a forward slash / will start a prompt at the bottom of the page, and you can then type in text that is searched for below the point in the document you were at. Typing in a ? also searches for a text string you enter, but it searches in the document above the point you were at. Hitting the n key during a search looks for the next instance of that text in the file.

With less (but not more), you can use the arrow keys to scroll up and down the page, and the b key to move back up the document if you wish to.

Exercise

Read the file **/usr/local/bioinf/staden/staden-ubuntu-1-7-0/course/data/mutations/HS14680.embl** using the commands **cat**, **more** and **less**. Don't forget that tab completion can save you typing effort.
Use the spacebar to scroll down
Press q to quit.
Press the / key and search for the letters ovarian in the file.
Press the ? key and search for the letters BRCA in the file.
Press the n key to search for other instances of BRCA in the file.
There are many command line options available for each of the above commands, as well as functionality we do not cover here. To read more about them, consult the manual pages:

man cat
man more
man less

---

10. Copying files

The basic command used to copy files using the command line is cp. At a minimum, you must also specify the name of the file(s) to be copied, and the destination location. The main things to know about using the cp command are:

- if you provide a directory name as the last argument to the command, the files will be copied into that directory

- if you provide a name that cannot be found from your current working directory as the last argument to the command, it will be assumed that this is the filename you wish the copied file to be called.
- if you provide multiple filenames to cp, the final filename provided needs to be the name of a directory that already exists – all the files will be copied into this directory.

cp firstfile destinationdir copies firstfile to a directory called destinationdir
cp file1 file2 file3 location copies file1, file2 and file3 to a directory called location
cp destdir/* location copies all files in the directory called destdir to the directory called location

To move whole directories, with all the subfiles and subdirectories, use the –R option, (meaning recursive).

cp –R mydir location

This command means "Copy all files/directories under mydir to the directory called location".

Also useful is the shorthand for someone's home account. e.g. instead of having to know and type the location of their account, you can use ~username. In the case of your own account, you need only use ~

cp ~user2/somefile .

Means "copy the file somefile from user2's home directory here. Note that user2 would have to have given you permission to do this!

cp ~/somedir/somefile .

Means "copy the file somefile from within my account/somedir to here."

Exercise

Copy the file **myfile.txt** to another file called **myfile2.txt**

Copy the file **myfile.txt** to the directory you created, **testdir**
Move into your directory testdir.
List the files in this directory.

---

11. Removing files and directories

To remove a file or files, use the rm command followed by the name of the file(s) you wish to delete.

rm file1
rm file2 file3 file4
rm cat* (remove all files starting with the letters cat)

To remove an empty directory, you can use the rmdir command:

rmdir thisdir

If that directory contains any files, you will not able to delete the directory using rmdir until you have deleted all the files within it.

To delete a directory and all the files in it, use the rm command with the option -r (recursive)

rm –r fulldir

Note

When working with the Linux command-line, there is no recycle bin or undo button. Once it is deleted, it's gone and there is nothing you can do.

Be very careful with the rm and rmdir commands!

Exercise
Move into the testdir directory.
Delete myfile2.txt
Move back into your home directory.
Delete testdir

---

12. Piping and outputting to files
An incredibly powerful facility on Linux/Unix systems is the ability to take the output of one command and use it directly as the input to another command. This is referred to as “piping” the output of one command into another
command. The symbol used for this is called a pipe and looks like: |

An example of when you might use a pipe would be if you wanted to list all the files in a directory, but there are too many to fit on a single page. This is probably what you saw when you listed the contents of /usr/local/bioinf in an exercise earlier. You can pipe the output of the ls command (a list of files) into the less command, which will allow you to view the list page by page. To list the files in /usr/local/bioinf and view them page by page, the command would be written:

ls /usr/local/bioinf | less

A useful little command is the wc command, which stands for wordcount. By default, wc returns the number of newlines, words and bytes in a file (or in information given to it via a pipe). Using command line options, you can get wc to return just the number of lines, just the number of words or just the number of bytes. There are other options available for obtaining information from a file that can be found by reading the manual page for wc. For example, you could find out how many files you had in a directory by typing:

ls | wc -l

---

13. Grep

In this section, we look briefly at two very useful commands: grep and sort. As with all the commands covered today, we recommend that you read the manual page for more information about how these work and what options are available.

grep

grep stands for global regular expression, which is a not very intuitive term telling you that you use this command to search for text patterns in a file (or list), and you can use flexible search terms, known as regular expressions, in your searching. **grep** requires a regular expression as input, and returns all the lines containing that pattern to you as output. grep is especially useful in combination with pipes as you can search through the results of other commands. For example, perhaps you only want to see only the information in an embl file relating to the origin of the sequence, that is, the DE line. You do not need to open the file, you can just cat it and grep for lines beginning in DE.

grep DE *filename*

Will print out all lines in filename that match DE

Adding "-v" to the grep command inverts the logic - so grep will print out lines that *don't* match the regular expression. For example:

grep -v DE *filename*

Will print out all lines in filename that *do not* match DE.

Exercise

Move to your home directory

Read the manual page for grep

Use the grep command to find all lines in /usr/local/bioinf/staden/staden-ubuntu-1-7-0/course/data/mutations/HS14680.embl that contain the word "gene"

Try the command **cat /usr/local/bioinf/staden/staden-ubuntu-1-7-0/course/data/mutations/HS14680.embl | grep gene**

What is the difference?

Try the command **grep DE /usr/local/bioinf/staden/staden-ubuntu-1-7-0/course/data/mutations/HS14680.embl**

Try the command **grep ^DE /usr/local/bioinf/staden/staden-ubuntu-1-7-0/course/data/mutations/HS14680.embl**

What is the difference?

Try the command **grep -v DE /usr/local/bioinf/staden/staden-ubuntu-1-7-0/course/data/mutations/HS14680.embl**

The ^ symbol means “at the beginning of a line”. The $ symbol can be used similarly to mean “at the end of a line”. There are many useful commands available on Linux and we cannot begin to cover them in this course. We
recommend that you consider buying a book to help you learn how to use Linux efficiently.

---

14. What permissions mean

Under the Linux/Unix operating system, there is a very sophisticated method of determining who is allowed to look at and read files and directories. More sepcifically, permissions are split up into **user**, **group** and **others**

- User: this is you, the user logged into the machine
- Group: this is a group of users. You may be in several groups, a long with other people who make a logical group
- Others: this is everyone else

Each file will be owned by **one user** and **one group**.

Then, to each of "User", "Group" and "Others" we can grant **r**, **w** and **x** priveleges:

- r: allowed to read the file
- w: allowed to edit, or write, to the file (also allowed to delete etc)
- x: allowed to execute the file

If you type:

cd

ls -l

You should see a listing of your home directory. You may see something similar to this:

```
drwxrwxr-x 5 ubuntu ubuntu   4096 Oct 26 13:19 install
lrwxrwxrwx 1 root   root       21 Jun 18 11:07 lib -> ../../usr/proftpd/lib
lrwxrwxrwx 1 root   root       25 Jun 18 11:07 libexec -> ../../usr/proftpd/libexec
lrwxrwxrwx 1 root   root       20 Jun 18 11:06 logs -> ../../usr/nginx/logs
-rw-rw-r-- 1 ubuntu ubuntu     34 Nov 26 16:42 myfile.txt
drwxrwxr-x 3 ubuntu ubuntu   4096 Oct 26 13:26 R
drwxr-xr-x 2 root   root     4096 Jun 18 11:07 sbin
lrwxrwxrwx 1 root   root       23 Jun 18 11:07 share -> ../../usr/proftpd/share
-rw-r--r-- 1 ubuntu ubuntu 990263 Oct 22 09:38 snappy-java-1.0.3-rc3.jar
drwxrwxr-x 2 ubuntu ubuntu   4096 Jun 18 15:10 tmp
```

```
drwxrwxr-x 2 ubuntu ubuntu   4096 Oct 26 13:46 training
lrwxrwxrwx 1 root   root       21 Jun 18 11:07 var -> ../../usr/proftpd/var
```

Now, what does all of this mean? Well, the very first column of data refers to the permissions. Let's take "myfile.txt" as an example:

```
-rw-rw-r-- 1 ubuntu ubuntu     34 Nov 26 16:42 myfile.txt
```

- The dash at the start inidcates that this is a file - if it was a "d" it would be a directory, if it was an "l" it would be a link (or shortcut).
- The next three letters are the **user** permissions. We have **rw-** which means that the user can read the file, can edit/write the file, but cannot execute it
- The next three letters are the **group** permissions. Again, we have **rw-** which means that members of the group can read the file, can edit/write the file, but cannot execute it
- The next three letters are the **others** permissions. Here we have **r--** which means that everyone else can read the file, but cannot edit/write or execute it

The next two columns show the **user** (=ubuntu) and **group** (=ubuntu).

To change the permissions of files, we can use the **chmod** command

- The **chmod** command uses the letters u, g and o to represent user, group and others.
- The **chmod** command uses the letters r, w and x to represent read, write and execute.

We can then combine these like so:

- **chmod u=rw *filename*** gives the user read/write access to filename
- **chmod o=rwx *filename*** gives everyone else (others) read, write and execute permissions to filename

This may seem complex at first but this will become easier with practice.

Exercises

Make the file ‘myfile.txt’ readable and writeable by ‘you’ and your ‘group’ but only readable for the ‘rest of the world’
Make the file ‘myfile.txt’ read-only for ‘you’ and with no permissions for anyone else

---

15. Head and tail

The commands **head** and **tail** show the top and bottom of files respectively. Both take a -n parameter that explains how many lines to show:

head *filename*

head -n 20 *filename*

tail *filename*

tail -n 20 *filename*

Exercises

Use **head** to list the top of the **/usr/local/bioinf/sampledata/nucleotide_seqs/seq_format_examples/sulfolobus_solfataricus_pol1.embl** file

Use **tail** to list the top of the **/usr/local/bioinf/sampledata/nucleotide_seqs/seq_format_examples/sulfolobus_solfataricus_pol1.embl** file

Repeat the above, but this time get the top 20 and bottom 20 lines

Use **head**, the **|** symbol and **wc -l** to figure out how many lines head prints by default

---

16. Redirection

Redirection is a way of sending data to and from files. In this section we will deal with the former - i.e. what do you do when you want to save the output of a Linux command? This is done using the **>** symbol. The format of the command is:

command > output.txt

So whatever the results of command are, they will be saved into output.txt. **WARNING: if output.txt already exists, it will be over-written**.

We can also append to existing files using **>>**:

command >> output.txt

In this case, the results of command will be appended to the end of output.txt.

A note of caution: the > character is used in biology, in particular at the start of fasta headers. If you find yourself wishing to search for > in a file, please ensure that you put it in quotes e.g.:

grep ">" seqs.fasta

This will search for instances of ">" in seqs.fasta. Without the quotes, the results of running grep will be written in to your seqs.fasta file, and the original contents destroyed!

Exercises

Change to your home directory by executing **cd**

Run the following command:

ls -l /usr/local/bioinf/ > bioinf_listing.txt

Look at what's in **bioinf_listing.txt** using **less**

Run the following commands in sequence:

date >> bioinf_listing.txt

cat /usr/local/bioinf/sampledata/nucleotide_seqs/seq_format_examples/sulfolobus_solfataricus_pol1.embl >> bioinf_listing.txt

Look at the file **bioinf_listing.txt** using **less**

Run the following command:

echo 'oops' > bioinf_listing.txt

Look at the file **bioinf_listing.txt** using **less**

---

17. Working with zipped data

Often in bioinformatics, and increasingly with NGS data, we find ourselves working with zipped data. This cannot be read directly and needs to be processed. Here are some simple commands to deal with different types of zipped data.

Working with .tar.gz files (sometimes names .tgz)

A tape archive, or tar file, is a common format in Linux whereby entire directory structures and all of the files within them have been placed into a single file. The fact that we also have a gz extension means that this .tar file has then been zipped as well! The easiest way to access the contents of this is to execute:

gunzip < *filename.tar.gz* | tar xvf -

gunzip < *filename.tgz* | tar xvf -

Once executed, look in the current directory and you should see the contents of the .tar.gz file

Working with just .gz files

There are two options here - unzip the file completely, or look at the contents whilst leaving it zipped. To unzip the contents, type:

gunzip *filename.gz*

To look at the contents, we can use a version of cat called **zcat**:

zcat *filename.gz*

Often you will want to combine this with **more** or **less:**

zcat *filename.gz* | less

Exercise

cd to your home directory

Use **zcat** to display the contents of **training/rnaseq/ERR022486_chr22_read1.fastq.gz** (Press Ctrl-C when you get bored)

Use **zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | less** to access the data in a more controlled fashion

Execute the following command:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | grep ^@

What is it doing? Have I made a mistake? Try:

zcat training/rnaseq/ERR022486_chr22_read1.fastq.gz | grep ^@ERR

Do you think this is this doing what I want now?

How could you combine the above command with **wc -l** to count the number of reads in the file?

---

18. Some other useful information

A very basic way to stop a process
Sometimes a command or program goes on too long, or is obviously doing something you did not plan. If this is not a situation where there is an obvious way (e.g. a menu option) to stop the program running, try using Control-c i.e. press the Control key and c-key at the same time.

Logging out of a session
To exit your terminal simply type "exit" (don't do this until the end of the day!)

Clearing your terminal of text
Your terminal windows can fill up with lots of text, and it can become difficult to see the information you wantbecause of all the clutter. You can clear the terminal window of all previous text by typing

clear

The result is a prompt in a nice clean window.

Copying and pasting text
From within putty, highlight the text you wish to copy, then right-click the mouse where you want the text to go - simple!

From windows to Linux(Putty), highlight the text in windows, press ctrl-c to copy the text, go to your putty window and right-click the mouse where you want the text to go

---

Next - Linux and Bioinformatics

Our Sponsors

ROSLIN

ROSLIN FOUNDATION

BBSRC
bioscience for the future

Technology Strategy Board
Driving Innovation

EUROPE & SCOTLAND

Privacy and cookies policy