Introduction to
Sequence Informatics

Institute of Infection and Global Health
Introduction to sequence informatics
Session 1 - Introduction to the Unix Command Line

If you haven't used linux before it can appear daunting, but at its core it is no different from using a Windows or Mac computer. You have files, folders and programs that can be run and most of the time you can do so using a mouse and graphical interface. Unlike Windows or OSX, linux is free and open source. OSX on Mac and linux started out from a common set of code and still share a lot of similarities, but have also diverged significantly over the years.

The major difference with linux, however, is that it is still common to use the command line (it is also possible to do this on a Mac if you wish). The command line is a text only interface, accessed through a Terminal. It provides a number of advantages.

Firstly, it is lightweight and uses very little computing power, there are no graphical elements that the computer is continually having to spend resources on. This is especially useful for servers such as the one that you are using, which may have multiple people logged in at the same time and who would otherwise each require their own graphics being streamed over a network connection.

Secondly, it provides direct access to program options, you can specify the variables for each command and chain multiple commands together using a variety of shortcuts. While this can seem daunting at first, with practise it provides an extremely powerful set of tools.

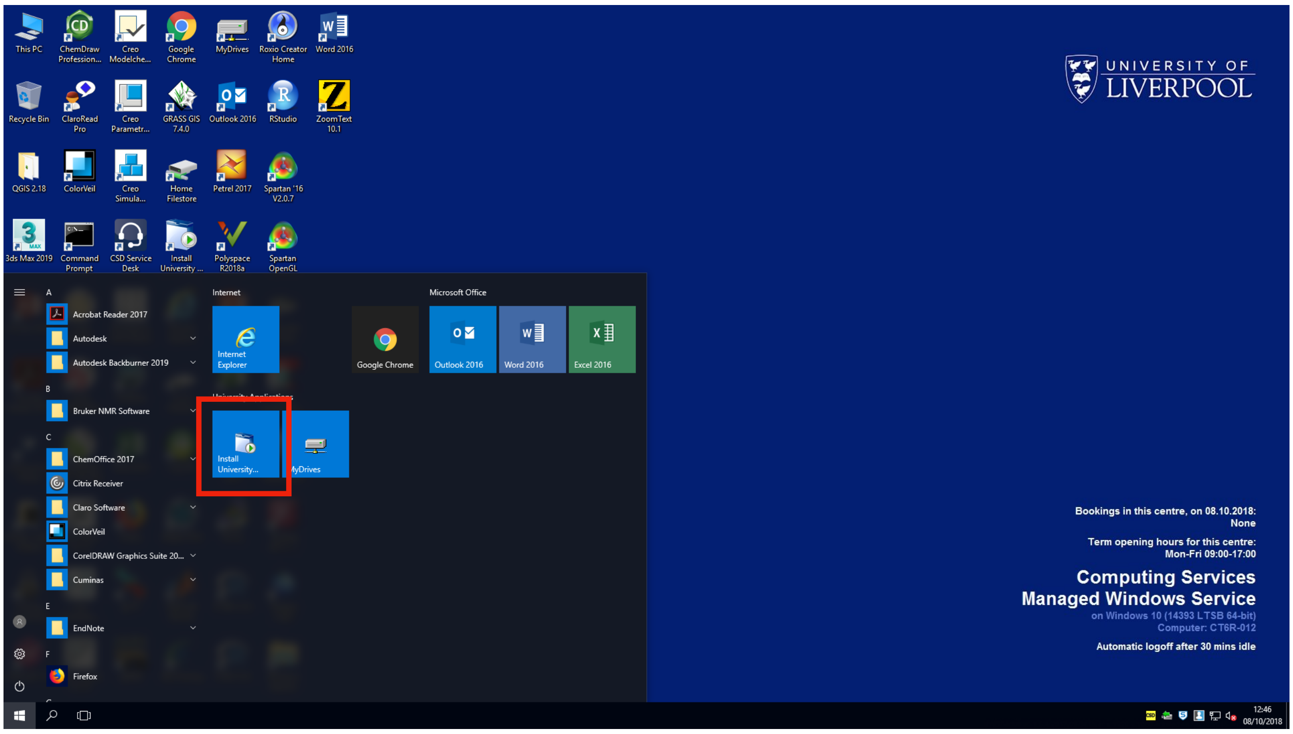
Finally, the command line gives direct access to a number of scripting languages, allowing commands and functions to be combined together. With a single line it is possible to take a set of input data, analyse it with a program and filter the results down to only those of interest. During this session we are going to cover some basic commands, these will allow you to move from folder to folder, create and delete files and run some of the most common commands. Throughout this course you will see some text highlighted in bold, these are

Commands to type into the terminal window

Logging in to the terminal

For the duration of this course you will be using a linux server located elsewhere in the University. As the computers you are using run Windows we will be using a piece of software called MobXterm. This combines two useful features - ssh and X11 forwarding.

Ssh is a network protocol that can be used to securely access a remote computer. In this case that is our server, which is called tryps01, and it is located at tryps01.liv.ac.uk within the University network. X11 is the system used by linux computers to draw basic graphics, including those used for individual windows. By using X11 forwarding we can tell tryps01 to draw those windows on the computer you are using, as opposed to the non-existent server screen.



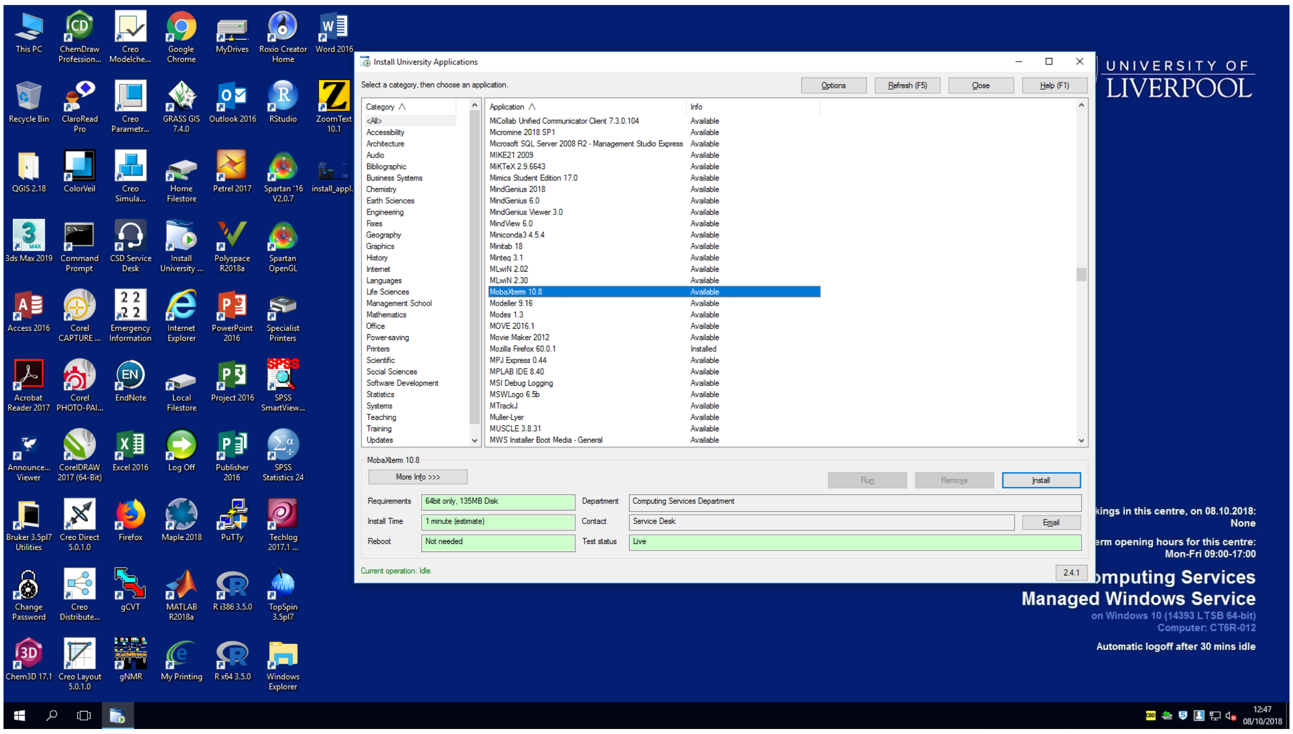
Bookings in this centre, on 08.10.2018:
None

Term opening hours for this centre:
Mon-Fri 09:00-17:00

Computing Services
Managed Windows Service
on Windows 10 (14393 LTSB 64-bit)
Computer: CT6R-012

Automatic logoff after 30 mins idle

MobaXterm can be installed using the ‘University Applications’ service, which can be found in the start menu (highlighted in red).



Install University Applications

Select a category, then choose an application.

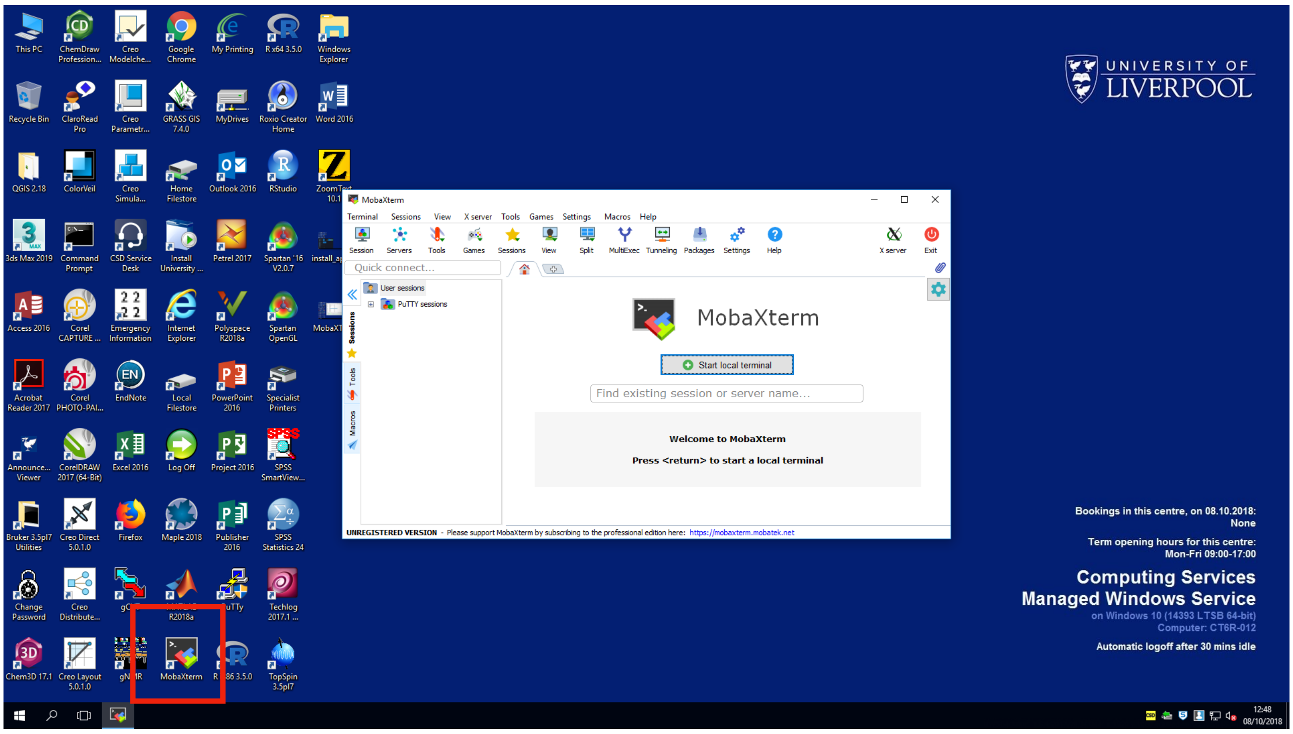
Application	Info
MiCollab Unified Communicator Client 7.3.0.104	Available
Micromine 2018 SP1	Available
Microsoft SQL Server 2008 R2 - Management Studio Express	Available
MIKE21 2009	Available
MiKTeX 2.9.6643	Available
Mimics Student Edition 17.0	Available
MindGenius 2018	Available
MindGenius 6.0	Available
MindGenius Viewer 3.0	Available
MindView 6.0	Available
Miniconda3 4.5.4	Available
Minitab 18	Available
Minteq 3.1	Available
MLwiN 2.02	Available
MLwiN 2.30	Available
MobaXterm 10.8	Available
Modeller 9.16	Available
Modes 1.3	Available
MOVE 2016.1	Available
Movie Maker 2012	Available
Mozilla Firefox 60.0.1	Installed
MPJ Express 0.44	Available
MPLAB IDE 8.40	Available
MSI Debug Logging	Available
MSWLogo 6.5b	Available
MTrackJ	Available
Muller-Lyer	Available
MUSCLE 3.8.31	Available
MWS Installer Boot Media - General	Available

MobaXterm 10.8

Requirements	64bit only, 135MB Disk	Department	Computing Services Department
Install Time	1 minute (estimate)	Contact	Service Desk
Reboot	Not needed	Test status	Live

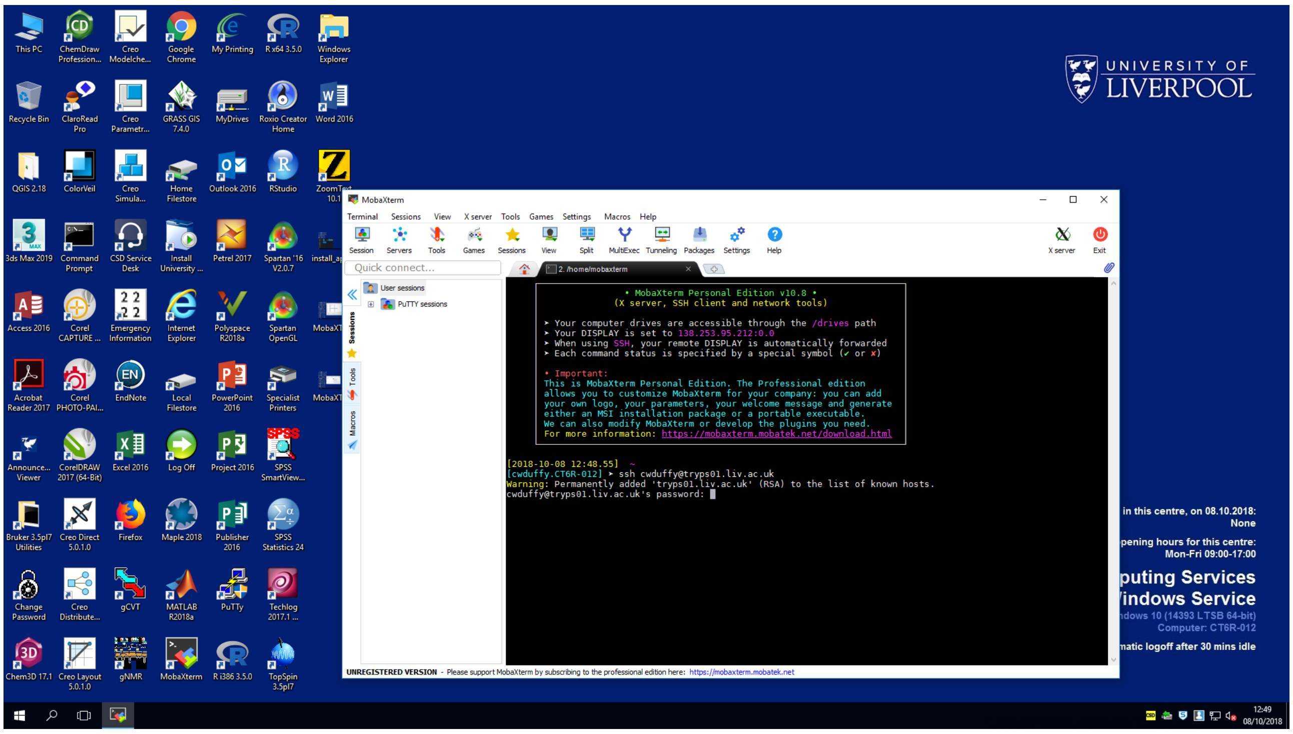
Current operation: Idle.

Find MobaXterm in the list of available applications and click install. This may take a couple of minutes.



MobaXterm
Terminal
Sessions
View
X server
Tools
Games
Settings
Macros
Help
Quick connect...
User sessions
PuTTY sessions
Start local terminal
Find existing session or server name...
Welcome to MobaXterm
Press <return> to start a local terminal
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
UNIVERSITY OF LIVERPOOL
Bookings in this centre, on 08.10.2018: None
Term opening hours for this centre: Mon-Fri 09:00-17:00
Computing Services
Managed Windows Service
on Windows 10 (14393 LTSB 64-bit)
Computer: CT6R-012
Automatic logoff after 30 mins idle

Once the software has installed it will create an icon on your desktop. Double click it to open MobaXterm and familiarise yourself with the options before clicking on the ‘Start local terminal’ button in the centre of the window

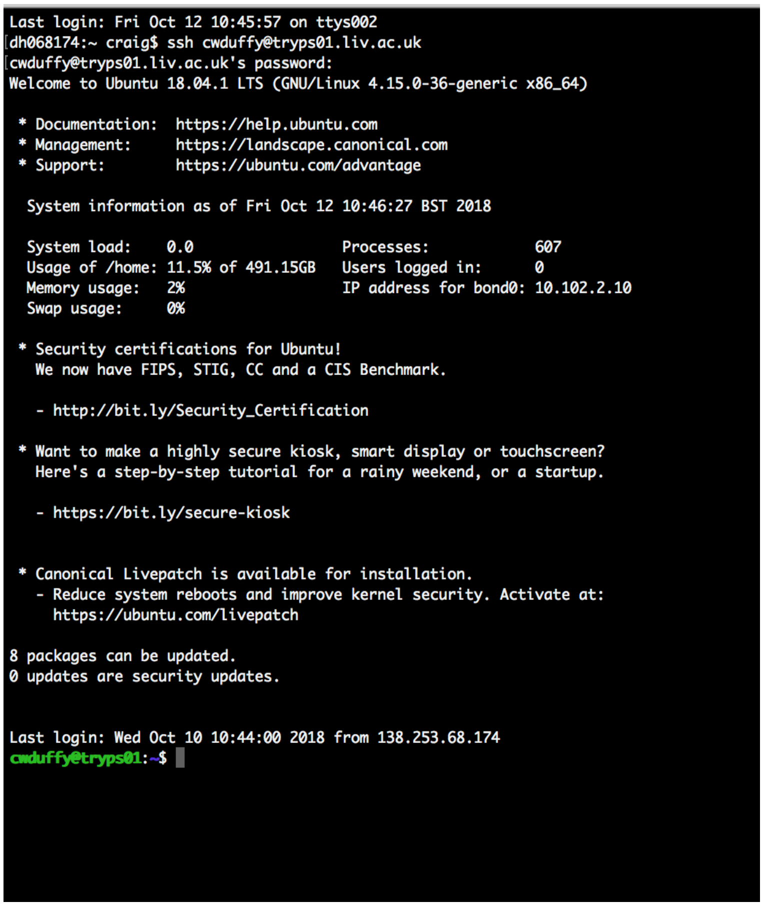


```
         • MobaXterm Personal Edition v10.8 •
       (X server, SSH client and network tools)

 ➤ Your computer drives are accessible through the /drives path
 ➤ Your DISPLAY is set to 138.253.95.212:0.0
 ➤ When using SSH, your remote DISPLAY is automatically forwarded
 ➤ Each command status is specified by a special symbol (✓ or ✗)

 • Important:
 This is MobaXterm Personal Edition. The Professional edition
 allows you to customize MobaXterm for your company: you can add
 your own logo, your parameters, your welcome message and generate
 either an MSI installation package or a portable executable.
 We can also modify MobaXterm or develop the plugins you need.
 For more information: https://mobaxterm.mobatek.net/download.html

[2018-10-08 12:48.55]  ~
[cwduffy.CT6R-012] ➤ ssh cwduffy@tryps01.liv.ac.uk
Warning: Permanently added 'tryps01.liv.ac.uk' (RSA) to the list of known hosts.
cwduffy@tryps01.liv.ac.uk's password:
```

You will be greeted by a black screen. This is the terminal window and it contains the command line, which you will be using extensively during this course. We now need to log on to the server. For this we need to provide the name of the command we wish to run (ssh) and our options (our username and the name of the server we wish to access). This should look like the following:

ssh username@tryps01.liv.ac.uk

Replace username with the username you have been provided and press enter. You will be asked whether the server should be permanently added to the list of known hosts. Type y for yes, press enter and then provide the password when prompted.



```
Last login: Fri Oct 12 10:45:57 on ttys002
dh068174:~ craig$ ssh cwduffy@tryps01.liv.ac.uk
cwduffy@tryps01.liv.ac.uk's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Oct 12 10:46:27 BST 2018

  System load:    0.0                Processes:           607
  Usage of /home: 11.5% of 491.15GB  Users logged in:     0
  Memory usage:   2%                 IP address for bond0: 10.102.2.10
  Swap usage:     0%

 * Security certifications for Ubuntu!
   We now have FIPS, STIG, CC and a CIS Benchmark.

   - http://bit.ly/Security_Certification

 * Want to make a highly secure kiosk, smart display or touchscreen?
   Here's a step-by-step tutorial for a rainy weekend, or a startup.

   - https://bit.ly/secure-kiosk


 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

8 packages can be updated.
0 updates are security updates.


Last login: Wed Oct 10 10:44:00 2018 from 138.253.68.174
cwduffy@tryps01:~$
```

You should now see text similar to the above within the terminal window. This indicates that you have successfully logged on to the server. You are currently located within your home directory (folder). You should see a line that looks like this:

username@tryps01.liv.ac.uk:~$

This is the prompt. It lists the following:
username - your username
@tryps01.liv.ac.uk - the name and location of the server
: - A separator between the user and location information
~ - Your current location. ~ is used as a shorthand for your home directory.
$ - The start of the command line, anything you type will be entered after this symbol.

Now that you have logged in we are going to introduce some basic commands. Towards the end of this handout you will find a longer list of some of the most commonly used linux commands.

You can check that you have logged into the correct account by typing:

```
whoami
```

This will return your current username. You can also check the location of the folder you are in using:

```
pwd
```

Which stands for Print Working Directory. This should return /home/yourusername, ~ is the shorthand for this.

Now that we know who we are and where we are we need to see what is in our current folder using

```
ls
```

Which tells the terminal to *list* the contents of the current directory. At the moment this directory will be empty, we need to copy the data for this session from its current location. First we will make a new directory called session1 using the following command:

```
mkdir session1
```

Check that this directory exists using ls. We will then copy the data for session one into this directory using the command:

```
cp -R /archived/Informatics/session1/. session1/.
```

This command breaks down as:
cp - Copy
-R - Use the recursive option to copy all the files within a folder.
/working/Informatics/session1 - The location of the files to be copied.
session1/. - Where the files should be copied to.

We have already seen that we can list the contents of our current directory using ls on its own.

What if we wanted to do the following:

List the contents of the session1 folder
List the full details of each file (such as the size, who owns it, when it was created)
List the size in a human readable format (Mb, Gb etc as opposed to bytes)
Order the list by the date that the file was created

The command for this would be:

ls -lht session1/

Which breaks down as:

ls - list the contents of the input

-l - show the full details of the files

-h - display file sizes in a human readable format (so Mb, Gb etc as opposed to the default of bytes)

-t - sort the files by time of creation

session1/ - Use session1/ as the input. In this case we are telling ls that this is the folder we wish to list the contents of.

As you can see above we have been able to combine the options into a group using only a single -. You do not have to take this approach, we could have typed:

ls -l -h -t session1/

But in general programmers and bioinformaticians will try and use shorthand. It is both faster to type and reduces the chance of introducing typos. The command line is very sensitive to typing errors and requires the correct use of capital letters. Ls and ls are interpreted as two different commands, only one of which is valid. Most of the time, a typo will cause the command to fail but the computer will always try to run what you have typed, even if this causes the output to be wrong.

After ls the next most important command is cd, which stands for change directory and allows you to move from location to location. This command is typically written as:

```
cd path
```

With the path being the location of the directory we wish to move to. If we are in the home directory (/home/username) and want to move to the session1 directory we only need to type

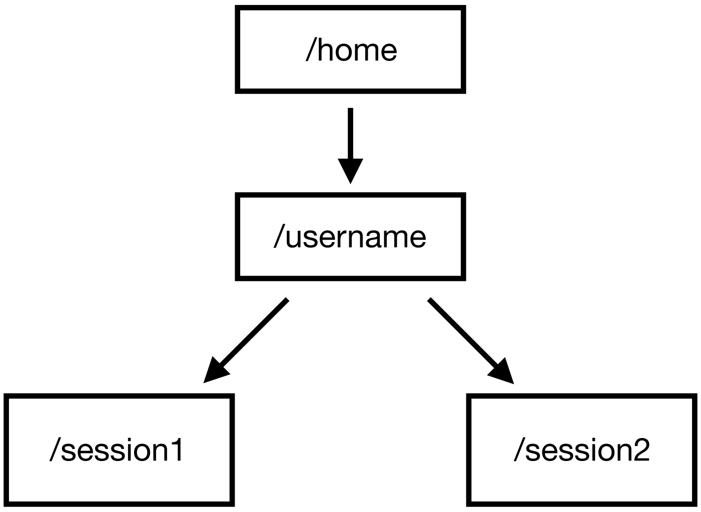
cd session1

And not the full

cd /home/username/session1

Although that would also work. If we wish to navigate to the directory up from our current location we can use .. which is shorthand for 'the directory above our current one'

```
cd ..
```



Directories nest within one another and it is possible to navigate up and down them using cd. If we were in session1 but wanted to move to session2 we could type the full command:

```
cd /home/username/session2
```

Or we could use .. to go up one level (to /username) and then go back down to /session2

```
cd ../session2
```

Tab complete and history

You may have noticed that the commands used so far have names that are short for what they do. This is fairly standard for many of the built in linux commands. Where possible programmers (and bioinformaticians!) try to minimise how much typing they have to do. One of the other ways to achieve this is using TAB complete and history

The TAB key is one of the most useful tools available from the command line as it reduces both the amount of typing required and consequently the chance of making an error. To use tab complete simply start typing the name of a command or file/folder name and press TAB. The computer will then attempt to finish the command or file name. For this to work there must be only a single possible command or file/folder name that matches what you have already typed. If we had a folder containing ‘my.txt’, ‘mypicture.gif’, ‘myfolder’ and ‘armyfile’ we could start typing

```
ls ar
```

Then pressing TAB would complete the line with

```
ls armyfile
```

If there are multiple options then the line will not be completed, however, pressing TAB for a second time will list all of the possible options. So typing

```
ls my
```

Then pressing TAB TAB will return

```
my.txt          myfolder        mypicture.gif
```

You would then see that the file you are looking for is called mypicture.gif, add one extra letter and hit TAB to complete the file name.

Later on in the course we will be using a popular piece of software called samtools.

What is the minimum number of letters that you would have to type before TAB complete will return just the samtools command?

History

You will often find yourself repeating the same commands over and over, or wanting to check exactly what you typed previously. Thankfully, the terminal stores a log of what has been previously run, which you can access through a number of methods.

Up and Down arrows - Using the Up and Down arrows on the keyboard you can scroll through recent commands, edit them or run them again.

The history logfile can be accessed using the simple command of

history

Which lists recent commands alongside an ID number. You can use this ID to repeat a command by using ! Followed by the ID number. So

!3

Would repeat command number 3 from the history.

Working on the command line

Being comfortable working from the command line is an essential skill set for modern bioinformatics. To help familiarise you with it we are going to work through a simple exercise that will use many of the most common commands.

The majority of the exercise will describe what you need to do but not which commands to use. We have provided a list of common commands at the end of this exercise, all of the commands you need for this exercise can be found there, in addition to a number of others. It is important to remember that running a command usually requires writing it in the following way:

Command -[option] [argument]

Where options include things such as the -lrht we encountered earlier and inputs include file names or data associated with an option. A command may have multiple options and inputs For example the following:

head -n 3 myfile.txt

Can be broken down as:

head	The command to run. In this case that we want to display the first lines of a file
-n	The option that tells the computer we are going to specify how many lines
3	The argument associated with the -n option. So we want to display the first 3 lines
myfile.txt	The file where those 3 lines should be found

There are multiple ways you can check how to run a command. The simplest is often to run the command on its own, without any options or inputs. As most commands require that these are set leaving them out will return a list of options. For example, the samtools help text is:

```
Program: samtools (Tools for alignments in the SAM format)
Version: 1.8-6-g7188f59 (using htslib 1.8-7-gf100074)

Usage:   samtools <command> [options]

Commands:
  -- Indexing
     dict           create a sequence dictionary file
     faidx          index/extract FASTA
     index          index alignment

  -- Editing
     calmd          recalculate MD/NM tags and '=' bases
     fixmate        fix mate information
     reheader       replace BAM header
     targetcut      cut fosmid regions (for fosmid pool only)
     addreplacerg   adds or replaces RG tags
     markdup        mark duplicates

  -- File operations
     collate        shuffle and group alignments by name
     cat            concatenate BAMs
     merge          merge sorted alignments
     mpileup        multi-way pileup
     sort           sort alignment file
     split          splits a file by read group
     quickcheck     quickly check if SAM/BAM/CRAM file appears intact
     fastq          converts a BAM to a FASTQ
     fasta          converts a BAM to a FASTA

  -- Statistics
     bedcov         read depth per BED region
     depth          compute the depth
     flagstat       simple stats
     idxstats       BAM index stats
     phase          phase heterozygotes
     stats          generate stats (former bamcheck)

  -- Viewing
     flags          explain BAM flags
     tview          text alignment viewer
     view           SAM<->BAM<->CRAM conversion
     depad          convert padded BAM to unpadded BAM
```

We can get this text just by typing samtools into the terminal, we could have also gotten the same text by typing:

samtoools --help

One of the quirks of linux development is that there are no fixed defaults, some programs may use --help, -help or -h. Unfortunately, it is also common for programs to lack any help listings, leaving you to rely on manuals or the web for instructions.

The second approach to getting help are manual pages. These are available for almost all of the core linux commands and can be accessed by typing:

man name_of_command

Or

info name_of_command

Bring up the manual page for the ls command using:

man ls

You can move through the page with the space bar and close it by pressing q.

READ THIS BEFORE YOU CONTINUE

At the end of this document you will find two pages of common linux commands - You will need to use some of these during the next two exercises. If you cannot work out the command to use search online before asking for assistance - learning how to solve a problem is a valuable skill and you'd be surprised how often experienced bioinformaticians rely on online resources to solve a problem.

Exercise One

1. If you have not already done so change your working directory to session1 and list the contents of that location. Try and list the full details of the files to work out when they were created and who owns them.

2. Within session1 there should be two zip files, called ExerciseOne.tar.gz and ExerciseTwo.tar.gz.

   The .tar indicates that these files have been collected into a tar archive, while the .gz indicates that they have been compressed using gzip.

3. Extract the contents of these folders, the file extensions will give you a hint as to the name of the program to use. The command you use will require the options -vxzf

4. Check what -vxzf actually did by bringing up the manual page or help text for the command you just used.

5. Enter the new ExerciseOne directory and list the files that are in the directory - how big is the largest file?

6. Determine which of the files are plain text files by attempting to display the contents of each of then. There are multiple commands available to do this, try and use more than one of them.

7. There is one file that contains the single line ‘Delete me’. Remove that file.

8. Concatenate the remaining plain text files into a single file called combined_data.

   You will need to use a new option here, the > symbol. This indicates that we want to take the output from a command and save it into a new file. For example:

   ```
   ls my* > myfiles
   ```

   would list all the files starting with ‘my’ and then save the result into a file called myfiles.

9. Search ‘combined_data’ for lines containing the word ‘the’ and save those lines in a new file called ‘lines_with_the’

10. Work out how many words and how many lines your new file has.

11. Without creating a new file, count how many lines in ‘combined_data’ have the word ‘and’ in them. You may need to combine the commands from the previous 2 steps by piping the output from one to the other.

    The pipe, represented by a vertical line |, is a special command. It takes the output from one command and allows us to use it as the input for a second command. We can, therefore, quickly chain commands together and produce only a final output as opposed to having to generate multiple temporary outputs that need to be deleted at a later date.

12. Move ‘lines_with_the’ into the directory above the one you are currently in. For this you will need to use the .. notation. We have already introduced the idea of paths and that folders are located inside one another. When we refer to a file or location linux assumes its location is relative to our current directory unless we tell it otherwise. If we are working in a folder called session1 then its full path may be:

    ```
    /home/User/informatics/session1/
    ```

    but if we wish to look at the top 5 lines of a file in that folder we just need to type

    ```
    head -n 5 mytextfile
    ```

    and the terminal will look in session1 for the file. But if we wanted to look in the folder ‘informatics’ for that file we need to tell the terminal this. We could type it the long way:

    ```
    head -n 5 /home/User/informatics/mytextfile
    ```

    or we can use the shortcut of .. which means the folder above our current location. So the command would be:

    ```
    head -n 5 ../mytextfile
    ```

    Use the .. approach to move lines_with_the into the session1 directory.

13. Change directory back to session1.

14. Delete the ExerciseOne directory using the recursive option.

15. One of the big advantages of linux based systems is that it is relatively easy for individuals to develop, write and distribute their own scripts and programs. The majority of the software used in bioinformatics is developed this way. Because of how linux is developed it is common to provide scripts in a raw, uncompiled format. This is for two reasons.

    Firstly, it allows for individuals to modify the scripts if they wish.

    Secondly, it allows for software to adapt to the configuration of each computer it is installed on.

    While each linux distro maintains its own repository of software that has already been checked for compatibility it is still common to compile software obtained elsewhere, such as the widely used GitHub service. We are going to demonstrate this process with a popular piece of software called samtools, which you will use later in the course.

    The first step to compiling software is to obtain it. We are going to download it using the following command:

    ```
    wget https://github.com/samtools/samtools/releases/download/1.9/samtools-1.9.tar.bz2
    ```

16. Unzip the file you have just downloaded, this time you will need to use the options -xvjf instead of -xvzf as the file has been compressed using the bzip format as opposed to the gzip format.

17. Enter the directory you have just created and examine the files. Are any of them familiar? Are there any that you think you should read before proceeding?

18. The first step in compiling software is to see if it needs to be configured using the command:

    ```
    ./configure
    ```

    This command is actually running a script within the folder. Remember we previously used .. to refer to the directory above our current one. Here the single . tells the computer to look in our current directory for the script called configure. The configuration script performs an important step - checking that the software samtools depends on is present and working.

19. Once samtools has been configured we need to make it, which pulls all the components into the final executable program. This command is simply:

    make

20. After make has completed you should have a new file present called samtools. Check that it runs using the ./ notation we used in step 19. If you were compiling this software for your own computer the final step would be to install it, however, we have already done this step to ensure samtools can be run by every user of the server.

21. Once you have samtools working from your local build leave the directory and delete it.

Introduction to blast

The Basic Local Alignment Search Tool (BLAST) is a mainstay of informatics and molecular biology. It allows us to take a query sequence and determine if the gene, protein or unknown sequence is related to other known sequences. BLAST is an extensively used tool because it can quickly identify regions of local similarity and there are numerous online databases containing sequences from thousands of organisms. The largest collection of these databases can be found on the NCBI website, with the main BLAST page located at http://blast.ncbi.nlm.nih.gov

It is important to understand the different type of BLAST searches that are available. These are:

Blastn - Nucleotide blast that takes a nucleotide query sequence and searches a database of nucleotide sequences.

Blastp - Protein blast that takes a protein query sequence and searches a database of protein sequences.

Blastx - Translated blast that takes a nucleotide sequence, translates it into a protein sequence and searches against a database of protein sequences.

Tblastn - Translated blast that takes a protein sequence and searches against a database of nucelotide sequences which have been translated into protein sequences.

Tblastx - Translated blast that takes a nucleotide sequences, translates it into a protein sequences and searches against a database of nucleotides which have also been translated into protein sequences.

Exercise Two

For this exercise we will be performing a BLAST using the command line. To do this we will first create a database from a set of reference sequences, before blasting our sequences against them and examining the results.

1. Enter the ExerciseTwo directory. If you have not already unzipped the ExerciseTwo.tar.gz file do that now.

2. List the contents of this directory, there should be four files ending in .fasta and a 5th file ending in .txt

   Whenever you run blast the software takes your query and searches for it in a subject database. The NCBI servers maintain the biggest database, but for the purpose of this exercise we are going to create our own local database. The first step in this process is to use the cat command to combine our raw fasta files into a single file.

   Use cat to concatenate the following files into a new file called allspecies.fasta:
   927-01.fasta
   LinJ-20.fasta
   Y486-01.fasta

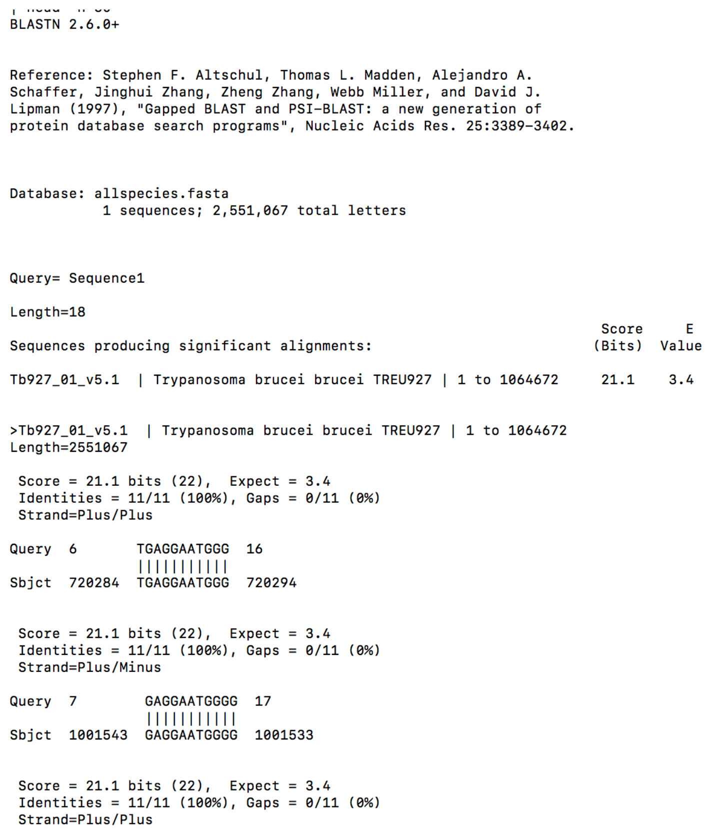
3. Create a blast database by adapting the following generic command using allspecies.fasta as the input file:

   makeblastdb -in fasta_file -parse_seqids -dbtype nucl

4. After creating the database we need to blast our query sequences against it. These sequences are in sequences_to_blast.txt and the command to run the blast search is:

   blastn -query sequences_to_blast.txt -db allspecies.fasta -task blastn

5. Running the command as is will have printed the results directly to the screen, which should look like the figure below.



```
BLASTN 2.6.0+


Reference: Stephen F. Altschul, Thomas L. Madden, Alejandro A.
Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J.
Lipman (1997), "Gapped BLAST and PSI-BLAST: a new generation of
protein database search programs", Nucleic Acids Res. 25:3389-3402.



Database: allspecies.fasta
           1 sequences; 2,551,067 total letters



Query= Sequence1

Length=18
                                                                      Score     E
Sequences producing significant alignments:                          (Bits)  Value

Tb927_01_v5.1  | Trypanosoma brucei brucei TREU927 | 1 to 1064672     21.1    3.4


>Tb927_01_v5.1  | Trypanosoma brucei brucei TREU927 | 1 to 1064672
Length=2551067

 Score = 21.1 bits (22),  Expect = 3.4
 Identities = 11/11 (100%), Gaps = 0/11 (0%)
 Strand=Plus/Plus

Query  6       TGAGGAATGGG  16
               |||||||||||
Sbjct  720284  TGAGGAATGGG  720294


 Score = 21.1 bits (22),  Expect = 3.4
 Identities = 11/11 (100%), Gaps = 0/11 (0%)
 Strand=Plus/Minus

Query  7        GAGGAATGGGG  17
                |||||||||||
Sbjct  1001543  GAGGAATGGGG  1001533


 Score = 21.1 bits (22),  Expect = 3.4
 Identities = 11/11 (100%), Gaps = 0/11 (0%)
 Strand=Plus/Plus
```

This is useful if there are only a few hits, but if there are numerous hits it is better to save the results to an output file. We can do this by adding the -out option followed by an output file name. Alternatively, we could instead redirect the output using the > symbol as we did in exercise one.

Repeat the blast but save the results to a file called sequence1-blast

6. Examine the output file that you have just created using the less command. The default output format summarises each of the blast hits for your three input sequences. This includes the alignment score, the expect value, the number of identities (number of matches between query and the database) and the number of gaps, arising from insertions/deletions. It also provides a visual summary of the alignment.

INSTITUTE OF INFECTION AND GLOBAL HEALTH

7. The default format is useful if you only have a small number of hits to examine, but your result file will contain almost 100 hits. This is because the default expect value (e-value) for blast is 1. The e-value represents the number of hits we would expect to get by chance alone. It is determined by a combination of the length of the query, the complexity of the sequence and the size of the database.

   We can change the minimum e-value for a blast hit by setting the option

   ```
   -evalue NUMBER
   ```

   Where NUMBER is the value we wish to set it to. It is typical to set the e-value very low, to levels such as 1e-10 or lower in order to return only significant hits.

8. It is also possible to change the format of our output file. There are 12 different output formats. One of the most useful is format 7, which summaries the results into an easy to read table. The output format can be set using

   ```
   -outfmt NUMBER
   ```

9. Repeat your blast analysis but reduce the e-value to 1e-15 and set the format to 7 before examining the results.

   How many hits do you get for each input sequence?
   What is the minimum e-value required to return at least one hit for each input sequence?

10. If you have time at the end of the session try and BLAST your sequences against the NCBI database using the web portal. Do you get different results when using their databases?

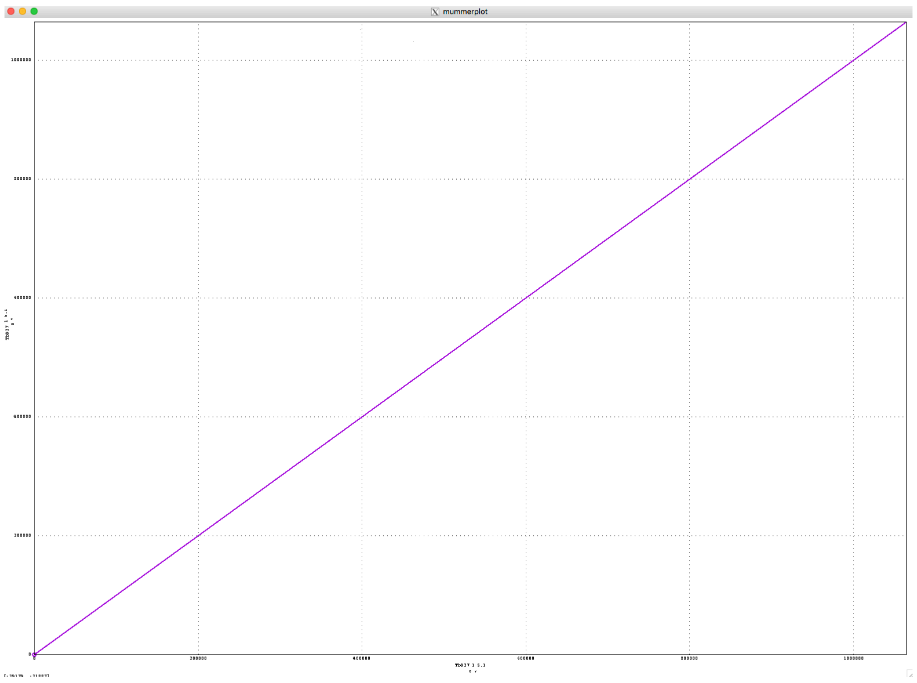
Exercise Three

BLAST isn't the only way to find similarities between sequences. It is only one type of local alignment, during this course you will make use of a number of different types of sequence alignment.

For example, at a chromosomal level, we may wish to determine not only whether sequences share regions of similarity, but whether these regions are collinear, or arranged in the same order. This often indicates that the sequences share common ancestry and can be used to identify rearrangements that have arisen since their divergence.

When sequencing a new organism we may also need to rely on a reference sequence from another species to scaffold small sequences into a larger pseudo-contig. Here we make the assumption that the chromosomes share common ancestry and use that information to order our pieces. While this approach is a useful tool, it is vital that this ordering is validated later on!

1. One piece of software used to assess collinearity is MUMmer. Using a local alignment approach it identifies regions of similarity between two sequences and allows us to plot the alignment. For example in the figure below the x-axis plots the position along the first sequence, the y-axis the position along the second sequence.



   A diagonal line such as the above indicates that the two sequences are completely collinear.

2. Within your ExerciseTwo directory check that you have the following four sequence files:
   444-01.fasta
   Y486-01.fasta
   927-01.fasta
   LinJ-20.fasta

3. Running MUMmer requires two steps - identifying regions of similarity and plotting the results. For the first we have a choice of two programs, nucmer and promer.

   Nucmer looks for nucleotide - nucleotide similarity and is best for comparing sequences that are closely related to one another (such as two samples from the same species where one may have rearrangements).

   Promer converts the sequences into amino acids before looking for regions of similarity. It is better at aligning sequences that may have diverged from one another, such as when comparing sequences from two different species. Why might a protein alignment be better for divergent species?

4. First, we will run a basic nucmer alignment using the following command:

   nucmer -p 927-927 927-01.fasta 927-01.fasta

   which compares 927-01.fasta against itself and writes the output to a file called 927-927.delta

5. Run mummerplot and examine the plot that it produces. The command is:

   mummerplot -x11 --layout --filter 927-927.delta -R 927-01.fasta -Q 927-01.fasta -p 927-927

6. Repeat this process for the other files which are present changing the name of the output file each time. Which sequences appear to be most related based on nucleotide similarity? Do any of the sequences have rearrangements?

7. Instead of using nucmer use promer to run the alignment. The basic command is the same, but replace nucmer with promer at the start. How does this change your results? Why do you think the results have changed?

8. As with most programs we can change the majority of the default options. Use nucmer -h or promer -h to get the list of options and see what effect changing them has.

Basic commands

pwd	Print the current working directory
cp	Copy a file from one location to another with the option of renaming it in the process.
rm	Remove a file or directory. To remove a directory that is not empty the -r option must be specified.
mv	Move a file from one location to another (equivalent to copying it to the new location and then deleting the original). This command can also be used to rename files
history	Display the history log of previously run commands
cd	Change your working directory to the specified folder
mkdir	Make a new directory with the specified name
whoami	Display the username of the current user
wget	Download a file from a web address
ls	List the contents of a folder. By default this will be the current working directory but other locations may be specified.
nano	Open the nano text editor. You can chose to open an existing file in nano or to start a new, blank file
head	Print the starting lines of a text file to the terminal
tail	Print the end lines of a text file to the terminal
less	Display a text file starting from the beginning but with the option of scrolling or searching within the file once it is open
rmdir	Remove a directory. The directory must be empty, if it is not then an error will be returned.

touch	Create a new empty file with the specified name
cat	Print the contents of the specified files to the terminal in the order of the
tar	Create and extract tar archive files with optional compression / decompression using gzip
chmod	Change the permissions of a file
man	Display the manual page for a command
grep	Search a file for a given pattern (which could be a word or phrase). By default grep will return any line that contains a match to the search pattern.
touch	Create a new empty file of any type
wc	Display the number of words, lines, characters or bytes of a file
Ctrl-c	Kill a job by forcing it to terminate. Note that this replaces the copy command, which is Ctrl+Shift+C in a terminal (with Ctrl+Shift+V for paste)
Ctrl-z	Suspend a job
&	Run the command in the background
fg	Bring the most recent suspended or background job into the foreground
bg	Resume a suspended job but keep it in the background
jobs	List jobs which have been suspended or are running in the background
;	Used to separate commands. After the first is complete run the second command
|	Pipe (send) the output from the first command to the second command as input
>	Take the output from the command and save it to the following output file