

Introduction to Linux

Introduction

Linux is the standard operating system on most large computer systems in scientific research, in the same way that Microsoft Windows is the dominant operating system on desktop PCs.

Linux and MS Windows both perform the important job of managing the computer's hardware (screen, keyboard, mouse, hard disks, network connections, etc...) on your behalf. They also provide you with tools to manage your files and to run application software.

They both offer a graphical user interface (desktop). The desktops look different, call things different names and have different pictures but they mostly can do the same things.

Linux is a powerful, secure, robust and stable operating system which allows dozens of people to run programs on the same computer at the same time. This is why it is the preferred operating system for large-scale scientific computing. It runs on all kinds of machines, like mobile phones (Android), desktop PCs, kitchen appliances,..., all the way up to supercomputers. The majority of the Internet is powered by Linux.

Aims

The aim of this module is to introduce Linux and cover some of the basics that will allow you to run some of the programs used in this workshop. Several of the programs that you are going to use during the workshop, plus many others that are useful for bioinformatics analyses, are run in Linux. This module is only designed to provide a very brief introduction to some of the features and useful commands of Linux.

During this module we will also obtain a genome sequence and examine the basic structure of an EMBL entry.

Why use Linux?

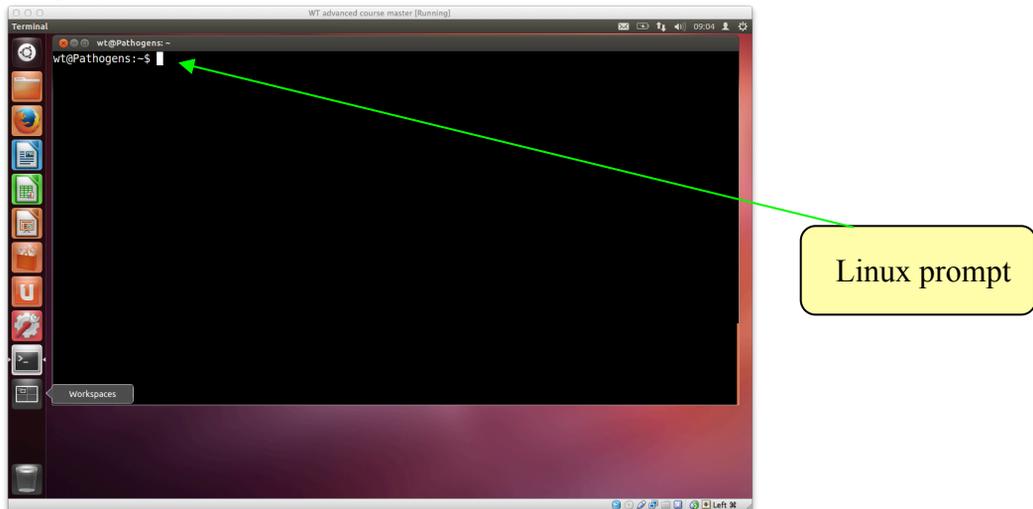
- Linux is a well established, very widespread operating system. You probably have a device running on Linux in your home without realising it (e.g. playstation, TV box, wireless router, android tablets/phones,...)
- Command line driven, with a huge number of often terse, but powerful commands
- In contrast to Windows, it is designed to allow many users to run their programs simultaneously on the same computer
- Designed to work in computer networks - for example, most of the Internet is Linux based
- It is used on many of the powerful computers at bioinformatics centres and also on many desktops and laptops.
- The major difference between Linux and Windows is that it is free (as in freedom) and you can modify it to work however you want. This same principle of freedom is also used in most bioinformatics software.
- There are many distributions of Linux such as Ubuntu, RedHat, Fedora, Mint,...). These are all Linux, but they bundle up extra software in a different way or combinations. Some are known for being conservative and reliable, whilst others are known for being on the cutting edge (and less reliable).

Getting started

In this workshop, we will be using Ubuntu, a version of Linux which was specially designed for PCs.

We will use a terminal window to type in our Linux command line. This is similar to the "Command Prompt" window on MS Windows systems, which allows the user to type DOS commands to manage files.

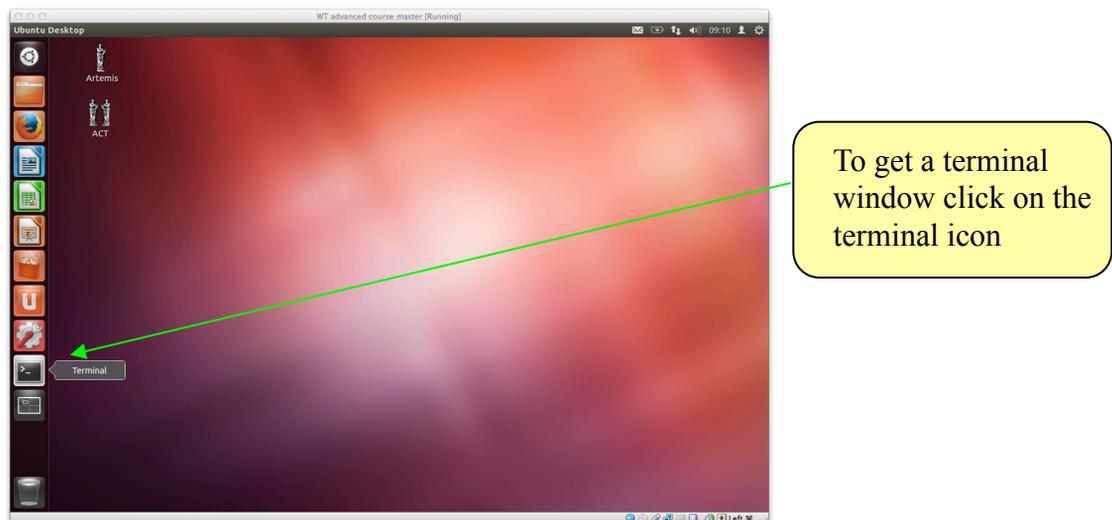
You should see a window labelled "Terminal" which will be empty except for a '\$' character at the top left. The '\$' character is the Linux prompt, similar to "C:\>" in DOS. Note: the prompt will often be different on different Linux computers, for example it may be displayed as a '%' character.



You can type commands directly into the terminal at the '\$' prompt.

A list of useful commands can be found on the next page.

Many of them are two- or three-letter abbreviations. The earliest Linux systems (*circa* 1970) only had slow Teletype terminals, so it was faster to type 'rm' to remove a file than 'delete' or 'erase'. This terseness is a feature of Linux which still survives.



The command line

All Linux programs may be run by typing commands at the Linux prompt `$`. The command line tells the computer what to do.

You may subtly alter these commands by specifying certain options when typing in the command line.

Command line Arguments

Typing any Linux command for example `ls`, `mv` or `cd` at the Linux prompt with the appropriate variables such as files names or directories will result in the tasks being performed on pressing the enter key.

```
command options arguments
```

The 'command' is separated from the options and arguments by a space. Additional options and/or arguments can be added to the commands to affect the way the command works. Options usually have one dash and a letter (e.g. `-h`) or two dashes and a word (`--help`) with no space between the dash and the letter/word. Arguments are usually filenames or directories.

For example:

List the contents of a directory

```
ls
```

List the contents of a directory with extra information about the files

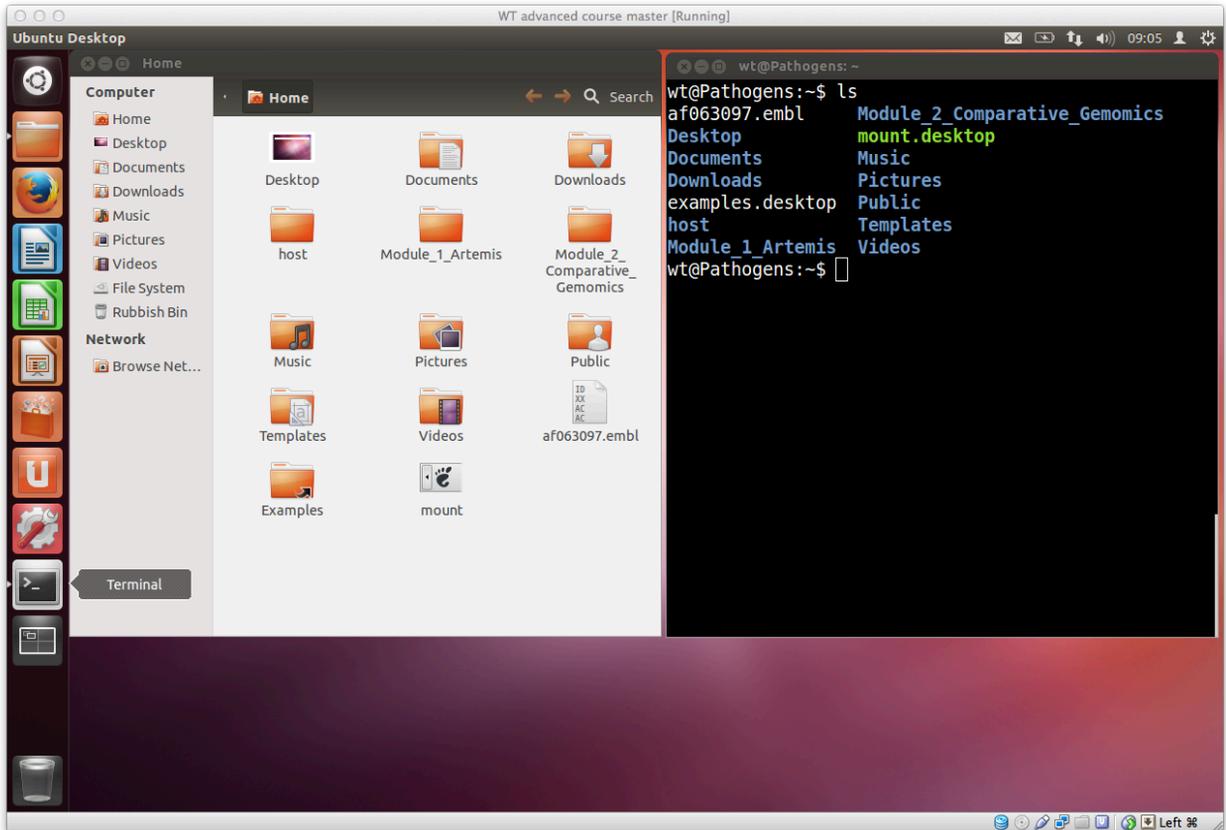
```
ls -l
```

List all contents including hidden files & directories

```
ls -a
```

List the contents of the directory called `Module_1_Artemis` with extra information

```
ls -l Module_1_Artemis
```



To get a list of files in the terminal you can use the `ls` command with no other options. This tells the computer you want a list of all the files in the current directory. The same information is also displayed in the file browser but with nice looking pictures.

Finding out more about your files with ls -l

```

wt@Pathogens: ~/Module_1_Artemis/exercise_1
wt@Pathogens:~/Module_1_Artemis/exercise_1$ cd Module_1_Artemis/exercise_1
wt@Pathogens:~/Module_1_Artemis/exercise_1$ ls -l
total 3384
drwxr-xr-x 2 wt wt 4096 Oct 7 11:42 fasta
-rw-r--r-- 1 wt wt 2282640 Oct 7 11:42 Pf3D7_03.embl
-rw-r--r-- 1 wt wt 1085781 Oct 7 11:42 Pf3D7_03.fasta
-rw-r--r-- 1 wt wt 83209 Oct 7 11:42 PF.tab
wt@Pathogens:~/Module_1_Artemis/exercise_1$

```

File permissions

Who owns the file

Size of file in bytes

Date file last updated

File name

By using the `-l` option we can change the behaviour of the `ls` command. Instead of printing out a simple list, it will print out additional information about each file. There is a space between the command `ls` and the `-l` option. There is no space between the dash and the letter `l`.

Permissions

Every file has permissions which restrict what can be done with a file or directory.

Read (r): permission to read from a file/directory

Write (w): permission to modify a file/directory

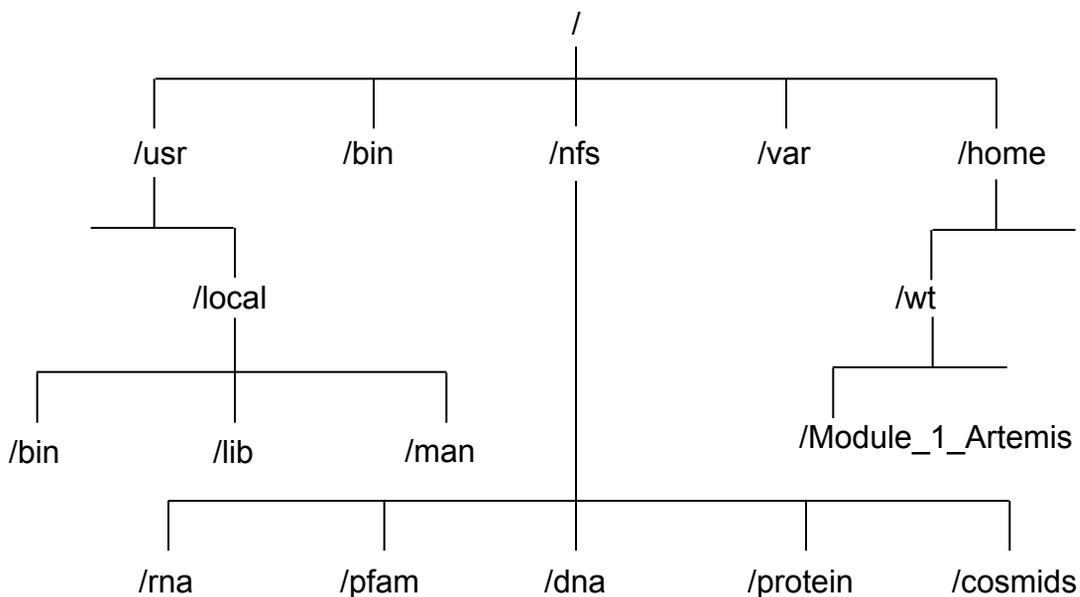
Execute (x): Tells the operating system that the file contains code for the computer to run, as opposed to a file of text which you open in a text editor.

The first set of permissions (characters 2,3,4) refer to what the owner of the file can do, the second set of permissions (5,6,7) refers to what members of the Linux group can do and the third set of permissions (8,9,10) refers to what everyone else can do.

Files and Directories

Directories are the Linux equivalent of folders on a PC or Mac. They are organised in a hierarchy, so directories can have sub-directories and so on. Directories are very useful for organising your work and keeping your account tidy - for example, if you have more than one project, you can organise the files for each project into different directories to keep them separate. You can think of directories as rooms in a house. You can only be in one room (directory) at a time. When you are in a room you can see everything in that room easily. To see things in other rooms, you have to go to the appropriate door and crane your head around. Linux works in a similar manner, moving from directory to directory to access files. The location or directory that you are in is referred to as the current working directory.

Directory structure example



Therefore if there is a file called `genome.seq` in the **dna** directory its location or full pathname can be expressed as `/nfs/dna/genome.seq`.

General Points

Linux is pretty straightforward, but there are some general points to remember that will make your life easier:

Linux is case sensitive - typing "ls" is not the same as typing "LS".

You need to put a space between a command and its argument - for example, "more myfile" will show you the contents of the file called myfile; "moremyfile" will just give you an error!

Linux is not psychic! If you mis-spell the name of a command or the name of a file, it will not understand you.

Many of the commands are only a few letters long; this can be confusing until you start to think logically about why those letters were chosen - ls for list, rm for remove and so on.

Often when you have problems with Linux, it is due to a spelling mistake, or perhaps you have omitted a space.

If you want to know more about Linux and its commands there are plenty of resources available that provide a more comprehensive guide (including a cheat sheet at the end of this chapter):-

- <http://Linuxhelp.ed.ac.uk>
- <http://Linux.t-a-y-l-o-r.com/>

In what follows, we shall use the following typographical conventions:

Characters written in **bold typewriter font** are commands to be typed into the computer as they stand.

Characters written in *italic typewriter font* indicate non-specific file or directory names.

Words inserted within square brackets [**Ctrl**] indicate keys to be pressed.

So, for example,

\$ ls anydirectory [Enter]

means "at the Linux prompt \$, type ls followed by the name of some directory, then press the key marked Enter"

Don't forget to press the [Enter] key: commands are not sent to the computer until this is done.

Some useful Linux commands

Command	What it does
ls	Lists the contents of the current directory
mkdir	Makes a new directory
mv	Moves or renames a file
cp	Copies a file
rm	Removes a file
cat	Concatenates files
less	Displays the contents of a file one page at a time
head	Displays the first ten lines of a file
tail	Displays the last ten lines of a file
cd	Changes current working directory
pwd	Prints working directory
find	Finds files matching an expression
grep	Searches a file for patterns
wc	Counts the lines, words, characters, and bytes in a file
kill	Stops a process
jobs	Lists the processes that are running

Exercise

The following exercise introduces a few useful Linux commands and provides examples of how they can be used.

Many people panic when they are confronted with an Linux prompt! Don't! The exercise is designed to be step-by-step, so all the commands you need are provided in the text. If you get lost ask a demonstrator. If you are a person skilled at Linux, be patient it is only a short exercise.

Finding where you are and what you've got

pwd Print the working directory

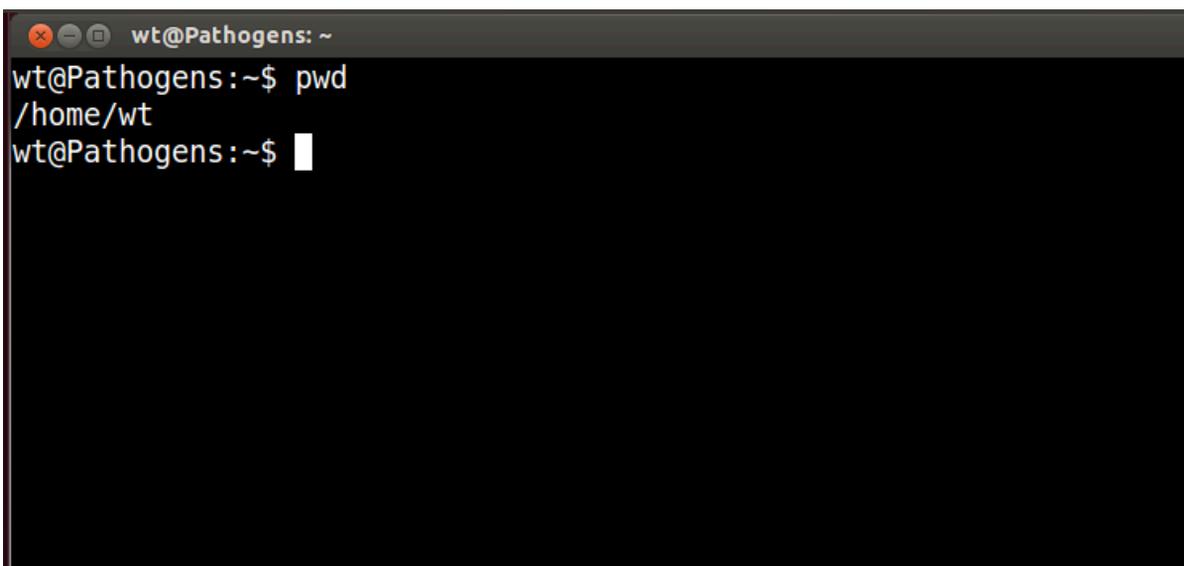
As seen previously directories are arranged in a hierarchical structure. To determine where you are in the hierarchy you can use the **pwd** command to display the name of the current working directory. The current working directory may be thought of as the directory you are in, i.e. your current position in the file-system tree

To find out where you are type

```
pwd [enter]
```

You will see that you are in your home directory. We need to move into the Linux directory

Linux is case sensitive, **PWD** is not the same as **pwd**



```
wt@Pathogens: ~  
wt@Pathogens:~$ pwd  
/home/wt  
wt@Pathogens:~$
```

cd Change current working directory

The **cd** command will change the current working directory to another, in other words allow you to move up or down in the directory hierarchy. First of all we are going to move into the Linux directory below. To do this type:

```
cd Linux[enter]
```

Now use the **pwd** command to check your location in the directory hierarchy.

ls List the contents of a directory

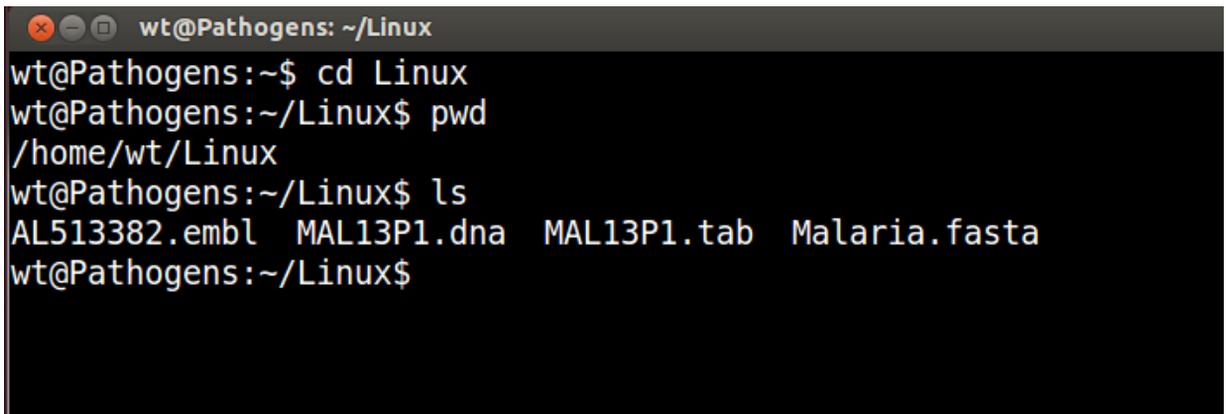
To find out what are the contents of the current directory type

```
ls [enter]
```

The **ls** command lists the contents of your current directory, this includes files and directories

You should see that there are 4 files called:

AL513382.embl, MAL13P1.dna, MAL13P1.tab, Malaria.fasta



```
wt@Pathogens: ~/Linux
wt@Pathogens:~$ cd Linux
wt@Pathogens:~/Linux$ pwd
/home/wt/Linux
wt@Pathogens:~/Linux$ ls
AL513382.embl MAL13P1.dna MAL13P1.tab Malaria.fasta
wt@Pathogens:~/Linux$
```

Changing and moving what you've got

cp Copy a file

`cp file1 file2` is the the command which makes a copy of `file1` in the current working directory and calls it `file2`

What you are going to do is make a copy of `AL513382.embl`. This file contains the genome of *Salmonella typhi* strain CT18 in EMBL format. The new file will be called `S_typhi.embl`

```
cp AL513382.embl S_typhi.embl [enter]
```

If you use the `ls` command to check the contents of the current directory you will see that there is an extra file called `S_typhi.embl`.

```
wt@Pathogens: ~/Linux
wt@Pathogens:~/Linux$ cp AL513382.embl S_typhi.embl
wt@Pathogens:~/Linux$ ls
AL513382.embl MAL13P1.dna MAL13P1.tab Malaria.fasta S_typhi.embl
wt@Pathogens:~/Linux$
```

rm Delete a file

This command removes a file permanently so take care!

You are now going to remove the old version of *S. typhi* genome file, `AL513382.embl`

```
rm AL513382.embl [enter]
```

The file will be removed. Use the `ls` command to check the contents of the current directory to see that `AL513382.embl` has been removed.

Linux as a general rule does exactly what you ask, and does not ask for confirmation. Unfortunately there is no "recycle bin" on the command line to recover the file from, so you have to be careful.

```
wt@Pathogens: ~/Linux
wt@Pathogens:~/Linux$ rm AL513382.embl
wt@Pathogens:~/Linux$ ls
MAL13P1.dna MAL13P1.tab Malaria.fasta S_typhi.embl
wt@Pathogens:~/Linux$
```

cd Change current working directory

As before the **cd** command will change the current working directory to another, in other words allow you to move up or down in the directory hierarchy. First of all we are going to move into the directory above, type:

```
cd .. [enter]
```

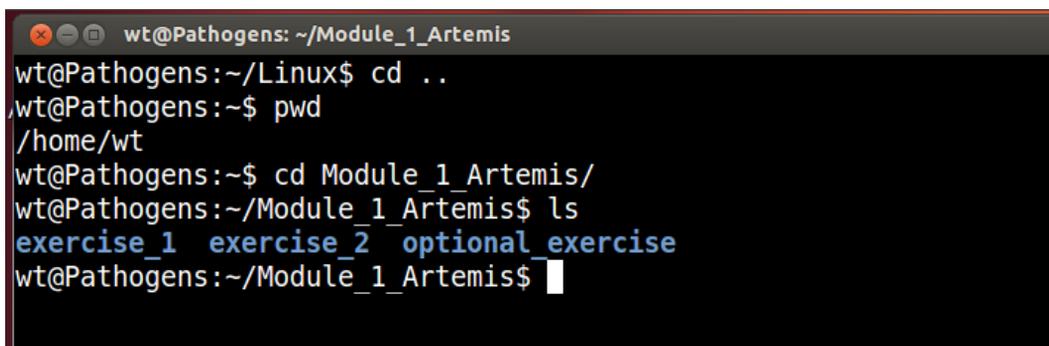
Now use the **pwd** command to check your location in the directory hierarchy.

Next, we are going to move into the `Module_1_Artemis` directory.

To change to the `Module_1_Artemis` directory type:

```
cd Module_1_Artemis [enter]
```

use the **ls** command to check the contents of the directory



```
wt@Pathogens: ~/Module_1_Artemis
wt@Pathogens:~/Linux$ cd ..
wt@Pathogens:~$ pwd
/home/wt
wt@Pathogens:~$ cd Module_1_Artemis/
wt@Pathogens:~/Module_1_Artemis$ ls
exercise_1 exercise_2 optional_exercise
wt@Pathogens:~/Module_1_Artemis$
```

Tips

There are some short cuts for referring to directories:

- `.` Current directory (one full stop)
- `..` Directory above (two full stops)
- `~` Home directory (tilda) which is `/home/wt`
- `/` Root of the file system (like `C:\` in Windows)

Pressing the tab key twice will try and autocomplete what you've started typing or give you a list of all possible completions. This saves a lot of typing and typos.

Pressing the up/down arrows will let you scroll through the previous commands.

If you highlight some text, middle clicking will paste it on the command line.

mv Move a file

To move a file from one place to another use the **mv** command. This moves the file rather than copies it, therefore you end up with only one file rather than two.

When using the command the path or pathname is used to tell Linux where to find the file. You refer to files in other directories by using the list of hierarchical names separated by slashes. For example, the file *bases* in the directory *genome* has the path **genome/bases**

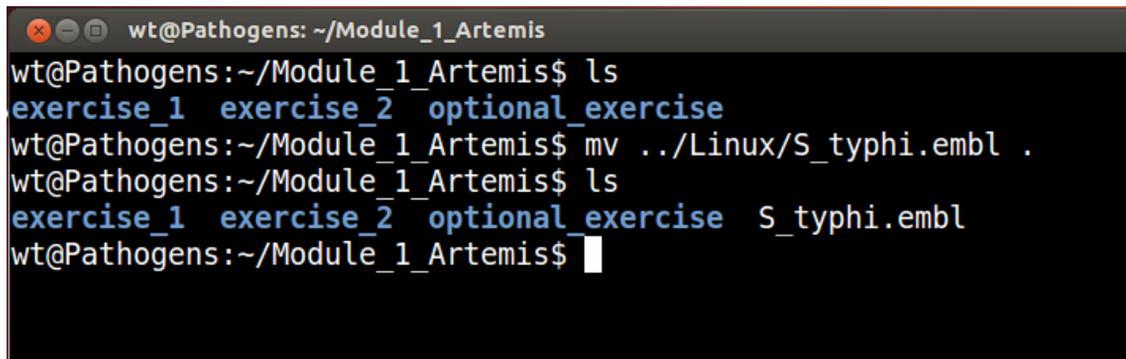
If no path is specified Linux assumes that the file is in the current working directory.

What you are going to do is move the file *S_typhi.embl* from the *Linux* directory, to the current working directory

```
mv ../Linux/S_typhi.embl . [enter]
```

Use the **ls** command to check the contents of the current directory to see that *S_typhi.embl* has been moved.

../Linux/S_typhi.embl specifies that *S_typhi.embl* is in the *Linux* directory. If the file was in the directory above, the path would change to: *../S_typhi.embl*



```
wt@Pathogens: ~/Module_1_Artemis
wt@Pathogens:~/Module_1_Artemis$ ls
exercise_1 exercise_2 optional_exercise
wt@Pathogens:~/Module_1_Artemis$ mv ../Linux/S_typhi.embl .
wt@Pathogens:~/Module_1_Artemis$ ls
exercise_1 exercise_2 optional_exercise S_typhi.embl
wt@Pathogens:~/Module_1_Artemis$
```

The command can also be used to rename a file in the current working directory. Previously we used the **cp** command, but **mv** provides an alternative without the need to delete the original file. Therefore we could have used:

```
mv AL513382.embl S_typhi.embl [enter]
```

instead of:

```
cp AL513382.embl S_typhi.embl [enter]
rm AL513382.embl [enter]
```

Viewing what you've got

less Display file contents

This command displays the contents of a specified file one screen at a time.

You are now going to look at the contents of `S_typhi.embl`.

```
less S_typhi.embl [enter]
```

The contents of `S_typhi.embl` will be displayed one screen at a time, to view the next screen press the **space bar**. **less** can also scroll backwards if you hit the **b** key. Another useful feature is the **slash key**, `/`, to search for a word in the file. You type the word you are looking for and press enter. The screen will jump to the next occurrence and highlight it.

As `S_typhi.embl` is a large file this will take a while, therefore you may want to escape or exit from this command. To exit press the letter `'q'`. If you really need to exit from a program and it isn't responding press **'control'** and the letter **'c'** at the same time.

```
wt@Pathogens: ~/Module_1_Artemis
ID   STYPHCT18   standard; circular DNA; PRO; 4809037 BP.
XX
AC   AL513382;
XX
SV   AL513382.1
XX
DT   18-JUL-2002 (Rel. 72, Created)
DT   18-JUL-2002 (Rel. 72, Last updated, Version 2)
XX
DE   Salmonella enterica serovar Typhi (Salmonella typhi) strain CT18, c
omplete
DE   chromosome
XX
KW   .
XX
OS   Salmonella enterica subsp. enterica serovar Typhi
OC   Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriaceae;
OC   Salmonella; Salmonella enterica.
XX
S_typhi.embl
```

head Display the first ten lines of a file
tail Display the last ten lines of a file

Sometimes you may just want to view the text at the beginning or the end of a file, without having to display all of the file. The head and tail commands can be used to do this.

You are now going to look at the beginning of `S_typhi.embl`.

```
head S_typhi.embl [enter]
```

To look at the end of `S_typhi.embl` type:

```
tail S_typhi.embl [enter]
```

The amount of the file that is displayed can be increased by adding extra arguments. To increase the number of lines viewed from 10 to 100 add the `-100` argument to the command. For example to view the last 100 lines of `S_typhi.embl` type:

```
tail -100 S_typhi.embl [enter]
```

Do this for both head and tail commands. What type of information is at the beginning and end of the EMBL format file?

```
wt@Pathogens: ~/Module_1_Artemis
wt@Pathogens:~/Module_1_Artemis$ head S_typhi.embl
ID   STYPHCT18   standard; circular DNA; PRO; 4809037 BP.
XX
AC   AL513382;
XX
SV   AL513382.1
XX
DT   18-JUL-2002 (Rel. 72, Created)
DT   18-JUL-2002 (Rel. 72, Last updated, Version 2)
XX
DE   Salmonella enterica serovar Typhi (Salmonella typhi) strain CT18, complete
wt@Pathogens:~/Module_1_Artemis$ tail S_typhi.embl
cgacgcgctc cacgatgtgg attttaccgt agcgcgcaca gcccgagcc gggcaaaatt   4808580
tcattactac gcttcgcccg ctgaactggt tccttatta caggaaaaat cacgctggat   4808640
ggtcatgcc  gcgctggttt ttggccgtga ggattccggt ctgaccaacg acgagctggc   4808700
gctggcggat gtattgaccg gcgtgccgat ggcggcggat tacccttcgc tcaatctggg   4808760
tcaggcggtc atggtgtatt gctatcaatt agcaggttta atgcaacaga ccccggaatc   4808820
cgttgatatt gctgatgaat cgcagttaca ggcgttacgc ggcgccttt tacgctgct   4808880
aaccactctg gaggcggccg atgaccacaa attaaccgac tggctacaac agcgaatcgg   4808940
cctgctggga cagcgagata cggcaatggt gcaccgtttg gtccatgata ttgaaaaaaa   4809000
actaacaaaa taacgtgttg taatttttaa aataata   4809037
//
wt@Pathogens:~/Module_1_Artemis$ █
```

cat Join files together

Having looked at the beginning and end of the `S_typhi.embl` file you should notice that in EMBL format files the annotation comes first, then the DNA sequence at the end.

If you had two separate files containing the annotation and the DNA sequence, both in EMBL format, it is possible to concatenate or join the two together to make a single file like the `S_typhi.embl` file you have just looked at. The Linux command **cat** can be used to join two or more files into a single file. The order in which the files are joined is determined by the order in which they appear in the command line.

For example, we have two separate files, `MAL13P1.dna` and `MAL13P1.tab`, that contain the DNA and annotation, respectively, from the *P. falciparum* genome.

Return to the Linux directory using the **cd** command:

```
cd ../Linux [enter]
```

By typing the command line:

```
cat MAL13P1.tab MAL13P1.dna > MAL13P1.embl [enter]
```

`MAL13P1.tab` and `MAL13P1.dna` will be joined together and written to a file called `MAL13P1.embl`

The `>` symbol in the command line directs the output of the **cat** program to the designated file `MAL13P1.embl`

```
wt@Pathogens: ~/Linux
wt@Pathogens:~/Module_1_Artemis$ cd ../Linux/
wt@Pathogens:~/Linux$ cat MAL13P1.tab MAL13P1.dna > MAL13P1.embl
wt@Pathogens:~/Linux$ ls
MAL13P1.dna MAL13P1.embl MAL13P1.tab Malaria.fasta
wt@Pathogens:~/Linux$
```

wc Counts the lines, words or characters

By typing the command line:

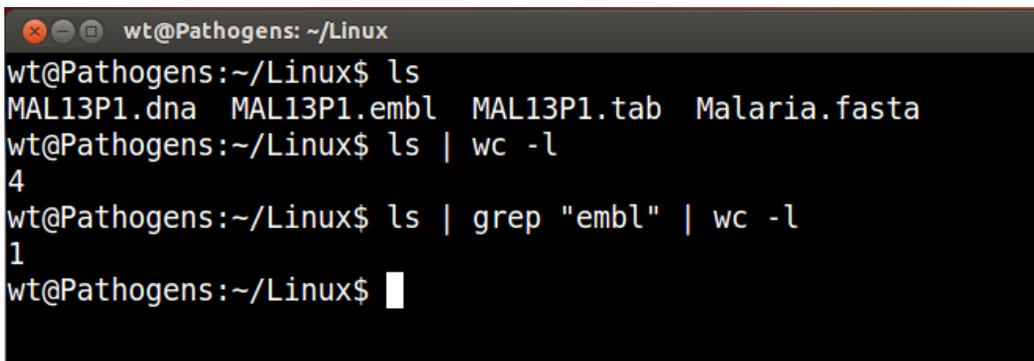
```
ls | wc -l [enter]
```

The above command uses **wc** to count the number of files that are listed by **ls**. The '-l' option tells **wc** to return a count of the number of lines.

The | symbol (known as the 'pipe' character) in the command line connects the two commands into a single operation for simplicity. You can connect as many commands as you want:

```
ls | grep ".embl" | wc -l
```

This command will list out all of the files in the current directory, then send the results to the **grep** command which searches for all filenames containing the 'embl', then sends the results to **wc** which counts the number of lines (which corresponds to the number of files).



```
wt@Pathogens: ~/Linux
wt@Pathogens:~/Linux$ ls
MAL13P1.dna MAL13P1.embl MAL13P1.tab Malaria.fasta
wt@Pathogens:~/Linux$ ls | wc -l
4
wt@Pathogens:~/Linux$ ls | grep "embl" | wc -l
1
wt@Pathogens:~/Linux$
```

grep Searches a file for patterns

grep is a powerful tool to search for patterns in a file.

In the examples below, we are going to use the file called *Malaria.fasta* that contains the set of *P. falciparum* chromosomes in FASTA format. A FASTA file has the following format:

```
>Sequence Header
CTAAACCTAAACCTAAACCCTGAACCCTAA...
...
```

Therefore if we want to get the sequence headers, we can extract the lines that match the '>' symbol:

```
grep '>' Malaria.fasta [enter]
```

By typing the command line:

```
grep -B 1 -A 1 'aagtagggttca' Malaria.fasta [enter]
```

This command will search for a nucleotide sequence and print 1 line before (-B 1) and 1 line after (-A 1) any match. It won't find the pattern if it spans more than 1 line.

```
wt@Pathogens: ~/Linux
wt@Pathogens:~/Linux$ grep '>' Malaria.fasta
>MAL1 Plasmodium falciparum, chromosome 1, contig MAL1
>MAL2 Plasmodium falciparum, chromosome 2, contig MAL2
>MAL3 Plasmodium falciparum, chromosome 3, contig MAL3
>MAL4 Plasmodium falciparum, chromosome 4, contig MAL4
>MAL5 Plasmodium falciparum, chromosome 5, contig MAL5
>MAL6 Plasmodium falciparum, chromosome 6, contig MAL6
>MAL7 Plasmodium falciparum, chromosome 7, contig MAL7
>MAL8 Plasmodium falciparum, chromosome 8, contig MAL8
>MAL9 Plasmodium falciparum, chromosome 9, contig MAL9
>MAL10 Plasmodium falciparum, chromosome 10, contig MAL10
>MAL11 Plasmodium falciparum, chromosome 11, contig MAL11
>MAL12 Plasmodium falciparum, chromosome 12, contig MAL12
>MAL13 Plasmodium falciparum, chromosome 13, contig MAL13
>MAL14 Plasmodium falciparum, chromosome 14, contig MAL14
wt@Pathogens:~/Linux$ grep -B 1 -A 1 'aagtagggttca' Malaria.fasta
ttcacattattatgatcaacatatatcacccatagggtgtttttcatcatctatatcg
aaataagtaatatcatcctcataaatatcatagaagtagggttcagatgttttccata
ccattcaagatattatccatagtggaattatcagaatttgatatacatatttgtaaatggg
--
atccatagggtggttaacatcatctgtatcaaaataagtgatatcatcatcataaatac
atagaagtagggttcagatgttttccagatcattcaatatagtatccttaatgaaatt
atcagggtttgtatccatattcgtaaattcgttctttggttaggattatgcatactat
--
tcccagggtataatgaactattgcagcagttggatgaaacggaggagagtgtaactattgg
tgatccgaaagaagggcatatttcacaagaggataggatgcaagtagggttcatatgaa
tgacaataatacatgtgatgtcaacaatgtgatgtaatatagggtgatgtaacaatg
wt@Pathogens:~/Linux$
```

find Finds files matching an expression

The **find** command is similar to **ls** but in many ways it is more powerful. It can be used to recursively search the directory tree for a specified path name, seeking files that match a given Boolean expression (a test which returns true or false)

```
find . -name '*.embl'
```

This command will return the files which name has the .embl suffix.

```
mkdir test_directory
```

```
find . -type d
```

This command will return all the subdirectories contained in the current directory.

These are just two basic examples but it is possible to search in many other ways:

```
-mtime      search files by modifying date
-atime      search files by last access date
-size       search files by file size
-user       search files by user they belong to
```

You need to be careful with quoting when using wildcards.

The wildcard * symbol represents a string of any character and of any length.

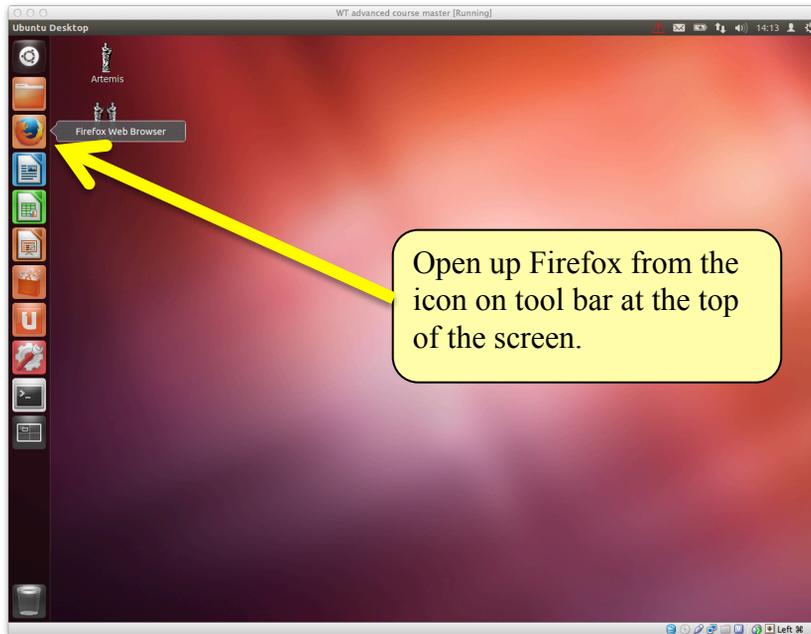
```
wt@Pathogens: ~/Linux
wt@Pathogens:~/Linux$ find . -name '*.embl'
./MAL13P1.embl
wt@Pathogens:~/Linux$ mkdir test_directory
wt@Pathogens:~/Linux$ find . -type d
.
./test_directory
wt@Pathogens:~/Linux$
```

Obtaining and transferring information

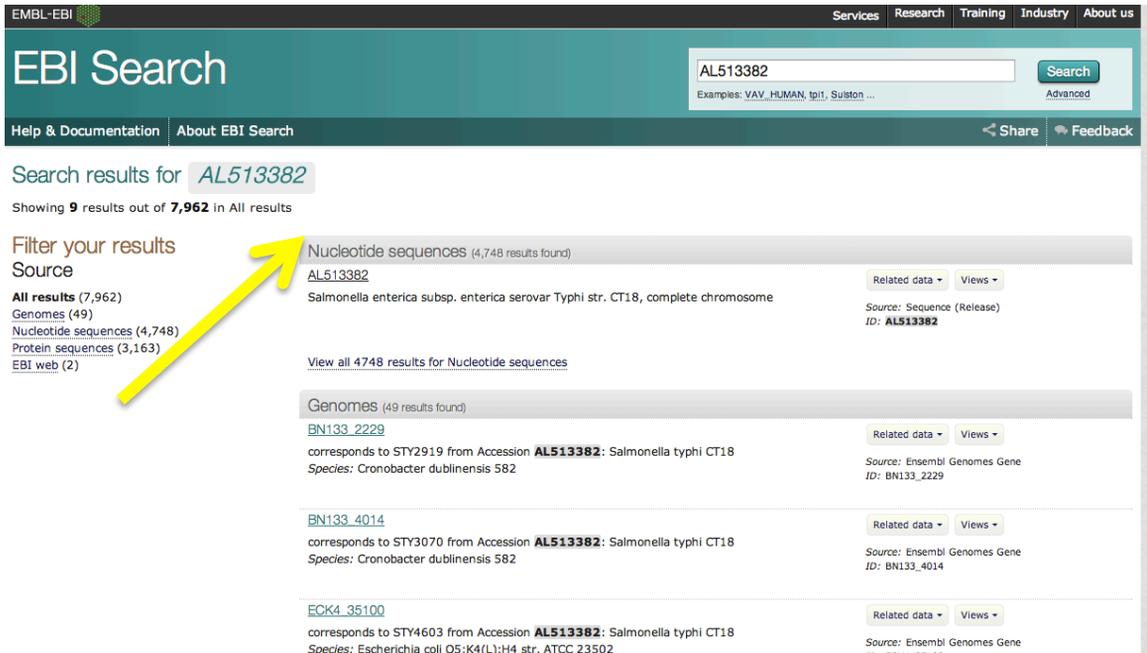
The first step in exploiting genome sequences is obtaining your genome sequence. As time goes by there are more and more genome sequences available, from an ever increasing number of locations. Typically a complete genome sequence project is quite large, and therefore the files containing the data are going to be quite unwieldy. Two methods for retrieving the genomic sequence of Salmonella Typhi are presented below. The first involves using your web browser, the second involves using the command line.

Obtaining sequences using a web browser

Open firefox by clicking on the icon at the side of the screen.



In the location box type in the address <https://www.ebi.ac.uk> and press return. In the search box type **AL513382** and click on the first result.



EMBL-EBI Services Research Training Industry About us

EBI Search

AL513382 Search Advanced

Examples: VAV_HUMAN, tp11, Sulston ...

Help & Documentation About EBI Search Share Feedback

Search results for **AL513382**

Showing 9 results out of 7,962 in All results

Filter your results

Source

- All results (7,962)
- Genomes (49)
- Nucleotide sequences (4,748)
- Protein sequences (3,163)
- EBI web (2)

Nucleotide sequences (4,748 results found)

AL513382

Salmonella enterica subsp. enterica serovar Typhi str. CT18, complete chromosome

Related data Views

Source: Sequence (Release)
ID: **AL513382**

[View all 4748 results for Nucleotide sequences](#)

Genomes (49 results found)

BN133_2229

corresponds to STY2919 from Accession **AL513382**: Salmonella typhi CT18
Species: Cronobacter dublinensis 582

Related data Views

Source: Ensembl Genomes Gene
ID: BN133_2229

BN133_4014

corresponds to STY43070 from Accession **AL513382**: Salmonella typhi CT18
Species: Cronobacter dublinensis 582

Related data Views

Source: Ensembl Genomes Gene
ID: BN133_4014

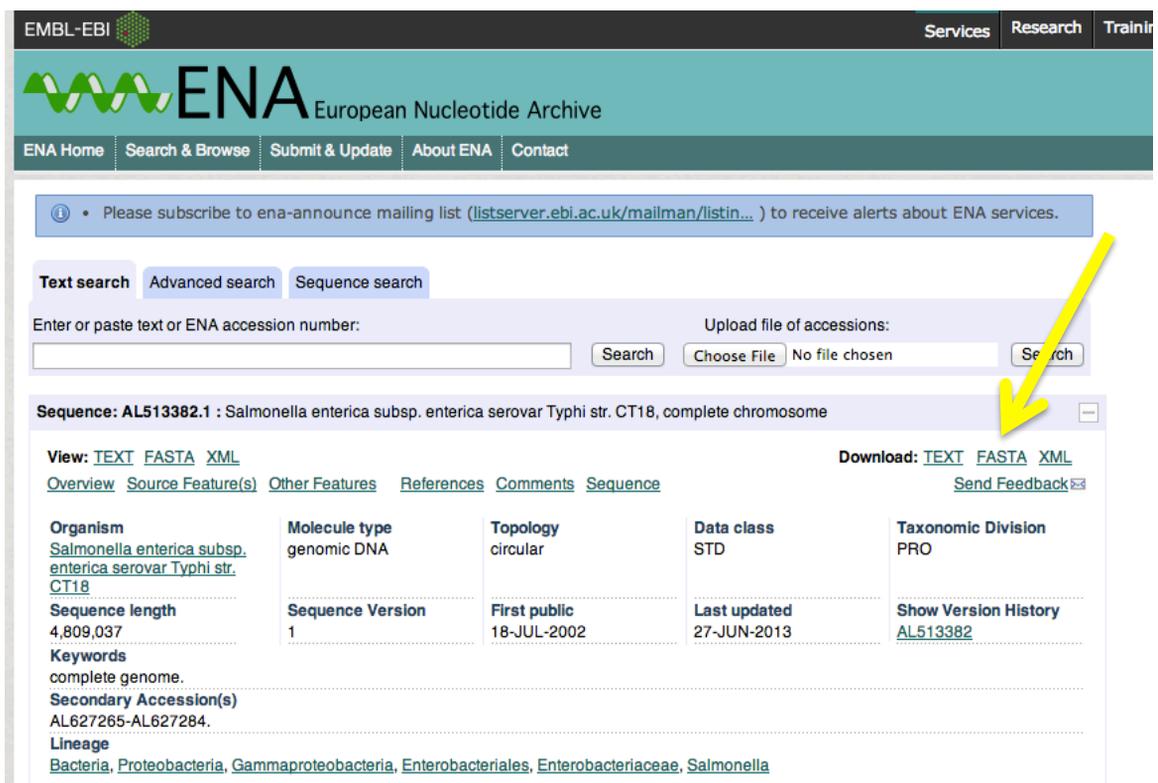
ECK4_35100

corresponds to STY4603 from Accession **AL513382**: Salmonella typhi CT18
Species: Escherichia coli O5:K4(L):H4 str. ATCC 23502

Related data Views

Source: Ensembl Genomes Gene
ID: ECK4_35100

In the download section, click on the FASTA link.



EMBL-EBI Services Research Training

ENA European Nucleotide Archive

ENA Home Search & Browse Submit & Update About ENA Contact

Please subscribe to ena-announce mailing list (listserv.ebi.ac.uk/mailman/listin...) to receive alerts about ENA services.

Text search Advanced search Sequence search

Enter or paste text or ENA accession number: Upload file of accessions: No file chosen

Sequence: AL513382.1 : Salmonella enterica subsp. enterica serovar Typhi str. CT18, complete chromosome

View: [TEXT](#) [FASTA](#) [XML](#) **Download:** [TEXT](#) [FASTA](#) [XML](#)

[Overview](#) [Source Feature\(s\)](#) [Other Features](#) [References](#) [Comments](#) [Sequence](#) [Send Feedback](#)

Organism Salmonella enterica subsp. enterica serovar Typhi str. CT18	Molecule type genomic DNA	Topology circular	Data class STD	Taxonomic Division PRO
Sequence length 4,809,037	Sequence Version 1	First public 18-JUL-2002	Last updated 27-JUN-2013	Show Version History AL513382

Keywords
complete genome.

Secondary Accession(s)
AL627265-AL627284.

Lineage
[Bacteria](#), [Proteobacteria](#), [Gammaproteobacteria](#), [Enterobacteriales](#), [Enterobacteriaceae](#), [Salmonella](#)

Save the file. It is automatically saved to the Downloads directory.

The screenshot shows the EMBL-EBI ENA (European Nucleotide Archive) website. A dialog box titled "Opening AL513382.fasta" is open, asking the user what to do with the file. The dialog indicates the file is a plain text document from http://www.ebi.ac.uk. The "Save File" option is selected in the "What should Firefox do with this file?" section. The background shows the ENA interface for accession AL513382.1, including search options, organism details (Salmonella enterica subsp. enterica serovar Typhi str. CT18), and a navigation table.

Navigation		Top
↑	Study:	PRJNA236
🏠	Taxon:	Taxon:220341
→	ENA	ERS000058

There are many different ways to download files using the command line. One command is called **wget**. It takes in a url and saves the file to disk (which in this case is a FASTA file from the ENA).

```
wget "https://www.ebi.ac.uk/ena/data/view/AL513382&display=fasta"
```

Database Entries

The `S_typhi.embl` file we previously looked at was obtained from the EMBL database at the European Bioinformatics Institute (<https://www.ebi.ac.uk/ena/>) and is presented in a specific format with a series of defined qualifiers and keys (see below and Appendix III) to help identify the different components of an entry.

Below is an example of a small EMBL entry with the different features of the entry highlighted

```

ID AF060869 standard; DNA; PRO: 27290 BP.
XX
AC AF060869;
XX
SV AF060869.1
XX
DT 17-JUL-1998 (Rel. 56, Created)
DT 17-JUL-1998 (Rel. 56, Last updated, Version 1)
XX
DE Salmonella typhimurium excision nuclease UvrA (uvrA) gene, partial cds;
DE single-strand binding protein (ssb) gene, complete cds; tRNA-Thr gene,
DE complete sequence; pathogenicity island SPI-4 operon, complete sequence;
DE yjcB gene, complete cds; and yjcC gene, partial cds.
OS Salmonella typhimurium
OC Bacteria; Proteobacteria; gamma subdivision; Enterobacteriaceae;
OC Salmonella.
XX
RN [1]
RP 1-27290
RX MEDLINE: 98298059.
RA Wong K.K., McClelland M., Stillwell L.C., Sisk E.C., Thurston S.J.,
RA Saffer J.D.;
RT "Identification and sequence analysis of a 27-kilobase chromosomal fragment
RT containing a Salmonella pathogenicity island located at 92 minutes on the
RT chromosome map of Salmonella enterica serovar typhimurium LT2";
RL Infect. Immun. 66(7):3365-3371(1998).
DR SPTREMBL: 085309; 085309.
DR SPTREMBL: 085310; 085310.
XX
FH Key Location/Qualifiers
FH
FH source 1..27290
FH /db_xref="taxon:602"
FH /organism="Salmonella typhimurium"
FH /strain="LT2"
FH /map="92 minutes"
FH CDS complement(1..312)
FH /codon_start=1
FH /db_xref="SPTREMBL:085309"
FH /transl_table=11
FH /gene="uvrA"
FH /product="excision nuclease UvrA"
FH /protein_id="ARC26637.1"
FH /translation="MKIEVRGARTHNLKINNFVIPRODKLIVVTGLSGSGKSSLAFTDL
FH YAEQRRYVESLSAYARQFLSMEKPDVDHIEGLSPAISIEQKSTSHNPRSTVGTITEI
FH "
FH CDS 583..1083
FH /codon_start=1
FH /db_xref="SPTREMBL:085310"
FH /transl_table=11
FH /gene="ssb"
FH /product="single-strand binding protein"
FH /protein_id="ARC26638.1"
FH /translation="MILVGNPQDPEVRYMPSGGAVANLTLATSESWRDKQTGEKKEQT
FH EIJHRVVMFGKLAEVAGEYLRLLSQVYIEGQLRTRKRTDQNCQERYTTELTSADRRVMIQI
FH LGGPKGGGAPAGGHNRLGSPQPQQPQGGNQFNGGAQSRPQQSAPAPSKPEPMDFFDD
FH IPF"
FH tRNA 1898..1970
FH /note="putative"
FH /product="tRNA-Thr"
XX
SQ Sequence 27290 BP: 7965 A: 5530 C: 6661 G: 7134 T: 0 other:
gatctcggta atagtaccga cgttagagcgc cggttgtgc gatglogatt tctgttcaat 60
tgaatcgcgc ggcgatagcc cctcaatagc gtcgaatccc ggttttcca tgagcagaaa 120
aaactgcgcg cgttaagcgg agagcgattc aactgaaga cctgcccctt cggcaacag
tgttattac gtatcatgco agctcgagaa gcttttcaga agaggacact tataaaaata 2520
aggcttggtt agaagacaaa atcaatagta atttattgat agaaatggtt attcctcagg 2580
//

```

Key

Qualifier

Two-character line code indicates the type of information contained in the line

EMBL Header

Annotation

Sequence

In addition to the EMBL database, there are the mirror databases, Genbank (NCBI) and DDBJ (National Institute of Genetics, Japan), which contain the same sequence entries, but have slight differences in the way in which the information is presented. The next two pages contain the text of the complete entries for same sequence from the EMBL and GenBank databases, compare the two entries and identify the differences.

EMBL Entry

```

ID   ECRSMA          standard; DNA; PRO; 500 BP.
XX
AC   L40173;
XX
SV   L40173.1
XX
DT   10-AUG-1995 (Rel. 44, Created)
DT   04-MAR-2000 (Rel. 63, Last updated, Version 4)
XX
DE   Erwinia carotovora repressor (rsmA) gene, complete cds.
XX
KW   repressor; rsmA gene.
XX
OS   Pectobacterium carotovorum
OC   Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriaceae;
OC   Pectobacterium.
XX
RN   [1]
RP   1-500
RA   Cui Y., Chatterjee A., Liu Y., Dumenyo C.K., Chatterjee A.K.;
RT   "Identification of a global repressor gene, rsmA, of Erwinia carotovora
RT   subsp. carotovora that controls extracellular enzymes,
RT   N-(3-oxohexanoyl)-L-homoserine lactone, and pathogenicity in soft-rotting
RT   Erwinia spp";
RL   J. Bacteriol. 177(17):0-0(1995).
XX
DR   GOA; Q47620; Q47620.
DR   SWISS-PROT; Q47620; CSRA_ERWCA.
XX
FH   Key           Location/Qualifiers
FH
FT   source        1..500
FT                 /db_xref="taxon:554"
FT                 /organism="Pectobacterium carotovorum"
FT                 /strain="71"
FT                 /sub_species="carotovora"
FT                 /gene="rsmA"
FT   CDS           246..431
FT                 /codon_start=1
FT                 /db_xref="GOA:Q47620"
FT                 /db_xref="SWISS-PROT:Q47620"
FT                 /note="putative"
FT                 /transl_table=11
FT                 /gene="rsmA"
FT                 /function="global repressor"
FT                 /protein_id="AAA74502.1"
FT                 /translation="MLILTRRVGETLIIGDEVTVTVLGVKGNQVRIGVNAPKEVSVHRE
FT                 EIYQRIQAEKSQPTSY"
XX
SQ   Sequence 500 BP; 140 A; 101 C; 120 G; 139 T; 0 other;
      ggatccggca agcaggatag aaagtgtggtt accttcagat attctgaagc tttacatgct          60
      cagttctggt gttgtgataa caaaagcaca agctactgat atcgactaaa ctaacaagta          120
      gtgacaaacc ggagtgtgat ggtgtgggta taccatcgtc taggtttacg ttttcacagc          180
      acatgatgga taatggcggg gagacagaga gacccgactc tttataatct ttcaaggagc          240
      aaagaatgct tattttgact cgtcagagtg gcgaaaccct catcatcggc gatgaggtaa          300
      cggttaccgt attaggagtg aaaggcaacc aggtgcgat  tgggtgtaat gcacctaag          360
      aggtttctgt ccaccgtgaa gagatctatc agcgtattca ggccgaaaaa tctcaaccaa          420
      cgtcatattg attgacaatg cgtctcgtgt tcgcgggacg caattgttat ttccggtttt          480
      tccccacac atttatcgat
//

```

GenBank Entry

```

LOCUS      ERWRSMA                      500 bp    DNA        linear    BCT 19-AUG-1995
DEFINITION Erwinia carotovora repressor (rsmA) gene, complete cds.
ACCESSION  L40173
VERSION    L40173.1  GI:927031
KEYWORDS   repressor; rsmA gene.
SOURCE     Pectobacterium carotovorum
           ORGANISM  Pectobacterium carotovorum
           Bacteria; Proteobacteria; Gammaproteobacteria; Enterobacteriaceae;
           Pectobacterium.
REFERENCE  1 (bases 1 to 500)
AUTHORS    Cui,Y., Chatterjee,A., Liu,Y., Dumenyo,C.K. and Chatterjee,A.K.
TITLE      Identification of a global repressor gene, rsmA, of Erwinia
           carotovora subsp. carotovora that controls extracellular enzymes,
           N-(3-oxohexanoyl)-L-homoserine lactone, and pathogenicity in
           soft-rotting Erwinia spp
JOURNAL    J. Bacteriol. 177(17) (1995) In press
COMMENT    Original source text: Erwinia carotovora (strain 71, sub_species
           carotovora) DNA.
FEATURES   Location/Qualifiers
           source          1..500
                               /organism="Pectobacterium carotovorum"
                               /strain="71"
                               /sub_species="carotovora"
                               /db_xref="taxon:554"
           gene            107..431
                               /gene="rsmA"
           -10_signal      107..112
                               /gene="rsmA"
           RBS             235..239
                               /gene="rsmA"
           CDS             246..431
                               /gene="rsmA"
                               /function="global repressor"
                               /note="putative"
                               /codon_start=1
                               /transl_table=11
                               /protein_id="AAA74502.1"
                               /db_xref="GI:927032"
                               /translation="MLILTRRVGETLIIGDEVTVTVLGVKGNQVRIGVNAPKEVSVHR
           EEIYQRIQAEKSQPTSY"
BASE COUNT      140 a      101 c      120 g      139 t
ORIGIN
      1 ggatccggca agcaggatag aaagtgtggtt accttcagat attctgaagc tttacatgct
      61 cagttctggt gttgtgataa caaaagcaca agctactgat atcgactaaa ctaacaagta
     121 gtgacaaacc ggagtgtgat ggtgtgggta taccatcgtc taggtttacg ttttcacagc
     181 acatgatgga taatggcggg gagacagaga gaccogactc ttataaatct ttcaaggagc
     241 aaagaatgct tattttgact cgtcgagttg gcgaaacctt catcatcggc gatgaggtaa
     301 cggttaccgt attaggagtg aaaggcaacc aggtgcgtat tgggtgtaat gcacctaaag
     361 aggtttctgt ccaccgtgaa gagatctatc agcgtattca ggccgaaaaa tctcaaccaa
     421 cgtcatattg attgacaatg cgtctcgtgt tcgcgggacg caattgttat ttccggtttt
     481 tccccacac atttatcgat

```

//

The two entries shown above contain the same biological information but differ in the format and presentation of this information. One of the most obvious difference is in the header region of the file that gives the background information to the submitted sequence. Another clear difference is that the EMBL entry has an additional **two letter line code** on the left hand margin.

EMBL entries are structured so as to be usable by human readers as well as by computer programs. The explanations, descriptions, classifications and other comments are in ordinary English for readability. At the same time, the structure is systematic enough to allow computer programs to easily read, identify, and manipulate the various types of data included.

Each line begins with a **two letter line code**, which indicates the type of information contained in the line. The currently used line types, along with their respective line codes, are listed below.

ID - identification	(begins each entry; 1 per entry)
AC - accession number	(>=1 per entry)
SV - new sequence identifier	(>=1 per entry)
DT - date	(2 per entry)
DE - description	(>=1 per entry)
KW - keyword	(>=1 per entry)
OS - organism species	(>=1 per entry)
OC - organism classification	(>=1 per entry)
OG - organelle	(0 or 1 per entry)
RN - reference number	(>=1 per entry)
RC - reference comment	(>=0 per entry)
RP - reference positions	(>=1 per entry)
RX - reference cross-reference	(>=0 per entry)
RA - reference author(s)	(>=1 per entry)
RT - reference title	(>=1 per entry)
RL - reference location	(>=1 per entry)
DR - database cross-reference	(>=0 per entry)
FH - feature table header	(0 or 2 per entry)
FT - feature table data	(>=0 per entry)
CC - comments or notes	(>=0 per entry)
XX - spacer line	(many per entry)
SQ - sequence header	(1 per entry)
bb - (blanks) sequence data	(>=1 per entry)
// - termination line	(ends each entry; 1 per entry)

Unix/Linux Command Reference

File Commands

ls - directory listing
ls -al - formatted listing with hidden files
cd *dir* - change directory to *dir*
cd - change to home
pwd - show current directory
mkdir *dir* - create a directory *dir*
rm *file* - delete *file*
rm -r *dir* - delete directory *dir*
rm -f *file* - force remove *file*
rm -rf *dir* - force remove directory *dir* *
cp *file1 file2* - copy *file1* to *file2*
cp -r *dir1 dir2* - copy *dir1* to *dir2*; create *dir2* if it doesn't exist
mv *file1 file2* - rename or move *file1* to *file2*
 if *file2* is an existing directory, moves *file1* into directory *file2*
ln -s *file link* - create symbolic link *link* to *file*
touch *file* - create or update *file*
cat > *file* - places standard input into *file*
more *file* - output the contents of *file*
head *file* - output the first 10 lines of *file*
tail *file* - output the last 10 lines of *file*
tail -f *file* - output the contents of *file* as it grows, starting with the last 10 lines

Process Management

ps - display your currently active processes
top - display all running processes
kill *pid* - kill process id *pid*
killall *proc* - kill all processes named *proc* *
bg - lists stopped or background jobs; resume a stopped job in the background
fg - brings the most recent job to foreground
fg *n* - brings job *n* to the foreground

File Permissions

chmod *octal file* - change the permissions of *file* to *octal*, which can be found separately for user, group, and world by adding:

- 4 - read (r)
- 2 - write (w)
- 1 - execute (x)

Examples:

chmod 777 - read, write, execute for all
chmod 755 - rwx for owner, rx for group and world
 For more options, see **man chmod**.

SSH

ssh *user@host* - connect to *host* as *user*
ssh -p *port user@host* - connect to *host* on port *port* as *user*
ssh-copy-id *user@host* - add your key to *host* for *user* to enable a keyed or passwordless login

Searching

grep *pattern files* - search for *pattern* in *files*
grep -r *pattern dir* - search recursively for *pattern* in *dir*
command* | grep *pattern - search for *pattern* in the output of *command*
locate *file* - find all instances of *file*

System Info

date - show the current date and time
cal - show this month's calendar
uptime - show current uptime
w - display who is online
whoami - who you are logged in as
finger *user* - display information about *user*
uname -a - show kernel information
cat /proc/cpuinfo - cpu information
cat /proc/meminfo - memory information
man *command* - show the manual for *command*
df - show disk usage
du - show directory space usage
free - show memory and swap usage
whereis *app* - show possible locations of *app*
which *app* - show which *app* will be run by default

Compression

tar cf *file.tar files* - create a tar named *file.tar* containing *files*
tar xf *file.tar* - extract the files from *file.tar*
tar czf *file.tar.gz files* - create a tar with Gzip compression
tar xzf *file.tar.gz* - extract a tar using Gzip
tar cjf *file.tar.bz2* - create a tar with Bzip2 compression
tar xjf *file.tar.bz2* - extract a tar using Bzip2
gzip *file* - compresses *file* and renames it to *file.gz*
gzip -d *file.gz* - decompresses *file.gz* back to *file*

Network

ping *host* - ping *host* and output results
whois *domain* - get whois information for *domain*
dig *domain* - get DNS information for *domain*
dig -x *host* - reverse lookup *host*
wget *file* - download *file*
wget -c *file* - continue a stopped download

Installation

Install from source:

./configure
make
make install
dpkg -i *pkg.deb* - install a package (Debian)
rpm -Uvh *pkg.rpm* - install a package (RPM)

Shortcuts

Ctrl+C - halts the current command
Ctrl+Z - stops the current command, resume with **fg** in the foreground or **bg** in the background
Ctrl+D - log out of current session, similar to **exit**
Ctrl+W - erases one word in the current line
Ctrl+U - erases the whole line
Ctrl+R - type to bring up a recent command
!! - repeats the last command
exit - log out of current session

* use with extreme caution.

