**Bioinformatics**

**The command-line**Top of Form

**Table of contents**

1. [**Introduction**](https://www.hadriengourle.com/tutorials/command_line/#introduction)
2. [**Why use Unix?**](https://www.hadriengourle.com/tutorials/command_line/#why-use-unix)
3. [**Getting started**](https://www.hadriengourle.com/tutorials/command_line/#getting-started)
4. [**The command line**](https://www.hadriengourle.com/tutorials/command_line/#the-command-line_1)
   1. [**Command line Arguments**](https://www.hadriengourle.com/tutorials/command_line/#command-line-arguments)
   2. [**Files and Directories**](https://www.hadriengourle.com/tutorials/command_line/#files-and-directories)
   3. [**General Points**](https://www.hadriengourle.com/tutorials/command_line/#general-points)
   4. [**Some useful Unix commands Command and What it does**](https://www.hadriengourle.com/tutorials/command_line/#some-useful-unix-commands-command-and-what-it-does)
   5. [**Firts steps**](https://www.hadriengourle.com/tutorials/command_line/#firts-steps)
   6. [**Changing and moving what you’ve got**](https://www.hadriengourle.com/tutorials/command_line/#changing-and-moving-what-youve-got)

**The command-line**

The aim of this module is to introduce Unix and cover some of the basics that will allow you to be more comfortable with the command-line. This module is only designed to provide a very brief introduction to some of the features and useful commands of Unix. During this module we will also obtain a genome sequence and examine the basic structure of an own data/EMBL entry.

**Introduction**

Unix 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. Unix 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 by different names but they mostly can do the same things. Unix is a powerful, secure, robust and stable operating system that 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 is run on all kind of machines, like mobile phones (Android), desktop PCs, kitchen appliances,. all the way up to supercomputers. Unix powers the majority of the Internet.

**Aims**

The aim of this course is to introduce Unix and cover the basics. The programs that you are going to use during the courses, plus many others that are useful for bioinformatics analyses, are run in Unix. This module is designed to provide a very brief introduction to some of the features and useful commands of Unix. During this module we will also obtain a genome sequence and examine the basic structure of an own data /EMBL entry.

**Why use Unix?**

* Unix is a well-established, very widespread operating system. You probably have a device running on Unix in your home without realising it (e.g. playstation, TV box, wireless router, android tablets/phones etc.
* 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 Unix based.
* It is used on many of the powerful computers at bioinformatics centres and also on many desktops and laptops (MacOS is largely UNIX compatible).
* The major difference between Unix 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 Unix such as Ubuntu, RedHat, Fedora, Mint,...). These are all Unix, but they bundle up extra software in a different way or combinations. Some are known for being conservative and reliable; whilst others are know for being cutting-edge (and less reliable).
* ~~The MacOSX operating system used by the [eBioKit](https://www.hadriengourle.com/tutorials/command_line/77.235.253.122) is also based on Unix.~~

Getting started

For this course, you will have to connect to the eBiokit using SSH. SSH stands for Secure Shell and is a network protocol used to securely connect to a server. To do so, you will need an SSH client:

* On Linux: it is included by default, named Terminal.
* On MacOS: it is included by default, also named Terminal.
* On Windows: you'll have to download and install [MobaXterm](http://mobaxterm.mobatek.net/), a terminal emulator.

~~Once you've opened your terminal (or terminal emulator), type~~

~~ssh username@ip\_address~~

~~replacing username and ip\_address with your username and the ip address of the server you are connecting to. Type your password when prompted. As you type, nothing will show on screen. No stars, no dots. It is supposed to be that way. Just type the password and press enter!~~

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 Unix 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 Unix that still survives.

The command line

All Unix programs may be run by typing commands at the Unix 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 Unix command for example **ls**, **mv** or **cd** at the Unix prompt with the appropriate variables such as files names or directories will result in the tasks being performed on pressing the enter key.

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 List the contents of a directory with extra information about the files **ls –l** List the contents of a directory with extra information about the files **ls –a** List all contents including hidden files & directories **ls -al** List all contents including hidden files & directories, with extra information about the files **ls –l /usr/** List the contents of the directory /usr/, with extra information about the files

Files and Directories

Directories are the Unix equivalent of folders on a PC or Mac. They are organised in a hierarchy, so directories can have sub-directories. 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. Unix 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**

Unix is pretty straightforward, but there are some general points to remember that will make your life easier: most flavors of UNIX are case sensitive - typing **ls** is generally not the same as typing **LS**. You need to put a space between a command and its argument - for example, **less my\_file** will show you the contents of the file called my\_file; **lessmyfile** will just give you an error! Unix is not psychic: If you misspell 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 Unix, it is due to a spelling mistake, or perhaps you have omitted a space. If you want to know more about Unix 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.

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\*\* \*any\_directory\* [Enter] means "at the Unix prompt $, type ls followed by the name of some directory, then press Enter" Don't forget to press the [Enter] key: commands are not sent to the computer until this is done.

Some useful Unix commands Command and What it does

| Command | What it does |
| --- | --- |
| **Ls** | Lists the contents of the current directory |
| **mkdir** | Creates 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 |

Firts steps

The following exercise introduces a few useful Unix commands and provides examples of how they can be used. Many people panic when they are confronted with an Unix 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 Unix, 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 ngs\_course\_data directory. Remember, Unix is case sensitive **PWD** is not the same as **pwd**

**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 "ngs\_course\_data" directory below. To do this type:

cd ngs\_course\_data [enter]

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

Change again the directory to Module\_Unix

**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 several other directories.

Now use the cd command again to change to the Module\_Unix directory.

Changing and moving what you’ve got

**cp**  
Copy a file. **cp file1 file2** is 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 (we'll learn more about file formats later during the course). 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.

**rm** Delete a file. This command removes a file permanently, so be careful! You are now going to remove the old version of the *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.

Unix, 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.

**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\_Artemis directory. To change to the Module\_Artemis directory type: cd Module\_Artemis [enter] use the ls command to check the contents of the directory.

**Tips**

There are some short cuts for referring to directories:

.  Current directory (one full stop)

.. Directory above (two full stops)

~  Home directory (tilde)

/  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 Unix 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 Unix 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 Module\_Unix directory, to the current working directory. mv ../Module\_Unix/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. ../Module\_Unix/S\_typhi.embl specifies that S\_typhi.embl is in the Module\_Unix directory. If the file was in the directory above, the path would change to: ../ S\_typhi.embl

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.

**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 number of lines that are 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?

**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 Unix 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 Module\_Unix directory using the cd command: cd ../Module\_Unix [enter]

and type 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

**wc**  
Counts the lines, words or characters of files. 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).

**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 nucliotide sequence and print 1 line before and after any match. It won’t find the pattern if it spans more than 1 line.

**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.

**Tips**

You need to be careful with quoting when using wildcards!

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

 For more information on Unix command see EMBNet UNIX Quick Guide.