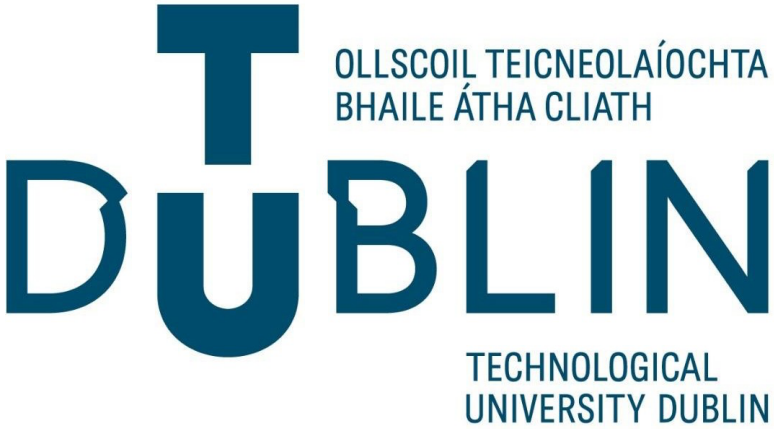


# Goals

In this lab we're going to

- Get familiar with common unix/linux commands in Bash

# Bash and common unix/linux commands

This is a short introduction to commands, arguments and options.

Command Syntax
The general syntax followed by any Bash command is:

command_name [-option(s)] [argument(s)]
Let's divide this syntax and understand what each of the terms mean:

Command Name:
A command name is a unique word which is used to indicate to the system what action needs to be performed. Each command has its own set of arguments and options to narrow down the functionality even more. For example, `ls` is a very commonly used command.

Argument:
Any Bash command takes a list of arguments to indicate to the system which objects to look for while performing the action. An argument could be a string, set of string or a token passed to the command. For example, the command ls can take a directory's path as an argument.

ls /home/user/Downloads/Music

Options:
An option is also referred as the “mode” of the command. It controls the behavior of the command. It is a single character carry that carries a unique meaning. Most of the commands run without the option. Some commands can also take multiple sets of options simultaneously. For example, the command ls can take -a as an option which generally means “all” and is used to show hidden files.

# Tasks

Open up a terminal window in your codespace.

**Task 1:** Find out where you are in the filesystem by typing

```
pwd
```

What does this information tell you?

**Task 2:** Next, find out your own username that you are logged in as.

```
whoami
```

Find out your own username

Getting info about commands

`man` is used to format or display the manual pages. Manual pages or man pages are the official documentation of all shell commands that comes along with the Linux distributions. You use the `q` command to exit.

**Task 3:** Get information about the pwd command (use q to exit)

**Listing files in a directory**

`ls` is used to list files in a directory.

The syntax is `ls [option] file_name`

These are some of the options:

| Option | Meaning |
| --- | --- |
| -a | List all the files, even those starting with `.` |
| -C | List files column-wise |
| -i | To show the index number of each file |
| -s | To show the size of each file |
| -1 | To show one file per line |

**Task 4:** What command would I use to find the rest of the options?

**Task 5:** How would I list all files in my current directory, including those that are hidden and start with a `.`?

### Navigate between directories

cd — short for “change directory” — is used to change Shell’s current working directory. We use this command frequently to work around the hierarchy of our file system.

Example:
To move from current directory to parent directory and list it’s contents:

```
cd ..
ls
```

`cd` on it’s own will bring you back to your home directory. Try it.

**Task 6: How can you verify this? What command can you use to print out the current directory you are in?**

### Creating Directories

mkdir

Definition:
mkdir is one of the most important directory manipulation commands. It’s short for ‘make directory’ and as the name suggests, it is used for creating new directories within the current directory. You can create any number of directories with this command. It takes the directory name as an argument.

Syntax:
mkdir [options][dir_name]

some mkdir options:

| Option | Meaning |
| --- | --- |
| -p | Means Parent or Path. It is used when we want to create a directory within a directory that doesn't already exist |
| -m | Means Mode. It is used to specify and control permission modes |

Example:
**Task 7:** Run the following

```
mkdir my_dir_1 my_dir_2 my_dir_3
ls
```

You should see three directories in the directory

### Remove a directory in Bash

```
rmdir
```

Definition:
rmdir is the opposite of mkdir. It is used to remove the directory name given in the argument. This command is mainly used to remove empty directories. Just like you can make muliple directories at the same time, you can also delete multiple directories simultaneously by passing their names as arguments.

Syntax:

```
rmdir [option] [dir_name]
```

**Task 8**: Run all the following commands from the same directory as Task 7

```
echo "Current Directory Structure"
 ls

 rmdir my_dir_1
 echo "Updated Directory Structure"
 ls
```

### Create a File in Bash

```
touch
```

`touch` command is considered one of the easiest ways to create new files because it does not overwrite the existing files unlike `cp`, `rm` etc.

Syntax:

```
touch [option] [file_name(s)]
```

**Task 9**: Display a message using the echo command 'Current directory files'. List the directory. Create a new file called new_file.
Display another message using the echo command 'Updated directory files'. List the updated directory.

**Task 10**: Play around with the different options of touch. Do a `man touch` to find out what they are and mean. Do some examples of your learnings with some new files.

### Remove a file in Bash

```
rm
```

`rm` is used to delete files and directories. By default, it is only set to remove files and not directories for safety. It basically unlinks the file name with the data stored in so that it cannot be accessed anymore.

Example: Deleting multiple files at once:

**Task 11**: Run the following commands in your terminal:

```
touch file_1 file_2 file_3 file_4 #Create 4 files
echo "Current Directory Files:"
ls

rm file_1 file_2 file_3
echo "Updated Directory Files:"
ls
```

### Copying a file in Bash

```
cp
```

This command is one of the most frequently used commands. It is used to copy files and directories. By default, `cp` copies files but not directories.

Syntax:
cp [option] [file_name] new_file_name

Options

| Option | Meaning |
| --- | --- |
| -r or -R | Means *Recursive*. It is used to copy directories including all its content |
| -i | This option is only used to warn the user about overwrite issues |
| -b | Used to make backup copies |
| -f | Means *Force*. This option is used to force open the destination files |
| -u | Means *Update*. As the name suggests, it only updates the file if any changes are made in the file |
| -x | This option is to indicate `cp` to stay on the same file system |
| -s | By using this option you can only make references rather than deep copying everything |

Wildcards

Wildcards are commonly used in Shell commands. Some of the most frequently used are: ?, #, [] etc. The star wildcard simply means “any” or “every”. Thus, the star wildcard, wherever used, would mean to return “any” of the possible result or object.

Example: Copy all files from current directory to “my_directory” with only .txt extension

cp *.txt my_directory

**Task 12**:

Type the following commands
touch a.md
touch b.txt
touch c.java
mkdir test

Using the wildcard, copy all java files to the test directory.
Ensure there is just one file in the directory.

We can also prefix a wildcard with more specific information. Ie.

cp b* test

This will copy all files starting with b to the folder test.

**Task 13:** Try this out. Using a more specific wildcard, copy all files beginning with c to the test directory. What is the command for this? Execute it.